LTE Monitoring

Master Thesis

Author(s):
Kotuliak, Martin

Publication date:
2020

Permanent link:
https://doi.org/10.3929/ethz-b-000462184

ETH
Eidgenössische Technische Hochschule Zürich
Swiss Federal Institute of Technology Zurich

inf | Informatik
Computer Science

LTE Monitoring

Master Thesis

Martin Kotuliak

19th December, 2020

Advisors:
Dr. Marc Röschlin
Patrick Leu
Prof. Dr. Srdjan Čapkun

Department of Computer Science, ETH Zürich

Abstract

As mobile networks, specifically LTE networks, are a key infrastructure nowadays, the security of them becomes a priority. A passive attacker is undetectable and uses flaws in the protocol against victims. In this work, we build a sniffer, essential passive attacker tool recording all the communication happening between the parties. Our sniffer is both affordable and white-box. Compared to other sniffers it can be used for both *downlink* and *uplink* communication.

We show the capabilities of Sniffer by introducing two new attacks. *UE Fingerprinting* attack detects the victim's phone model helping identification and *Timing Advance Attack* localizes a victim.

In our experiment, the passive attacker estimates a distance of the phone from the sniffer. We measure the distance at six different locations up to 60m away. The attacker can 90% of the time estimate the distance with less than 6m error. For the fingerprinting, we build both supervised and unsupervised machine learning models which perfectly classify manufacturers of mobile phones.

Acknowledgements

Firstly, I would like to thank Patrick and Marc for their support and guidance throughout the whole Master thesis. Secondly, I would like to thank Prof Čapkun for the opportunity to work on such a cool project with the best hardware. Finally, I want to thank my co-worker Simon for interesting discussions and fun breaks while working on the thesis.

I would not be able to finish my Master's without my best flatmates, with whom we had such a good time in Zurich. The biggest thanks go to my Mom, Dad and Sister for always being there for me.

Contents

Chapter 1

Introduction

Mobile networks are now core infrastructure for our day-to-day life. We use mobile networks constantly, always being connected to them by at least one device if not more. Even though infrastructure for 5G networks is being built right now, it is still LTE protocol which plays the main role.

With new data breaches publicized in the media, people care about privacy more than ever before. Mobile networks are a perfect way how to access private information. Cellular providers need to know user identity and location to enable them the service. Security of mobile network protocols is an utmost priority. It was not always the case, as there are multiple attacks against 2G and 3G protocols. We investigate if the LTE protocol, as the most used protocol, is vulnerable to privacy leaks.

Papers [1, 2, 3, 4] showed how LTE is vulnerable by an active attacker. It was shown at the same time (i.e. [5]) that active attackers are possible to detect. Passive attackers, on the other hand, cannot be observed, making them more powerful in real-world scenarios.

A passive attacker cannot use already existing devices such as rogue base stations or relays to perform the attacks. New tools must be developed for future work. The most important is a sniffing device which records the traffic between the users and base stations. Open-source libraries for LTE protocols, such as srsLTE [6], make it possible to build such tools. In the past, wireless networks were often inaccessible by attackers due to a simple reason: cost. Nowadays, software defined radios are cheap, and anyone can afford them. In this work, we build new sniffer for both uplink and downlink communication with only open-source tools and software defined radios. This is the first affordable and white-box solution for passive attacks against LTE networks. To showcase its functionality we propose two attacks which leak location and mobile phone model of the users.

Each phone model has different capabilities which can be used by the LTE

protocol. Since there are many possible combinations of capabilities, an attacker can figure out model just by receiving them and comparing them to her database. We show the details in UE Fingerprinting attack.

The phone needs to be tightly synchronized to the base station, otherwise, it would not be able to receive or transmit the messages. Base station estimates the round trip time and informs the UE about the propagation delay. A passive attacker can use the same procedure to learn the distance to the phone, locating the user in the world. We call this attack a Timing Advance Attack.

These two attacks show how the LTE protocol is flawed by design. We propose solutions to lower the risk of these attacks in the future.

1.1 Contributions

We identify following points as our key contributions:

- First affordable and white-box uplink and downlink sniffer built on top of open-source libraries. We tested our sniffer against real LTE networks.
- Introduction and evaluation of passive attack which leaks location of the users based on Timing Advance commands. The simplest version can be performed with just one device, by exploiting unencrypted communication from base stations.
- Introduction and evaluation of passive User Equipment (UE) fingerprinting based on uplink NAS messages.
- Analysis of weakly configured real-world network which leaks users' information.

1.2 Organization

We organize the document into eight chapters. We follow Introduction chapter with background in chapter 2, where we introduce the most relevant functions of LTE protocols. In chapter 3, we propose two attacks and how the attacker can combine them to track the user. Chapter 4 gives implementation details for both the sniffer and the attacks. We show our results in chapter 5. Finally, we finish with related work in chapter 6, discussion in chapter 7, and conclusion in chapter 8.

Chapter 2

Background

The 3GPP is an umbrella organisation for seven standardisation organisations designing telecommunication protocols. Standards are structured as releases. First releases specified GSM and later 3G networks. Release 8 was a first LTE specification and current Release 16 details 5G protocol. Each release spans multiple documents covering every detail of the complex procedures. We only cover the main components of the LTE protocol relevant to our work.

2.1 LTE Network Architecture

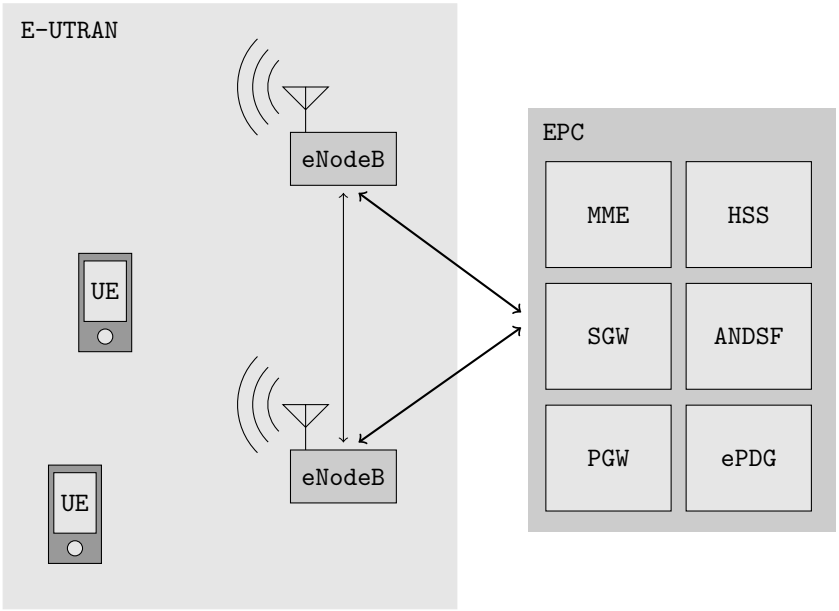


Figure 2.1: LTE Network Architecture.

The network architecture of LTE is split into Evolved Packet Core (EPC) and Evolved UMTS Radio Access Network (E-UTRAN) as seen in Figure 2.1. E-UTRAN contains cells (eNodeB), the hardware used for direct wireless connection of the mobile devices, also called User Equipment, to the network. ENodeB connects to the EPC, which handles most of the mobile network functionality. The overall description of LTE is specified in [7]

As explained in [8], users are identified in the mobile network by a persistent IMSI number. To conceal this number from attackers, temporary identifiers such as TMSI or GUTI are used. They are assigned to the users by Mobility Management Entity, a part of EPC. First 5 digits of IMSI number identify issuing mobile operator. This number is called PLNM and it is used when searching for cells by UE. UE does not connect to competitors cells.

The mobile network architecture is split into Tracking Areas consisting of multiple eNodeBs [9]. All the eNodeBs in one Tracking Area communicate with the same Mobility Management Entity. If a user switches Tracking Areas, new temporary numbers must be generated.

Since a user might use multiple devices, each device is uniquely identified by IMEI number [8]. IMEISV number specifies the software version of the phone as well. First 8 digits of IMEI number identify a device model.

In this document, we primarily focus on communication between eNodeB and UE. There is a two-way communication stream between the two entities. Downlink, going from eNodeB to UE, and uplink going in the opposite direction.

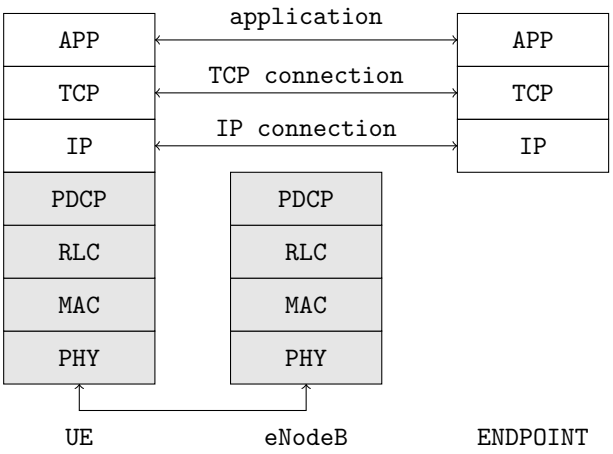
2.2 Protocol Stack

LTE protocol differentiates into control and data plane [7]. Former handling the inner working of the system, and latter the data transmission of the users. Figure 2.2 shows the two planes next to each other. First four layers of LTE radio protocol stack are the same between the two planes. Control stack then has extra RRC layer on top of PDCP.

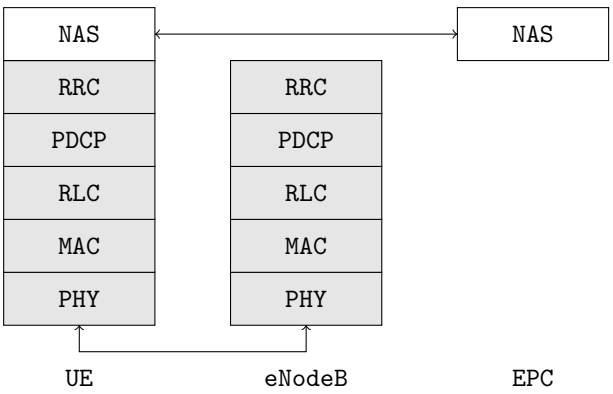
PHY handles the transmission of MAC transport channels over the air interface. It manages power control, radio channel measurements, cell search.

MAC controls mapping between logical and transport channels. The layer also handles error correction, scheduling, MAC Control Elements.

RLC transfers upper-layer data, handles error correction, performs reordering, discarding, re-segmentation, and duplicate detection of datagrams.



(a) LTE data plane protocol stack.



(b) LTE control plane protocol stack.

Figure 2.2: LTE protocol stack.

PDCP handles header compression and decompression of IP data, the transmission of higher-layer data, ciphering, and integrity protection. Data on lower layers are sent in clear-text.

RRC broadcasts system information, handles paging or key management. It manages establishment, maintenance and release of an RRC connection between the UE and E-UTRAN.

IP, TCP and APP layer in Figure 2.2a are an example for internet connection on top of LTE protocol. NAS layer in the control plane (Figure 2.2b) is used for communication between EPC and UE.

In the next section, we describe the PHY layer in more detail, as this is the most complex layer and also it is the layer we utilize the most for our project.

In our project, rest of the layers is mostly handled by the srsLTE library [6], and we did not need to make big alterations to them. We, therefore, omit their detailed description and only highlight important procedures. [7] gives overall descriptions of these layers.

2.3 PHY Layer

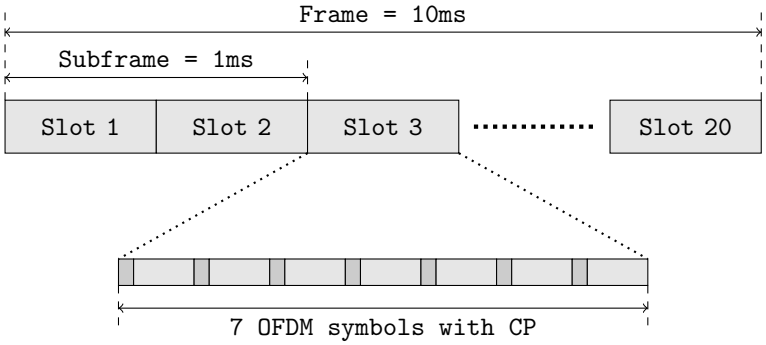


Figure 2.3: LTE Frame Structure.

Specified in [10], PHY layer data transmission is scheduled in 10ms long frames for both downlink and uplink. Frames are indexed from 0 to 1023. Each frame is split into 10 subframes each lasting 1ms. Subframes consist of two slots. By default, a slot consists of 7 OFDM symbols with cyclic prefixes (CP). Figure 2.3 visualizes the frame structure.

As mentioned in section 2.1, the transmission is two way, uplink and downlink. Currently, the providers in Switzerland use frequency division duplex. In FDD, uplink and downlink use two separate RF carriers according to [11, 12]. There are two different multiplexing methods used. OFDMA for downlink and SC-FDMA for uplink.

In both OFDMA and SC-FDMA, data is modulated onto orthogonal subcarriers. Modulated samples are called frequency samples. Using inverse fast Fourier transformation, frequency samples are transformed into time samples and transmitted over the radio. Smallest allocation element is a resource block (RB) [10] which spans over 12 subcarriers lasting one slot. RB consists of $7 \times 12 = 84$ frequency samples. Some of the frequency samples in RBs are used as PHY signals, therefore actual capacity is smaller. The number of resource blocks used on downlink and uplink is specific to each cell.

2.3.1 PHY Channels

Data on the PHY layer is sent over channels [10], which specify the type of the information. Physical shared channels are used for data transmission and control channels manage them. The Physical Random Access Channel is used for new UE connections.

On the PHY layer, UEs are identified by a temporary identifier called RNTI. Each message on the downlink is addressed by 16-bit RNTI number. This number specifies the recipient of the message. Depending on the function, the RNTI number specifies one UE or multiple UEs. The following table [13] shows the main types of RNTIs, their functionality, and the recipients.

RNTI	Values	Function	Recipient
RA-RNTI	`0x1 - 0xa`	initial message	connecting UEs
C-RNTI	`0x46 - 0xfff3`	UE communication	single UE
P-RNTI	`0xfffe`	paging messages	idle UEs
SI-RNTI	`0xffff`	system information	all UEs

Downlink

A base station sends Downlink Control Information (DCI) to the UE over Physical Downlink Control Channel (PDCCH). Format of the DCI specifies its function.

- Format 0 allocates resource blocks on the uplink to UEs. UE can transmit on PUSCH channel only if it received Format 0 DCI.
- Format 1 and 2 define which RBs UE should decode, and what parameters it should use to get the message on PDSCH.

Physical Downlink Shared Channel (PDSCH) carries data intended to the UE, System Information Blocks (SIB) or paging messages. SIB contains the configuration of the cell. Paging messages signal idle UEs to connect back when there is a message addressed to them. We further discuss states of the UE in section 2.5.

There is a special broadcast channel carrying the Master Information Block. This block contains information needed for UE to further decode SIB and connect to the eNodeB.

Uplink

UE can transmit on the Physical Uplink Shared Channel (PUSCH) only if it received DCI Format 0 on PDCCH. DCI Format 0 also specifies parameters for the encoding of the messages such as modulation scheme.

The Physical Uplink Control Channel is used for transmission of Uplink Control Information (UCI). UCI carries channel state information, acknowledgements and/or scheduling request. ENodeB specifies when UE can send UCI in the RRC configuration messages.

2.3.2 PHY signals

Physical layer signals [10] are only used for the internal working of the PHY layer.

On the downlink, there is Primary and Secondary Synchronization Signal (PSS/SSS), and Reference Signal (RS). Primary and Secondary Synchronization Signals are used for synchronization of UE to the eNodeB. Moreover, they carry an identity of the eNodeB. Reference Signal is known to both parties, by deriving it from the cell identity. UE uses it to estimate the radio channel on the downlink to decrease the number of decoding errors.

On the uplink, for the eNodeB to decode the PUSCH and PUCCH messages, Demodulation Reference Signal is sent. This is a known sequence which help eNodeB to estimate the radio channel on the uplink.

2.3.3 Cell Search

Procedure for finding neighbouring cells is not defined in 3GPP specification. According to [14], most of the UE implementations perform following, when they search for a cell:

1. UE searches for a frequency used by a neighbour cell. UE measures RSSI for frequencies and checks if the measurement is above a threshold.
2. UE uses PSS for the slot synchronization and SSS for the frame synchronization. From the values of PSS and SSS, UE computes Physical Cell Identity.
3. UE acquires quality of the service from the Reference Signal. If the quality is above a threshold, UE decodes MIB, and later SIB to learn PLNM.
4. If UE can use found PLNM, UE camps on the cell.

Once the UE is camping on the cell, it runs the synchronization process every 5ms such that it is synced to the base station even if its clock is drifting.

2.4 Random Access Procedure

Once UE selects cell it tries to connect to it. The procedure for initiating data transfer is called Random Access Procedure and it is defined in [15, 13].

