RUHR-UNIVERSITÄT BOCHUM

Analysing Security-related Signals Using Software-defined Radio

Thomas Rudolph

Bachelor's Thesis. February 12, 2013.
Chair for Embedded Security – Prof. Dr.-Ing. Christof Paar
Advisor: Dipl.-Ing. David Oswald

Abstract

Software-defined radio (SDR) has grown to be a serious alternative to traditional radio architectures. With the higher flexibility of software based filtering and demodulation (compared to hardware modifications necessary in traditional radios), much less expensive receivers and transmitters for specialised broadcast protocols can be built using SDR. With this simplification comes a higher risk of eavesdropping and manipulation of signals. As a result, security-related wireless communication is highly advised to integrate modern security mechanisms to cope with the new threats. In this thesis, we use SDR devices to analyse several security-related signals to point out possible threats.

A spectrum comparator software developed for this thesis allows on the one hand, manual generation of frequency spectra and on the other hand, automatic detection and logging of signals. The software supports two SDR devices, i.e., the USRP series by Ettus Research Inc., and USB sticks usually sold for receiving digital television using the Realtek RTL2832U chipset.

Furthermore, in this thesis, we analyse signals of remote controls and we look at the design of the TETRA standard, where we point out security weaknesses by analysing real world TETRA networks. In addition, we set up a GSM base station using the OpenBTS software and the USRP2.

Declaration

I hereby declare that this submission is my own work and that, to the best of my knowledge and belief, it contains no material previously published or written by another person nor material which to a substantial extent has been accepted for the award of any other degree or diploma of the university or other institute of higher learning, except where due acknowledgment has been made in the text.

Erklärung

Hiermit versichere ich, dass ich die vorliegende Arbeit selbstständig verfasst und keine anderen als die angegebenen Quellen und Hilfsmittel benutzt habe, dass alle Stellen der Arbeit, die wörtlich oder sinngemäß aus anderen Quellen übernommen wurden, als solche kenntlich gemacht sind und dass die Arbeit in gleicher oder ähnlicher Form noch keiner Prüfungsbehörde vorgelegt wurde.

Thomas Rudolph

Contents

1 Introduction

The development of Software-defined Radio (SDR) systems has made reasonable progress during the recent years due to increased efficiency of computer hardware [Han09]. As an effect, SDR receivers and transmitters are by now sold at a comparable price to traditional radio hardware of the same quality class. This allows new operational fields in low-cost environments, as SDR devices have a higher flexibility and with this more applications, e.g., a low-cost spectrum analyser.

1.1 Motivation

With this flexibility comes an increased risk of unauthorised eavesdropping and signal manipulation. Unauthorised eavesdropping and publishing of secret information may cause severe financial damage to companies. Transmitting manipulated data using SDR devices can cause safety threats to the population, e.g., provoking a power failure by manipulating signals in a TETRA network of an electricity operator. By now, an SDR stick for the USB port sold for approximately 15 EUR is sufficient to receive and decode GSM network signals as well as signals of TETRA networks, which are often operated in an unencrypted mode in order to save hardware costs. Besides the USB stick, professional SDR devices are available, e.g., the USRP series by Ettus Research. The Osmocom project currently developes a semiprofessional SDR device. A developer preview edition is sold at 180 EUR [Sys].

1.2 Related Work

Autonomous spectrum monitoring with USRP hardware has been successfully demonstrated, e.g., by [Lui12] to record spectrum changes of analogue television broadcasts. Research has also been done to understand radio wave propagation and reflection on small scales, e.g., houses and the surrounding property. In doing so, USRP SDRs are used to simultaneously record the levels of a signal at different positions of the property with a receiver software written in GNU Radio and a subsequent analysis of the collected data [Muh12].

Systems for analysing TETRA networks for professional usage already exist built with conventional radio hardware, e.g., by Boger Electronics [Bog] and RSI [RSI13]. Such systems allow automatic detection and recording of TETRA signals, including voice calls.

Due to the cost of this specific hardware and commercial software, private researchers are only known to have tested TETRA networks using SDR and open source software

decoders written by the Osmocom community. It is reported that people have successfully been eavesdropping on the network of the Berlin subway (BVG) [hei11]. One can assume that rather few details of successful TETRA reception are shared, as many countries, e.g., Germany, prohibit unauthorised reception of authority communication channels [Bun].

1.3 Contribution

In this thesis, we describes the development of a spectrum analyser software that allows autonomous detection and recording of signals appearing at unpredictable points in time. Furthermore, local TETRA signals are received and analysed. Also, a set-up of a GSM base station running on SDR hardware is tested and described.

All tests in this thesis are performed using SDR hardware. On the one hand, a USRP2 by Ettus Research Inc. with radio daughterboards for miscellaneous frequency ranges is used. On the other hand, an inexpensive DVB-T receiver stick for the USB port by ezcap is used. It employs a RTL2832U chipset by Realtek (that can be switched into SDR mode) and a Fitipower FC0013 tuner. Besides the price and the difference in flexibility, the USB stick also differs in the possible maximum sample rate of only about 3.6 MHz compared to 100 MHz of the USRP2 [Ettc]. Furthermore, the USRP hardware provides not only reception but also the transmission of signals when equipped with a suitable radio daughterboard [Ettd].

1.4 Organisation of this Thesis

The remainder of this thesis is organised as follows. In Chapter 2 we describe the benefit of SDR hardware over traditional radios and present a software that allows autonomous or manual spectrum recording (and simple filtering) using SDR devices. The required theoretical background is given in this section as well. Next, in Chapter 3 we describe the TETRA protocol and security design and explain the decoding of TETRA networks using Osmocom software. This is followed by a classification of the TETRA networks operating in the city of Bochum and the respective network structure. In Chapter 4 a practical look at the OpenBTS software and results of a short test run of this software is given. Finally, we conclude in Chapter 5.

2 Spectrum and Signal Analysis

The SDR technology allows a flexible usage of one single device for various signals which differ in bandwidth, frequency, and modulation mode. When using SDR devices, the properties of the devices can be adjusted from within a software instead of exchanging hardware components like in traditional radio designs. SDRs are therefore more suitable for performing signal analysis.

2.1 Advantages of SDR for Signal Analysis

One feature of SDRs is the transfer of a complete signal spectrum in a selected frequency range with a defined sample rate to the computer. This means that all received data is available in a raw format and can be used without the restrictions and information losses of traditional radio hardware, e.g., caused by a fixed filter bandwidth or signal demodulation. Therefore, one single device can work as a receiver for very different types of signals, e.g., for receiving a car key and for receiving television broadcasts. The same properties hold for the transmission of signals.

With traditional radio hardware, a special receiver for almost each radio communication standard is needed. One the one hand, because of different signal properties, e.g., the signal bandwidth. On the other hand, in many devices, the radio hardware and the decoder hardware are merged on the same board or are not intended to work separately. As a result, using existing radio hardware for an "unintended" purpose turns out to be difficult or even impossible, which means high costs for specialised radio hardware, e.g., for research purposes. In contrast, for SDR devices, in many cases only software changes are necessary to support another signal standard.

2.2 Working Principles of Radio Devices

Like common traditional radio hardware, many existing SDR devices use a Variable-Frequency Oscillator (VFO) in order to down convert a frequency band selected by the user to a constant Intermediate Frequency (IF), where it can be filtered and amplified. From the IF the signal is again downmixed to a low-frequency baseband signal and handed over to the Analogue to Digital Converter (ADC). In traditional hardware, instead of the ADC, a demodulator and filters for a specific mode of modulation, e.g., FM, are put in place.

A VFO can be freely tuned within a certain frequency range, depending on the specifications of the receiver or transmitter hardware. The oscillator signal of a VFO always has a constant offset to the frequency range that is actually being received. Thus

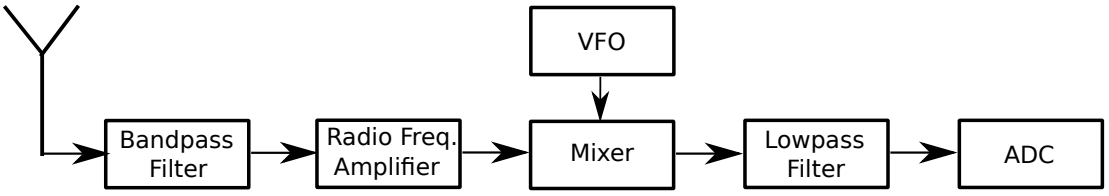

Figure 2.1: Block diagramme of a DCR as recreated from an example circuit [The].

Table 2.1: Available USRP daughterboards in the tests.

Board Name	Frequency Range	Capabilities
LFTX-LF rev 2.2	0 - 30 MHz	Transmitter
LFRX-LF rev 2.2	0 - 50 MHz	Receiver
MICROTUNE Tuner 4937 DI5	50 - 860 MHz	Receiver
FLEX400	400 - 500 MHz	Receiver and Transmitter
FLEX900	750 - 1050 MHz	Receiver and Transmitter

the difference between the VFO's frequency and the desired frequency results is the IF:

$$f_{IF} = f_{VFO} - f_{actual}$$

The DVB-T stick used in this thesis and shown in Figure 2.3 using the RTL2832U chipset supports two different IFs, i.e., 4.56 MHz and 36.125 MHz [Rea]. Considering the USRP2 shown in Figure 2.2, the intermediate frequency is zero, which makes it an direct-conversion receiver (DCR), i.e., the downmixed signal is already the low-frequency or baseband signal so that only one frequency conversion has to take place instead of two. A block diagramme of an DCR can be seen in Figure 2.1. The advantages are a less complex circuit and the suppression of "second-channel frequencies" which otherwise are likely to arise in the receiver and may disturb wanted signals [Ash01]. The DCR principle is not very common in AM and FM receivers due to problems of locking the local oscillator to the signals by means of analogue technology. However, this receiver concept became more popular in receivers for digital transmission standards [Wan06], particularly linked to the usage of digital modulations, e.g., QAM and FSK, which allow for direct processing of the baseband signal.

The USRP receiver family uses exchangeable daughterboards as receivers and transmitters. These boards allow the usage of more specialised receiver and transmitter hardware for certain purposes, i.e., receivers supplying a higher quality within a small frequency range or broadband boards covering several hundred MHz at the cost of sensitivity. For the experiments in this thesis, the daughterboard listed in Table 2.1 are available.

The daughterboards are plugged onto the USRP main board using one or two connectors, depending on the number of receivers supported on the daughterboard. Hence, one or

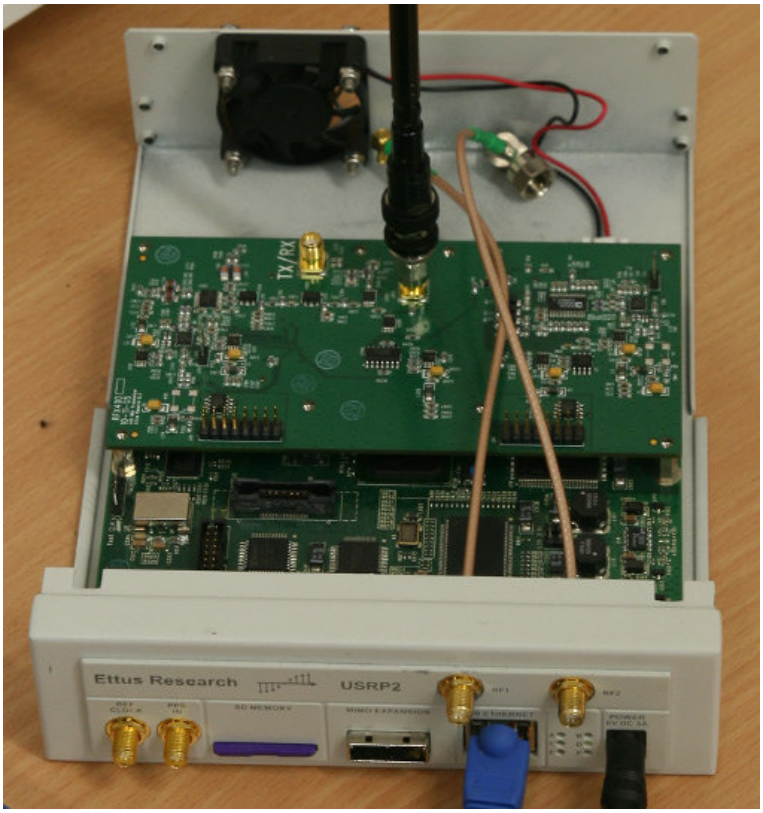

Figure 2.2: USRP2. The small board on top is the FLEX400 daughterboard plugged into the corresponding connectors of the main board.

Figure 2.3: RTL2832U USB stick, various models exist.

two antenna connectors are supplied on each daughterboard, which can be connected with short cables visible in Figure 2.2 to fixed connectors on the casing box of the USRP so that operation in a closed case is possible. The USRP2 runs firmware supplied on a SD memory card that has to be inserted into the integrated card reader. The connection with the computer is realised using Gigabit Ethernet. The USRP2 needs a power supply with a voltage of 6 Volts.

2.3 Discrete Fourier Transformation

As an SDR device provides the original raw signal spectrum to the computer, the reconstruction of this spectrum within the received frequency range is possible by applying a Discrete Fourier Transformation (DFT) on this data [Doub].

The DFT is commonly used in digital signal processing. It converts a discrete-time signal into a frequency spectrum, or to be more precise, into a frequency domain representation. Whenever an analogue signal is passed to an ADC, the digitalised result is a quantised version of the original signal, because converters only sample the analogue signal with a finite sample rate and a finite amplitude resolution. The Nyquist–Shannon sampling theorem [Phi07] implies that for sampling a signal without loss of information, the sampling rate must be larger than twice the maximum frequency of the signal:

$$f_{samp} > 2 \cdot f_{max}.$$

The usage of DFT is necessary in most applications that deal with processing of analogue data, e.g., not only radio signals but also image processing when using lossy image formats, e.g., JPEG.

A DFT uses a finite number of samples of a discrete-time signal as input. It generates a discrete frequency spectrum by transforming the original data (that consists of the real and imaginary part of the signal) into absolute values. These absolute values resemble the signal levels at all frequencies of the sample. Therefore, the discrete-time signal is multiplied with a root of unity (i.e., coefficients equally distributed on the unit circle of the complex plane):

$$X_k = \sum_{n=0}^{N-1} x_n e^{-\frac{2\pi i}{N} kn} \qquad k = 0, \ldots, N-1$$

where x_n are the input values, i.e., the measured values of the discrete-time signal and $e^{-\frac{2\pi i}{N} kn}$ are the unity roots on the complex plane with N the total number of input samples. X_k is called the Fourier coefficient. [Uni05] The result of this transformation is a periodical function of discrete equidistant real number values.

In a computer application for spectrum analysis, the DFT can be implemented as a subroutine of the software. A numerical computation environment, e.g., MATLAB can be used to perform the DFT, too. For running the transformation, a certain number of signal samples is captured by a receiver and transferred to the computer. After that these samples are used to calculate the DFT.

2.4 Practical Signal Analysis

Radio receivers may suffer from influences like overdriving the receiver. The elimination of such influences is one criteria for the quality of the devices.

During the tests with the RTL2832U chipset, influences from intermodulations were noticed, i.e., a mixture of several existing radio signals which are mapped to another frequency where they may cause disturbance of real signals. These intermodulations do not exist in reality, but are only generated in the receiver caused by non-linear components of a radio receiver, e.g., diodes, that mix signals by adding or subtracting frequencies of existing radio signals or their harmonics [Wik13]. Usually, a certain, rather strong signal of a transmitter is necessary to produce mixing products. This can be verified experimentally with the Realtek USB stick where an inappropriate signal gain setting causes the appearance of rather many intermodulations.

