

IMSI Catcher



By

GC M Bilal Mughal

GC Afaq Ahmad

GC Faraz Kanwar

GC Khan Nabi

Supervised by:

Lt Col M Imran Javaid

Submitted to the faculty of Department of Electrical Engineering,
Military College of Signals, National University of Sciences and Technology,
Islamabad,
in partial fulfillment for the requirements of B.E Degree in Electrical (Telecom)
Engineering.

June 2023

In the name of ALLAH, the Most benevolent, the Most Courteous

CERTIFICATE OF CORRECTNESS AND APPROVAL

This is to officially state that the thesis work contained in this report

“IMSI Catcher”

is carried out by

GC M Bilal Mughal

GC Afaq Ahmad

GC Faraz Kanwar

GC Khan Nabi

under my supervision and that in my judgement, it is fully ample, in scope and excellence, for the degree of Bachelor of Electrical (Telecom.) Engineering in Military College of Signals, National University of Sciences and Technology (NUST), Islamabad.

Approved by

Supervisor

Lt Col. M Imran Javaid

Department of EE, MCS

Date: _____

DECLARATION OF ORIGINALITY

We hereby declare that no portion of work presented in this thesis has been submitted in support of another award or qualification in either this institute or anywhere else.

ACKNOWLEDGEMENTS

Allah Subhan'Wa'Tala is the sole guidance in all domains.

Our parents, colleagues and most of all supervisor, Lt Col. M Imran Javaid without
your guidance.

The group members, who through all adversities worked steadfastly.

PLAGIARISM CERTIFICATE (Turnitin Report)

This thesis has **16%** similarity index. Turnitin report endorsed by Supervisor is attached.

GC M Bilal Mughal
325601

GC Afaq Ahmed
325041

GC Faraz Kanwar
325032

GC Khan Nabi
325037

Signature of Supervisor

ABSTRACT

Electronic Warfare is one of the major deciding factors in success of conventional and unconventional warfare. As RF technology is advancing with a very high pace, the conventional Electronic Warfare equipment are getting obsolete along with. Pakistan Army has indoctrinated a large quantity of Electronic Warfare equipment and still more is required to compete current requirement, owing to high tension scenario at borders and operational areas. These high-cost equipment are prone to frequent upgradation and maintenance as well as they are highly resource dependent. Considering above mentioned issues related to the EW equipment that Pak Army is using, they are seldom utilized for difficult terrains and border areas and are to be kept as reserve for conventional warfare. To aid heavily strained sector of EW, we utilized Software Defined Radios (SDR) to form a small portable detachment that can analyze RF spectrum, demodulate, decode and Identify Radio Sets being used in the vicinity of operator. These detachments being highly mobile and very less resource demanding, can be moved to any location for operation. We made considerable efforts in demodulation of non-encrypted radio channels and RF finger printing of radio by analyzing their frequency spectral density and frequency-time graphs. As Software Defined Radios are entirely computer dependent, the possibility of obsolescence of equipment is considerably reduced. New protocols can be programmed using opensource software and they can be implemented by SDR connected to computer via USB. All the computation and Digital Signal Processing (DSP) is to be handled by computer attached to SDR, hence performance of SDR is directly dependent on processing power of computer they are attached with, which is very cheap in comparison to traditional EW equipment upgradation. The RF fingerprinting is an important enhancement to EW sector, as this technology was not available to Pak Army, and we proudly have for the first time provided possible methodology and promising result.

Table of Contents

PLAGIARISM CERTIFICATE (Turnitin Report).....	6
ABSTRACT.....	7
Table of Contents.....	8
<i>Chapter 1</i>	11
INTRODUCTION.....	11
1.1 Background, Scope and Motivation	11
1.2 SDR:.....	12
1.2.1 RTL-SDR.....	13
1.3 Hack-RF One.....	14
1.4 Flow Graph of Hack-RF Jammer.....	16
1.4.1 Osmocom Sink:	16
1.4.2 Noise Source:	17
1.4.3 WX GUI Slider:.....	17
1.6.4 USRP Sink:.....	18
1.4 Waveform Generators	20
1.5 Modulators.....	20
1.7 Filters:	21
<i>Chapter 2</i>	22
LITERATURE REVIEW.....	22
2.1 Limitations and Areas of Improvement.....	23
2.2 Existing solutions and their drawbacks.....	25
<i>Chapter 3</i>	27
PROBLEM DEFINITION	27
3.1 Working Principle.....	27
3.2 Components and Functionality:.....	28
3.3 Project Procedure:	29
3.3.1 Design and Planning	29
3.3.2 Development and Implementation.....	29
3.3.3 Testing and Evaluation.....	30
<i>Chapter 4</i>	31
METHODOLOGY	31
4.1 Project Steps:.....	31
4.1.1 Hardware Setup:	31
4.1.3 Creating a Downgrade Script	31
4.1.4 Capturing the Downgraded Signal	32
4.1.5 Extracting IMSI Information.....	32