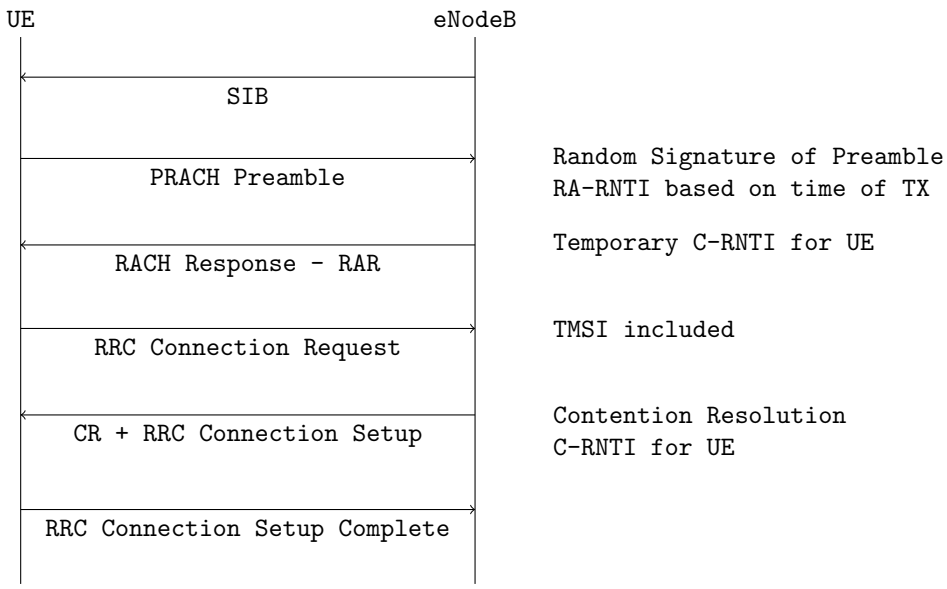


Figure 2.4: Random Access Procedure. All of the messages are sent in cleartext.

Figure 2.4 shows the protocol message flow.

It uses PRACH to transmit a preamble. The preamble is chosen randomly by UE from 64 sequences. The choice of the preamble is called the signature of the preamble. UE computes RA-RNTI it should listen for from the TX time of preamble.

Once eNodeB receives the UE transmission, it replies with Random Access Response (RAR). The message contains temporary C-RNTI which UE should use for the following messages. Moreover, RAR contains Timing Advance command (see section 2.6), and an uplink resource allocation for the next uplink message.

There are two states of UE on the RRC layer. RRC Connected and RRC Idle. For actual data transmission between UE and eNodeB, the RRC state is Connected. UE initiates RRC connection with RRC Connection Request message. ENodeB replies with Contention Resolution and RRC Connection Setup. The goal of Contention Resolution is to choose only one of UEs who transmit the same signature at the same frame over PRACH. RRC Connection Setup contains common and dedicated configurations for various layers and channels. This message confirms to UE that the temporary C-RNTI is assigned to it, and C-RNTI won't change.

Finally, UE replies with RRC Connection Setup Complete to confirm the connection.

2.5 Registration procedure

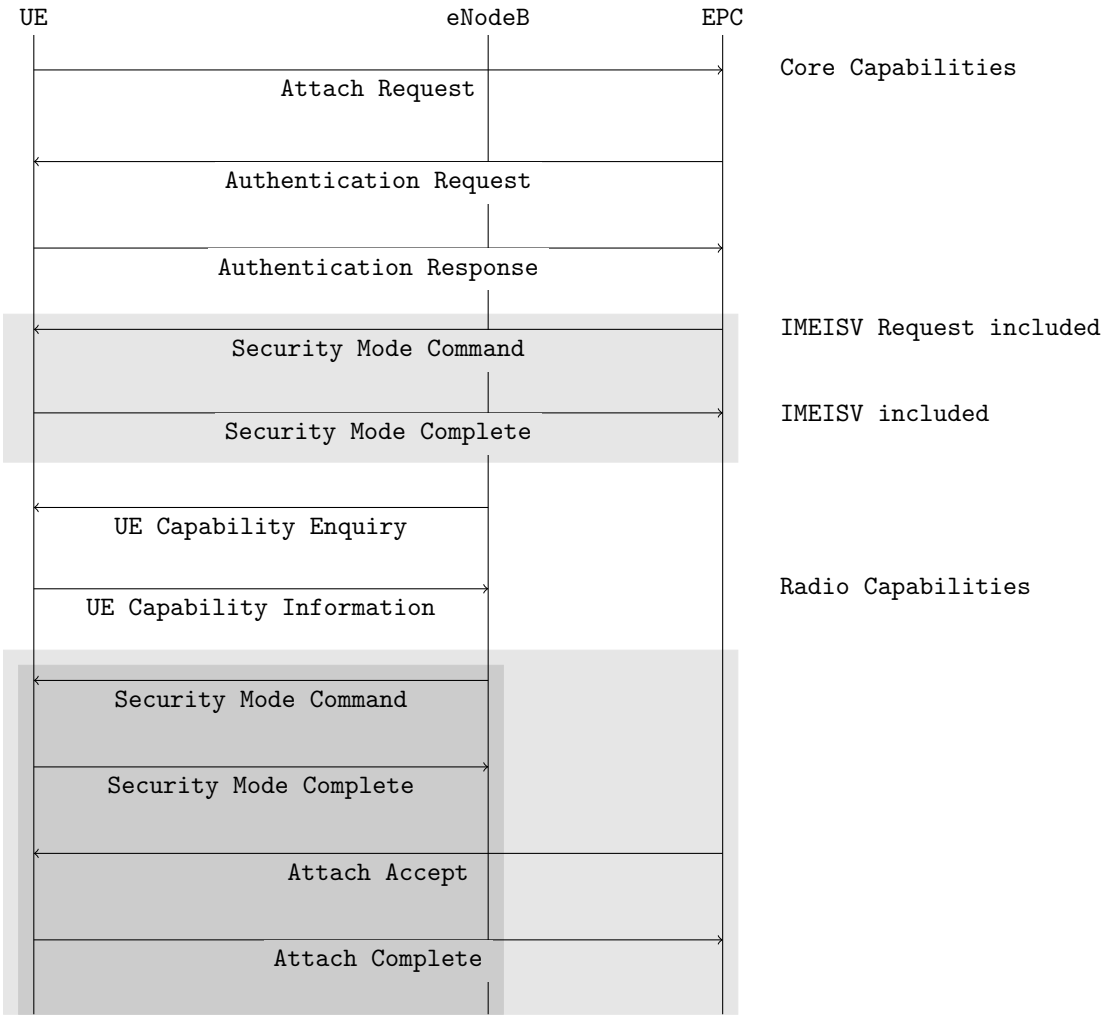


Figure 2.5: Attach Procedure. Messages in light gray area are encrypted and integrity protected with UE-EPC security context. Dark gray area corresponds to UE-eNodeB security context. The rest of the messages is sent in clear-text.

Under the Registration Procedure, we mean a series of procedures which register UE in the network and allow normal communication both between UE and eNodeB, and UE and EPC. You can follow the procedure in Figure 2.5. Some of the messages might be sent in a different order, Figure 2.5 corresponds to an actual communication intercepted between UE and eNodeB. Messages between UE and EPC are on the NAS layer, and messages

between UE and eNodeB are on the RRC layer.

The procedure is initiated by Attach Request message [16]. Attach Request contains core capability information of the UE, specifying implemented functionality EPC can utilize. Individual UEs differ and these capabilities are widely different between devices. The network first authenticates UE using keys stored in the USIM. The authentication happens in the Authentication Request and the Authentication Response.

Afterwards, Security Mode Command is sent from EPC which creates a security context [16]. From then on, all the messages between UE and EPC are ciphered and integrity protected. Security Mode Command includes a request for IMEI number which is sent back in Security Mode Complete.

Concurrently, the security context is created between UE and eNodeB with Security Mode Command on RRC level [17]. Following messages between them are ciphered and integrity protected.

Radio capabilities are exchanged between UE and eNodeB in UE Capability Information [17]. Radio capabilities differ between devices as well. UE Capability Information is sent either before or after RRC Security Mode Command. Figure 2.5 shows it before as it was recorded on an actual network. In this instance, both core and radio capabilities were sent in clear-text.

2.6 Timing Advance

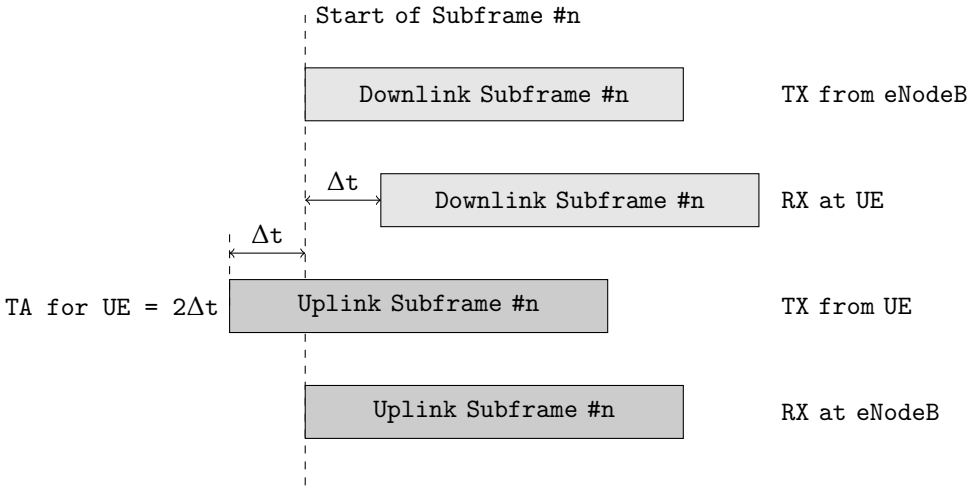


Figure 2.6: Timing Advance is used to align uplink transmissions.

Multiple UEs are connecting to eNodeB at the same time. Each UE is at

a different distance. Due to propagation delay, without any mechanism, uplink messages would come with a different delay. ENodeB needs to help correct each UE's timing to ensure alignment of all uplink messages within the resource grid as observed by the eNodeB.

Figure 2.6 shows a situation where propagation delay between the UE and eNodeB is Δt. Due to the propagation delay of the downlink message, frame synchronization of the UE is being shifted by Δt from the eNodeB time. Propagation delay of uplink message is again Δt. Therefore, uplink message arrives at eNodeB with delay of $2\Delta t$. ENodeB estimates the value of the delay and signals it to UE with Timing Advance (TA) command.

The delay is constantly measured by eNodeB with each uplink message. If the delay changes, i.e. due to change of the location, eNodeB sends to UE a new TA command.

Apart from using timing advance for synchronization, it is also used in LTE positioning protocol [18]. The feature is called Enhanced Cell ID and works by reporting channel characteristics to EPC which computes the location of UE. One of the characteristics is the Timing Advance.

2.6.1 Timing Advance Estimation

Since UE synchronizes to the eNodeB it has no notion of the propagation delay. ENodeB computes the timing advance from the uplink transmission of reference signals. The 3GPP standard does not specify how estimation of TA should be computed. It is implementation-specific.

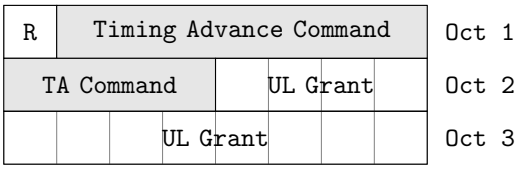
A user sends DM-RS, demodulation reference signal, which is a sequence known to both parties. Course grained timing advance computation is computed by eNodeB computing cross-correlation between the known and received signal. The offset of the correlation peak is the timing offset. This approach gives us timing advance with a granularity of one sample.

To increase the precision, eNodeB has to compare phase offset between received and known reference signal. This approach gives us finer granularity than one sample. [19] explains the estimation in more detail.

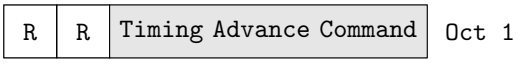
2.6.2 Timing Advance Command

[20] defines that Timing Advance is expressed as $N_{TA} \times T_S$, where $T_S = 1/30720ms$. N_{TA} is value signalled by eNodeB. Initial 11-bit unsigned value T_A is transmitted as a part of Random Access Response. $N_{TA} = T_A \times 16$.

The following alterations T_A to the TA are sent as part of the MAC Control Element (MAC CE). Alterations are expressed as a relative change to the



(a) Timing Advance Command in MAC RAR.



(b) Timing Advance Command in MAC CE.

Figure 2.7: Timing Advance command signalisation.

previous timing advance.

$$N_{TAnew} = N_{TAold} + (T_A - 31) \times 16$$

T_A is a positive 6-bit value. A constant value of 31 is subtracted to account for both increase and decrease in timing advance. Figure 2.7 shows two types of TA commands and their location in the downlink as specified in [13].

The granularity of the TA is $T_S \times 16 = 0.5208\mu s$. UE does not receive a more precise value for propagation delay. Given the propagation speed is the speed of light, UE can estimate its distance from eNodeB in a range of length $78.07m$.

Chapter 3

Attack

To show capabilities of the uplink and downlink sniffer we analyse two attacks which passive attacker can perform against LTE networks. In this chapter, we first outline the attacker and assumptions about the environment. We then introduce the two passive attacks: Timing Advance Attack and User Equipment Fingerprinting. These two attacks can be merged into a method for tracking people.

3.1 Attacker Model

A Passive attacker can receive RF samples on both uplink and downlink but cannot transmit anything. For our attacks, we use UL/DL Sniffer as the source of packets. The sniffer records all communication between UEs and eNodeBs. It does not break encryption. We detail UL/DL Sniffer in section 4.1. The attacker's goal is to find the location of a particular person in the real world. In this work, we show how our two attacks, Timing Advance Attack and User Equipment Fingerprinting, help her accomplish the goal.

Our assumption is that for each possible eNodeB where a victim might be, the attacker has at least one, preferably more, USRP devices which have a stable clock. The clock can be stabilized by synchronizing to the GPS as is the case for all eNodeBs' clocks. The attacker's radio must be synchronized with an error of at most $\sim 5\mu s$ from victim UEs otherwise the messages won't be aligned to the correct resource blocks. Length of cyclic prefix is $\sim 5\mu s$. Analysis of the delay and its impact on decoding of uplink messages can be found in the Appendix in Figure 1.

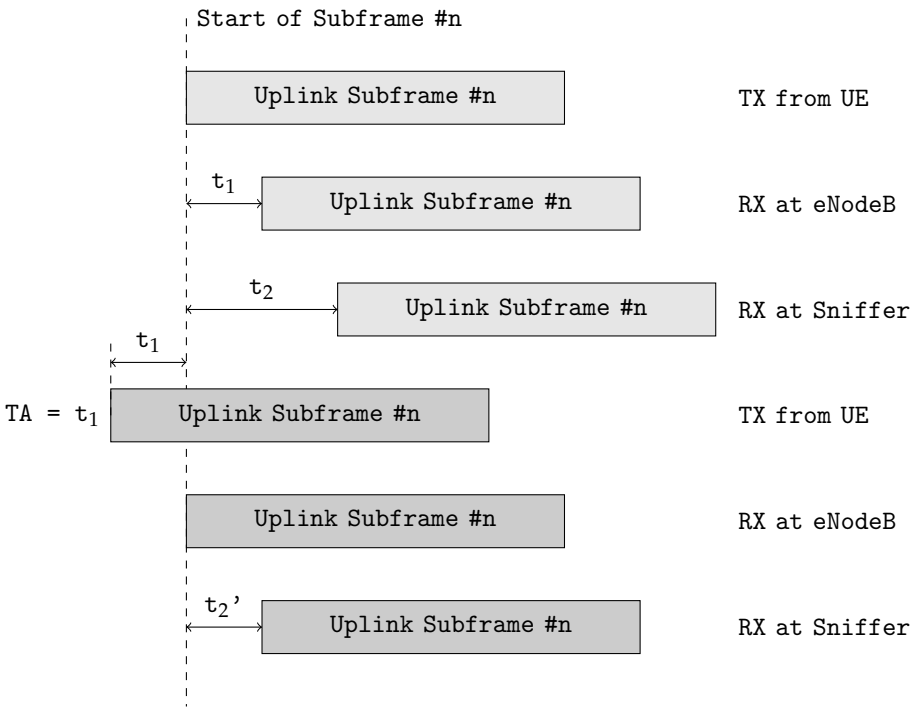


Figure 3.1: Delay of UL message at the sniffer with and without TA Command for UE.

3.2 Timing Advance Attack

The most basic localization attack works by sniffing TA commands. Since they are transmitted in clear-text on the MAC layer of LTE protocol, they are visible to our DL Sniffer. TA command localization constricts possible location to a ring around the DL sniffer with width of 78m. In Figure 3.1, TA command has a value t_1.

Adding a UL sniffer, allows us to run the same Timing Advance algorithm as eNodeB is running (subsection 2.6.1). Timing Advance measured on the sniffer is propagation delay between eNodeB and UE plus propagation delay between the UE and Sniffer. Measured TA gives us a more precise location of a victim. The location is now constricted to intersections of a wide circular ring and thinner ellipsoid ring as shown in Figure 3.2.