Another effect of a strong local signal source is overdriving the receiver, i.e., the sum of received signal levels exceeds the level range where the circuits of the receiver work linearly. As a result, an increase of the general noise level and thus a decrease in the signal-to-noise-ratio (SNR) is noticed. One incident noticed to occasionally cause overdriving of the receiver is buses driving past the test site in approx. 10 metres distance. The buses transmit a short TETRA signal. Overdriving even occurs if the receiver gain is disabled. Even for weak and temporary signals, the automatic gain control of the stick tends to overdrive the receiver. Thus, the receiver seems not to have a sophisticated mechanism to regulate its gain automatically.

Having optimised a dipole antenna to the wavelength corresponding to the FM radio band at approx. 100 MHz by using the formula

$$\lambda = \frac{c}{f}$$

where λ is the wavelength, c the speed of light and f the desired frequency, another source causing overdriving was found to be the large radio and TV transmitter site at Langenberg in a distance of approx. 12 km to the test site, broadcasting a total sum of 639 kW Effective Radiated Power (ERP) analogue FM radio and 400 kW ERP of DVB-T signals [UKW13]. Placing the antenna at a site where more walls are in the path between the transmitter and the antenna, the noise level decreased by up to 20 dB, depending on the selected gain of the receiver's signal amplifier.

Figure 2.4 shows a possible result of overdriving the receiver. Only the left large green peak is caused by a bus transmitting while driving past the receiver site. The large green signal on the right originates only in the receiver at the same time.

2.5 Spectrum Comparator Software

In order to control SDR devices and to retrieve and process data, an application was developed in this thesis. The application allows manual and autonomous comparison and analysis of signal spectra. In many cases, numerical computation environments like

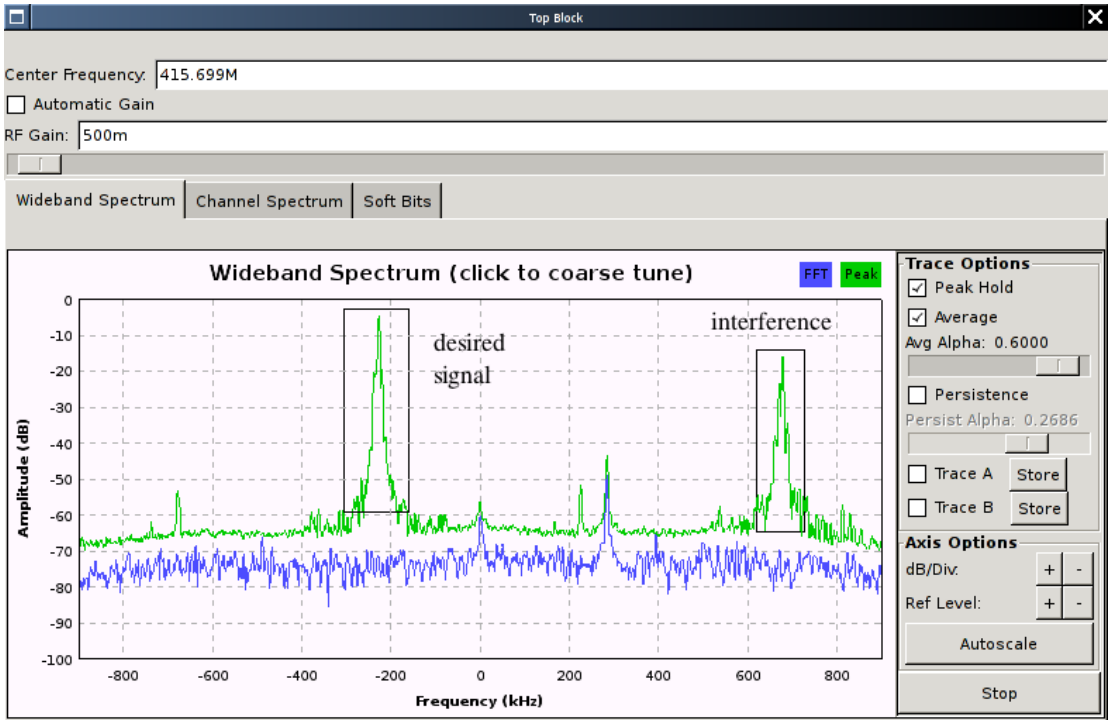

Figure 2.4: Bursts (green) during a local TETRA transmission in the observed frequency range. The tall peak at approx. -220 kHz is the actual signal, the large peak on the right is caused by overdriving the RTL2832U receiver during the transmission.

MATLAB are used for this purpose, but the complexity of such environments can be seen as a benefit and a disadvantage at the same time, because usually a longer period of time is necessary to get familiar with the functionality and find the relevant functions of these environments. Besides, many computing environments are proprietary and not free of charge. On the other hand, the benefit of a specialised software is a fast and flexible usage (and a low cost).

The application written for this thesis "`spectracom`" limits its functionality to receiving, calculating, and comparing signal spectra. It runs on Linux platforms and is compatible with the RTL2832U USB stick as well as USRP devices using the UHD C++ libraries. `spectracom` hands over the data to the DFT function to calculate the signal spectrum. By comparing different samples, it is decided whether a significant change beyond a certain threshold has occurred in the spectrum. These events can either be shown in a graph or logged autonomously, while the retrieved data is stored on disk to allow manual review by the user later. The software can be run in three different modes to serve several purposes:

- **GUI mode**: Allows manual adjustments of settings, i.e., frequency, number of

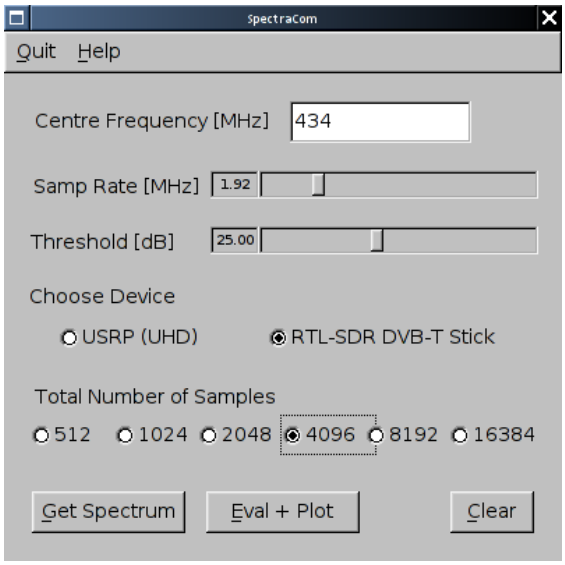

Figure 2.5: Main window of the developed spectrum comparator software.

samples, threshold, and sample rate with a few mouse clicks or keyboard inputs. A graph of the frequency spectra including the signal difference between two spectra can be drawn.

- **Command Line mode**: Can perform the same adjustments as the GUI mode, although a graph is not being shown automatically. Settings are given as command line parameters to the application.

- **Autonomous mode**: Is configured the same way as the command line mode, but allows to autonomously run for a defined or infinite number of events. If a signal difference beyond a certain threshold is detected, the signal is saved and an entry in a logfile is written.

The application is written in C++ using FLTK 1.3 as GUI toolkit. Pieces of the code were taken from the samples written by the developers of the UHD libraries, i.e., the code for communicating with the USRP and code to perform the DFT. As the code by Ettus is released under the GNU General Public Licence Version 3 or later, also `spectracom` is under this licence. The application makes use of parts of the example code in `ascii_art_dft.hpp` for the DFT and `rx_samples_to_file.cpp` for communicating with the USRP. Both files are attached to the UHD driver package [Etta]. The communication with the RTL2832U USB stick takes place by calling the external `rtl_sdr` application by the Osmocom project [Osmb].

The GUI mode window shown in Figure 2.5 lets the user set the centre frequency of the spectrum to be received. The sample rate slider is used to adjust the bandwidth

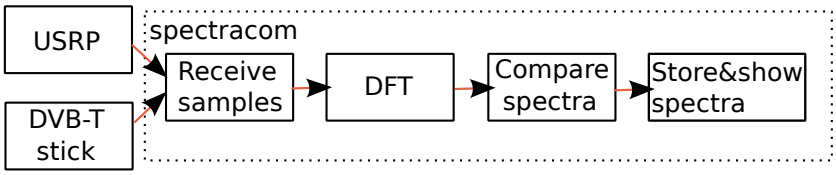

Figure 2.6: Sketch of the data processing order in `spectracom`.

of the received samples to a desired value. Whereas the RTL2832U USB stick can only handle sample rates up to 3.6 Mbit/s, the USRP2 works with up to 100 Mbit/s. The threshold value has no impact on the received signal but on the printed graph only. It is used to point out frequencies in the graph that differ by more than the selected value in two successive sample snapshots. The capture device and total number of samples has to be selected, too. To run a measurement, one clicks twice onto the *Get Spectrum* button to take two distinct snapshots. One example of utilisation is taking one spectrum snapshot with and one without a device transmitting in order to find out changes in the spectrum and the frequency the device uses. After performing these two measurements, a click on *Eval + Plot* shows a graph with both spectra and all spectral differences above the threshold value. To display the graph, the application relies on the plotting software *Gnuplot*, so the user can zoom in and out using the mouse in order to do further investigations on the spectrum. After clicking the *Clear* button in `spectracom`, it is possible to perform further measurements without the need of restarting the application. Together with the plot, the application shows a window with some statistics about the average noise level and the average frequency of all signals that exceed the threshold. Figure 2.6 shows a very basic sketch of data processing in `spectracom`.

For the command line mode, all implemented parameters can be displayed by typing `./spectracom --help`. As an example, the command `./spectracom --freq 99 --rate 2 --d 1 --samp 2048 --nogui` uses a centre frequency of 99 MHz with a bandwidth of 2 MHz and a total of 2048 samples. It uses device 1, i.e., the USRP, and does not start the GUI. Leaving out the `--nogui` option, these parameters can be used to preset some values in the GUI. The number of samples must be a power of 2. If a deviant value is given, the application calculates the closest lower power of 2 to the given value.

The application uses several files to cache its data. `dft_samp1` and `dft_samp2` contain the received raw samples. `dft1` and `dft2` contain the results of the DFT for the two given samples in a table showing the corresponding signal level for each recorded frequency in dBm in the spectrum. Finally, `dft_log` is again a table that sets every frequency in reference to the signal level of both samples. Besides, in the last line `dft_log` it shows the level difference between both signals if it exceeds the threshold. This is the file that is used by Gnuplot to display the graph. All files are saved in the path, from where the application is invoked. An example for the file format showing the frequency, then the signal level of spectrum 1 and 2 in dBm, and the difference between both spectra is:

```
434.127 -42.8189  -72.4649  29.646
```

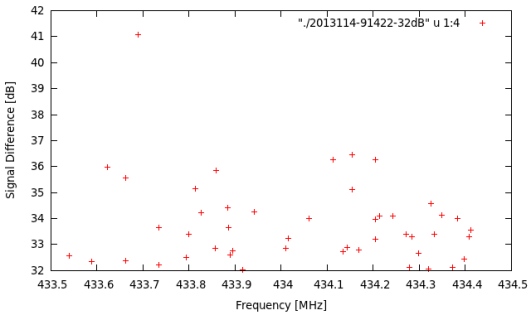

Figure 2.7: Results of autonomous logging, showing no occurance of a local signal during the run.

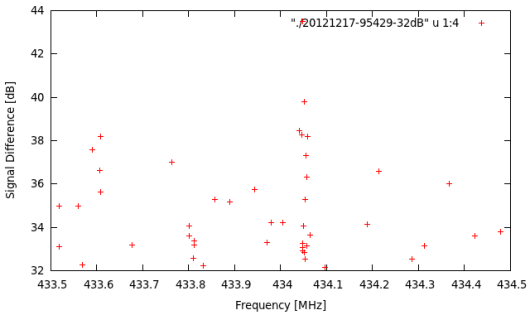

Figure 2.8: A local signal is received during the autonomous run at approx. 434.05 MHz.

The autonomous mode is started in a similar way as the command line mode, as it can only be invoked from command line. To start an automatic monitoring with a centre frequency of 434 MHz using the USRP device and a spectrum of 1 MHz width with a sample size of 2048 each, one uses the command `./spectracom --auto 0 --freq 434 --rate 1 --d 1 --samp 2048`. The application will run infinite and can be stopped pressing *Ctrl-C*. Appending a number larger than zero behind `--auto` limits the sample snapshots to the given number. During the autonomous run, `spectracom` saves files in the directory where it is started. The filenames use the format *year month day-hour minute second-threshold dB.sam*. These files are only saved if at least one value exceeds the threshold compared to the previous sample. Additionally, one single logfile is written for each run. It uses the same name format but without the ending *.sam* and the starting time is used as timestamp. Examples for the filenames are:

```
20121218-184046-35dB     // Logfile
20121218-184350-35dB.sam // Sample
```

To get an overview of received signals during the run, the logfile can be read in by *Gnuplot* or other plotting software. It is recommended to plot the first line against the fourth line of the logfile, to obtain a frequency versus signal difference graph. Figure 2.7 shows a log record with no visible local signal appearing. Hence, the measurement points are to some extent equally distributed. Figure 2.8 shows another record with a clearly visible accumulation of measurement points around one frequency. In addition to the values saved in the `dft_log`, the autonomous logs contain columns with time and date of incidents exceeding the threshold are given. The data of the logfile successively contains information about the frequency, the signal level of spectrum 1 and 2 in dBm, the difference between both signal levels, and finally date and time. An example line:

```
434.44  -94.3013  -126.338  32.0367 2013-1-17  9:53:50
```

Tests with the USB stick reveal a receiver quality which turns out not to give useful results in autonomous mode. This is caused by too high noise fluctuations at a level of wanted signals. Therefore, in contrast to captures with the USRP, graphical analysis of automatic recordings with the USB stick does hardly show a correlation with test signals of, e.g., wireless remote controls.

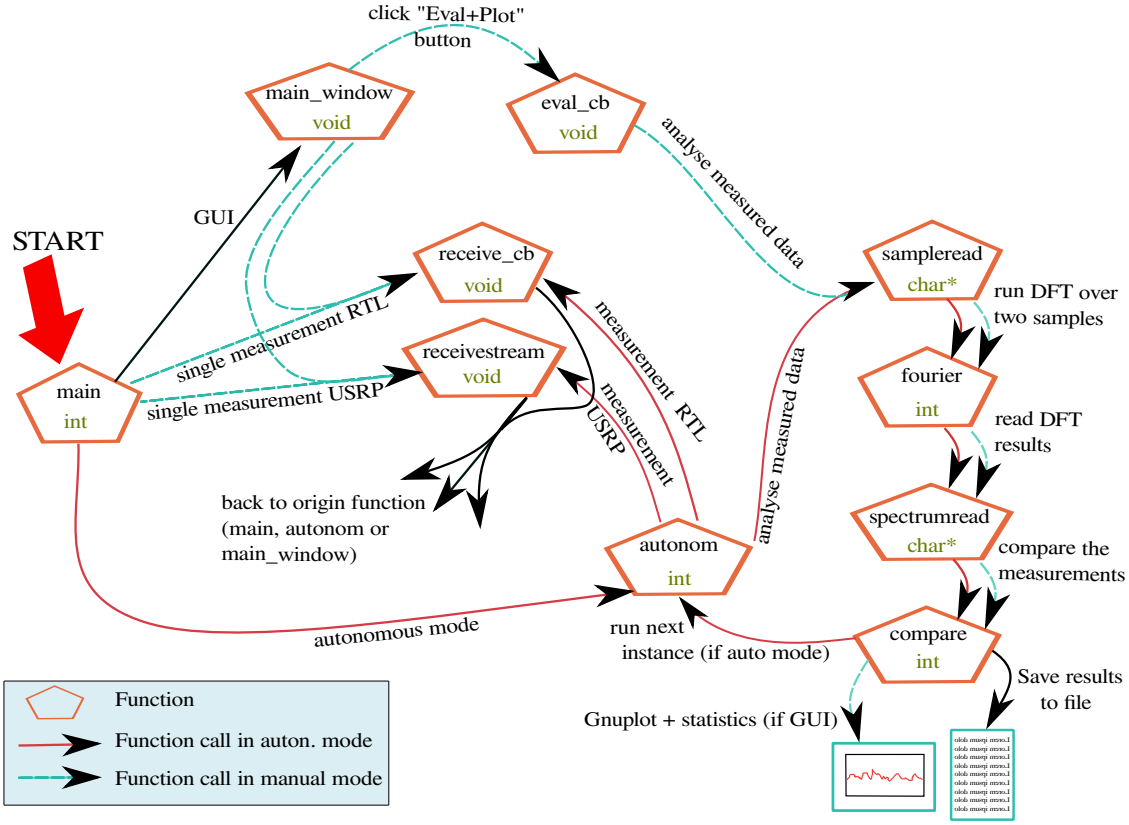

Figure 2.9: Structural sketch of the functions of the `spectracom` software.