4.1.6 Storing IMSI Information	33
4.1.7 Creating a User Interface	33
4.1.8 Testing	33
4.2 Working Principle	34
4.2.1 The Hack-RF Jammer	34
4.2.2 Downgrading the 4G Signal	34
4.2.3 The RTL-SDR	34
4.2.4 Capturing the Downgraded Signal	34
4.2.5 Extracting IMSI Information	35
4.2.6 Storing IMSI Information	35
4.2.7 User Interface	35
Chapter 5	36
DETAILED DESIGN AND ARCHITECTURE	36
5.1 System Architecture:	36
5.2 Hardware and Software Components:	36
5.3 Signal Processing	38
5.4 User Interface	38
5.5 Security and Privacy	39
Chapter 6	40
IMPLEMENTATION AND TESTING	40
6.1 IMSI Catcher:	40
6.1.1 API Code: This code will connect frontend of the project with its backend. It will tell the front end that IMSI CATCHER has started and storing the IMSI number in its database and will give it a signal to start showing the output on GUI.	55
6.2 Using GSMTAP Library:	59
6.2.1 Setup and Initialization:	60
6.2.2 Signal Processing and Capture:	60
6.2.3 User Interface	60
6.2.4 Security and Privacy	66
6.3 HackRF Customized Jammer:	66
6.3.1 Code:	67
Chapter 7	75
RESULTS AND DISCUSSION	75
7.1 Results:	75
7.2 Discussion:	77
Chapter 8	79
CONCLUSIONS AND FUTURE WORK	79
8.1 Future Work	79
References	81
Meeting Logs & Plagiarism Report	82

LIST OF FIGURES

Figure 1: SDR workflow.....	12
Figure 2: Hack-RF One.....	14
Figure 3: Hack-RF Jammer	16
Figure 4: Example of Osmocom Sink.....	17
Figure 5: Example of USRP Sink.....	19
Figure 6: Components of desired IMSI Catcher	29
Figure 7 A and B: Extracting the IMSI Information.....	31 & 32
Figure 8: Flowchart of Project	35
Figure 9: HackRF ONE & RTL SDR.....	36
Figure 10: Flowgraph of Jammer.....	37
Figure 11: Graphical User Interface	38
Figure 12: Flowgraph of Jammer.....	65
Figure 12: Storing IMSI Numbers to Database	73
Figure 12: Locating GSM Tower.....	74
Figure 12: Cell Tower Location.....	74

INTRODUCTION

The research work in this dissertation has been presented in three main parts. First part is related to the wireless spectrum monitoring and investigation of unknown signal frequencies. The objective of this part is to formulate possible methods for wireless spectrum monitoring and interception of any unencrypted analogue and digitally modulated signal. The second part includes jamming and other wireless attacks that can be launched to signal of interest. Finally, third part revolves around signal identification and fingerprinting of radio sets by various techniques that can be adopted using SDR.

1.1 Background, Scope and Motivation

In today's world, mobile phones have become an essential part of our lives. It has become imperative for various organizations and governments to monitor the communication activities of individuals for national security purposes, intelligence gathering, and law enforcement activities. To cater to these needs, the concept of IMSI Catcher was introduced. IMSI Catcher has gained significant importance and popularity due to its ability to provide real-time monitoring of mobile phones in a specific area.

Software Defined Radios are highly dynamic radio sets that can perform any digital signal processing via computer attached to them with USB or ethernet interface. Since the signal processing is done using a computer software, this gives the fluidity to software defined radios in comparison to traditional hardware radios in which signal processing is usually done using analogue elements and circuitry. A traditional hardware RF device can fulfill a single purpose for which it was manufactured, for example a Wi-Fi modem cannot demodulate FM radio signals and same applies for an FM radio. But a single circuitry of SDR can perform both functions because the signal processing depends only on software and is independent of hardware. The computational power, being cheaper than complex hardware radios provide very cost-effective alternatives to its counterparts.

This feature or dynamicity in Software Defined Radios motivated us to research methods that could aid EW, particularly jamming and interception. As only software is required to change its function, hence a single SDR and a laptop is enough to provide all functions that were required. Moreover, new methods of encoding and modulating data over wireless signal can be adopted easily by few amendments in software. The SDR which we have chosen for our project is HackRF One which is a complete opensource equipment and is available in market easily for a very low cost. It has ability to transmit as well as receive signals and is half duplex in nature.

1.2 SDR:

Concept of SDR is briefly explained above. In this paragraph I would like to explain conceptual working methodology of SDR, which will further improve overall understanding of this equipment. An ideal SDR converts desired signal into bits of information and feeds it to computer for further processing, the computer is mainly responsible for all digital signal processing. Responsibility of SDR lies with catching, pre-filtering, pre-amplifying, and converting an analogue signal into digital signal and then packing digital signals into predefined bits. After this, it sends these signals to computer via USB interface.