We can describe ellipse by its two focal points F_1 and F_2, and its semi-mayor axis a. For any point P on the ellipse, following holds:

$$2a = |F_1P| + |PF_2|$$

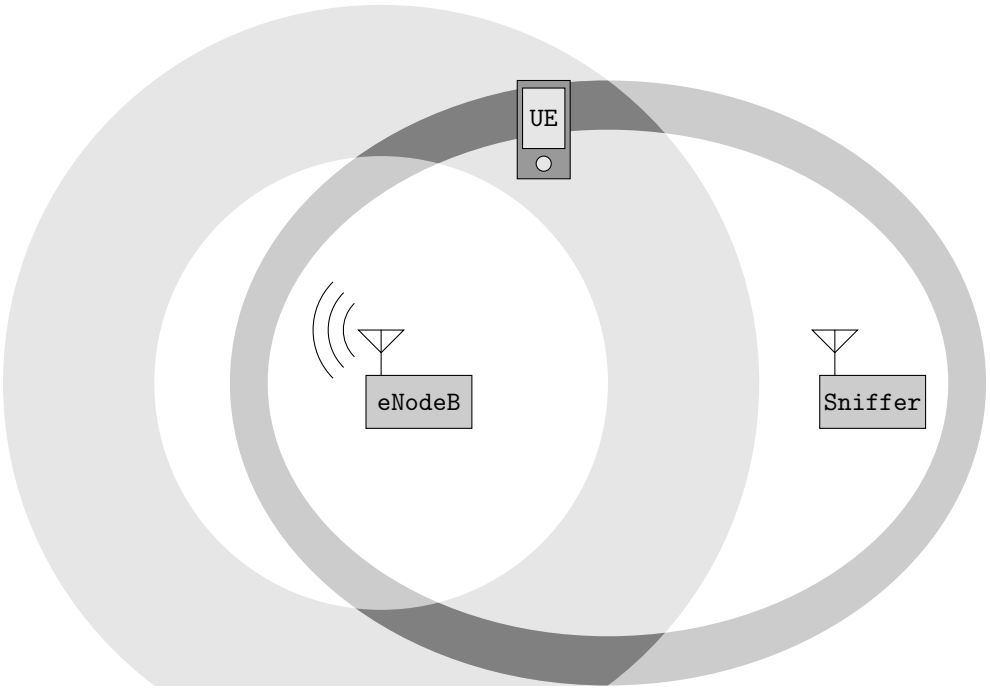


Figure 3.2: Location leak using eNodeB TA command and a sniffer.

In case of Figure 3.1, ellipse ring where UE is located has two focal points eNodeB and Sniffer. If no TA command was sent, $t_2 = 2a$. However, given TA command was set on UE, the new measured time is t_2'. Since TA command just delays transmission of UE messages it is clear that $t_2 = t_2' + t_1$.

As Sniffer gives more fine-grained location than eNodeB's TA command, we can employ multiple sniffers to increase the accuracy of the localization. Figure 3.3 shows the precise location of UE using two UL sniffers.

3.3 User Equipment Fingerprinting

The Attacker's goal is to identify a user from UL or DL messages. If the attacker learns one of the persistent identifiers (IMSI or IMEI) she can map them to the victim. In a correctly configured network, IMEI is never sent in the clear-text, and IMSI is only sent the first time UE connects to the network. In section 5.5 we show a real example of weakly configured mobile network which leaks IMEISV, therefore, the UE fingerprinting attack is not needed at all. Fortunately, only TMSI or GUTI is sent in clear-text for most of the networks.

In our fingerprinting attack, we consider two scenarios.

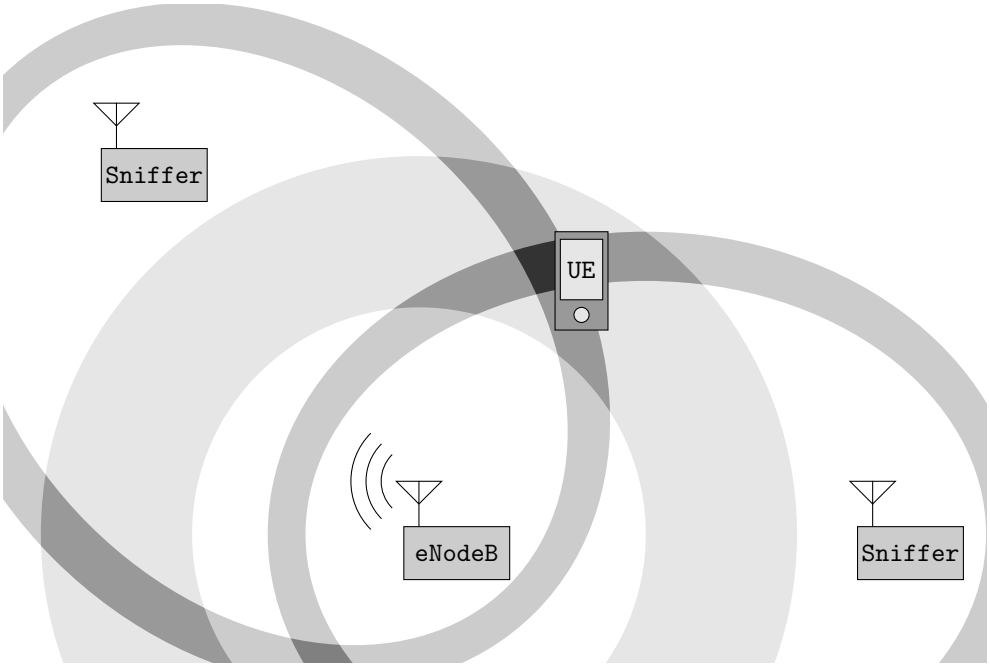


Figure 3.3: TA attack using multiple sniffers.

1. Passive attacker intercepted victim's communication with eNodeB (section 2.5) and saved the fingerprint. The attacker can now use the fingerprint of the communication to match the future unknown connections to the fingerprint.
2. The attacker learns the UE model (iPhone X, Samsung Galaxy s10, etc.) of the victim. She uses a machine learning model to predict the UE model from a new recorded connection and try to match it to the known model of the victim's UE.

Both of these attacks significantly narrow down the search space of possible victims' UEs in the network.

As a feature vector used in fingerprinting, we use capabilities sent during the Attach Request and UE Capability Information. Each phone has different capabilities implemented therefore these messages differ significantly. Core capabilities are always sent in clear-text during attach procedure. UE capabilities (radio capabilities) are sent during the Registration Procedure either before or after the security context is created.

After identifying the victim, the attacker can use TMSI or GUTI, she recorded during the Registration Procedure, for further tracking. [21] shows that a lot of Mobile Networks do not update the temporary identifiers often enough.

Even though during the next RRC connection UE only sends the Service Request instead of the Attach Request, the attacker already knows the TMSI UE is going to use. The attacker loses track of TMSI when the victim goes to a different tracking area and new TMSI is assigned to UE.

3.4 Combined Attack

TA Attack tells us the location for an RNTI. The attacker does not know anything about the UE apart from the temporary identifier. Combining the attack with fingerprinting, the attacker learns both the location information with identity information. This combination allows her to precisely track people in the local area without the need of following them.

Moreover, as shown in subsection 5.2.3, we observed that a UE's LTE modem adds a constant time offset to the Timing Advance measurements. It is, probably, due to hardware and software design of the chips. LTE modems are black-box devices and we do not see their hardware or software. We observed that the constant offset depends on the chip model. We use UE fingerprinting to obtain the chip model and apply the constant time offset to the TA Attack measurement increasing its precision.

Chapter 4

Design and Implementation

In this chapter, we detail how individual parts are designed and implemented. First, we introduce the main and most complex part of the thesis: UL/DL Sniffer. The whole software is programmed in C/C++. Its code can be found in: https://gitlab.ethz.ch/komartin/lte-monitoring. Next, we introduce the design of our two attacks. Data collection is done by the UL/DL Sniffer. Postprocessing is implemented in Python. You can find the code and saved data in: https://gitlab.ethz.ch/komartin/lte_other.

4.1 UL/DL Sniffer

For the implementation we use srsLTE [6], an open-source library for LTE protocol. Three programs are included with the library: srsUE, srsENB, and srsEPC, corresponding to the three entities of LTE protocol shown in Figure 2.1.

4.1.1 DL Sniffer

DL Sniffer has similar functionality to UE. UE synchronizes to a base station, connects to the base station, and then listens for the messages addressed to it. DL sniffer also synchronizes to the base station. Instead of connecting to the base station, it needs to sniff RNTI of victim UEs. Finally, it receives messages intended for the victim UE.

In algorithm 1 we show how DL sniffer works during each subframe. The procedure takes as input:

- array `rntis` specifying global RNTIs (RA-RNTI, SI-RNTI) and victim UEs' RNTIs. In the next section, we show how victims' RNTIs are found.
- `timeSamples` received on the USRP.

Algorithm 1: DL sniffer

```
Procedure DL Sniffer (One subframe)
    Input: rntis, timeSamples /* Time samples for current Subframe
           */, ULDci, subframeIndex
1   ULDci[subframeIndex + 4] ⟵ [ ];
    resourceBlocks ⟵ OFDMAReceiver(timeSamples);
    allLocations ⟵ allPDCCHLocations();
    for location ∈ allLocations do
        for rnti ∈ rntis do
            /* Decode PDCCH. If CRC check fails, skip.          */
            dci ⟵ decodePDCCH(resourceBlocks, location, rnti);
            if dci.format = 1 or dci.format = 2 then
                data ⟵ decodePDSCH(resourceBlocks, dci);
                /* Do something with the PDSCH data here.       */
2           else if dci.format = 0 then
3               ULDci[subframeIndex + 4].append(dci);
            end
            if rnti = RA-RNTI then
                /* New UE connecting to eNodeB.                 */
                rntis.append(getRNTIFromRAR(data));
            end
        end
        existingRNTI, probability ⟵
          getExistingRNTI(resourceBlocks, location);
        if probability > 0.9 then
            rntis.append(existingRNTI);
        end
    end
```

- `subframeIndex`.
- `ULDci`, the data structure to save received DCI for UL transmission (format 0). `ULDci` array, and procedures on lines 1, 2, and 3 are the preparation for the UL sniffer.

First, DL Sniffer performs inverse OFDMA transformation to receive frequency samples. It performs other functions such as channel correction or frequency offset correction. It then goes over all possible locations of DCI and tries to decode them. If DCI is included DL Sniffer parses it. If DCI specifies downlink message, DL Sniffer uses DCI to decode the PDSCH message. If DCI specifies resource allocation for the future uplink message, DL Sniffer saves the DCI and later passes it to UL Sniffer.

Since a lot of the functionality is similar to UE, we base our implementation on srsUE code-base. Challenging part of DL sniffer is the complexity and time efficiency. Our implementation works with real-world base stations and performs higher layer decoding of data as well.

4.1.2 RNTI Interception

There are two ways how to get currently connected UEs' RNTIs.

Random Access Response is sent with RNTI derived from the time of TX of the PRACH preamble. Inside the RAR, eNodeB specifies new C-RNTI to use for the UE. RAR message is sent in cleartext and is visible to the DL sniffer. In algorithm 1 we call the function `getRNTIFromRAR`. This method can be used only for the new connections. For existing connections assigned C-RNTI is not exchanged in plain text.

CRC of the DCI is scrambled with C-RNTI of the UE and then transmitted with the DCI. UE checks if DCI is addressed to it by descrambling the received value and checking if the descrambled CRC is valid for the DCI. Scrambled CRC is computed as bit-wise XOR of C-RNTI with the original CRC. We can use the scrambled CRC to get C-RNTI of the user. We show the function in algorithm 2. This approach was first introduced in [22].

Algorithm 2: Get Existing RNTI from DCI CRC

```
Function getExistingRNTI(resourceBlocks, location):
    dci, scrambledCRC ⟵ getDCIWithCRC(resourceBlocks, location);
    CRC ⟵ computeCRC(dci);
    RNTICandidate ⟵ CRC ⊕ scrambledCRC;
    codedDci ⟵ encodeDci(dci);
    /* Compare two convolutional coded messages.  If the
       difference is too large, chances are the DCI was just a
       noise.                                                  */
    probability ⟵ compareCoded(codedDci, resourceBlocks, location);
    return RNTICandidate, probability;
```

4.1.3 USRP Synchronization

Timings of the subframes on both downlink and uplink are the same, as are the subframe numbers. UE gets the timings from the synchronization to the eNodeB. UE can then transmit at the right time at the beginning of UL subframe.

Algorithm 3: Synchronize two USRPs

```
Procedure Synchronize USRPs
    Input: syncWindowDL[windowSize ], syncWindowUL [windowSize ],
           lastDLsubframeIndex, lastULsubframeIndex /* Thread safe
           data structures                                          */
    Thread DLthread
        if not DLUSRP.isSynced() then
            lastDLsubframeIndex ⟵ synchronizeDLUSRPtoENodeB();
        end
        timeSamplesDL, subframeTimeDL ⟵ receiveDLSubframe();
        inc(lastDLsubframeIndex);
        syncWindowDL[lastDLsubframeIndex % windowSize] ⟵
         subframeTimeDL;
        /* Call DL Sniffer here.                                    */
    Thread ULThread
        timeSamplesUL, subframeTimeUL ⟵ receiveULSubframe();
        inc(lastULsubframeIndex);
        syncWindowUL[lastULsubframeIndex % windowSize] ⟵
         subframeTimeUL;
        lastCommonIndex ⟵
         min(lastDLsubframeIndex,lastULsubframeIndex);
        timeDifference ⟵ syncWindowUL[lastCommonIndex] −
         syncWindowDL[lastCommonIndex];
        if |timeDifference| > threshold then
            if timeDifference < 0 then
                receiveULSamples(timeDifference / samplingRate);
            else
                extraSamples ⟵
                 subframeLength − timeDifference / samplingRate;
                receiveULSamples(extraSamples);
                inc(lastULsubframeIndex);
            end
        end
        /* Call UL Sniffer here.                                    */
    ULThread.start();
    DLthread.start();
```

For our implementation, we use two USRPs. One for RX on the frequency of the downlink. Another one for RX on the frequency of the uplink. Downlink USRP is synchronized to the eNodeB, but the uplink USRP is not. They need to exchange their timings and subframe number for UL Sniffer to RX from the beginning of the subframe. The two USRPs need to have the same global clock to achieve this. This can be solved by either using GPSDO on both of the USRPs to have the same GPS clock or by using an Octoclock a device intended for this problem. In algorithm 3 we show how the synchronization is done for each subframe. In the final UL/DL Sniffer, synchronization of two USPRs would be called in an infinite loop. We specify in comments, where DL and UL Sniffers would be called. The procedure takes as input:

- array `syncWindowDL` where timestamps of last `windowSize` DL subframes are saved.
- array `syncWindowUL` where timestamps of last `windowSize` UL subframes are saved.
- `lastDLsubframeIndex` specifying last DL subframe index.
- `lastULsubframeIndex` specifying last UL subframe index. At the launch of the sniffer, `lastULsubframeIndex` is initialized to the value of `lastDLsubframeIndex`.

In essence, the algorithm 3 is simple. For each subframe, both UL and DL Sniffer record subframe index and the exact time they received it. If the timestamps for the same subframe index do not match, UL Sniffer has to adjust its RX time by receiving extra time samples.

4.1.4 UL Sniffer

Contrary to DL sniffer, UL sniffer is similar to eNodeB. It receives the samples and then decodes scheduled information. ENodeB controls the scheduling in the LTE protocol. In our case, we have to get the scheduling information from the downlink sniffer. This is where DL Sniffer comes into play. DCIs with format 0 carry the scheduling for uplink transmission. During the registration procedure, configuration for PUCCH is sent on NAS level which DL Sniffer obtains as well.

In algorithm 4 we show the procedure of UL Sniffer for each subrame. It takes the same inputs as DL Sniffer. It takes saved Format 0 DCIs and uses them to receive and decode correct resource blocks to get uplink messages. Similarly, from intercepted PUCCH configuration on downlink, UL Sniffer knows when to intercept Uplink Channel Information. To make the whole UL/DL Sniffer work we merge two large code-bases of srsUE and srsENB together.

Algorithm 4: UL sniffer

```
Procedure UL Sniffer (One subframe)
    Input: rntis, timeSamples /* Time samples for current Subframe
           */, ULDci, subframeIndex
    resourceBlocks ⟵ SC-FDMAReceiver(timeSamples);
    for dci ∈ ULDci[subframeIndex] do
        data ⟵ decodePUSCH(resourceBlocks, dci);
        /* Do something with the PUSCH data here.            */
    end
    for rnti ∈ rntis do
        uci ⟵ decodePUCCH(rnti, resourceBlocks, subframeIndex);
        /* Do something with the PUCCH control information
           here.                                             */
    end
```

4.2 Timing Advance Attack

The goal of the attack is to learn the location of the user in the area. For TA attack we employ UL/DL Sniffer from previous section 4.1. The attacker knows the distance of the sniffer from eNodeB and victim's RNTI.

For example, the attacker can send a message to the victim, which prompts the victim to re-connect to the network. The attacker learns the user's RNTI as it corresponds to the next UE connecting to the network after the paging message is sent. Therefore, this is a valid assumption.