Figure 2.9 gives an overview over the structural design of the `spectracom` application. Some callback functions necessary for the GUI but without major impact on the functionality are neglected. A brief description of the functions is given below.

- **main**: This function is responsible for initialisation, processing command line parameters, calling the routines to receive and processing a signal if in command line mode.
- **main_window**: This function is mainly used to show the GUI window.
- **receive_cb**: Here samples are received from the RTL2832U USB stick.
- **receivestream**: This function receives samples from the USRP.

- **eval_cb**: Here inputs from the GUI are fetched and then either *receive_cb* or *receivestream* is called.
- **sampleread**: In this part of the code the received samples are loaded from files into memory.
- **fourier**: This function performs the DFT and saves the results back to files with the names `dft1` and `dft2`. The DFT takes place in the code of the `ascii_art_dft.hpp` file.
- **spectrumread**: Here the result of the two last DFTs are loaded into memory.
- **compare**: This function compares two spectra, calculates the signal difference between both for every frequency and checks whether the difference exceeds the threshold. The function writes the result back into the `dft_log` file.
- **autonom**: This code controls the autonomous mode and calls all necessary functions to perform an automatic measurement.

A known issue in `spectracom` is a problem with memory leaks on some Linux distributions, e.g., Ubuntu, while using the autonomous mode. As a workaround, the environment variable `MALLOC_CHECK_` can be adjusted in order to ask the operating system to take care of the memory allocation. The variable is set with the `export MALLOC_CHECK_=0` command on the console. Another issue occurs when using the USRP in autonomous mode. The connection verifiably fails after almost 1000 signal snapshots which causes the application to stop. A workaround could be running `spectracom` in a shell script within an infinite loop.

To compile the application, it is recommended to use the attached `fltkcomp` file. Two versions of this file exist, for 32-bit and 64-bit systems. The development libraries for FLTK [FLT] (package `libfltk1.3-dev` on Debian based systems) have to be installed as well as the UHD libraries provided by Ettus Research Inc.

2.6 Analysis of Security-related Signals

Additionally to frequency bands exclusively allocated to a specific radio communication system, e.g., DVB-T, TETRA or UMTS, there are further bands reserved to miscellaneous short range radio services, i.e., ISM bands, which are listed in Table 2.2. These systems can serve quite different purposes and may use various signal standards but they share common frequency ranges. Examples of common signals are external sensors of thermometers, car keys, wireless headphones, and Bluetooth. All services have in common that there is no need for their signals to exceed several ten to hundred metres when used for their original purpose.

Within Europe, a further frequency band at 868 to 870 MHz, the Short Range Device (SRD) band, is allocated. It is used for similar purposes as the band at 433 MHz.

Many of the exemplarily stated services may contain security-related information. E.g., the data of a remote temperature sensor might be used to trigger machines or regulate

Table 2.2: ISM bands as allocated by the ITU for Region 1, which includes Europe [Int].

Frequency Range [MHz]	Bandwidth [MHz]	Common Usage
6.765 - 6.795	0.03	
13.553 - 13.567	0.014	RFID, NFC
26.957 - 27.283	0.326	Babyphones
40.66 - 40.7	0.04	
433.05 - 434.79	1.64	Car keys, thermometer sensors
2400 - 2500	100	Bluetooth, wireless LAN
5725 - 5875	150	Wireless LAN
24000 - 24250	250	Radar
61000 - 61500	500	
122000 - 123000	1000	
244000 - 246000	2000	

the cooling in industrial facilities. Using SDR devices, it is possible to design a receiver software for such signals without the need to own specialised hardware. Furthermore, once an original sample of a signal is successfully recorded, it can be used for replay attacks using a SDR transmitter that broadcasts the received signal in order to compromise the receiver system of the service. However, replay attacks only work in simple transmission protocols that do not verify their data using integrity or authentication features. Successfully compromising a communication system that uses a more sophisticated protocol needs more effort. The signal has to be decoded and the protocol design specifications must be public or have to be reverse-engineered. Furthermore, as in every cryptographic protocol, design weaknesses are necessary to successfully manipulate a message.

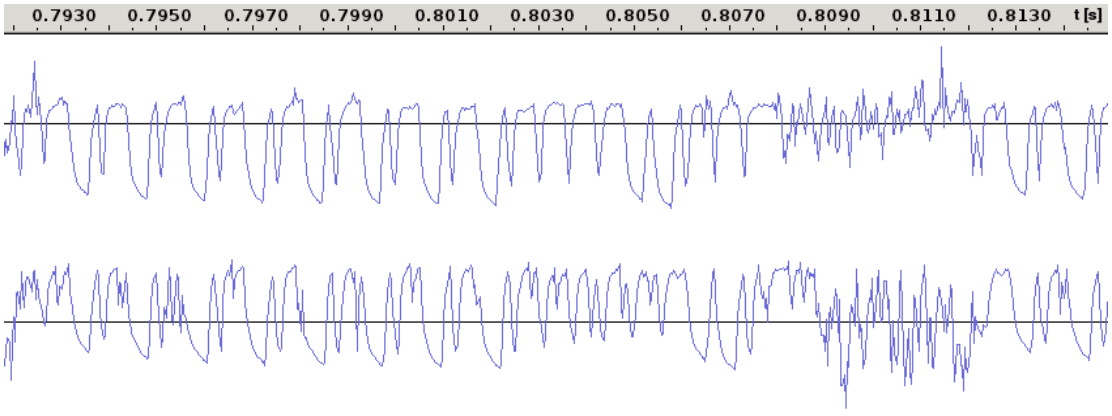

Figure 2.10: Signal of a Christmas decoration's remote control. On top is the *on* signal, below the noisier *off* signal. The signal is recorded with the RTL2832U USB stick.

For this thesis, experiments were performed with several remote controls. One of these remote controls belongs to a decoration candle working at 434 MHz. The signals for the two functions *on* and *off* are captured with the USRP2 using the RFX400 daughterboard and the `rx_samples_to_file` application shipped with UHD. Afterwards, the recorded signals are rebroadcasted using the same parameters with the `tx_samples_from_file` application. These retransmissions are successful, it is possible to turn on and off the candle with the USRP.

The original remote control broadcasts at a centre frequency of 434.060 MHz. Attempts show that the recorded signal is still successfully received if the recording is reduced to only 10 kHz sample bandwidth and merely 16384 samples. Besides, the tests show that the candle still recognises signals 1.3 MHz besides its centre frequency, but with a decreasing success rate. A test series of 40 broadcasts at 433.060 MHz, 1 MHz off the centre frequency, shows only 55 % success rate in a distance of 5 metres. Moreover, the transmission sample rate can be reduced to 95 Hz and increased to 300000 Hz before the signal is not decoded properly anymore. This means that the signal is very robust against variation.

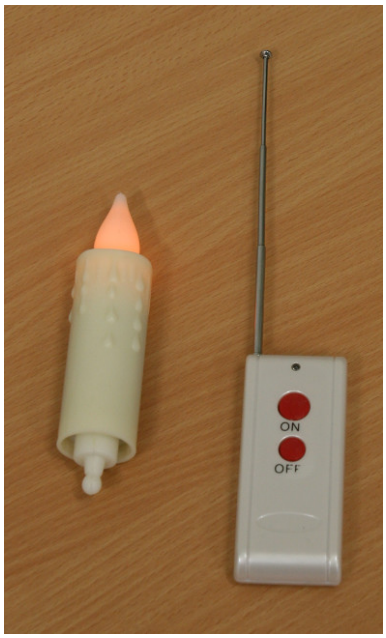

Figure 2.11: Wireless decoration candle and associated remote control for 433 MHz.

Figure 2.12: Nissan car key. Buttons, chip, and antenna can be seen on the board.

A record of the decoration remote control's signal loaded into the signal editor *Audacity* [Aud] reveals that the data signal shown in Figure 2.10 repeats every 195 ms as long as a button keeps being pressed. The message, broadcasted in frequency modulation, seems to consist of approx. 16 Bit, with a device ID at the start and the command to

switch on or off the candle at the end, where two short and two long signals indicate *on* and two long followed by two short signals describe the *off* command. A power remote control with identical design is not only sold for controling rather security-irrelevant decoration, but can be ordered in large amounts from China to be used for a freely chosen field of operation. The data sheet claims that the signal code can be fixed by soldering. A variety of other remote controls is also available, varying in design and number of buttons [Mad]. One can assume that similar straightforward signals are used in these remote controls and that such remote controls are used in various products.

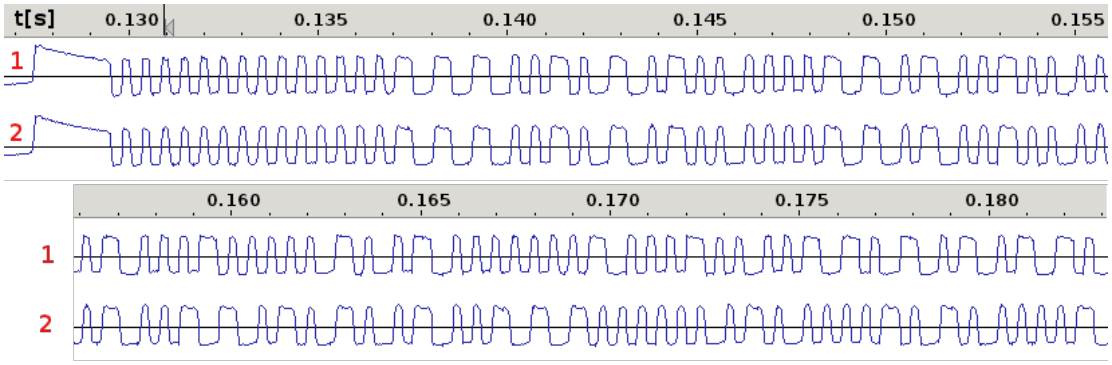

Figure 2.13: Two successive signals (1 and 2) of the Nissan car key signal.

Car keys are also likely to transmit in the 433 MHz frequency band, but the protocol is more secure, though. We analysed the signal of a Nissan key sent using amplitude modulation and shown in Figure 2.13. The key broadcasts three different signals after a button is pressed. At first, a sequence of 14 zero-bits is transmitted, presumably to make the receiver wake up and calibrate. Afterwards, a longer sequence which contains data that differs in some parts after each use is broadcasted. This sequence has a length of 81 to 82 Bit and is repeated three times. Finally, a short sequence of 22 to 23 Bit is repeated as long as the user continues to press the button. Examples of three different captures are given below. The frequency spectrum of the key is depicted in Figure 2.14. The feature of changing the key code after every usage is called a *rolling key*. As can be seen from the time spectrum or sequence, not a simple counter is used that increments the broadcasted message by one bit, but a cryptographic protocol generates a message differing in many bits. In effect, an attacker is not able to create a future value as long as he does not know the algorithm. The receiver hardware in, e.g., a car, will dismiss a wrong message.

Middle sequence of the car key:

```
000000000000001110010101000000110100100000100 010000010010000000100010000110101011O
000000000000001110010101000000110100100000100 110101001001110000000100000101000001
000000000000001110010101000000110100100000100 100010100101000001000110000001110001
```

Final sequence of the car key:

```
000000000000001100 01000
000000000000001100 1100
000000000000001100 10000
```

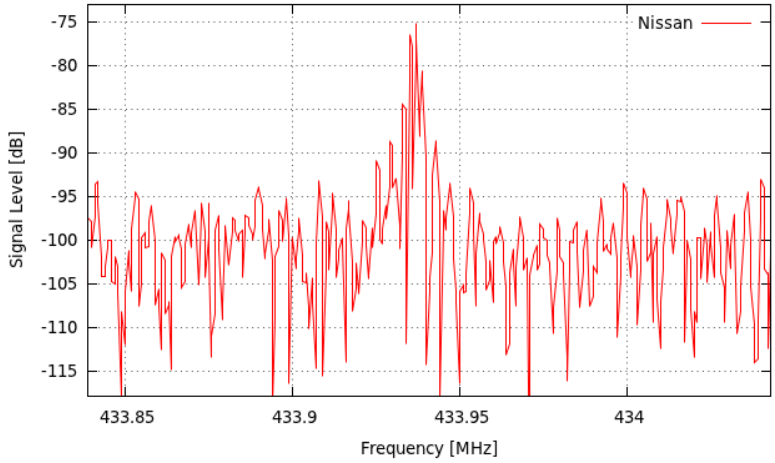

Figure 2.14: Frequency spectrum of a Nissan car key.

Two more wireless remote controls are investigated for this thesis. One of them is used to open and close a garage door. The other one switches a low voltage relay used, e.g., for turning on and off a LED light. Both remote controls use frequencies in the 433 MHz band. The garage door opener is produced by the German company Adolf Tedsen GmbH & Co. KG. This company is specialised on radio controls and RFID transponders [Ted]. It can be assumed that similar products are sold with other security-related components as well. The opener's model name is SKX1WD and it is still sold by the company. It has only one button. Analysing the time spectrum of the garage door opener, it turns out that the signal shown in Figure 2.15 is constant each time the button is pressed, which means that a simple replay attack would be successful. The signal code has a length of approx. 40 ms and repeats until the button is released again.

The remote control of the low voltage relay is sold by Pollin Electronic for approx. 20 Euros including the receiver board. Its model number is HD2RX and it consists of two buttons to control two separate relays. The analysis shows that after a button is pressed, first an initialisation signal containing 12 bit is broadcasted. Afterwards, a sequence of 66 bit is broadcasted containing information. As shown in Figure 2.16, this sequence if different after every usage. A simple replay attack would not be successful in this case.

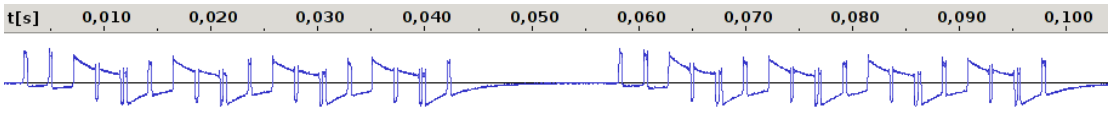

Figure 2.15: Time spectrum of the garage door opener's signal. This signal is constant during each usage. The code is shown two times.

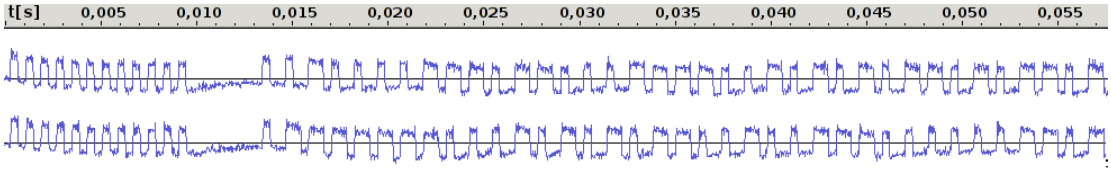

Figure 2.16: Excerpt of two successive time spectra of the remote control by Pollin Electronic using a rolling key. To the left, the initialisation sequence. Then the data sequence which differs in both samples.

3 TETRA

The first plans for developing the TETRA standard as a system for digital radio communication within public safety authorities and organisations date back to the year 1989. TETRA is intended to allow a transition of the radio communication of authorities and organisations from analogue to digital, while implementing additional features like data transfer and signal encryption. Two major disadvantages of analogue systems are a low capacity and the lack of secure encryption methods. The acronym TETRA first meant *Trans European Trunked Radio* and was later changed to *Terrestrial Trunked Radio*, as TETRA services are today on air around the world. TETRA is standardised by the ETSI, a European non-profit organisation that develops other standards in IT and communication as well [ETSa] [Doua].

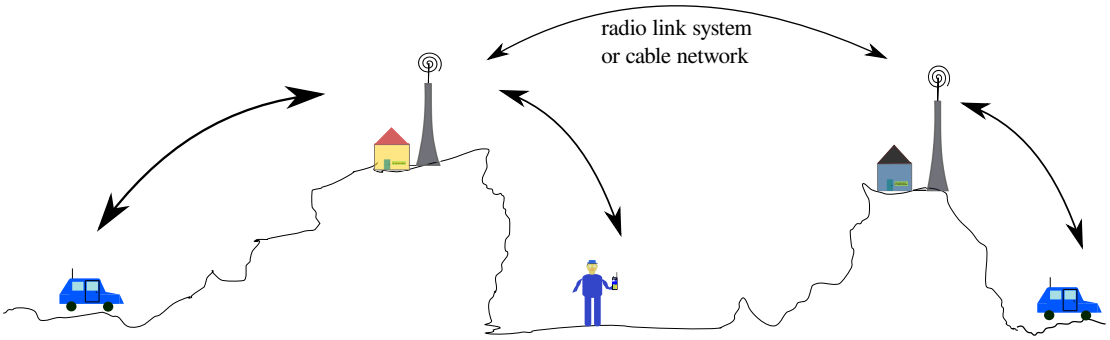

Figure 3.1: A complex trunked network with interconnected base stations and several users can only be realised in terms of a trunked radio system like the one sketched here.

The first working TETRA network was established in 1997. The communication requirements of organisations comprise some features that are not included in GSM, UMTS, or LTE standard, e.g., group calls, location update, and direct mode operation, i.e., communication from one mobile station to others in a local range although no network coverage is available [Doua]. This can be the case in large buildings, valleys or during power outages. Today, TETRA is used by, e.g., police, bus operators, and electricity providers.

The term trunked radio in general describes a radio communications system that supplies commonly shared frequencies for a closed user group. The users do not have their own fixed frequency each, but share a finite amount of channels with all other participants. This is possible as every member only actively uses the system for a short

Table 3.1: Frequency ranges allocated by the CEPT [ETS13] for TETRA.

Uplink Range [MHz]	Downlink Range [MHz]
410 - 420	420 - 430
450 - 460	460 - 470
870 - 888	915 - 933

fraction of time. Bundling the frequencies this way saves frequency resources that can be used by other services and simplifies the installation of base stations or repeaters that help, e.g., improving signal coverage and network infrastructure as depicted in Figure 3.1.

Like GSM and UMTS, TETRA uses unique mobile country and network codes (MCC and MNC) to identify networks. Furthermore, there is a 24-bit Short Subscriber Identity (SSI) for each device as well as for closed groups and foreign participants when roaming.

The frequency ranges given in Table 3.1 are assigned by the European Conference of Postal and Telecommunications Administrations (CEPT) and thus valid in most European countries. Nevertheless, some authorities use a further frequency range at 380 to 400 MHz.

3.1 Physical Working Principles

TETRA may not be confused with TETRAPOL, a system for very similar purposes developed by EADS. Both systems are not compatible with each other, despite the similar name. In some regions, a coexistence of TETRA and TETRAPOL can be observed, for the reason that in many countries operators can freely choose between the standards. While police and fire fighters in Germany use TETRA, the Federal Armed Forces employ TETRAPOL [Fra06].

TETRA uses the Time Division Multiple Access (TDMA) mode with four slots to allow several parallel calls or data transfers by different users as shown in Figure 3.2. This means that the data of up to four different calls or data connections is broadcasted on the same frequency in successive time slots of 14.167 ms length. This is a time period short enough not to be noticed by the user. One device can handle a voice call parallel to data transfers. Voice calls can be duplex, i.e., both parties can hear the opposite party while they are talking like on a regular phone call but unlike most analogue radio systems.

The network is responsible for structure and management. Like other modern mobile networks, TETRA uses a geographic network structure separated in location areas and cells. While a Base Transceiver Station (BTS) usually controls one to three network cells, a location area usually comprises some dozen adjacent cells and thus also several BTS. One benefit of location areas is that mobile stations do not need to reregister to the network when they change to another cell as long as they stay within one location area. In case of an incoming call, the network tries to reach the mobile station within

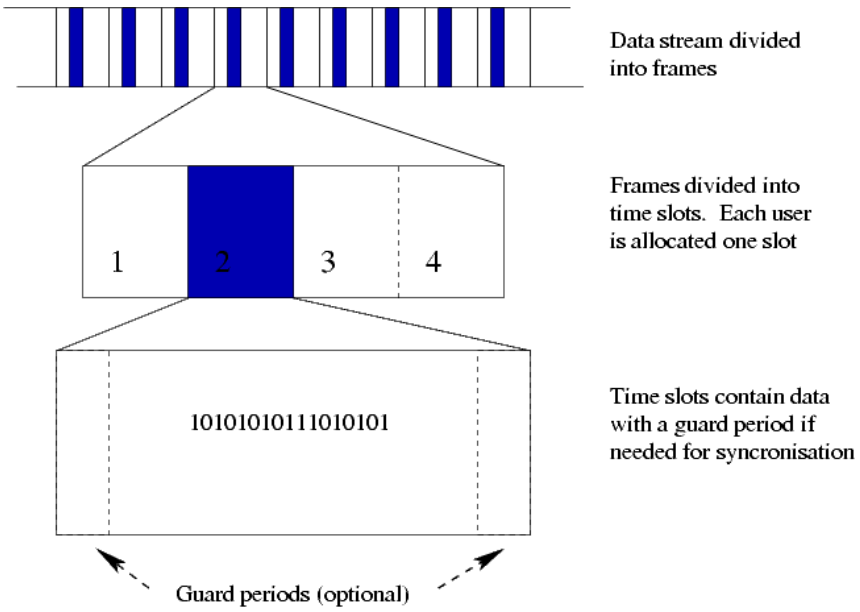

Figure 3.2: TDMA working principle: shared usage of frequencies by using successive time slots [Use04].

the whole location area. On the other hand, a mobile station saves energy, because it rarely needs to transmit in idle mode. Management comprises establishing connections, a location register of the mobile stations, data transfer to other networks, cell handover, and authentication.

It is possible to use a mixture of direct mode and trunked mode operation in some cases. One mobile station in direct mode can work as a repeater for other stations so that two stations that are not in range of each other but of the repeater station can communicate with each other, though. A more sophisticated case is implemented as well: if one of these mobile stations can reach a TETRA network then other mobile stations can use this mobile station as a repeater to reach the network as well.

In the original TETRA standard, data transfer can only be performed with rather slow data rates of up to 28.8 kBit/s. There is also a service similar to SMS on typical phone networks called Short Data Service SDS [ETS13].

3.2 Security

Encryption is not mandatory in TETRA networks and, in fact, many networks have turned out to broadcast without encryption. One reason is that equipment manufacturers sell encryption as an add-on, to make devices with encryption capabilities to be more expensive [Tim11]. As depicted in Figure 3.3, TETRA supports end-to-end encryption and encryption on the air interface, i.e., the wireless connection between the network and the mobile station. For the air interface encryption a session key, i.e., the *derived cipher*

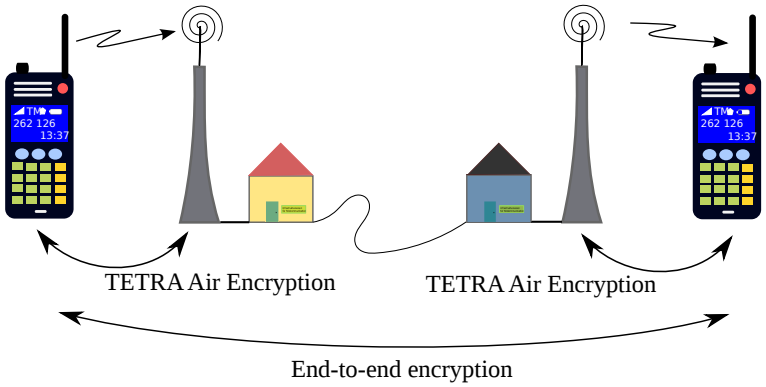

Figure 3.3: Structural view of a TETRA network pointing out the difference between end-to-end encryption and air interface encryption.

key generated from the key given on the SIM card or in the mobile station as well as negotiations between mobile station and network is used. This key encrypts subsequent communications to exchange other sensitive data like further keys listed below:

- **Common Cipher Key**: A common key for communicating with all other mobile stations in one local area.
- **Group Cipher Key**: A joint key for a fixed closed user group of more than two members.
- **Modified Group Cipher Key**: A joint key for a certain user group where users can flexibly join or leave.
- **Static Cipher Key**: A key that does not need authentication by the network, therefore it is the only key that can also be used in direct mode operation. Although staticly saved on the SIM card, it may be changed.

For the distribution of new keys, the Over The Air Re-keying (OTAR) mechanism is used as a secure means, as can be seen in Figure 3.4.

An end-to-end encryption can be implemented in order to encrypt the complete data path. While eavesdropping within the network and on the transfer links between network components might be possible when only air interface encryption is used, there is no possibility of eavesdropping an end-to-end encryption. The selection of the algorithm of the end-to-end encryption is only limited by the capabilities of the mobile devices. Besides, the algorithm has to fit into the design of the TETRA standard's cryptographic functions. Recommendations have been released that explain how to implement public cryptography standards like AES in TETRA [TET]. The TETRA standard comprises some standards for air interface encryption called TETRA Encryption Algorithm (TEA), but alternative standards can be implemented in the same manner as with end-to-end encryption. By default standards implemented are not publicly documented. They can be

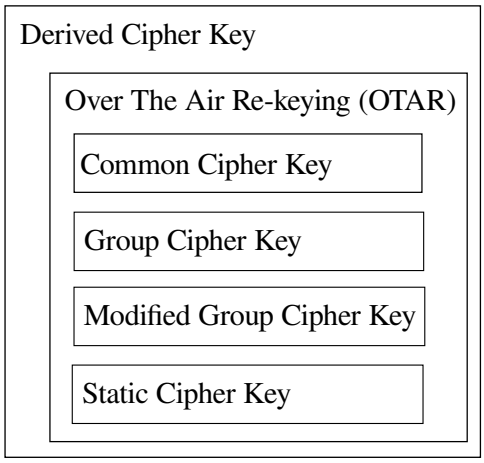

Figure 3.4: Structure of OTAR key exchange in TETRA. Key exchange using OTAR is secured by the derived cipher key.

ordered from ETSI for 1000 EUR each and only after signing a non-disclosure agreement. Standards developed for use by safety organisations within the European Union cannot be ordered officially at all. Furthermore, devices capable of these standards may not be exported outside the European Union [ETSb].

Bidirectional authentication between the mobile station and the network is included in the TETRA standard but can be overridden by user-specific protocols as well. Like in mobile phone networks, the network infrastructure checks whether the mobile station is allowed to log in to the network using a whitelist. The authentication key is stored in the network as well as on the mobile station and, if used, SIM card. The built-in algorithm for authentication is called TAA1 and is a proprietary algorithm provided by ETSI. Authentication does not work within direct mode operation, because of the lack of a central certification authority. This means that only the static cipher keys mentioned before can be used in this mode [Doua]. Table 3.2 gives an overview over all encryption and authentication algorithms implemented in TETRA.

According to a paper by Parkinson, the main key stored on the SIM card and mobile station has a length of 128 Bit. Besides, the TEA encryptions use 80 Bit stream ciphers. [D. 01]

3.3 Modulation

TETRA uses a $\frac{\lambda}{4}$ Differential Quadrature Phase Shift Keying (DQPSK) modulation. It is a variety of the Phase Shift Keying (PSK) modulation family, which shifts a sine wave by a certain phase value in order to transmit information. With this method, DQPSK modulations can transmit four different states (00, 01, 10, 11) in one shift process, i.e., in one symbol. Reading the (current) real and imaginary part of a DQPSK signal, the real part can be compared to the x-axis and the imaginary part to the y-axis of Figure 3.5 to recover the data. [Joh12]

DQPSK modulation means that values are represented twice in the complex plane

Table 3.2: Overview of encryption and authentication algorithms implemented in the TETRA standard.

Algorithm	Purpose	Application	Restriction
TEA1	Encryption	Commercial Usage	European Union only
TEA2	Encryption	Public Safety Organisations	European Union only
TEA3	Encryption	Public Safety Organisations	Outside European Union
TEA4	Encryption	Commercial Usage	Outside European Union
TAA1	Authentication	all	none

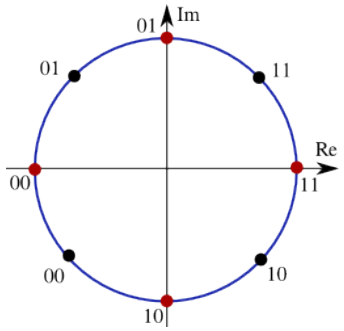

Figure 3.5: The four states of a QPSK signal equally distributed on a circle of the complex plane. The states with black dots are the redundant states of DQPSK.

scheme to reduce phase shifts from a maximum of 180° to 135° which allows decoding a signal without knowing the phase of the previous signal. The value of $\frac{\lambda}{4}$ indicates that phase shifts are only possible in discrete steps of $\frac{\lambda k}{4}, \quad k \leq 0 < 8$. This modulation is also deployed, e.g., in the Bluetooth Enhanced Data Rate (EDR) standard [Dav04].

3.4 Protocol Structure

TETRA makes use of several logical channels to transfer data as well as network control messages [Act01] [ETS13]. We only consider the control channels of the lower MAC layer, as only these will be dealed with in this thesis. In a brief overview, these are:

- **Access Assignment Channel (AACH):** Transmitted by the base station, informs about the occupancy of the current cell channel and when the channel can be accessed by other mobile stations.
- **Broadcast Network Channel (BNCH):** Transmitted by the base station, containing system information about the network and cell.

- **Broadcast Synchronisation Channel (BSCH):** Is a training sequence to synchronize the receiver of the mobile station to the signal of the base station.
- **Common Linearisation Channel (CLCH):** Used for calibrating the power amplifier of a mobile station after changing the frequency.
- **Signalling Channel (SCH)**: Used for, e.g., (re)registering the mobile station, e.g., after moving to another location area.
- **Stealing Channel (STCH):** Used to interrupt traffic broadcasts to transmit protocol information like cell handovers.
- **Traffic Channel (TCH):** This channel contains the actual user data.
- **Traffic Channel / Speech (TCH/S):** This channel contains user speech data.

3.5 Detected Networks

Within Germany, public security organisation use a frequency range between 380 and 385 MHz for the uplink to the network and 390 to 395 MHz for the downlink to the mobile station. Commercial networks are observed in the range 410 to 420 MHz uplink and 420 to 430 MHz downlink. In February 2013, the city zone of Bochum is covered by three distinct TETRA networks:

Freq. Range [MHz]	MCC and MNC	Network User	Encrypted
410 - 430	262 108	STEAG encotec GmbH	–
410 - 430	262 126	Dortmunder Stadtwerke AG	–
380 - 395	262 1001	BDBOS	✓

In total, 122 TETRA networks are currently registered with the German Federal Network Agency *Bundesnetzagentur* [Bun13]. The network by *Dortmunder Stadtwerke AG* is used for the bus service in Dortmund and neighbouring cities. The bus operators use the networks within the framework of their ITCS network to communicate between the buses and the headquarter. One service derived from this system is using the location data of buses to inform passengers on the internet or at stations about delays. Another profit is the realisation of phased traffic lights, where the system can switch a traffic light to green when a bus in approaching. The aims of the TETRA network by *Dortmunder Stadtwerke AG* are roughly described in a presentation given in 2009 [F.J09].