Algorithm 5: TA Attack

```
Procedure TA Attack
    Input: rnti /* Victim's RNTI                              */
    currentTACommand ⟵ 0;
    while true do
        dataUL, dataDL ⟵ sniffer(rnti);
        if containsTACommand(dataDL) then
            currentTACommand ⟵
             currentTACommand + getTACommand(dataDL);
        end
        MeasuredTimingAdvance ⟵
         computeTAFromReferenceSignal(dataUL);
        timingAdvance ⟵ MeasuredTimingAdvance +
         getTimeFromTACommand(currentTACommand);
    end
```

The attack algorithm is shown in algorithm 5. Attacker measures Timing Advance with the same computation as eNodeB. It looks at the phase shift of the reference signal to get a precise Timing Advance. At the same time, the attacker sniffs TA commands to add the extra time to measured Timing Advance. Once Sniffer knows real timing advance to the UE, she can compute the distance constraint as:

$$\text{distance}_{\text{eNodeB}-\text{UE}} + \text{distance}_{\text{UE}-\text{Sniffer}} = c \times \text{timingAdvance}$$

where c is the speed of light in the air. If UL/DL Sniffer is at the same location as eNodeB:

$$\text{distance}_{\text{UE}-\text{Sniffer}} = c \times \text{timingAdvance}\ /2$$

Section 3.2 specifies how the ellipsoid rings are defined from the measured distance constraints and positions of eNodeB and UL/DL Sniffer.

4.2.1 Timing Errors

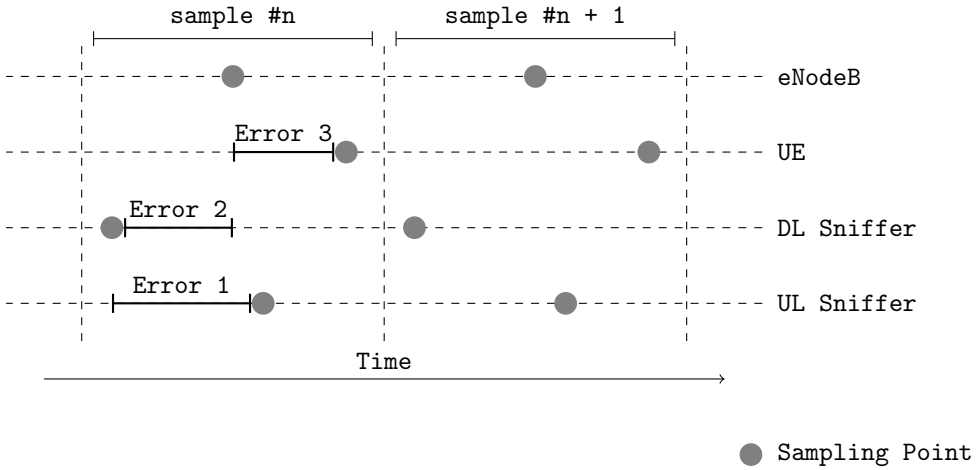


Figure 4.1: Visualisation of three timing errors in our system. ENodeB samples at the ideal time when there is no subcarrier interference.

The main challenge in this attack comes from having four different clocks, resulting in four different sampling points for each time sample. We identified three timing errors which occur in this system. The errors develop over time, however, their change is slow. Figure 4.1 visualizes the errors for one time sample. If we can correct the timing errors, we get a more precise time of flight measurement.

First radio is eNodeB. We assume the clock is precise by using a GPS to synchronize itself.

Two radios are used by the sniffer. Both of these also use precise clock, the time drift with respect to eNodeB's clock should be minimal. We get the most precise measurement of UE's Timing Advance, if the sampling point of UL sniffer is the same as eNode's sampling point. The offset between these two points cannot be measured directly. We can estimate Error 2 and Error 1 in Figure 4.1, and get the offset as their difference.

In algorithm 3 we show how the sniffer's two radios synchronize themselves to receive samples at the same time. However, the algorithm shows that the synchronization is only performed to the granularity of the length of one sample. The error can be a half a time sample in either direction. The difference in sampling time on the DL Sniffer and UL Sniffer is our first timing error. This error is at maximum the length of half of one time sample.

Second time error comes from how srsLTE performs synchronization to the eNodeB. If it registers drift from primary and secondary reference signal it corrects itself by receiving extra samples, similarly to the algorithm 3. Therefore, this synchronization is again with the granularity of length of one sample, half a time sample in both direction. We call the error synchronization error of DL Sniffer to the eNodeB. First two errors account for error of length of one time sample when measuring the time of flight.

The fourth radio is UE. The attacker does not know anything about its clock. Its clock might be less precise than the GPS clock. Moreover, the code for its synchronization procedure is unknown to the attacker. She cannot infer how precisely is the UE synchronized to the eNodeB. We identify this synchronization error as our third timing error.

We estimate the three time errors in the following way:

1. The low-level library for our USRP devices is called UHD. UHD library provides a timestamp for the reception of a sample. In our code, we receive timestamp both from DL Sniffer and UL Sniffer. The first time error is a time difference between the two timestamps.

2. Similarly to how Timing Advance is computed from phase shift introduced in subsection 2.6.1, we can compute the symbol time offset on the downlink reference signal. The measured offset is the synchronization error between DL sniffer and eNodeB.

3. Last time error cannot be measured by the attacker. The attacker can, however, look at the time of flight of UL messages continuously, and statistically infer the third time error. Different implementations of LTE protocol can have a different range for this error, some UEs might try to correct the synchronization error by themselves.

4.2.2 Improved Attack

The attacker can significantly improve the precision of TA Attack by employing multiple UL/DL Sniffers in different locations. Procedure for the improved TA Attack follows the original procedure in algorithm 5 running concurrently on multiple devices at the same time. The final UE location lays on the intersection of multiple ellipsoid rings as shown in Figure 3.3.

4.3 User Equipment Fingerprinting

To create a machine learning model we need to extract features from connections of UEs. We use our sniffer to sniff on real connections of UEs to the network. Afterwards, we save the uplink MAC layer packet capture and parse it in Wireshark [23] to get higher layer messages. Wireshark has built-in functionality to parse MAC layer to RRC or NAS layer. As explained in section 3.3 we are interested in core capabilities sent in Attach Procedure message on the NAS layer. We export the NAS message as JSON from the Wireshark and import it into a Python object.

We build a database of known UEs with their corresponding Attach Procedure messages.

4.3.1 Preprocessing

First, we need to extract the features. Core capability object is structured as a nested object with multiple optional information elements [24]. We filter information elements which carry temporary information, such as temporary identifiers or tracking area codes.

We use json-flatten [25] library for flattening of the object. It outputs a one level deep object with key-value pairs. We will refer to keys as capabilities even though some of them specify other information.

In order to run a machine learning model, each data-point needs to have the same features. Because of optional elements we define two types of features which are used for further feature extraction:

1. List of capabilities specified in all UEs
2. List of optional capabilities not specified in all UEs

For the first list, features are the values for all capabilities. For each capability in the second list, we have two features: boolean value if the capability is in the Attach Request for a given connection, and value of the capability. If the capability is not sent, the boolean value is false and we consider its value to be 0. Otherwise, the boolean value is True, and value is what the sniffer received. We put all the features explained in this paragraph into the first set of features.

Given we have a database of known UEs of size k, other features are differences of core capabilities to this known set. For each UE in the database, we have two features. For the first feature, we compare values of capabilities occurring in the new Attach Request and the saved Attach Request for the UE in database. The feature value is the number of capabilities with different values between the two Attach Requests. For the second feature, we look at capabilities occurring only in one of the Attach Requests. The second feature value is the number of such capabilities. We will call features comparing the Attach Requests to the database the second set of features.

Both sets of features create a fingerprint of a connection. If the attacker already knows the fingerprint of the victim's UE, she just needs to find the exact match to it from the recorded connections.

If the target fingerprint is unknown, the attacker needs to find more information about the UE model using Machine Learning approaches. She can use the following approaches.

4.3.2 Unsupervised Learning

Using unsupervised learning we want to find the inherent structure of the data and visualize the data-points. Given new UE, the attacker can find the most similar data-points in the dataset to infer more information about the device. To find similar phones we use Gaussian mixture model, a clustering algorithm. Using unsupervised learning, we do not add our biases to the training and clustering finds the most distinct groups of UEs. For example, the clustering model might cluster phones by year of production. New datapoint would end up in a cluster with phones from the same year.

We use principal component analysis to lower the number of features and plot UEs onto a 2D plane. PCA transforms data into lower dimensions by creating new features corresponding to as much variance in the dataset as possible. We plot four components into two scatter plots: the first plot with the first two most significant components, and the second plot with the second most significant components. We train the clustering algorithm on computed PCA components. Using PCA components makes the clustering algorithm more robust, as we de-noise the features and work with less dimensions.

4.3.3 Supervised Learning

We can label the known database of UEs according to a rule. For example, by manufacturer or year of the production. We use the database as a training set to the classifier. Afterwards, we can classify new UE fingerprints with it.

We use decision tree classifier since these are easy to interpret and they will help us find features which contribute the most to classification. Given the

attacker chooses sensible classes, they will have common features in core capabilities object. The decision tree model is fast and robust. We only use the first set of features, as they will directly point to differences in Attach Requests.

We decided to have only small trees with limited depth. Multiple different decision trees might create a good classifier. We build multiple possible trees to find all possible features differentiating the classes. We do this by running decision tree classifier. If the algorithm finds good enough tree, we save the decision making features chosen by the tree. We repeat the process without the saved features. We end up with a list of features defining the class chosen at the beginning.

Since we build multiple decision trees, we can use all the trees for classification as an ensemble. For the new data-point we predict the class independently by all the trees. The final class prediction is the most occurring class from the trees.

Chapter 5

Results

In this chapter, we evaluate our two attacks. First, we introduce hardware used in the experiments. Following, we show the design of the experiments and their results.

5.1 Setup

For the experiments in this chapter we used setup pictured in Figure 5.1. It consists of:

eNodeB running on software defined radio USRP N310, highlighted with blue colour in Figure 5.1.

UL/DL Sniffer running on two software defined radios USRP X310, highlighted with red colour in Figure 5.1. One X310 is used as a sniffer on DL frequency and the other on UL frequency. There is no antenna

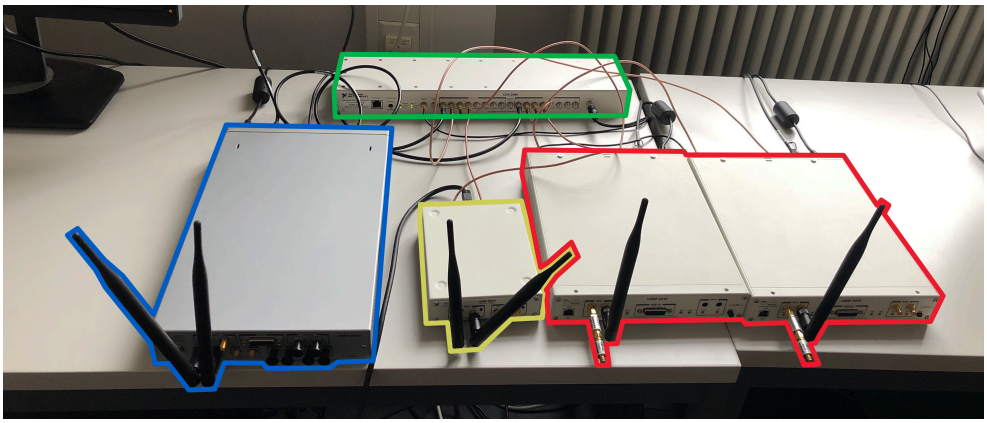


Figure 5.1: Our setup.

connected to the TX port of the radios confirming it is a passive device. Both devices are connected to the Octoclock to share the same clock.

UE running on software defined radio USRP B210, highlighted with yellow colour in Figure 5.1. In the experiment, we use multiple other mobile phones as UEs which are not pictured. The full list of UEs is in Table 1 in the Appendix.

Octoclock model CDA-2990, highlighted in green colour. The device is able to distribute the same clocking signal to all connected devices. It takes the GPS signal as the input. All connected devices have the same sense of time. All USRP devices are pictured connected to the Octoclock, however, in the experiments for Timing Advance this is not always the case and we always specify the details. The two sniffing USRPs are always connected to Octoclock.

Without using a precise clock on the eNodeB and the sniffer we saw a big noise in measured values. So in all our experiments, all three USRPs are connected to a precise clock. Either directly to GPS or to Octoclock.

5.2 Timing Advance Attack

5.2.1 Synchronization Error Estimation

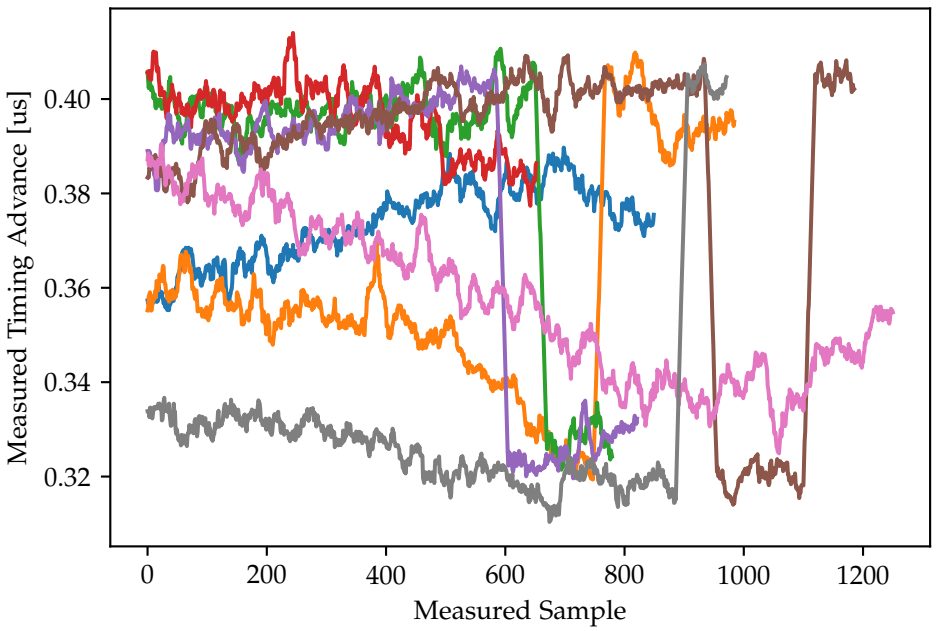


Figure 5.2: Timing Advance measured at the eNodeB.

As explained in subsection 4.2.1, we have three timing errors: the difference between RX time of UL/DL Sniffer's radios, synchronization error of DL Sniffer to the eNodeB, and synchronization error of a UE to the eNodeB. All three errors are visualized in Figure 4.1.

The first timing error is straightforwards to compute since it is just a difference of two timestamps. Since the two radios of the sniffer share the same clock input through Octoclock, this value is precise.

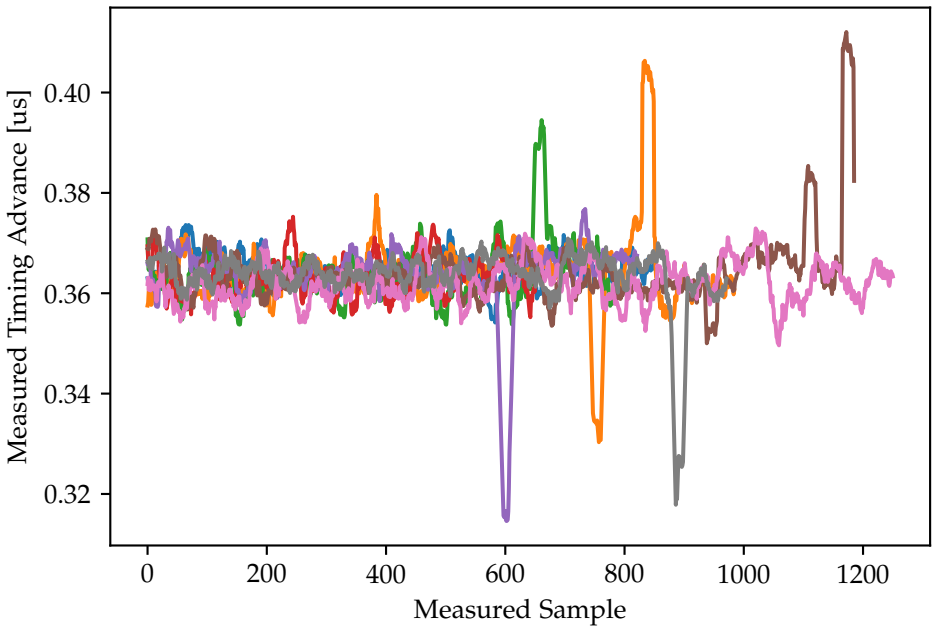


Figure 5.3: Timing Advance measured at the eNodeB with corrected synchronization error.

The main challenge is the synchronization error estimation. It is computed from phase shift as explained in [19]. To evaluate our estimation of the synchronization error, we ran a simple experiment. We had eNodeB running on USRP N310 and UE running on USRP X310, both using a clock synchronized with the GPS. We measured the Timing Advance of UE on the eNodeB. We measured the advance for each uplink message over multiple connections from the same distance in the same environment. We plot the measurements in Figure 5.2. Different lines correspond to different connections. Each point on the line is one measurement of TA from an PUSCH message. We can see that even though the clocks are synchronized with the GPS, they are not perfectly aligned. From multiple connections we can see a clock drifting slowly (i.e. the pink connection). The big jumps correspond

to the srsLTE synchronization algorithm aligning the RX time by receiving an extra sample. The median Timing Advance for a connection ranges from 0.33 to 0.4μs. This is an error of 70ns between the connections.

We at the same time estimated the synchronization error on the UE and corrected the error in raw Timing Advance in Figure 5.2. The corrected Timing Advances are in Figure 5.3. The median value for a connection now ranges from 0.3615 to 0.3655μs. We were able to improve the error between the connections to 4ns.

All of the following experiments in this section use a slightly idealized setup with both eNodeB and UL/DL Sniffer being connected to Octoclock. We showed in the previous experiment that we can fix the synchronization error even with the misaligned clocks. Introduced error from the misaligned clocks is negligible. For ease of use indoors, we opted for having both eNodeB and UL/DL Sniffer connected to Octoclock.

5.2.2 Timing Errors Estimation with UL/DL Sniffer

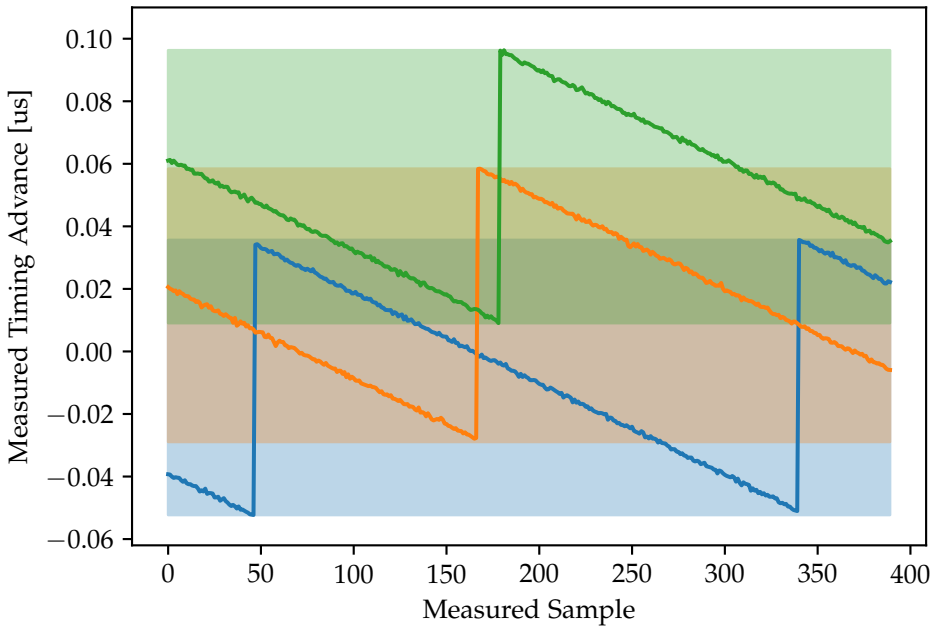


Figure 5.4: Raw estimation from the sniffer.

In Figure 5.4 we can see the Timing Advance estimations by our UL/DL Sniffer. We see three connections (blue, orange, and green) from the same UE (B210) to eNodeB. Each point on the three lines corresponds to one measurement of Timing Advance from the PUSCH message. For this experi-

ment, eNodeB, the sniffer and UE were at the same location. The error of the median Timing Advance between the connections is 60ns

Firstly, observations show a constant drift which gets corrected once it reaches a threshold as you can see on the blue connection. The band it creates (light-coloured band defining max and minimum for each connection), are of the same height, around 90ns. This is also a length of one time sample. Jumps show the synchronization algorithm aligning RX time, similar to the previous experiment. The drift comes from the imprecision of the B210 clock. This is the third timing error we defined in subsection 4.2.1, the synchronization error between UE and eNodeB. We cannot measure this error directly since neither UE nor eNodeB is under the control of the attacker.

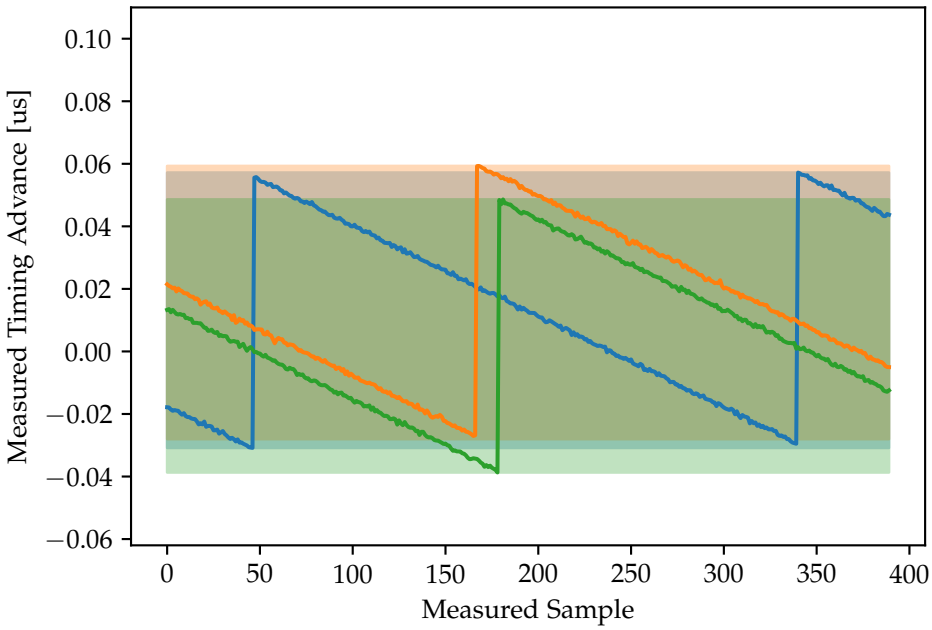


Figure 5.5: Fixed timing errors in the raw estimation from the sniffer.

Secondly, there is a constant shift between the three UE connections. As explained in subsection 4.2.1, we measure both the sampling time error between the two radios of the sniffer and the synchronization error between the DL Sniffer and eNodeB. We applied these measurements and got corrected results in Figure 5.5. Comparing the two figures, we can see that we were able to correct the constant offset, and the error between the connections decreased from 60ns to 10ns.

For the case in Figures 5.4 and 5.5, we can estimate the Timing Advance as the centre of the band of size 0.9μs bounding each connection. However, we

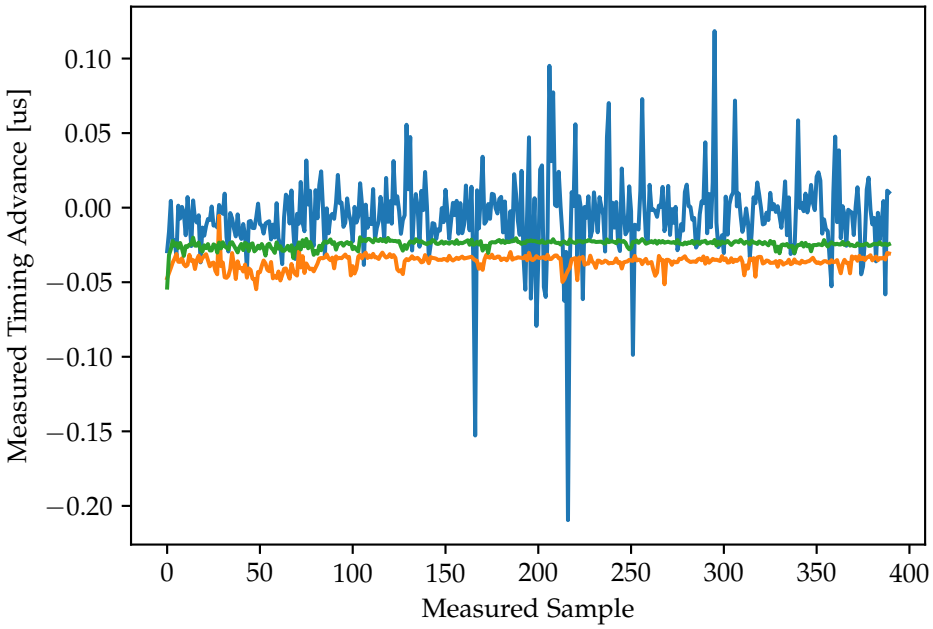


Figure 5.6: Estimation of Timing Advance for a mobile phone with unknown synchronization algorithm.

did not see this constant pattern of synchronization error for other UEs as we can see in Figure 5.6. Other UEs use different synchronization algorithms. We found the best choice, in general, is to use the median value from all the observations for one connection. The median value does not take into account outliers, being more robust than the mean value.

5.2.3 Distance Estimation

For the main experiment of our attack, we had an eNodeB and the sniffer at one location and we varied the distance of the UE. We ran the experiment with 5 different UEs: USRP B210, Huawei P20 Pro, Huawei P30, iPhone X, and iPhone 8. We positioned the UE at 6 different distances: $0m$, $7.5m$, $15m$, $30m$, $45m$, and $60m$. For each distance and UE, we reconnected multiple times to measure the Timing Advance over multiple connections. On top of that, for each measurement of distance and UE, we restarted the sniffer at least once, to reset the timing errors. We performed the experiment in a long indoor hall. Since eNodeB and UL/DL Sniffer are at the same location, the distance of the UE from the sniffer is:

$$\text{distance}_{\text{UE}-\text{Sniffer}} = c \times \text{timingAdvance} / 2$$

UE model	Correlation Value	p-value	Constant Offset [m]	90th Percentile Error [m]
USRP B210	0.980	1.2e-24	99.536	10.474
Huawei P20 Pro	0.988	4.1e-31	-8.526	5.659
Huawei P30	0.993	1.6e-33	-23.191	5.214
iPhone X	0.985	1.3e-27	-22.732	7.238
iPhone 8	0.995	3.9e-37	-20.930	4.672

Table 5.1: Summary statistics for distance estimation.

One data point corresponds to the median distance measurements during one connection. We do not consider connections for which we have less than 10 measurements. For each UE there is a constant time offset which comes from properties of UE chip, UL/DL Sniffer radios, and eNodeB radio. We estimate constant offset as a mean difference of estimated distances and real distances. Table 5.1 shows constant offset for each UE. If the attacker changes the radio of her sniffer, she would have to update the constant offsets for individual UEs.

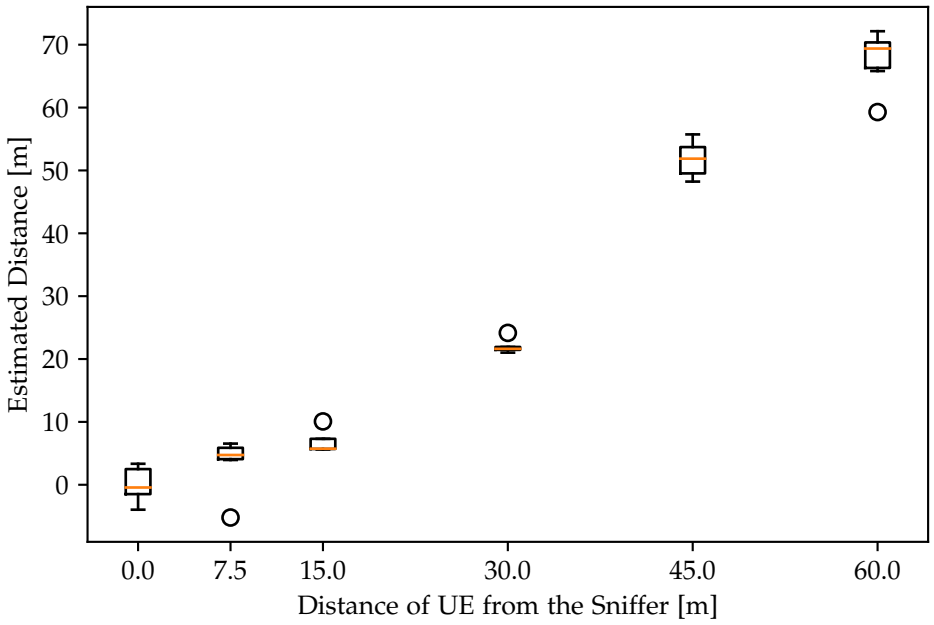
In appendix in Table 1, we quantify the constant offset for other phones as well. We only had a measurement at $0m$ distance which we use to estimate the offset. It is, therefore, less precise than in Table 5.1. We can identify that the constant offset is the same for all UEs with the same LTE modem.

We observed a large distance estimation error rising from UE not receiving TA command. If UE does not receive TA command, eNodeB resends the command. However, the sniffer receives it twice and applies the command again resulting in a mismatch. Since the N310 is not professionally graded eNodeB device, its TX power is lower. We can expect better performance in the real world. Possible fix in the future would be to monitor ACKs transmitted by the UE. UL/DL Sniffer would only apply TA commands which it received ACK for. Before further analysis, we removed connection outliers which were more than 10 times the interquartile range away from the median point. Out of 186 connections, we removed 4 data points.

We visualize data points with boxplots for each UE in Figure 5.7. Before plotting, we remove computed constant offset from these distance estimations.

For each UE we ran a separate Pearson correlation test. It tests whether there is a linear correlation between the estimated distances and the real distances of the UE from the sniffer. The null hypothesis is that there is no linear relationship between the two variables. We reject the null hypothesis

(a) USRP B210.



(b) Huawei P20.

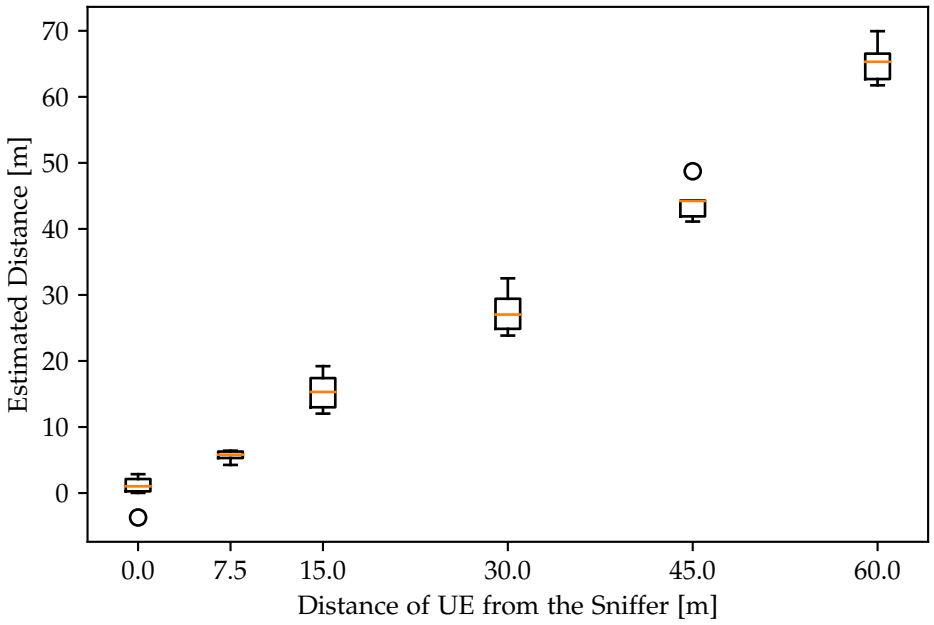
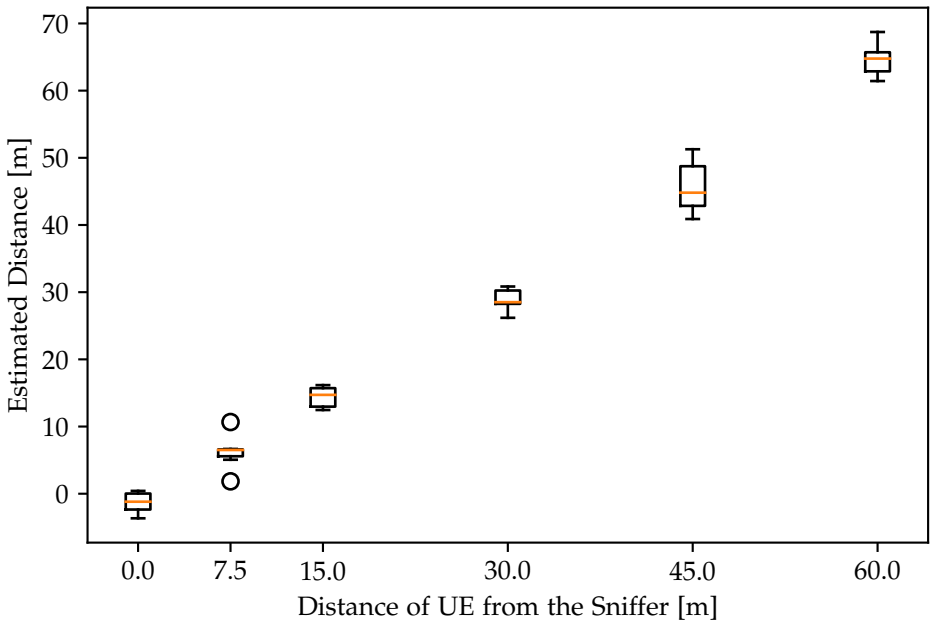


Figure 5.7: Distance measurements.