We use the applications provided by the Osmocom TETRA project to decode TETRA signals with either the RTL2832U USB stick or the USRP2 device. The application supports both, the UHD for connecting to the USRP and the RTL-SDR driver for using the RTL2832U USB stick. All software is running on a GNU/Linux system.

The Osmocom TETRA software consists of three applications. A receiver control software *tetra-demod.py* written in the software development toolkit GNU Radio, a conversion application that turns float values to bit values and finally a TETRA decoder

which also forwards the data to the Wireshark network analyser, where the data can be processed. Wireshark supports TETRA data since version 1.6.0 [Wir].

The split of Osmocom TETRA into three separated applications makes it possible to decode data after recording. As a benefit, signal recording and decoding do not have to take place at the same time but decoding samples of saved TETRA signals can be performed independently at a later time. This is recommended on slower hardware, as the receiver control software causes relatively high CPU load even on a 32-bit 2x 2 GHz processor, and some of the received data might get lost otherwise. Everytime the letter *O* appears in the console window, this means that *tetra-demod.py* has lost some data due to too much load.

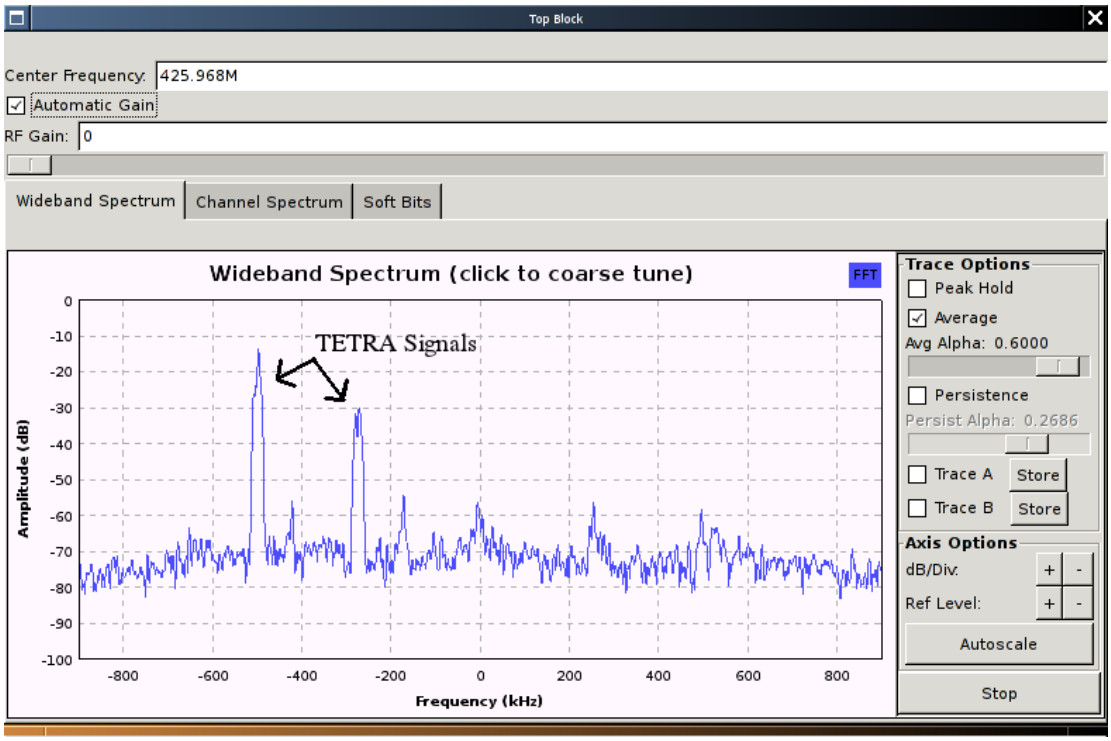

Figure 3.6: Receiver control software of osmo-tetra showing a spectrum with two TETRA signals with a high signal level.

A screenshot of the receiver control software is shown in Figure 3.6. The software allows settings like receiver gain and centre frequency and it can save peak signal levels on the shown spectrum. This helps the operator to, e.g., notice rarely broadcasted uplink signals.

To receive a signal at a certain frequency, the user has to click (with the left mouse button) on the desired frequency in the shown spectrum. For live decoding the Osmocom software has to be started with the following command in the `osmo-tetra/src` directory:

```
$ ./demod/python/osmosdr-tetra_demod_fft.py -o /dev/stdout |
./float_to_bits /dev/stdin /dev/stdout | ./tetra-rx /dev/stdin
```

The TETRA decoder displays details of successfully received data to the standard output. An example is:

```
BURST
CRC COMP: 0x1d0f OK
SB1 32/17/3/000 type1: 000001010011100011000000000000010000011000000000111111010001
TMB-SAP SYNC CC 010100(0x14) TN 11(3) FN 10001(17) MN 100000(32) MCC 0100000110(262)
MNC 00000001111110(126)
TMV-UNITDATA.ind 32/17/3/000 BSCH CRC=1 SYNC
TMV-UNITDATA.ind 32/17/3/000 AACH CRC=1 ACCESS-ASSIGN
ACCESS-ASSIGN PDU: DL_USAGE: Unallocated UL_USAGE: Unallocated
CRC COMP: 0x1d0f OK
SB2 32/17/3/000 type1: 10000011111110110100110000011010011010101100000110001001011111
01111010000110000000000000000010100111111111111111111101011101 01
TMV-UNITDATA.ind 32/17/3/000 UNKNOWN CRC=1 BROADCAST
BNCH SYSINFO (DL 425487500 Hz, UL 415487500 Hz), service_details 0x0d75 Hyperframe 6295
        Advanced link: 1
        Air encryption: 0
        SNDCP data: 1
        unknown 0x8: 0
        Circuit data: 1
        Voice service: 1
        Normal mode: 1
        Migration supported: 0
        Cell never uses minimum mode: 1
        Priority cell: 0
        De-registration mandatory: 1
        Registration mandatory: 1
```

This is a correctly decoded TETRA burst containing BSCH, AACH and BNCH channel data. Every channel has a CRC-16 checksum [Osma]. If the checksum is valid, the information is printed on the screen. On top, one can see the country and network number in the MCC and MNC fields. The BNCH system information displays information on the current frequencies for uplink and downlink and further properties of the network like if encryption is used and if the network supports voice services. A payload is not being broadcasted here.

```
BURST
TMV-UNITDATA.ind 52/16/2/000 AACH CRC=1 ACCESS-ASSIGN
ACCESS-ASSIGN PDU: ACCESS1: A/9 ACCESS2: A/9
CRC COMP: 0x1d0f OK
SCH/F 52/16/2/000 type1: 00100000101110011001100001101011100010000000010010011
11010001001001111000111100001100001011000010000000110000000000000101111011110 01
10111110010001000000000000000011000000001011101000000100000000000100001000000 00
00000000000000000000000000000000000000000000000000000000000000000000000000000
TMV-UNITDATA.ind 52/16/2/000 SCH/F CRC=1 RESOURCE
RESOURCE Encr=0, Length=23 Addr=SSI(9989000) TM-SDU(BL-UDATA,0,0): TL-SDU(CMCE):
01001111010001001001111000111100001100001011000010000000110000000000000101111 01
11100110111110010001000000000000000011000000001011101000000100000000000100001 00
00000000000000000000000 D-SDS DATA
```

The above burst contains payload data. First, a sequence on the AACH channel is broadcasted, followed by data on the SCH channel. Hence, we assume that a mobile

station has (re)registered to the network. Finally, SDS data is being broadcasted, addressed to the mobile device with the SSI number `9989000`.

```
BURST
#### SYNC burst at offset 206?!?
found SYNC training sequence in bit #1724
```

In case a burst is received but the signal is too weak or disturbed to be decoded, a message similar to the one shown above is printed.

When using Wireshark to analyse the TETRA data, the internal loopback device `lo` has to be selected as the capture device. Then, as long as the `tetra-rx` application (part of the Osmocom TETRA software) is decoding data, this data is instantly forwarded to Wireshark.

No.	Time	Source	Destination	Protocol	Length	Info
132	103.646382000	127.0.0.1	127.0.0.1	GSMTAP	92	D-STATUS
133	105.223261000	127.0.0.1	127.0.0.1	GSMTAP	92	D-RELEASE, D-RELEASE
134	107.506400000	127.0.0.1	127.0.0.1	GSMTAP	92	D-SDS-DATA
135	109.036288000	127.0.0.1	127.0.0.1	GSMTAP	92	
136	109.517794000	127.0.0.1	127.0.0.1	GSMTAP	92	D-SDS-DATA
137	109.520826000	127.0.0.1	127.0.0.1	GSMTAP	92	D-SDS-DATA
138	109.580921000	127.0.0.1	127.0.0.1	GSMTAP	92	D-STATUS
139	113.758437000	127.0.0.1	127.0.0.1	GSMTAP	92	

Figure 3.7: Various TETRA packets as displayed in Wireshark.

In Figure 3.7, a part of Wireshark's packet list overview is depicted. One can see that Wireshark detects some of the messages and thus shows their function in the *Info* column. Nevertheless, Wireshark can not recognize the function of all TETRA packets as of version 1.8.2. Figure 3.8 shows details of a SDS packet in Wireshark as well as the raw values of the packet. Information, e.g., the SSI of the device for which the packet in dedicated for and the encryption mode can be retrieved here. When highlighting a piece of data in the above window, Wireshark shows the raw value of the information in the lower part of the window. Using this method, we see that in the shown example packet only the last 20 bytes are used for payload.

It seems that the payload in the observed networks is using non-public proprietary structures, of which data can not be precisely deciphered. Besides, trying to decode data sent by buses instead of receiving just the base station downlink fails with the given equipment, mainly because of two reasons. First of all, the data bursts broadcasted by buses are very short and appear only approximately once a minute compared to a base station transmitting all the time. Second, the receiver tends to overdrive with these sudden strong signals, even after completely deactivating the receiver gain stage. Receiving a distinct signal of a bus could have made the protocol traceable better, because it probably would contain the current geographical coordinates of the bus or its current bus route number.

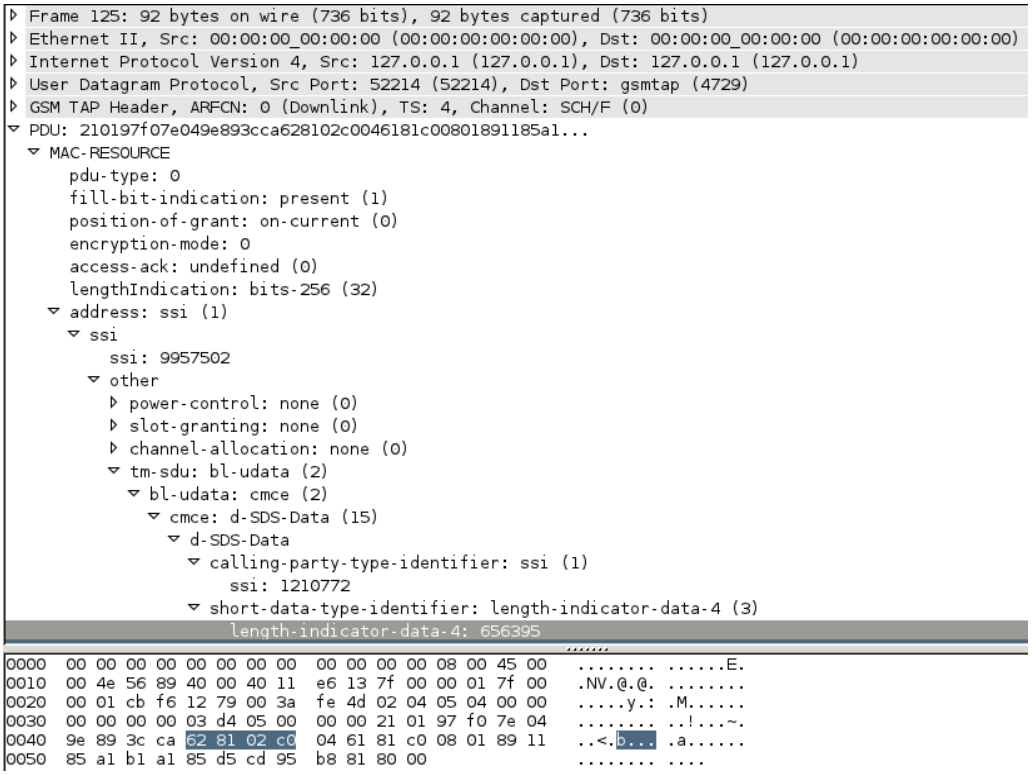

Figure 3.8: Detailed content of a SDS data packet in Wireshark.

3.6 TETRA Network Mapping

In January and February 2013, several spots in the city zone of Bochum were visited to analyse the TETRA frequency ranges. Exact frequencies, signal strength, and network numbers were recorded at different spots of the city and compared later. The tests are performed using a notebook computer running the osmo-tetra software and the RTL2832U USB stick. The aim is to roughly determine the location of each received base station by comparing the signal strength at each spot and to estimate the number of base stations necessary to cover the whole city. All logged networks cover neighbouring cities as well, which means that some base stations might be installed beyond the city borders but still officially or inofficially cover some margins of Bochum. Eight distinct spots were visited during this measurement campaign.

Table A.1 shows the downlink frequencies of successfully decoded base stations during the test including the corresponding MNC at tested spots. Moreover, the signal level in dBm is given for all tested locations. The abbreviations of the locations are described in Table 3.3, including latitude, longitude, and height above sea level of all positions. The geographic directions with free sight or sloping terrain are mentioned as well in order to

Table 3.3: Overview of all receiving locations in Bochum.

Abb.	Location	Latitude	Longitude	Height	Sight
KE	Kemnader Str./Surkenstr.	51.4389 N	7.2846 E	185 m	N,(S)
WH	Weitmarer Holz, playground	51.4280 N	7.2439 E	175 m	N,(W)
ST	Sternwarte, playground	51.4284 N	7.1914 E	155 m	S,(W)
OE	Oelbachteich/Hevener Str.	51.4389 N	7.2847 E	78 m	(W),E
SP	Stadtpark near Bismarckturm	51.4901 N	7.2258 E	120 m	S,W
WS	Westpark, south-west corner	51.4799 N	7.1953 E	83 m	S,W
WN	Westpark, north-east corner	51.4848 N	7.1994 E	84 m	N,(E)
BU	Querenburg, Buscheyplatz	51.4498 N	7.2624 E	145 m	(S),(N)

point out directions that are less shielded than others at each test location. Geographic directions put in brackets, e.g. (S) for south, mean that free sight to this direction is almost possible, only obstructed by a few houses or hills.

The results are shown in Figure 3.9. In rather many cases, an exact position of a base station is difficult to determine, because only a omnidirectional antenna was used during the tests. A directional antenna would have been helpful for investigation in some cases but would maybe have led to alertness by passing-by people. Therefore, for estimating the locations of the base stations, the assumption is made that base stations are likely to be in an approximately equal distance to other base stations of the same network. Because the test locations are at different heights and the conditions for receiving signals differ depending on how exposed the test spots are, the positions of the base stations cannot be found by simply doing a triangulation between the test spots. Therefore, the positions of the base stations are recorded after taking the attributes of the test locations into account, e.g., test spot WH is situated higher than most other spots and allows free sight to the north, so signals from a larger distance are more likely to be receivable here than elsewhere.

All observed networks turn out to have a closely meshed infrastructure with base stations in a distance of only two to four kilometres to each other, which is roughly comparable to the density of GSM and UMTS networks. The two base stations of the *BDBOS* network (green) at the bottom centre part of the map are likely to be located somewhat south of the map, because they are only received at one spot with relatively good sight to the south, but this cannot be determined exactly with the existing measurements. In the map, the network by *STEAG encotec GmbH* is marked with red dots and the network by *Dortmunder Stadtwerke AG* is depicted with blue dots.

The knowledge of the positions of base stations can be used to attack a network. One scenario is transmitting a disturb signal at the uplink frequency of base stations from spots close to the base station. This strong disturb signal will prevent the base station from receiving signals of mobile stations. Consequently, the network can not operate properly anymore.

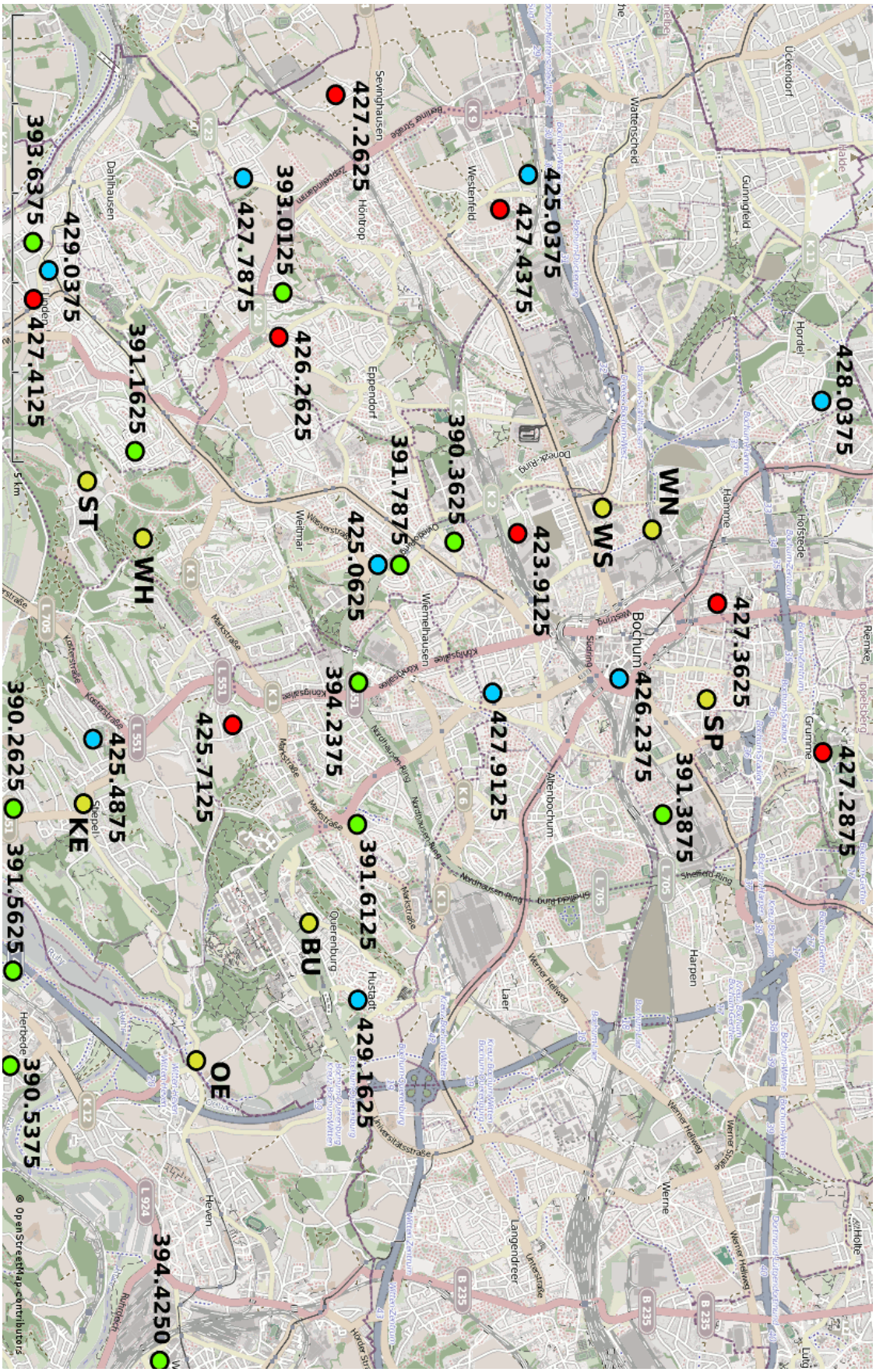

Figure 3.9: Estimated locations of TETRA base stations in Bochum. Yellow dots: Receiving spots, green dots: BDBOS stations, red dots: STEAG stations, blue dots: Dortmunder Stadtwerke stations. Map data by *openstreetmap.org*licensed under the Open Data Commons Open Database Licence.

4 OpenBTS

One application compatible to the USRP device is OpenBTS [Ope]. This software provides all components necessary to run a GSM base station and also the infrastructure needed for a small GSM network, e.g., databases for authentication of mobile phones and distributing text messages as well as phone call routing features. For phone calls to or from the outside world OpenBTS makes use of the open source phone routing software Asterisk [Ope]. OpenBTS is open source software, the development started in 2008 mainly by David A. Burgess [GNUb].

OpenBTS provides a freely configurable base station, thus, properly configured, it can also be used as an IMSI catcher, i.e., hardware that behaves like a phone to a base station and emulates a base station to a phone. As a result, the communication between phone and base station can be routed via the IMSI catcher without the phone user and the network taking notice of this incident. This man-in-the-middle-attack is only possible, because in the GSM standard the network does not need to authenticate itself to the phone. This was changed in newer telecommunication standards, e.g., UMTS and LTE.

The price of a USRP2 including a compatible daughterboard hardware seems to be much lower than that of a dedicated IMSI catcher system. Although no trustable documentation of costs for IMSI catchers is available, a presentation at the University of Freiburg claims that the cost of a IMSI catcher system by Rohde&Schwarz is about 100,000 USD [Den11]. The USRP can be bought for less than 2000 USD including a suitable daughterboard.

A unique network identification number is assigned to all official GSM network operators. It consists of the country identification number (MCC) and a network number (MNC). E.g., Germany is assigned the MCC 262 and the official networks' identifications are 01 for the Telekom network, 02 for Vodafone.de, 03 for E-Plus, and 07 for the network of O2 Germany.

4.1 Setup of OpenBTS

Using the OpenBTS software, all relevant configuration can be set by editing the attached `OpenBTS.config` file. In this file, one can set the MNC and MCC to user-defined values. This means that these values can also be set to those of existing operators. Phones will now log into the station without noticing that it is not an official station, hence the user can not distinguish this station from official ones. On a properly configured system that routes phone calls and text messages, it is possible to capture all outgoing communication in plaintext without the need to attack cryptography. On the other hand, it is not possible to capture incoming traffic as official routers of a phone network are not aware of the fact that the phone can only be reached via the fake base station, therefore,

the phone can not be called from the outside anymore. This applies to ordinary IMSI catchers as well.

OpenBTS can be configured to allow login requests from all phones without restrictions. In an area of bad coverage by the original network operator, phones tend to search for other networks and to log in if the signal level is higher. Thus, OpenBTS can be used as a honeypot if placed at the right locations and using higher output power. Every phone that tries to register with the base station submits its IMSI number. This is the unique number of the SIM card plugged into the phone with which the network recognizes the user. A phone number is not saved on the SIM but does only exist in the network, where a database translates phone numbers to IMSI numbers. The screenshot in Figure 4.1 shows a first phone trying to connect to the base station after the application has been started. Among other information, OpenBTS displays that the phone was earlier logged into a network with MCC 262, MNC 03 (i.e., E-Plus), and Location Area Code (LAC) 0x1be. *TimingAdvance=1* indicates that the phone is situated in a distance of less than 554 m to the BTS [GSM].

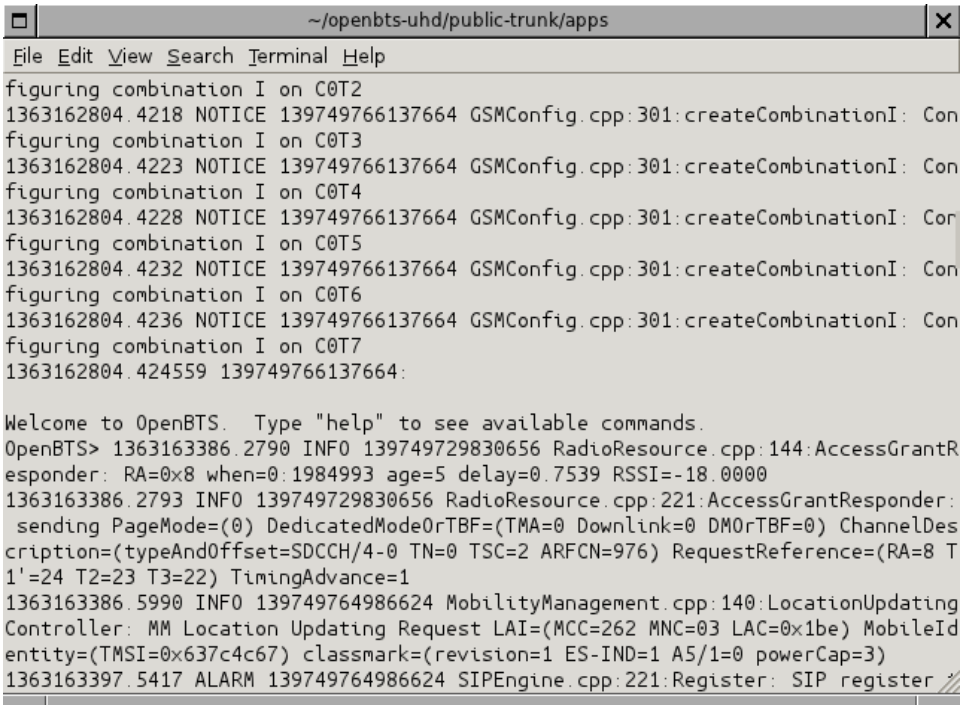

Figure 4.1: Screenshot of OpenBTS shortly after starting the application showing a first phone trying to connect to the base station.

During a test of the software with voluntary participants in a shielded room of the Ruhr-Universität Bochum, it was possible to make 35 phones log into the OpenBTS test system automatically, i.e., without user interaction, supplying the unique IMSI numbers

Table 4.1: Distribution of SIM card issuers for the participants of our test run.

Operator	Registered Phones
Telekom D	5
Vodafone.de	7
E-Plus	14
O2 Germany	9

of their SIM cards. This number, e.g., reveals the network operator that issued the SIM card, because it begins with the MCC and MNC of the issuer. The collected data shows the distribution between the operators given in Table 4.1 and Figure 4.2.

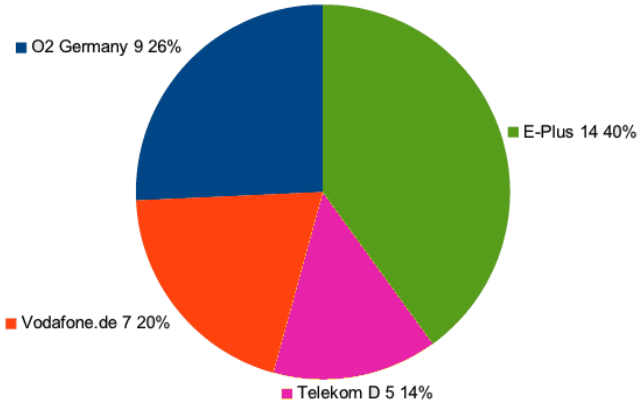

Figure 4.2: Distribution of SIM card issuers for the participants of our test run.

The test is performed using a simple vertically aligned telescope antenna directly connected to the FLEX900 daughterboard of the USRP. When setting the attenuation of the USRP output power in the OpenBTS configuration file to a range of 43 to 45 dB, it is possible to reduce the coverage range of the BTS to 10 metres. As the FLEX900 daughterboard of the USRP transmits with a maximum power of 23 dBm, i.e. 200 mW, the resulting output power is 10 μW. The telescope antenna has a gain of approx. 0 dBd, so the output power equals roughly the Effective Radiated Power (ERP). The experiment setup can be seen in Figure 4.3.

In general, the USRP2 uses an internal clock to achieve a frequency stability of around 20 ppm (which compares to the clock stability of common mobile phones), while the requirements to the accuracy of a BTS is to be better than 0.05 ppm. Calculating $0.05 \text{ ppm} \cdot 900 \text{ MHz} = 0.05 \cdot 10^{-6} \cdot 900 \cdot 10^{6}$ MHz, we get a tolerated variance of only 45 Hz for a base station working on 900 MHz, while it is around 18 kHz for the USRP2. Mobile phones calibrate their internal clock with those of a BTS. Now, if a mobile phone receives

Figure 4.3: Experiment setup of the USRP2 with a telescope antenna for running OpenBTS.

offical networks as well as the OpenBTS, it is likely not to find our network, because of a too large frequency offset. [GNUa] However, our tests have shown that older phones do only log in to our network when the signal level of official networks are very low. This applies in particular to a tested Nokia 5140 (released 2004) and a SonyEricsson K850i (released 2008). Modern smartphones are joining the network rather fast and early, in particular noticed with htc and Apple phones.

To supply the USRP2 with a more precise clock signal, a Philips PM 5715 pulse generator for a frequency range from 1 Hz to 50 MHz was connected to the USRP's

external reference clock connector. The pulse generator is adjusted to generate a 10 MHz square wave, as is confirmed by doing measurements with the Picoscope. However, an OpenBTS version compiled with support for an external reference clock does not recognize the reference signal. Besides, the LED *E* on the front cover of the USRP2 does not show up like it is explained in the manual [Ettb]. This LED indicates the lock of the USRP to the external source. So no tests could be performed with a possibly more accurate GSM signal. The analysis of the output signal of the pulse generator with the Picoscope shows a signal with amplitude variation by approximately 30 percent. The square waves are a bit rounded as well and although a signal amplitude of 5 Volts peak-to-peak is selected, the signal only has an amplitude of approximately 3 Volts, while the USRP2 requires an input value of 5 Volts peak-to-peak [Ettb]. This might cause the USRP2 not to detect the external reference source.

4.2 Recent Mobile Telephony Standards

GSM, while still being the most commonly used standard for mobile communication, must be considered out-of-date both in terms of technology and security. Hence, also OpenBTS is an implementation of an outdated standard, while two new standards, i.e., UMTS and LTE, for similar purposes have already been released and networks for these standards have been built over the last years in most countries of the world. Nevertheless, GSM still plays a leading role in mobile telephony. One of the main developers of the Osmocom Project and main developer of the related OpenBSC software, Harald Welte, claims that the implementation of those newer standards is tougher. He expects that there will rather be an open source implementation of the newer LTE standard than one for UMTS in the future [Tim11].

5 Conclusion

This thesis shows that SDR devices help to make signal analysis easier and less cost-intensive. This comes at the price of a higher risk of eavesdropping and signal manipulation, because the effort to capture digital wireless transmissions and to transmit fake signals that comply with protocol standards heavily reduces with SDR. As a consequence, the risk of signal manipulation increases for companies and authorities, which rely on wireless communication. Attacks can be performed, e.g., by spies who cause leakage of secret information. With it, financial damage may occur to companies. Besides, to authorities like police, leaking information and manipulated signals even pose a threat to public safety to the point of terror attacks.

Concerning remote controls and remote sensors, this thesis points out that many signals do not use security mechanisms. In quite some cases, these signals are not security-related at first sight, e.g., when regarding thermometer sensors. But there are cases where tampering with signals of remote thermometers can become a threat, e.g., when the thermometers are used to control other devices like a cooling system. It is reported that protocols of thermometers turn out not to use security mechanisms [Mem] [Hel]. In the scenario of manipulating temperature values, a manipulated signal could destroy machines in, e.g., industrial facilities.