(c) Huawei P30.



(d) iPhone X.

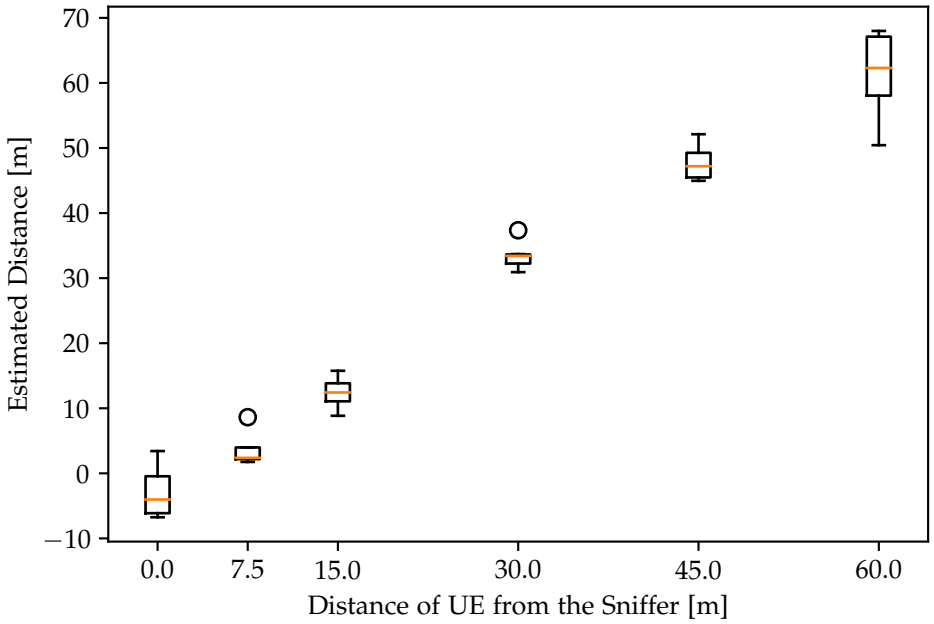


Figure 5.7: Distance measurements.

(e) iPhone 8.

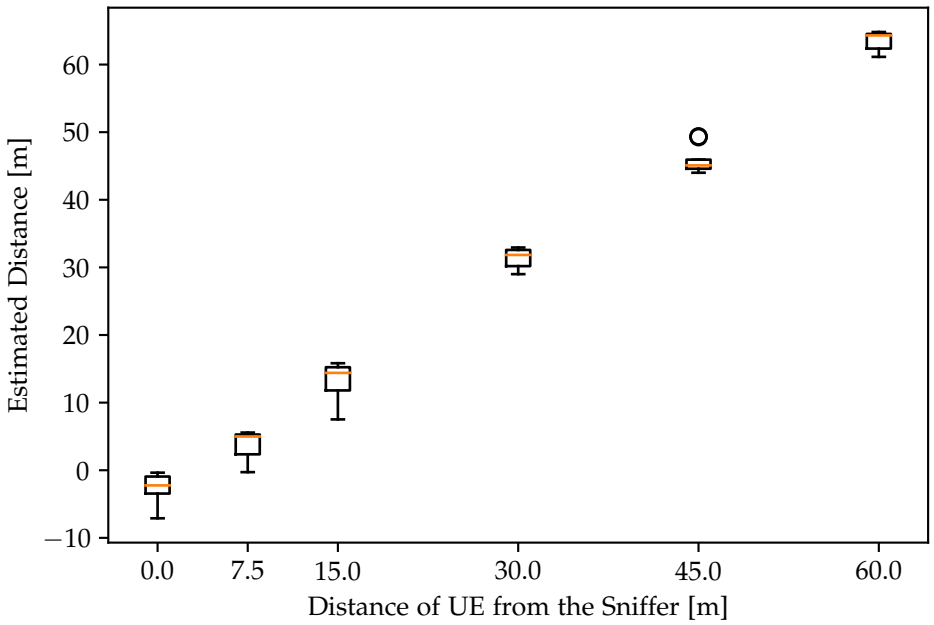


Figure 5.7: Distance measurements.

when p-value < 0.05. If we reject the null-hypothesis we conclude that there is sufficient evidence that there is a linear correlation between the two variables. Therefore, we can estimate the distance using UL/DL Sniffer. From Table 5.1, we see that for all UEs we reject the null hypothesis with high significance.

Finally, to quantify distance estimation error, we compute errors between estimated variables (with corrected constant offset) and real distances. We are interested in a threshold where 90% of all errors fall under. We observe that for all mobile phones the 90th percentile error is $\sim 6m$. For USRP B210 it is $\sim 10m$. Obviously, for lower percentile, the values get significantly better. Median error is $\sim 2m$ for phones and $\sim 7m$ for B210.

5.3 User Equipment Fingerprinting

As explained in the subsection 4.3.1, we use two sets of features for fingerprinting. Capabilities and their occurrences in Attach Request message as the first set of features. This set is very large with a lot of binary data (i.e. support for a capability is either true or false). And relative change to the

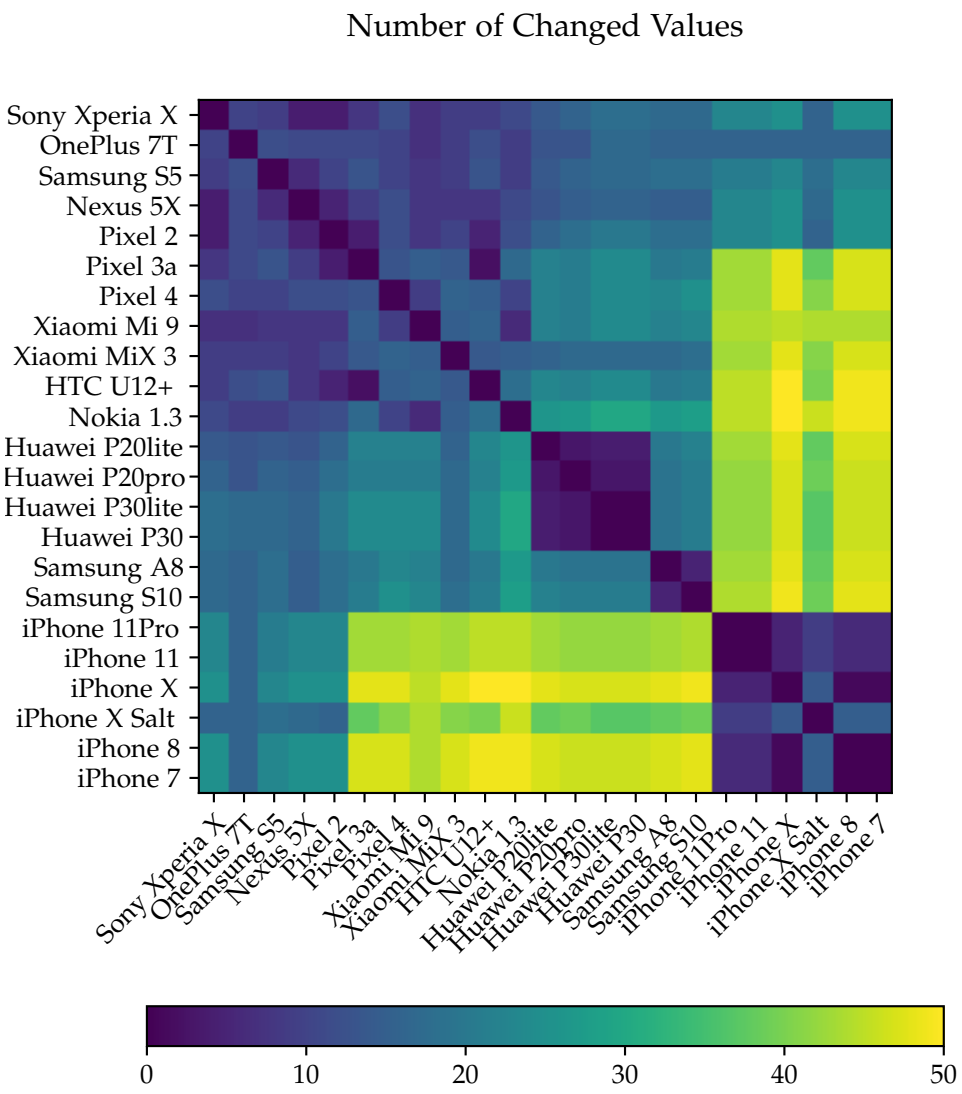


Figure 5.8: Heatmap of number of changed values between each two UEs in the database.

Attach Requests of UEs in the database as the second set of features. This second set of features is visualized in Figure 5.8 and Figure 5.9. From Figure 5.8 and Figure 5.9 we can directly see there is a relationship between the phones with the same wireless modem manufacturer.

In the Appendix in Table 1, you can see different modem manufacturers. In

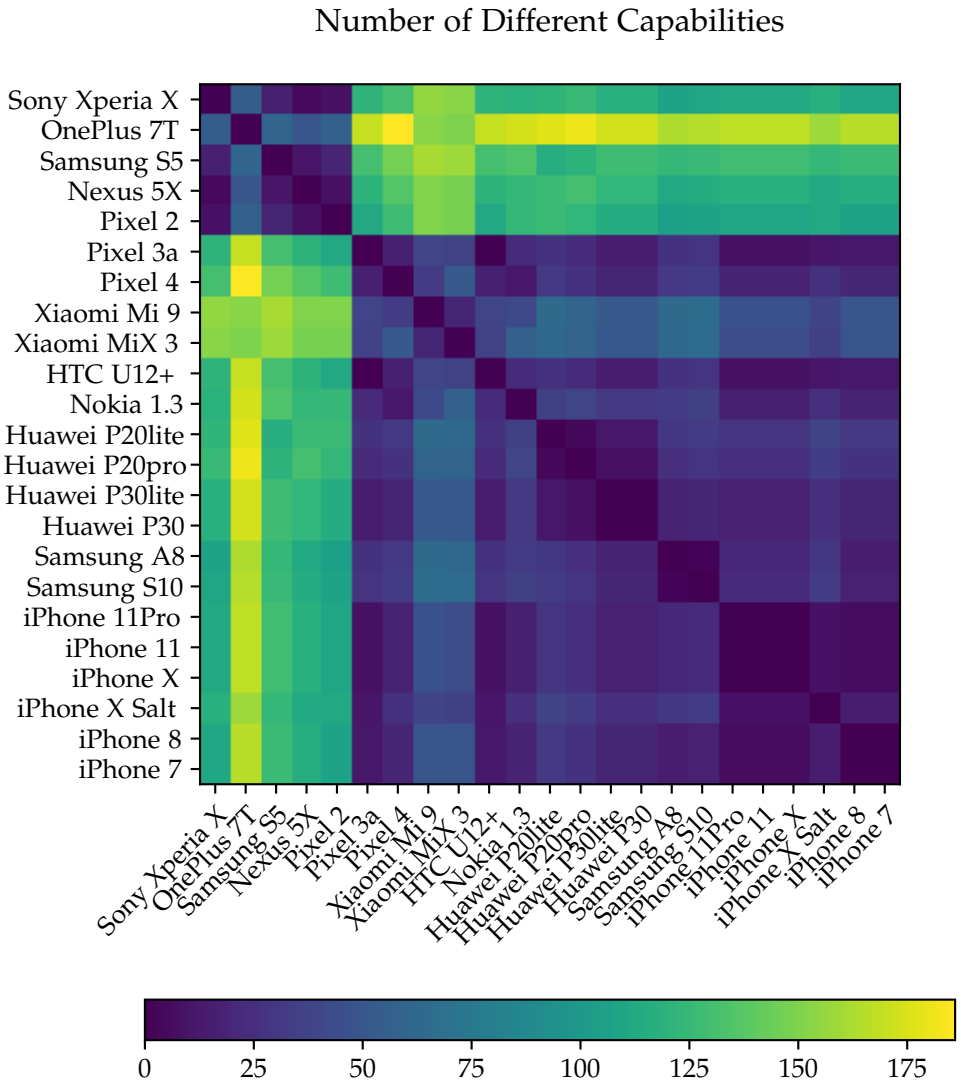


Figure 5.9: Heatmap of number of different capabilities between each two UEs in the database.

Figure 5.8, we can distinguish Qualcomm phones (first 11 phones), Huawei (next 4 phones) and Samsung phones (next 2 phones), and finally Intel chip manufacturer (last 6 iPhones). You could see similar things in Figure 5.9. Some of the Qualcomm phones do not send optional Mobile Station Classmark 3 information element. That is why the first five data points in Figure 5.8 are vastly different from other phones. Apart from OnePlus 7T, these

are very old phones, not being used frequently anymore. We can see in Figure 5.8 that Xiaomi phones are similar, as well as Huawei phones, Samsungs phones and iPhones. Xiaomi phones use Qualcomm modems but it is apparent that a manufacturer of the phone specifies some of the capabilities as well.

We observed that some of the capabilities also depend on the SIM card which you can also see in Figure 5.8 and Figure 5.9 between the iPhone X and the iPhone X Salt. Former contains programmable SIM, whereas the latter contains SIM from mobile network provider Salt. We do not research this phenomenon, as our main use case is a mobile network where most of the phones have a SIM from the same operator.

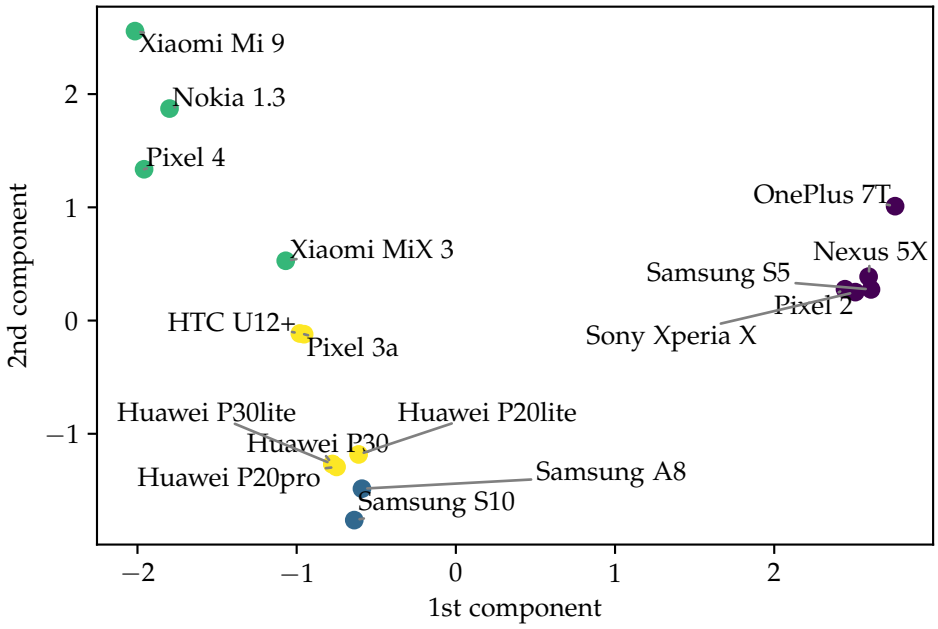
The main advantage of the second set of features over the first set is the number of the features. On the other hand, the first set of features give more interpretable features. We can directly see which capability impact the fingerprinting.

5.3.1 Unsupervised Learning

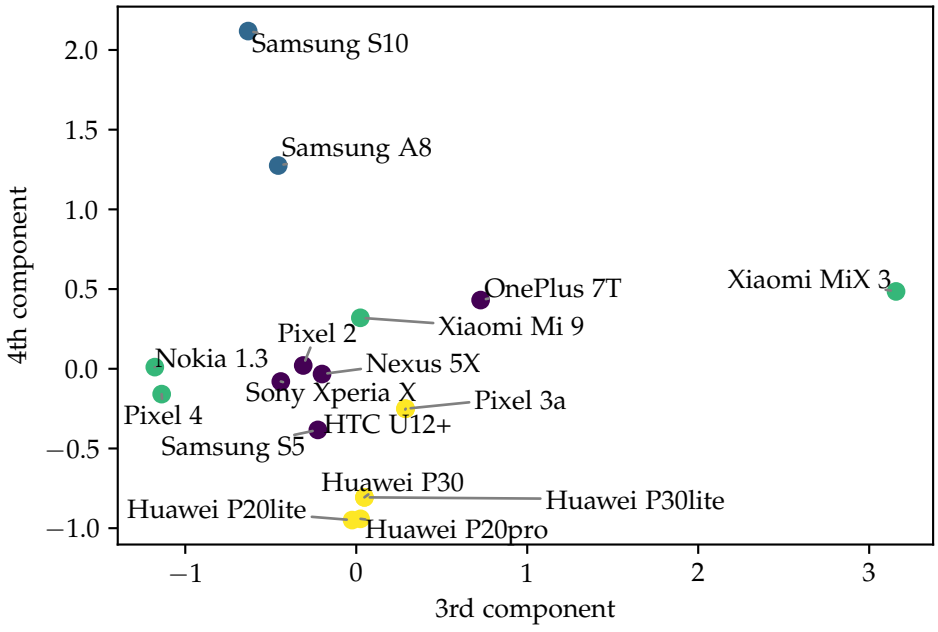
The first type of UE fingerprinting can be performed just by looking at the projection of the features onto a 2D plane. If these projections put similar types of phones close to each other, then the attacker can learn information about the UE from neighbouring phones in the projection. We use principal component analysis to extract the first four components from the first set of features and plot them as scatter plots in Figure 5.10. PCA is sensitive to outliers. Moreover, PCA gives a different result if the real and observed distributions of data points do not match. Therefore, we decided to not include iPhones in this visualisation. From heatmap in 5.8 we can see that they differ substantially from other phones and in the next section we show that it is easy to distinguish them from other UE models.

We ran a Gaussian mixture model with the top four components from PCA to confirm that similar phones are clustered together. Four different colours in Figure 5.10 distinguish the resulting clusters. We can see that two clusters, purple and green, specify phones with Qualcomm modem. Purple are newer models and green cluster is for older phones (apart from OnePlus 7T). Huawei phones are in yellow cluster. Finally, blue cluster is for Samsung modems. We see that HTC U12+ and Pixel3a ended up in the wrong cluster. From the first two components in Figure 5.10 we can see that these two phones are distant from other Huawei phones. We believe that with a larger data set, they would end up clustered with other Qualcomm phones instead.

Similarly to the first set of features, we can also visualize and cluster the second set of features describing relative differences between the models.



(a) First two components for the first set of features.



(b) Second two components for the first set of features.

Figure 5.10: PCA decomposition of the first set of features into four components.

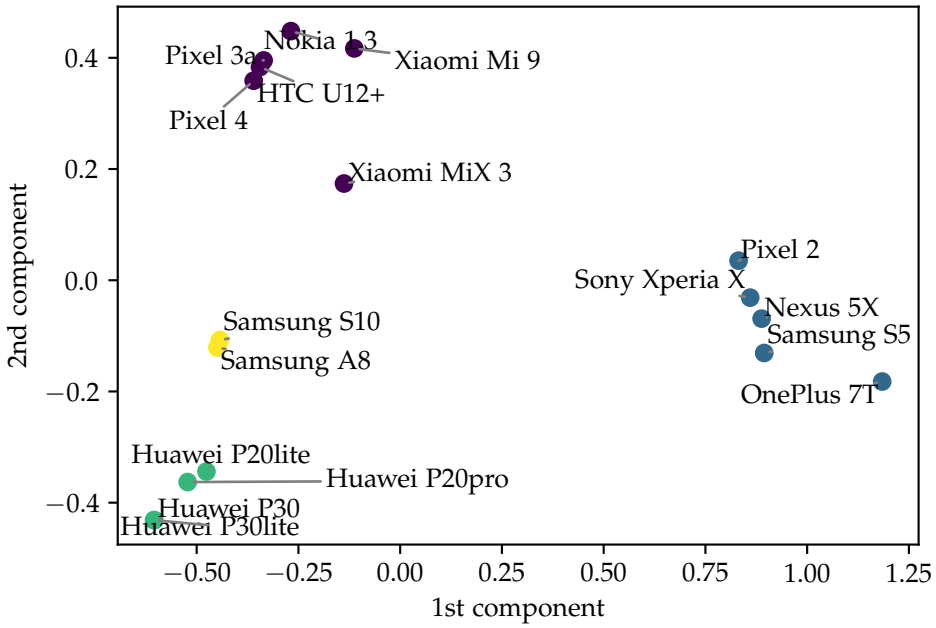


Figure 5.11: PCA decomposition of the second set of features into two components.

We can see the visualization and clustering in Figure 5.11. With just two principal components, Gaussian Mixture models perfectly differentiate the four groups of phones we saw previously. Blue cluster is for old Qualcomm phones, green is for the new Qualcomm phones, yellow is for the Samsung phones, and purple is for the Huawei phones. The only phone which is in the wrong cluster is OnePlus 7T, which weirdly does not send Mobile Station Classmark 3 information element. We conclude that second set of features performs better for the unsupervised learning.

Looking at the performance of the clustering method, we conclude that any machine learning model would perform well to distinguish the LTE modem manufacturer using PCA components.

5.3.2 Supervised Learning

With supervised learning, the goal is to identify features which separate classes of UEs. For this method, we use decision trees and only work with the first set of features, since the second set of features does not have as interpretable features. The ideal case is if the depth of the decision tree is one. That means a capability perfectly separates the two classes. One of the limitations of our work is a small data-set. We, therefore, try to find trees which separate the classes perfectly with as little depth as possible. If

we can find such trees we can be reasonably confident, we can build robust classifiers. Following are some categories we were able to differentiate with just one capability. This gives us the highest confidence it is the same for other unseen phones.

Intel Modems

Using the decision tree, we were able to identify features which perfectly differentiate Intel modems from other modems. In our case, it corresponds to differentiating iOS from Androids. Following are the differences for Intel modems:

- PS inter-RAT HO to E-UTRAN S1 mode supported in MS Network Capability information element.
- Support additional positioning capabilities in Mobile Station classmark 3 information element.
- SPLIT PG CYCLE CODE is equal to 8 in DRX Parameter Information Parameter.
- In Supported Codec List information element, codecs are sent first for GSM and then UMTS. For other manufacturers, it is the other way around.

Qualcomm Modems

Qualcomm modems are differentiated by two capabilities perfectly:

- Null integrity algorithm is not supported.
- Support location service value added location request notification capability in Mobile station classmark 2 information element.

Samsung Modems

Samsung modems are perfectly differentiated from other manufacturers by:

- High Multislot Capability value is present in Mobile station classmark 3 information element.
- ESM message container information element contains Device properties information element.
- Modems use maximum 8 seconds non-DRX mode after transfer state.

Huawei Modems

Finally, Huawei modems are perfectly differentiated from other modems by a single value:

- SPLIT PG CYCLE CODE is equal to 32 in DRX Parameter Information Parameter.

5.4 Combined Attack Use Case

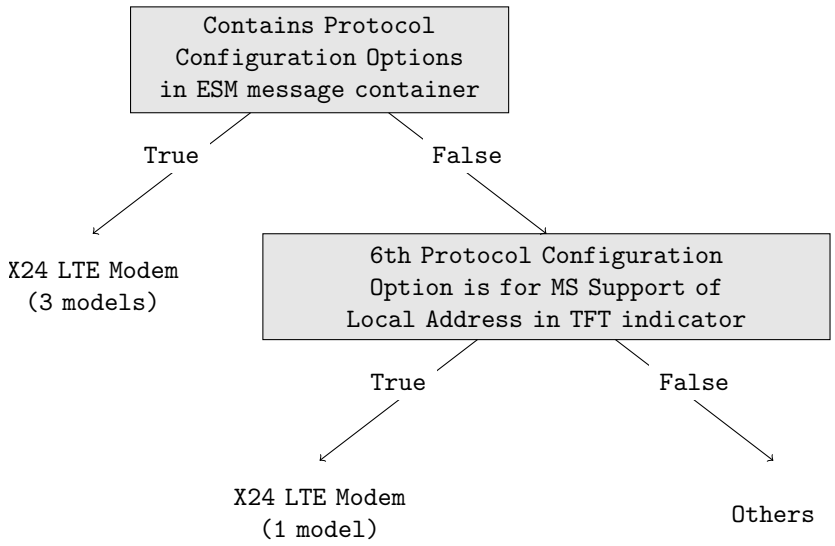


Figure 5.12: Decision Tree for Qualcomm X24 LTE Modem.

We create a fictional use case, how the attacker can use both of the attacks to find victims in the real world. Suppose, the attacker recorded victim's Attach Request but does not know what UE model the victim is using. The actual victim's UE model is LG V50 ThinQ with X24 LTE modem. The attacker wants to localize the victim when UE attaches to the network again.

The attacker has the fingerprint of the UE which she can use for the identification of the victim. However, for the localization, she is missing the constant offset to increase the precision.

The attacker computes a PCA of the recorded fingerprint and uses the pretrained unsupervised model to cluster the new data-point in the existing clusters. The attacker learns that it clustered the phone with recent Qualcomm phones.

From Table 1 in the Appendix, we see that LTE modems in that cluster add different constant offsets. We can now build a classifier for each LTE modem. Figure 5.12 shows a classifier for X24 Qualcomm modem in the cluster. The

attacker uses the decision trees and classifies the UE as a phone with X24 LTE modem. She now knows, that she should apply $\sim 12m$ constant offset when measuring the distance.

Without the fingerprinting attack, the attacker would not know the constant error and its distance estimation would be inaccurate.

5.5 Weak Network Configuration

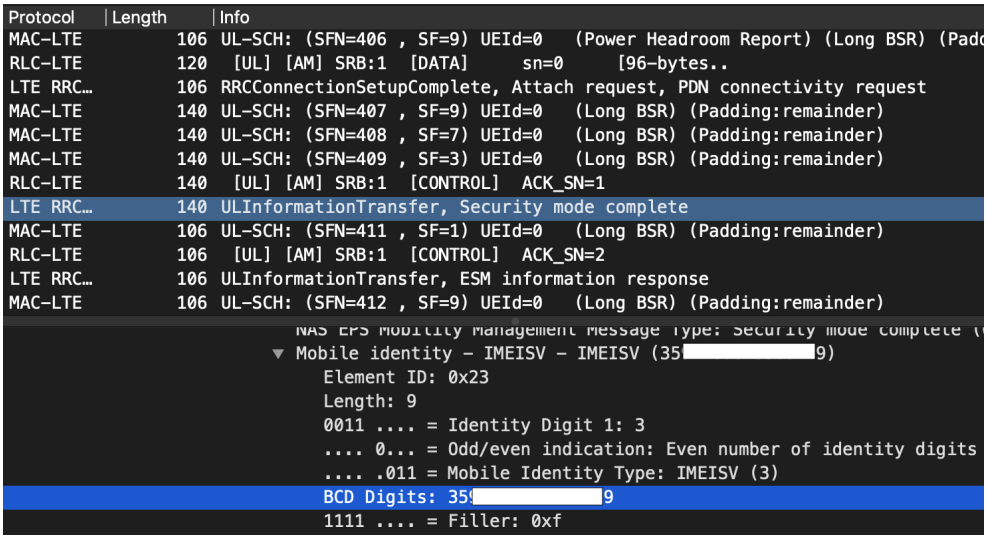


Figure 5.13: IMEISV number visible in clear-text for real network connection.

During the testing of our UL/DL Sniffer, we detected a weak configuration of an actual real-world network. This configuration allows user to learn IMEISV number, and track the user. The IMEISV number is sent in clear-text during each RRC connection, therefore we do not need the UE Fingerprinting to track the user. As mentioned before, first 8 digits of IMEI number identify a device model. We can use the weak configuration to get the constant offset of UE as well.

As seen in Figure 2.5, two security contexts are created. First one securing communication between UE and EPC, and second one securing communication between UE and eNodeB. In the first Security Mode Command, EPC requests an IMEISV. This message is integrity protected by the new context, but it isn't encrypted. Security Mode Complete is then both integrity protected and ciphered. It contains IMEISV as requested.

This network is configured such that ciphering is only applied for security context between UE and eNodeB. EPC security context performs only integrity protections which is a requirement by LTE protocol. The weak

configuration allows the sniffer to see IMEISV and following messages in clear-text before the second security context is created. Figure 5.13 shows a packet capture file in Wireshark. We can see the redacted IMEISV number. It matches with the phone's IMEI number.

Chapter 6

Related Work

In this chapter, we review literature related to three main contributions of our work: UL/DL Sniffer, TA Attack and UE Fingerprinting.

6.1 UL/DL Sniffer

Downlink sniffers for LTE are shown to work both in papers and commercial products. The first paper to implement downlink PDCCH sniffer was [22]. We thought we can use implementation of [26] since they also based it on srsLTE [6]. However, [26] based their work on one of the example projects of srsLTE with very limited functionality. We found that its performance in the real world was very limited. Compared to these two papers our implementation is robust and does not only PDCCH decoding but both data and control channel decoding up to higher layers. Airscope [27] is a commercial product by the company behind srsLTE. They also implement downlink sniffer. Due to its high price we were not able to compare our downlink sniffer with Airscope.

For uplink sniffer, we are the first paper to implement this functionality. Wavejudge [28] and thinkRF [29] provide uplink sniffer functionality for a very high price. We were not able to compare our programs to these commercial versions. We are the first implementation of both uplink and downlink sniffer based on open-source library srsLTE. We plan to release the code for it, creating an open-source alternative to otherwise expensive products.

6.2 Localization Attacks

Shaik et al. [2] show how an adversary may sniff on paging messages at different eNodeBs. An operator will first broadcast paging message for a

particular user from the last used eNodeB. Sniffer than learns the approximate location of the UE around eNodeB.

Moreover, they explain more advanced attack using a rogue base station to trigger Measurement Report. The report contains power measurement to neighbour cells. Using triangulation attacker can learn the precise location of the user.

Multiple papers (i.e. [30, 31, 32]) show how channel state information (CSI) can help user fingerprint her location in the environment. Similar way, an attacker can use CSI from uplink to fingerprint the location of users in the pre-mapped environment.

In LTEye [22], authors extend a synthetic aperture radar to capture the shortest and the most direct path of the radio signal going from User Equipment. Using multiple radars, the location of the users is at the intersection of the direct paths measured by the devices.

We believe our method for localization is more straightforward and gives better results than previously mentioned papers.

We got inspired for our TA Attack from theoretical work in [33]. Similar to us, they use both TA command from eNodeB and Timing Advance from the attacker to approximate geolocation of UE. We have to highlight, however, the differences between our and their work. First and most prominent contribution of our work is a working TA Attacker tested in real scenarios. In [33], they only do numerical analysis with simulated data. They had to make assumptions which might not be true in the real world. We miss in their work crucial details, such as an algorithm for measuring Timing Advance from uplink messages, mention constant offsets of the phones, or real distribution for TA command. They highlight how they work is successful in a setting with multiple eNodeBs. In our case, one eNodeB is sufficient, and instead we concentrate on a multi-attacker scenario. We further explore a setting with multiple attackers which they do not. Finally, we have to highlight the precision of our attack. In [33], they opted for approximating transmission time of UE from TA Command which introduces a large error. In our work, we estimate an ellipse with two focal points without an inherent error from TA Command.

[34] shows real-world TA attack against WiMax networks, which have different physical layer from LTE networks. They used a commercial device WaveJudge 4900A [28] to perform the attack. We, on the other hand, implemented both the downlink and the uplink sniffer to measure the distance and show the attack in the real-world scenario. Our distance estimation is vastly better than theirs. In [34], they miss the theoretical implication of attacker specifying ellipse of potential locations of the user.

6.3 User Equipment Fingerprinting

[1] shows how to use capabilities to fingerprint UE models. Their paper is base for our UE Fingerprinting attack. The main difference between the two approaches is that they use active attacker to receive the capabilities which in the real world is illegal and easy to detect. Moreover, they do not explain how their model works to identify the phone manufacturers or how they identified capabilities which distinguish them. We on the other hand gave a clear view of this information, and how an attacker can use machine learning to enhance future attacks. Finally, in our work we extend the list of capabilities distinguishing different modem manufacturers.

Chapter 7

Discussion

In this chapter, we first introduce mitigation techniques to decrease the risk of proposed attacks. Then we talk about the impact of our work in 5G networks. Finally, we highlight possible future projects related to our work.

7.1 Mitigation Technique

7.1.1 Timing Advance Attack

Reference signals are needed to estimate the channel and propagation delay. They cannot be omitted from uplink messages. There is, however, a fix to the current protocol which would solve most of the issues. If an attacker does not learn the initial TA Command in RAR message, he is not able to find the location, since all of the successive TA commands are relative to the current timing advancement. There are two ways how to achieve this:

1. Encrypt all MAC CE messages. All the MAC CE message would have to be sent after the security context. This is not ideal since the initial messages would not arrive at the correct time, resulting in degradation of performance. It would require a lot of changes to the current protocol as well.
2. Send the initial RA Preamble with a random offset. Since UE knows the offset, it can take received TA command in RAR, and modify it with the known random offset. Initial location of the user would be hidden to both eNodeB and the sniffer since only UE knows about its random offset. If necessary, it can disclose the random offset in an encrypted message to the eNodeB.

Both of these protection techniques only work against a small number of sniffers. If multiple sniffers are used, the attacker can infer the timing offset UE applies. The random offset is just another unknown which can be

estimated with more devices. Similar to random offset, if we use enough sniffers the attack does not have to record TA Commands, she can just compute TA from multiple observations.

7.1.2 User Equipment Fingerprinting

The only capabilities needed to create the security context are the ones describing crypto-suites implemented by UE. Only this information must be sent beforehand. We propose to send the rest of the capabilities after the security context is created. The same solution was already proposed by [1].

Response from 3GPP was that the radio capabilities are only sent after the security context is created. However, core capabilities are still planned to be sent in clear text even in 5G protocol.