Besides, the signals of remote controls do not necessarily comprise security mechanisms. Prefabricated remote controls without secure protocols can be ordered in large amounts [Mad], which increases the risk that these devices are built in, e.g., alarm systems. In this thesis, we have shown that even a recent garage door opener uses an unaltered signal without security mechanisms like a rolling key. This makes it vulnerable to simple replay attacks. Burglars may record the original signal once and can thereupon transmit this signal while, e.g., the residents of a property are not at home. In contrast to traditional burglary, this method does not even leave behind traces of violent intrusion to a property.

Concerning TETRA, this thesis shows that many operators underestimate the risks of unencrypted networks. One example is the network operated by STEAG encotec GmbH that is used to control an electricity network [EAD06]. It does not become clear by the information available, whether this network only transports status messages to a headquarter or whether the network is also used to remotely regulate distributed components of the electricity network, e.g., transformer stations. In the latter case, attackers may even cause power outages by transmitting manipulated signals containing commands to, e.g., shut down a transformer. Furthermore, faked status messages can be sent via the TETRA network to the headquarter in order to mislead operators and

to cause the operators to perform wrong countermeasures which can destabilise the electricity network.

Networks operated by a bus company can be manipulated in a similar way, although a damage to infrastructure or operation might seem less likely at first glance. But when taking into account that there are traffic lights which can be switched to green for one direction via the TETRA network of a bus operator [F.J09], incidents like traffic jams could be initiated by tampering with the TETRA network. Furthermore, one may wonder, whether the non-public encryption algorithms supplied with the TETRA standard and used by public security organisations throughout Europe can compete with the security level of modern public algorithms like AES, especially because not publishing the algorithm violates Kerckhoff's principle, which says that the security of a cryptosystem should rely on the secret key only. This implies that the algorithms of a cryptosystem should be secure although the workling principle of all algorithms and other details of the system are available to the public [PP10]. Besides, 80 bit stream ciphers can be seen as the lower edge of modern symmetric encryptions, when compared to, e.g., the AES standard which is available in a 128 bit, 192 bit, and 256 bit version [Nat].

5.1 Future Work

However, an open source software which does not only allow reception but also transmission of TETRA signals using SDR devices is not yet available, although there are plans by the Osmocom project to build a TETRA modulator application [Tim11]. Besides, there is no known attack against the default cryptography algorithms of the TETRA standard yet, but the availability of SDR technology helps to make attacks less expensive and thus more likely.

Besides, up to now, software which allows to set up a base station for UMTS or LTE using SDRs does not exist. Harald Welte, a founding member of the Osmocom project claims that the implementation of the UMTS and LTE standards is tougher compared to GSM [Tim11]. Due to the protocol standards of UMTS and LTE it is not possible to use such software as an IMSI catcher, because in contrast to GSM these networks require the base station to authenticate to the mobile station which would fail under usual circumstances when using a fake base station. However an attack vector could be setting up a false roaming network to which phones automatically connect if its signal level exceeds those of official networks.

A TETRA Network Mapping Table

Table A.1: Signal levels of received TETRA network base stations at distinct locations in Bochum. The abbreviations of the locations are explained in Table 3.3.

Downlink [MHz]	MNC	KE	WH	ST	OE	SP	WS	WN	BU
423.9125	108	-40	-18	–	–	-23	–	-17	–
425.7125	108	-20	–	–	–	–	-45	–	-55
426.2625	108	–	-30	–	–	–	–	–	–
427.2625	108	–	–	-50	–	–	–	-38	–
427.2875	108	-37	-25	–	–	–	–	-32	–
427.3625	108	–	–	–	–	-38	–	-30	–
427.4125	108	–	–	-40	–	–	–	–	–
427.4375	108	–	–	–	–	–	-37	–	–
425.0375	126	–	–	–	–	–	-38	–	–
425.0625	126	–	-22	–	–	–	–	-46	–
425.4875	126	-20	-20	-55	–	–	–	–	–
426.2375	126	-40	–	–	–	-20	-26	-27	–
427.7875	126	–	-30	-46	–	–	–	–	–
427.9125	126	–	-30	–	–	-33	-50	–	–
428.0375	126	–	–	–	–	–	–	-34	–
429.0375	126	–	–	-40	–	–	–	–	–
429.1625	126	-60	-30	–	–	–	–	–	-22
390.2625	1001	-50	–	–	–	–	–	–	–
390.3625	1001	–	-35	–	–	-42	-25	-30	–
390.5375	1001	-40	–	–	-60	–	–	–	–
391.1625	1001	–	-22	-30	–	–	–	–	–
391.3875	1001	-45	–	–	–	-17	–	–	–
391.5625	1001	–	-26	-50	-45	–	–	–	–
391.6125	1001	-30	–	–	–	–	–	-45	-30
391.7875	1001	–	-28	–	–	–	-46	–	–
391.9125	1001	-40	–	–	–	–	–	–	–
393.0125	1001	–	-20	–	–	–	–	–	–
393.6375	1001	–	–	-48	–	–	–	–	–
394.2375	1001	-45	-18	–	–	-32	–	-39	–
394.4250	1001	–	–	–	-47	–	–	–	–

B Listings of spectracom

In this chapter, some routines of the spectracom Software presented in Chapter 2.5 are printed, which contain code responsible for communicating with SDR devices and for signal processing.

```
// reads the original samples
char* sampleread(string filet, string type)
{
    char* dateiname = new char[filet.length()];
    strcpy(dateiname,filet.c_str());

    FILE* datei = fopen(dateiname,"rb");
    cout << "Read sample: " << dateiname << "\t" << type << endl;
    // get file size
    fseek (datei , 0 , SEEK_END);
    filesize = ftell (datei);
    rewind (datei);

    // test whether filesize is power of 2
    // source: http://www.gamedev.net/topic/406313-c-fast-power-of-2-test/
    if (!((filesize > 0) && ((filesize & (filesize - 1)) == 0))) {
        filesize = filesize/2;
      // find next larger power of 2
      // source: http://stackoverflow.com/questions/364985/algorithm-for
      // -finding-the-smallest-power-of-two-thats-greater-or-equal-to-a-giv
        if (filesize < 0) return 0;
        --filesize;
        filesize |= filesize >> 1;
        filesize |= filesize >> 2;
        filesize |= filesize >> 4;
        filesize |= filesize >> 8;
        filesize |= filesize >> 16;
        filesize++;
    }
    if (type == "float") samplelength = 8;
    if (type == "short") samplelength = 2;

    num_rx_samps = filesize/samplelength;

    // distinguish whether data is short (from USB stick) or float (from USRP)
    if (samplelength == 8) {
        size_t result = fread(&origdata,samplelength,num_rx_samps,datei);
        for (qa=0;qa<=num_rx_samps;qa++) origdata[qa] = origdata[qa]; }
    else {
        size_t result = fread(&orgdata,samplelength,num_rx_samps,datei);

        for(qa;qa<=num_rx_samps;qa++) {
            origdata[qa] = (float)orgdata[qa];
        }
    }
}
```

Listing B.1: The sampleread function which reads samples recorded with the SDR devices.

```
// receive samples with the USRP
template<typename samp_type> void receivestream() {
    // convert user variables
    int nsam = *((int*)&nums);
    if (nsam == 0) nsam = samples;
    qa = 0;
```

```
7
8     // code by Ettus Research for the communication with the USRP:
9     uhd::set_thread_priority_safe();
10    std::string args, file, type, ant, subdev, ref, wirefmt;
11
12    //create a usrp device
13    uhd::usrp::multi_usrp::sptr usrp = uhd::usrp::multi_usrp::make(args);
14
15    //Lock mboard clocks, print information only in manual mode
16    if (autonomous==0) cout << boost::format("Creating the usrp device with: %s...") % args <<
          endl;
17    usrp->set_clock_source("internal");
18    if (autonomous==0) cout << boost::format("Using Device: %s") % usrp->get_pp_string() << endl;
19
20    // set rate and frequency
21    usrp->set_rx_rate(rate);
22    rate = usrp->get_rx_rate();
23    usrp->set_rx_freq(freq);
24    freq = usrp->get_rx_freq();
25
26    if (autonomous==0) cout << boost::format("Actual RX Gain: %f dB...") % usrp->get_rx_gain() <<
          endl;
27
28    boost::this_thread::sleep(boost::posix_time::seconds(0.1)); //allow for some setup time
29
30    //Check Ref and LO Lock detect
31    std::vector<std::string> sensor_names;
32    sensor_names = usrp->get_rx_sensor_names(0);
33    if (std::find(sensor_names.begin(), sensor_names.end(), "lo_locked") != sensor_names.end()) {
34        uhd::sensor_value_t lo_locked = usrp->get_rx_sensor("lo_locked",0);
35        if (autonomous==0) cout << boost::format("Checking RX: %s ...") % lo_locked.to_pp_string()
              << endl;
36        UHD_ASSERT_THROW(lo_locked.to_bool());
37    }
38
39    sensor_names = usrp->get_mboard_sensor_names(0);
40    if ((ref == "mimo") and (std::find(sensor_names.begin(), sensor_names.end(), "mimo_locked") !=
          sensor_names.end())) {
41        uhd::sensor_value_t mimo_locked = usrp->get_mboard_sensor("mimo_locked",0);
42        if (autonomous==0) cout << boost::format("Checking RX: %s ...") % mimo_locked.to_pp_string
              () << endl;
43        UHD_ASSERT_THROW(mimo_locked.to_bool());
44    }
45    if ((ref == "external") and (std::find(sensor_names.begin(), sensor_names.end(), "ref_locked")
          != sensor_names.end())) {
46        uhd::sensor_value_t ref_locked = usrp->get_mboard_sensor("ref_locked",0);
47        if (autonomous==0) cout << boost::format("Checking RX: %s ...") % ref_locked.to_pp_string()
              << endl;
48        UHD_ASSERT_THROW(ref_locked.to_bool());
49    }
50
51    //create a receive streamer
52    uhd::stream_args_t stream_args("fc32","sc16");
53    uhd::rx_streamer::sptr rx_stream = usrp->get_rx_stream(stream_args);
54
55    uhd::rx_metadata_t md;
56    std::vector<std::complex<float> > buff(nsam);
57
58    const char* filename;
59    if (runnr == 1) filename = "dft_samp1";
60    else filename = "dft_samp2";
61
```

```
    std::ofstream outfile(filename, std::ofstream::binary);
    bool overflow_message = true;

    //setup streaming
        uhd::stream_cmd_t stream_cmd((nsam == 0)?
        uhd::stream_cmd_t::STREAM_MODE_START_CONTINUOUS:
        uhd::stream_cmd_t::STREAM_MODE_NUM_SAMPS_AND_DONE
    );

    //create a usrp device
    usrp = uhd::usrp::multi_usrp::make(args);

    stream_cmd.num_samps = nsam;
    stream_cmd.stream_now = true;
    stream_cmd.time_spec = uhd::time_spec_t();
    usrp->issue_stream_cmd(stream_cmd);

    size_t num_rx_samp;
    while(not stop_signal_called and (nsam != qa) or nsam == 0) {
        num_rx_samp = rx_stream->recv(&buff.front(), buff.size(), md, 3.0);
        qa += num_rx_samp;
    }
    // write spectrum into a file
    outfile.write((const char*)&buff.front(), num_rx_samp*sizeof(samp_type));
    outfile.close();
    usleep(1000);

    sampleread(filename,"float"); // load sample data into memory

    // call the fourier trafo respectively for the current sample
    if (runnr == 1) fourier("dft1"); // run DFT
        else fourier("dft2");
    return;
}
```

Listing B.2: The code for communicating with the USRP.

```
// this function compares two noise levels and does some additional statistics stuff
int compare(char* autofile) {
    // calculate the average noise level
    int z = 0;
    double noise1 = 0;
    double noise2 = 0;

    for (z;z<samples;z++) {
        noise1 += val1[z];
        noise2 += val2[z];
    }

    noise1 = noise1 / samples;
    noise2 = noise2 / samples;
    double sum = abs(noise2 - noise1);

    // for the autonomous part
    int autorun = 0;
    string tempstring;
    stringstream sstm1;

    // now find frequencies with a large level difference between
    // the probes and write the difference plot to file
    double diff = 0;
    freqnow = freqlow;
```

```
26      z = 1;
27      incidents = 0;
28
29      ofstream fout2(autofile, ios::app | ios::out);
30      ofstream fout ("dft_log", ios::out);
31
32      for (z;z<samples;z++) {
33          freqnow = freqnow + freqstep;
34          if (samplelength != 8) freqnow = freqnow + freqstep; // if RTL-SDR
35          diff = abs(val1[z] - val2[z]);
36          if (autonomous != 1) {
37              fout << freqnow << "\t" << val1[z] << "\t" << val2[z] << "\t" << 0 << "\n";
38          }
39
40      // if the difference is significant, tell it to the user directly
41      // to omit errors, filter values of less than -130 dBm, as they
42      // are below the real noise level of the device
43
44          if (diff > threshold && val1[z] > -130 && val2[z] > -130) {
45              if (autonomous == 0) {
46                  fout << freqnow << "\t" << val1[z] << "\t" << val2[z] << "\t" << diff << "\n";
47              }
48
49              if (autonomous == 1) {
50                  // fetch current date
51                  time_t sekunden = time(NULL);
52                  tm *uhr = localtime(&sekunden);
53                  tag = uhr->tm_mday;
54                  monat = uhr->tm_mon +1;
55                  jahr = uhr->tm_year + 1900;
56                  stunde = uhr->tm_hour;
57                  minute = uhr->tm_min;
58                  sekunde= uhr->tm_sec;
59                  fout2 << freqnow << "\t" << val1[z] << "\t" << val2[z] << "\t" << diff << "\t" \
60                    << jahr << "-" << monat << "-" << tag << "\t";
61                  if (jahr < 10) fout2 << 0;
62                  fout2 << stunde << ":";
63                  if (minute < 10) fout2 << 0;
64                  fout2 << minute << ":";
65                  if (sekunde < 10) fout2 << 0;
66                  fout2 << sekunde << "\n";
67
68                  // if something interesting was noticed: save a copy of the spectrum
69                  // but only once, because the spectrum already contains the other
70                  // interesting signals as well. Hence the autorun variable
71              if (autorun == 0) {
72                  sstm1 << jahr << monat << tag << "-";
73                  if (jahr < 10) sstm1 << 0;
74                  sstm1 << stunde;
75                  if (minute < 10) sstm1 << 0;
76                  sstm1 << minute;
77                  if (sekunde < 10) sstm1 << 0;
78                  sstm1 << sekunde << "-" << threshold << "dB.sam";
79                  tempstring = sstm1.str();
80
81                  char* copyfile = new char[tempstring.length()];
82                  strcpy(copyfile,tempstring.c_str());
83
84                  if (runnr % 2 == 0) {
85                      ifstream src("dft_samp1");
86                      usleep(1000);
87                      ofstream dst(copyfile);
```

```
88                  dst << src.rdbuf();
89                  usleep(1000);
90                  dst.close();
91              }
92              if (runnr % 2 == 1) {
93                  ifstream src("dft_samp2");
94                  usleep(1000);
95                  ofstream dst(copyfile);
96                  dst << src.rdbuf();
97                  usleep(1000);
98                  dst.close();
99              }
100         }
101
102         autorun++;
103     }
104   }
105
106     // remember frequency and intensity of signal
107     freqlist[incidents] = freqnow;
108     intlist[incidents] = diff;
109     treasurelist[incidents] = diff*freqnow;
110     incidents++;
111 }
112
113 // calculate average to find mean frequency
114 double added = 0;
115 double meanfreq = 0;
116 double h = 0;
117
118 for (z=0;z<incidents;z++) {
119     meanfreq += freqlist[z]*intlist[z];
120     h++;
121     added += intlist[z];
122 }
123
124 meanfreq = meanfreq/(added);
125
126 // surprise the user with some statistics (and convert them for the GUI...)
127 if (gui == 1) {
128     stringstream sstm1;
129     stringstream sstm2;
130     stringstream sstm3;
131     stringstream sstm6;
132     stringstream sstm7;
133
134     sstm1 << noise1;
135     sstm2 << noise2;
136     sstm3 << abs(sum);
137     sstm6 << freqstep;
138     sstm7 << meanfreq;
139
140     string statistic1 = sstm1.str();
141     string statistic2 = sstm2.str();
142     string statistic3 = sstm3.str();
143     string statistic6 = sstm6.str();
144     string statistic7 = sstm7.str();
145     const char* execcmd1 = statistic1.c_str();
146     const char* execcmd2 = statistic2.c_str();
147     const char* execcmd3 = statistic3.c_str();
148     const char* execcmd6 = statistic6.c_str();
149     const char* execcmd7 = statistic7.c_str();
```

```
150
151         Fl_Window* win = new Fl_Window(380,200, "Statistics");
152         win->begin();
153
154         Fl_Output*      out1 = new Fl_Output(218, 30, 120, 20, "Average level probe 1 ");
155         Fl_Output*      out2 = new Fl_Output(218, 60, 120, 20, "Average level probe 2 ");
156         Fl_Output*      out3 = new Fl_Output(218, 90, 120, 20, "Average level difference");
157         Fl_Output*      out6 = new Fl_Output(218, 120, 120, 20, "Frequency step      ");
158         Fl_Output*      out7 = new Fl_Output(218, 150, 120, 20, "Centre frequency    ");
159         win->end();
160         win->show();
161         out1->value(execcmd1);
162         out2->value(execcmd2);
163         out3->value(execcmd3);
164         out6->value(execcmd6);
165         // show mean frequency in the GUI
166         out7->value(execcmd7);
167     }
168     fout.close();
169     fout2.close();
170     return 0;
171 }
```