7.2 Relevance in 5G

5G protocol is built on top of LTE, however, the PHY layer has some differences. Our UL/DL Sniffer would not work in 5G networks out of the box. We do not see a reason why it would not be possible to build such sniffer. Beamforming for downlink messages might decrease the signal strength if the sniffer is far away from other UEs.

There is no planned counter-measure for TA Attack in 5G specification. The reference signal is still used. 5G base stations cover smaller areas due to higher frequencies. Just by knowing to which base stations UE is connected to leak information about its position. If larger subcarrier spacing is used, TA command is more precise, increasing the strength of the attack.

UE core capabilities are still being sent in cleartext for 5G networks as well. Therefore, UE Fingerprinting is still possible.

Tracking of the user through temporary identifiers gets impractical in 5G. As explained in [35], a mobile network has to refresh a temporary identifier on each connection to the network. After each Service Request attacker loses the temporary identifier of the victim. She can only fingerprint user the first time it attaches to the network. The attacker needs to implement active attacks for the identification of the users.

7.3 Future Work

There are multiple possibilities for an extension of our work. We propose the following:

- High-level parsing of the uplink messages. We would be able to skip Wireshark as our parser. We could then customize our code such that

it can identify phones in the real-time and perform operations with them, such as filtering to receive uplink messages only for some models.

- Extension for our UE fingerprinting model, such that we can fingerprint models outside of Attack Procedure. Possible new features are the timings of the PHY messages, ordering, or flags in control messages.
- Use Signal Overshadowing [36] attacker with uplink Sniffer to create a more powerful active attacker. We would be able to build an IMSI catcher, which is harder to detect than conventional rogue base stations approach. This or other approaches for identification are needed to further increase the tracking threat from attackers.
- Various localization tools, which would give us locations of users in the building. For example, lights would be turned on in the rooms where the user is located. Or localization of cheaters during the exam in a big hall.

Chapter 8

Conclusion

In this work, we built a robust uplink and downlink sniffer using open-source LTE library srsLTE. Our software is both affordable and white-box and is a good competition against commercial products. We showed its capabilities in two new attacks: UE Fingerprinting and localization TA Attack. The first attack helps the attacker reveal mobile phone model, which helps in identifying the user. In the second attack, we showed that an attacker can estimate the distance of UE from her device with better than $6m$ accuracy 90% of the time. Finally, we discovered a weak configuration of a real mobile network in Zurich, leaking IMEISV number. All the attacks together show, how the attacker can track users in the world. UL/DL Sniffer is an important tool for security analysis of LTE networks.

Bibliography

[1] A. Shaik, R. Borgaonkar, S. Park, and J.-P. Seifert, "New vulnerabilities in 4G and 5G cellular access network protocols: exposing device capabilities," in *Proceedings of the 12th Conference on Security and Privacy in Wireless and Mobile Networks*. Miami Florida: ACM, May 2019, pp. 221–231. [Online]. Available: https://dl.acm.org/doi/10.1145/3317549.3319728

[2] A. Shaik, R. Borgaonkar, N. Asokan, V. Niemi, and J.-P. Seifert, "Practical Attacks Against Privacy and Availability in 4G/LTE Mobile Communication Systems," *arXiv:1510.07563 [cs]*, Aug. 2017, arXiv: 1510.07563. [Online]. Available: http://arxiv.org/abs/1510.07563

[3] D. Rupprecht, K. Kohls, T. Holz, and C. Pöpper, "Breaking LTE on Layer Two," in *2019 IEEE Symposium on Security and Privacy (SP)*, May 2019, pp. 1121–1136, iSSN: 2375-1207.

[4] R. P. Jover, "LTE security, protocol exploits and location tracking experimentation with low-cost software radio," *arXiv:1607.05171 [cs]*, Jul. 2016, arXiv: 1607.05171. [Online]. Available: http://arxiv.org/abs/1607.05171

[5] J. Jin, C. Lian, and M. Xu, "Rogue Base Station Detection Using A Machine Learning Approach," in *2019 28th Wireless and Optical Communications Conference (WOCC)*, May 2019, pp. 1–5, iSSN: 2379-1276.

[6] I. Gomez-Miguelez, A. Garcia-Saavedra, P. D. Sutton, P. Serrano, C. Cano, and D. J. Leith, "srsLTE: an open-source platform for LTE evolution and experimentation," in *Proceedings of the Tenth ACM International Workshop on Wireless Network Testbeds, Experimental Evaluation, and Characterization - WiNTECH '16*. New York City, New York: ACM Press, 2016, pp. 25–32. [Online]. Available: http://dl.acm.org/citation.cfm?doid=2980159.2980163

[7] 3GPP, "Evolved universal terrestrial radio access (e-utra) and evolved universal terrestrial radio access network (e-utran); overall description; stage 2," 3rd Generation Partnership Project (3GPP), Technical specification (TS) 36.300, Oct. 2020. [Online]. Available: http://www.3gpp.org/DynaReport/36300.htm

[8] ——, "Technical Specification Group Core Network and Terminals; Numbering, addressing and identification," 3rd Generation Partnership Project (3GPP), Technical Specification (TS) 23.003, 2020. [Online]. Available: http://www.3gpp.org/DynaReport/23003.htm

[9] ——, "Technical Specification Group Core Network and Terminals; Network architecture," 3rd Generation Partnership Project (3GPP), Technical Specification (TS) 23.002, 2020. [Online]. Available: http://www.3gpp.org/DynaReport/23002.htm

[10] ——, "Evolved universal terrestrial radio access (e-utra); physical channels and modulation," 3rd Generation Partnership Project (3GPP), Technical specification (TS) 36.211, Oct. 2020. [Online]. Available: http://www.3gpp.org/DynaReport/36211.htm

[11] ——, "Evolved universal terrestrial radio access (e-utra); user equipment (UE) radio transmission and reception," 3rd Generation Partnership Project (3GPP), Technical specification (TS) 36.101, Oct. 2020. [Online]. Available: http://www.3gpp.org/DynaReport/36101.htm

[12] ——, "Evolved universal terrestrial radio access (e-utra); base station (BS) radio transmission and reception," 3rd Generation Partnership Project (3GPP), Technical specification (TS) 36.104, Oct. 2020. [Online]. Available: http://www.3gpp.org/DynaReport/36104.htm

[13] ——, "Evolved universal terrestrial radio access (e-utra); medium access control (MAC) protocol specification," 3rd Generation Partnership Project (3GPP), Technical specification (TS) 36.321, Oct. 2020. [Online]. Available: http://www.3gpp.org/DynaReport/36321.htm

[14] Jaeku Ryu, "ShareTechnote." [Online]. Available: http://www.sharetechnote.com/html/BasicProcedure_LTE_Cell_Search.html

[15] 3GPP, "Evolved universal terrestrial radio access (e-utra); physical layer procedures," 3rd Generation Partnership Project (3GPP), Technical specification (TS) 36.213, Oct. 2020. [Online]. Available: http://www.3gpp.org/DynaReport/36213.htm

[16] ——, "Universal Mobile Telecommunications System (UMTS); LTE; 5G; Non-Access-Stratum (NAS) protocol for Evolved Packet System (EPS);," 3rd Generation Partnership Project (3GPP), Technical Specification (TS) 24.301, 2020. [Online]. Available: http://www.3gpp.org/DynaReport/24301.htm

[17] ——, "Evolved universal terrestrial radio access (e-utra); radio resource control (RRC); protocol specification," 3rd Generation Partnership Project (3GPP), Technical specification (TS) 36.331, Oct. 2020. [Online]. Available: http://www.3gpp.org/DynaReport/36331.htm

[18] ——, "Evolved universal terrestrial radio access (e-utra); LTE positioning protocol (LPP)," 3rd Generation Partnership Project (3GPP), Technical specification (TS) 36.355, Jul. 2020. [Online]. Available: http://www.3gpp.org/DynaReport/36355.htm

[19] J. Liu, B. Wu, and P. Li, "On Timing Offset and Frequency Offset Estimation in LTE Uplink," in *Wireless Communications and Applications*, P. Sénac, M. Ott, and A. Seneviratne, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2012, vol. 72, pp. 265–274, series Title: Lecture Notes of the Institute for Computer Sciences, Social Informatics and Telecommunications Engineering. [Online]. Available: http://link.springer.com/10.1007/978-3-642-29157-9_25

[20] 3GPP, "Evolved universal terrestrial radio access (e-utra); physical layer; measurements," 3rd Generation Partnership Project (3GPP), Technical specification (TS) 36.214, Jul. 2020. [Online]. Available: http://www.3gpp.org/DynaReport/36214.htm

[21] C. Sørseth, S. X. Zhou, S. F. Mjølsnes, and R. F. Olimid, "Experimental Analysis of Subscribers' Privacy Exposure by LTE Paging," *Wireless Personal Communications*, vol. 109, no. 1, pp. 675–693, Nov. 2019. [Online]. Available: https://doi.org/10.1007/s11277-019-06585-7

[22] S. Kumar, E. Hamed, D. Katabi, and L. Erran Li, "LTE radio analytics made easy and accessible," in *Proceedings of the 2014 ACM conference on SIGCOMM - SIGCOMM '14*. Chicago, Illinois, USA: ACM Press, 2014, pp. 211–222. [Online]. Available: http://dl.acm.org/citation.cfm?doid=2619239.2626320

[23] "Wireshark · Go Deep." [Online]. Available: https://www.wireshark.org/

[24] 3GPP, "Digital cellular telecommunications system (Phase 2+) (GSM); Universal Mobile Telecommunications System (UMTS); LTE; Mobile

radio interface Layer 3 specification; Core network protocols;," 3rd Generation Partnership Project (3GPP), Technical Specification (TS) 24.008, 2020. [Online]. Available: http://www.3gpp.org/DynaReport/24008.htm

[25] "json-flatten: Python functions for flattening a JSON object to a single dictionary of pairs, and unflattening that dictionary back to a JSON object." [Online]. Available: https://github.com/simonw/json-flatten

[26] N. Bui and J. Widmer, "OWL: a reliable online watcher for LTE control channel measurements," in *Proceedings of the 5th Workshop on All Things Cellular Operations, Applications and Challenges - ATC '16*. New York City, New York: ACM Press, 2016, pp. 25–30. [Online]. Available: http://dl.acm.org/citation.cfm?doid=2980055.2980057

[27] "Products | SRS." [Online]. Available: https://www.softwareradiosystems.com/products/

[28] "WaveJudge 5000 Wireless test system for LTE and WiMAX." [Online]. Available: https://www.sanjole.com/our-products/wavejudge-test-system/

[29] "The Leader in Software Defined Spectrum Analysis." [Online]. Available: https://thinkrf.com/

[30] H. Zhang, Z. Zhang, S. Zhang, S. Xu, and S. Cao, "Fingerprint-Based Localization Using Commercial LTE Signals: A Field-Trial Study," in *2019 IEEE 90th Vehicular Technology Conference (VTC2019-Fall)*, Sep. 2019, pp. 1–5, iSSN: 2577-2465.

[31] X. Ye, X. Yin, X. Cai, A. Pérez Yuste, and H. Xu, "Neural-Network-Assisted UE Localization Using Radio-Channel Fingerprints in LTE Networks," *IEEE Access*, vol. 5, pp. 12 071–12 087, 2017, conference Name: IEEE Access.

[32] G. Pecoraro, S. Di Domenico, E. Cianca, and M. De Sanctis, "LTE signal fingerprinting localization based on CSI," in *2017 IEEE 13th International Conference on Wireless and Mobile Computing, Networking and Communications (WiMob)*, Oct. 2017, pp. 1–8.

[33] J. D. Roth, M. Tummala, J. C. McEachen, and J. W. Scrofani, "On Location Privacy in LTE Networks," *IEEE Transactions on Information Forensics and Security*, vol. 12, no. 6, pp. 1358–1368, Jun. 2017, conference Name: IEEE Transactions on Information Forensics and Security.

[34] B. A. Pimentel, "Passive Geolocation in a 4G WIMAX Single Base Station Scenario," p. 102, 2013.

[35] Nakarmi Prajwol Kumar, Noamen Ben Henda, and Vlasios Tsiatsis, "3GPP Release 15 and the battle against false base stations," Jan. 2019, last Modified: 2020-04-27T11:21:46+00:00. [Online]. Available: https://www.ericsson.com/en/blog/2019/1/3gpp-release15

[36] H. Yang, S. Bae, M. Son, H. Kim, S. M. Kim, and Y. Kim, "Hiding in Plain Signal: Physical Signal Overshadowing Attack on LTE," p. 19.

Appendix

UE model	Modem	Constant Error [m]	std [m]
Samsung Galaxy s10	Exynos 9820	11.29	7.22
Samsung Galaxy a8	Exynos 7885	-26.62	4.77
Samsung Galaxy s5	Qualcomm Gobi 4G	-	-
Huawei P20 Lite	Kirin 659	-24.47	2.13
Huawei P20 Pro	Kirin 970	-9.34	2.90
Huawei P30 Lite	Kirin 710	-10.27	0.98
Huawei P30	Kirin 980	-24.51	1.49
Xiaomi Mi9	Qualcomm X24 LTE	10.44	2.20
Xiaomi MiX 3	Qualcomm X24 LTE	11.57	1.60
Nokia 1.3	Qualcomm X5 LTE	-	-
Sony Xperia X	Qualcomm X8 LTE	-11.20	4.78
Google Nexus 5X	Qualcomm X10 LTE	5.08	2.51
Google Pixel 2	Qualcomm X16 LTE	-13.52	2.32
Google Pixel 3a	Qualcomm X12 LTE	4.46	2.14
Google Pixel 4	Qualcomm X24 LTE	12.88	1.67
HTC U12+	Qualcomm X20 LTE	-13.66	1.55
OnePlus 7T	Qualcomm X24 LTE	12.66	1.42
iPhone 7	Intel XMM7360	-23.86	0.88
iPhone 8	Intel XMM 7480	-23.65	2.28
iPhone X	Intel XMM7480	-25.64	3.75
iPhone 11	Intel XMM 7660	-23.19	2.49
iPhone 11 Pro	Intel XMM 7660	-25.35	2.46

Table 1: Mobile phones used in the experiments.

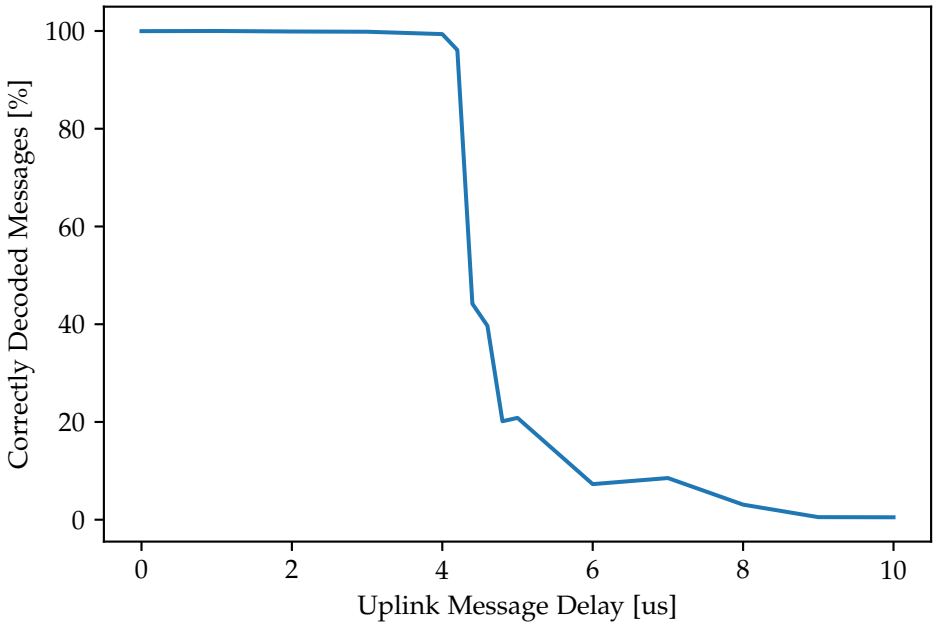


Figure 1: Percentage of correctly decoded uplink messages by our sniffer as a function of time delay of arrived messages from the start of a frame.

Eidgenössische Technische Hochschule Zürich
Swiss Federal Institute of Technology Zurich

Declaration of originality

The signed declaration of originality is a component of every semester paper, Bachelor's thesis, Master's thesis and any other degree paper undertaken during the course of studies, including the respective electronic versions.

Lecturers may also require a declaration of originality for other written papers compiled for their courses.

I hereby confirm that I am the sole author of the written work here enclosed and that I have compiled it in my own words. Parts excepted are corrections of form and content by the supervisor.

Title of work (in block letters):

LTE MONITORING

Authored by (in block letters):
For papers written by groups the names of all authors are required.

Name(s):	**First name(s):**
Kotuliak	Martin

With my signature I confirm that

- I have committed none of the forms of plagiarism described in the 'Citation etiquette' information sheet.
- I have documented all methods, data and processes truthfully.
- I have not manipulated any data.
- I have mentioned all persons who were significant facilitators of the work.

I am aware that the work may be screened electronically for plagiarism.

Place, date	**Signature(s)**
Zurich, 19.12.2020	

For papers written by groups the names of all authors are required. Their signatures collectively guarantee the entire content of the written paper.