Listing B.3: This code compares two spectra recorded earlier.

```
1  int autonom() {
2      // fetch current date
3      time_t sekunden = time(NULL);
4      tm *uhr = localtime(&sekunden);
5
6      tag = uhr->tm_mday;
7      monat = uhr->tm_mon +1;
8      jahr = uhr->tm_year + 1900;
9      stunde = uhr->tm_hour;
10     minute = uhr->tm_min;
11     sekunde= uhr->tm_sec;
12
13     stringstream sstm1;
14     sstm1 << jahr << monat << tag << "-";
15     if (jahr < 10) sstm1 << 0;
16     sstm1 << stunde;
17     if (minute < 10) sstm1 << 0;
18     sstm1 << minute;
19     if (sekunde < 10) sstm1 << 0;
20     sstm1 << sekunde << "-" << threshold << "dB";
21     string tempstring = sstm1.str();
22
23
24     char* autofile = new char[tempstring.length()];
25     strcpy(autofile,tempstring.c_str());
26     while (runs <= maxnr | maxnr == 0) {
27         usleep(1000);
28         cout << "Run: " << runs << endl;
29         if (devsel == 1) { receivestream<std::complex<float> >(); runnr++; }
30         if (runs == 0 && devsel == 1) { usleep(1000); receivestream<std::complex<float> >(); runnr
               ++; }
31         runs++;
32         if (devsel == 2) { // if USB stick
33             samples = sampconst; // restore orignal number of samples selected by the user
34             receive_cb(0,0);
35             sampleread("dft_samp1","short"); // load sample data into memory
36             fourier("dft1"); // run DFT
```

```
37              receive_cb(0,0);
38              sampleread("dft_samp2","short");
39              fourier("dft2");
40          }
41          spectrumread(file1,1);
42          spectrumread(file2,2);
43          usleep(1000);
44          compare(autofile);
45          if (runnr == 2 && devsel == 1) runnr = 0;
46      }
47      return 0;
48  }
```

Listing B.4: This code initiates the autonomous signal recording.

C Acronyms

AACH Access Assignment Channel

ADC Analogue to Digital Converter

AES Advanced Encryption Standard

AM Amplitude Modulation

BNCH Broadcast Network Channel

BSC Base Station Controller

BSCH Broadcast Synchronisation Channel

BTS Base Transceiver Station

CEPT Conférence Européenne des Administrations des Postes et des Télécommunications

CLCH Common Linearisation Channel

CRC Cyclic Redundancy Check

DCR Direct-Conversion Receiver

DFT Discrete Fourier Transformation

DQPSK Differential Quadrature Phase Shift Keying

DVB-T Digital Video Broadcasting - Terrestrial

EDR Enhanced Data Rate

ETSI European Telecommunications Standards Institute

ERP Effective Radiated Power

FLTK Fast Light Toolkit

FM Frequency Modulation

FSK Frequency Shift Keying

GNU GNU's Not Unix

GSM Global Standard for Mobile Communication

GUI Graphical User Interface

IF Intermediate Frequency

IMSI International Mobile Subscriber Identity

ISM Industrial, Scientific and Medical

ITCS Intermodal Transport Control System

ITU International Telecommunications Union

JPEG Joint Picture Expert Group

LAC Location Area Code

LAN Local Area Network

LED Light Emitting Diode

LTE Long Term Evolution

MAC Medium Access Control

MCC Mobile Country Code

MNC Mobile Network Code

NFC Near-Field Communication

Osmocom Open Source Mobile Communication

OTAR Over-the-Air Re-keying

PSK Phase Shift Keying

QAM Quadrature Amplitude Modulation

RFID Radio Frequency Identification

SCH Signalling Channel

SD Secure Digital

SDR Software-Defined Radio

SDS Short Data Service

SIM Subscriber Identity Module

SMS Short Message Service

SNR Signal to Noise Ratio

SRD Short Range Devices

SSI Short Subscriber Identity

STCH Stealing Channel

TCH Traffic Channel

TCH/S Traffic Channel / Speech

TDMA Time Division Multiple Access

TEA TETRA Encryption Algorithm

UHD USRP Hardware Driver

UMTS Universal Mobile Telecommunication Standard

USB Universal Serial Bus

USRP Universal Software Radio Peripheral

VFO Variable Frequency Oscillator

TETRA Terrestrial Trunked Radio

TV Television

List of Figures

List of Tables

Bibliography

[Act01] Acterna, LLC. TETRA Introduction Pocket Guide, 2001. http://www.angelfire.com/folk/johnrhett/TETRA_introduction.pdf.

[Ash01] Ashkan Mashhour, William Domino, Norman Beamish. On the Direct Conversion Receiver - A Tutorial, June 2001. http://www.microwavejournal.com/articles/3226-on-the-direct-conversion-receiver-a-tutorial.

[Aud] Audacity. Audacity: Download. http://audacity.sourceforge.net/download/.

[Bog] Boger Electronics. Analysis System of Terrestrial Trunk Radio. http://www.boger-electronics.de/en/comint-systems/tetra-analysing-system.html.

[Bun] Bundesnetzagentur. Unberechtigtes Abhören von Nachrichten Dritter mit Strafe belegt. http://www.bundesnetzagentur.de/cln_1931/SharedDocs/FAQs/DE/BNetzA/PresseFAQs/Funk-Scanner.html.

[Bun13] Bundesnetzagentur. Veröffentlichung zugeteilter ITSI, February 2013. http://www.bundesnetzagentur.de/SharedDocs/Downloads/DE/BNetzA/Sachgebiete/Telekommunikation/Regulierung/Nummernverwaltung/TechnNummern/ITSI/VeroeffentlichungITSIlId12879pdf.pdf.

[D. 01] D. W. Parkinson. TETRA Security, July 2001. http://link.springer.com/content/pdf/10.1023%2FA%3A1011942300054.

[Dav04] David McCall. Taking a Walk Inside Bluetooth EDR, December 2004. http://eetimes.com/design/communications-design/4012550/Taking-a-Walk-Inside-Bluetooth-EDR?pageNumber=1.

[Den11] Dennis Wehrle, Konrad Meier, Chair in Communication Systems, Department of Applied Sciences, University of Freiburg. GSM, 2011. https://entropia.de/wiki/images/a/a7/GSM-Vortrag_GPN11.pdf.

[Doua] Doug Gray. An Overview of TETRA. http://portal.etsi.org/workshopps/Presentations/0103TETRA.pdf.

[Doub] Douglas L. Jones. Spectrum Analysis Using the Discrete Fourier Transform. http://cnx.org/content/m12032/latest/.

[EAD06] EADS. EADS to build TETRA digital radio network for STEAG encotec GmbH, February 2006. http://www.eads.com/eads/int/en/news/press.20060208_steag1.html.

[ETSa] ETSI. About ETSI. http://www.etsi.org/about.

[ETSb] ETSI. ETSI algorithms. http://www.etsi.org/services/security-algorithms/etsi-algorithms.

[ETS13] ETSI. ETR 300-1 - Terrestrial Trunked Radio (TETRA), January 2013. http://www.etsi.org/deliver/etsi_etr/300_399/30001/01_60/etr_30001e01p.pdf.

[Etta] Ettus Research. Building UHD Software from Source. http://code.ettus.com/redmine/ettus/projects/uhd/wiki/UHD_Build.

[Ettb] Ettus Research. UHD - USRP2 and N2X0 Series Application Notes. http://files.ettus.com/uhd_docs/manual/html/usrp2.html#ref-clock-10mhz.

[Ettc] Ettus Research. USRP Datasheet. http://www.olifantasia.com/gnuradio/usrp/files/datasheets/ds_usrp2.pdf.

[Ettd] Ettus Research. USRP Daughterboards. https://www.ettus.com/product/category/Daughterboards.

[F.J09] F.J. Senf, DSW21. itcs im Ruhrgebiet: Status Quo des RBL-KöR, October 2009. http://www.itcs-info.de/download/ITCS_Treffpunkt_Okt_09/Senf.pdf.

[FLT] FLTK developers. Download - Fast Light Toolkit (FLTK). http://fltk.org/software.php.

[Fra06] Frank Bötel, Joachim Seuling. Quantensprung in Sachen Kommunikation, 2006. http://www.bundeswehr.de/portal/a/bwde/!ut/p/c4/NYqxDoJAEET_6JaLBqKdxJjYAB1Ct8BKNsIdWVZp_HjuCmaSV8wbaCHU4Y9HVPYOJ3hB0_O120y3DWRWFWL9CNJbCer4DmvvHWmkklMOHAXVi1m86BTNVyQYwwMOib3nNkuO2H9ap3lbXc6nZ_EoYZnn2w7-j2e4/.

[GNUa] GNU Radio. About BTS Clocks. http://gnuradio.org/redmine/projects/gnuradio/wiki/OpenBTSClocks.

[GNUb] GNU Radio. OpenBTS History. http://gnuradio.org/redmine/projects/gnuradio/wiki/OpenBTS/history?p=1.

[GSM] GSM Master. Timing Advance (TA). http://gsm-optimization.blogspot.de/2012/04/timing-advance-ta.html.

[Han09] Hans Elberskirch. Software Defined Radio - Methoden, Verfahren und Anwendung, June 2009. http://www.afu-sh.de/lokal/vortrag/sdr-vortrag-m05.pdf.

[hei11] heise online: Author ju. TETRA-Digitalfunk für Jedermann, June 2011. http://www.heise.de/security/meldung/TETRA-Digitalfunk-fuer-jedermann-1253092.html.

[Hel] Helmut Bayerlein. Sensorprotokoll. http://www.dc3yc.privat.t-online.de/protocol.htm.

[Int] International Telecommunications Union. Frequently Asked Questions: What is meant by ISM applications and how are the related frequencies used? http://www.itu.int/ITU-R/terrestrial/faq/index.html#g013.

[Joh12] Johannes Kunze, Chair for Integrated Systems, Ruhr-Universität Bochum. Versuch: IT-V10: digitale Übertragungen, 2012.

[Lui12] Luis Sanabria-Russo, Jaume Barcelo, Albert Domingo, Boris Bellalta. Spectrum Sensing with USRP-E110. Technical report, September 2012. http://arxiv.org/abs/1209.1246.

[Mad] Made-in-China.com, Shenzhen Royear Technology Co., Ltd. High Power Remote Control. http://royear.en.made-in-china.com/product/xMsmejgJrdVw/China-High-Power-Remote-Control-Wireless-Transmitter-Radio-Remote-PT009.html.

[Mem] Members of the mikrocontroller.net forum. Funkthermometer Controlbits. http://www.mikrocontroller.net/topic/38129.

[Muh12] Muhammad Nazmul Islam, Byoung-Jo J. Kim, Paul Henry, Eric Rozner. A Wireless Channel Sounding System for Rapid Propagation Measurements, November 2012. http://arxiv.org/abs/1211.4940.

[Nat] National Institute of Standards and Technology (NIST). Announcing the ADVANCED ENCRYPTION STANDARD (AES). http://csrc.nist.gov/publications/fips/fips197/fips-197.pdf.

[Ope] OpenBTS developer community. Building, Installing and Running OpenBTS. http://wush.net/trac/rangepublic/wiki/BuildInstallRun.

[Osma] Osmocom TETRA. crc_test.c source code of osmo-tetra software. http://tetra.osmocom.org/trac/wiki/osmo-tetra.

[Osmb] OsmocomSDR. rtl-sdr OsmoSDR. http://sdr.osmocom.org/trac/wiki/rtl-sdr.

[Phi07] Philip Lücke, Hochschule Harz, Fachbereich Medieninformatik. Referat: Das Nyquist-Shannon Theorem, January 2007. http://myweb3.hs-harz.de/mkreyssig/af/pdf/NyquistShannon.pdf.

[PP10] Christof Paar and Jan Pelzl. Introduction to Cryptography and Data Security Understanding Cryptography. In *Understanding Cryptography*, chapter 1, pages 1–27. Springer Berlin Heidelberg, Berlin, Heidelberg, 2010.

[Rea] Realtek Semiconductors Corp. RTL2832U General Description. http://www.realtek.com.tw/products/productsView.aspx?Langid=1&PNid=22&PFid=35&Level=4&Conn=3&ProdID=257.

[RSI13] RSI. Aries, designed for real-time TETRA network performance monitoring, January 2013. http://www.rsi-uk.com/aries.htm.

[Sys] Sysmocom Shop. OsmoSDR developer preview edition. http://shop.sysmocom.de/products/osmosdr.

[Ted] Tedsen GmbH und Co. KG. signaling movement - Products. http://www.tedsen.com/index.php?id=163&L=2.

[TET] TETRA MoU Association. TETRA Security. http://www.tandcca.com/Library/Documents/About_TETRA/TETRA%20Security%20pdf.pdf.

[The] The John Burroughs Earth and Space Science Club. Radio Jove FAQ. http://people.sunyulster.edu/ESC/club/radiojove.html.

[Tim11] Tim Pritlove, Harald Welte. cre.fm 183: TETRA-Bündelfunk, June 2011. http://cre.fm/cre183.

[UKW13] UKWTV-Arbeitskreis der AGDX. Sender-Tabelle - DVB-T und FM: Nordrhein-Westfalen, January 2013. http://www.ukwtv.de/sender-tabelle.

[Uni05] Universität Koblenz-Landau, Sebastian Thiel. Diskrete Fourier-Transformation, July 2005. http://www.uni-koblenz.de/~physik/informatik/DSV/DFT.pdf.

[Use04] User Mozzerati on the English-language Wikipedia. TDMA frame structure, licensed under the Creative Commons Attribution-ShareAlike 3.0 Licence, September 2004. http://en.wikipedia.org/wiki/File:Tdma-frame-structure.png.

[Wan06] Wang Xiaoning. Linear Zero-IF Direct Conversion Receiver, December 2006. http://hft.uni-duisburg-essen.de/arbeiten/Vortrag_Wang_Xiaoning.pdf.

[Wik13] Wikipedia collaborators. Intermodulationsprodukt, January 2013. http://de.wikipedia.org/wiki/Intermodulationsprodukt.

[Wir] Wireshark. Display Filter Reference: TETRA Protocol. http://www.wireshark.org/docs/dfref/t/tetra.html.