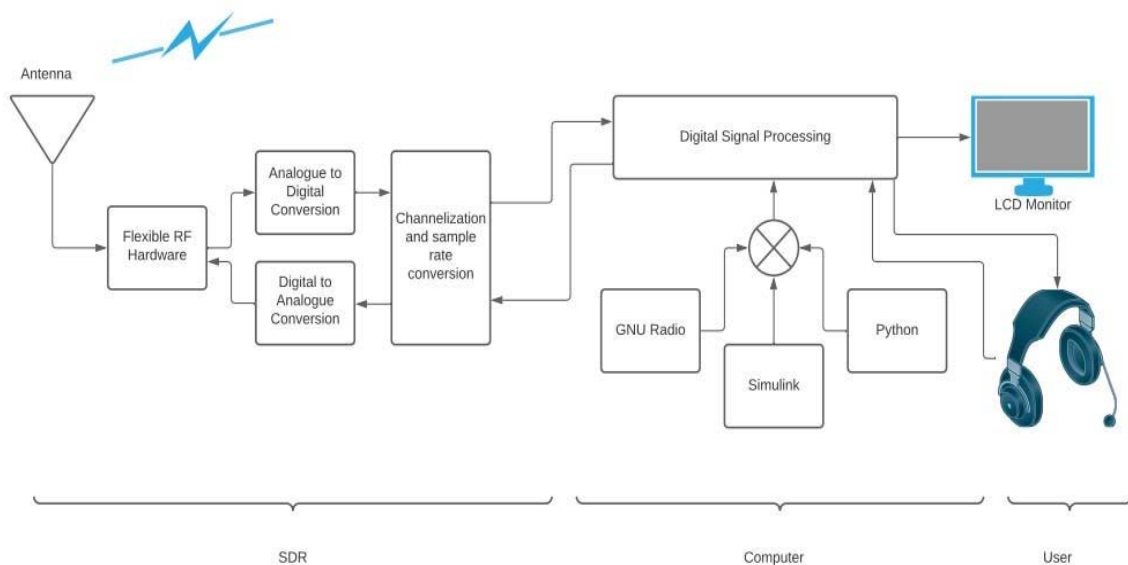


Figure 1: SDR workflow

1.2.1 RTL-SDR

Register Transfer Level is a hardware design methodology used in digital circuit design. It describes the behavior of a digital system by specifying the flow of data between registers.

In RTL design, the system is broken down into small components called registers, which are interconnected using data paths. These data paths are represented using logic gates and other electronic components. Each register in the system represents a specific state or value of the system.

The design process starts with defining the functionality of the system, which is then broken down into a set of operations that can be implemented using a combination of registers and logic gates. The registers are used to store the intermediate results of the operations.

Once the design is complete, it is verified using simulation tools that check whether the behavior of the system is consistent with the design specifications. The simulation also helps identify any potential errors or bugs in the design.

RTL is widely used in the design of digital circuits and systems, including microprocessors, digital signal processors, and other digital circuits. It is an important tool for creating complex digital systems that can perform a wide range of tasks with high efficiency and reliability.

RTL-SDR stands for "Realtek Software Defined Radio", and refers to a type of low-cost software-defined radio receiver that uses a Realtek RTL2832U chipset to receive and decode various types of radio signals. Here are some general specifications for RTL-SDR devices:

Frequency range: Typically between 24 MHz and 1766 MHz, although some devices can tune up to 2.4 GHz or higher.

Tuning resolution: Usually 1 kHz or better.

Bandwidth: Typically between 2.4 MHz and 3.2 MHz, although some devices can achieve up to 10 MHz or more.

Dynamic range: Varies depending on the specific device, but generally falls between

40 dB and 60 dB.

ADC resolution: 8 bits, although some devices may have an option for 12-bit sampling.

Connection: USB 2.0 or 3.0.

Compatibility: RTL-SDR devices are supported by many software-defined radio applications, including SDR#, HSDR, GQRX, and more.

Price: RTL-SDR devices are relatively inexpensive, with some models available for as little as \$20 USD.

1.3 Hack-RF One

Hack-RF One is an open source SDR designed and developed by Great Scott Gadgets which can transmit and receive radio signals from 1 Mega Hertz up to 6 Giga Hertz. It is specifically developed to provide testing and development platform for next generation radio technologies. It is to be connected to a computer via USB interface.



Figure 2: Hack-RF One

Following table explains important features of Hack-RF One SDR.

Table 1: Hack-RF One features

Operating Frequency	1Mhz – 6GHz
Mode of Operation	Half Duplex
Samples per second	20 million
Quadrature Samples	8-bit
Compatibility	GNU Radio, SDR#, GQRX
Antenna Port Power	50mA at 3.3V (Software Controlled)
Hardware	Open Source

1.4 Flow Graph of Hack-RF Jammer

Instrumentation are very useful blocks for analyzing received or transmitted signals visually.

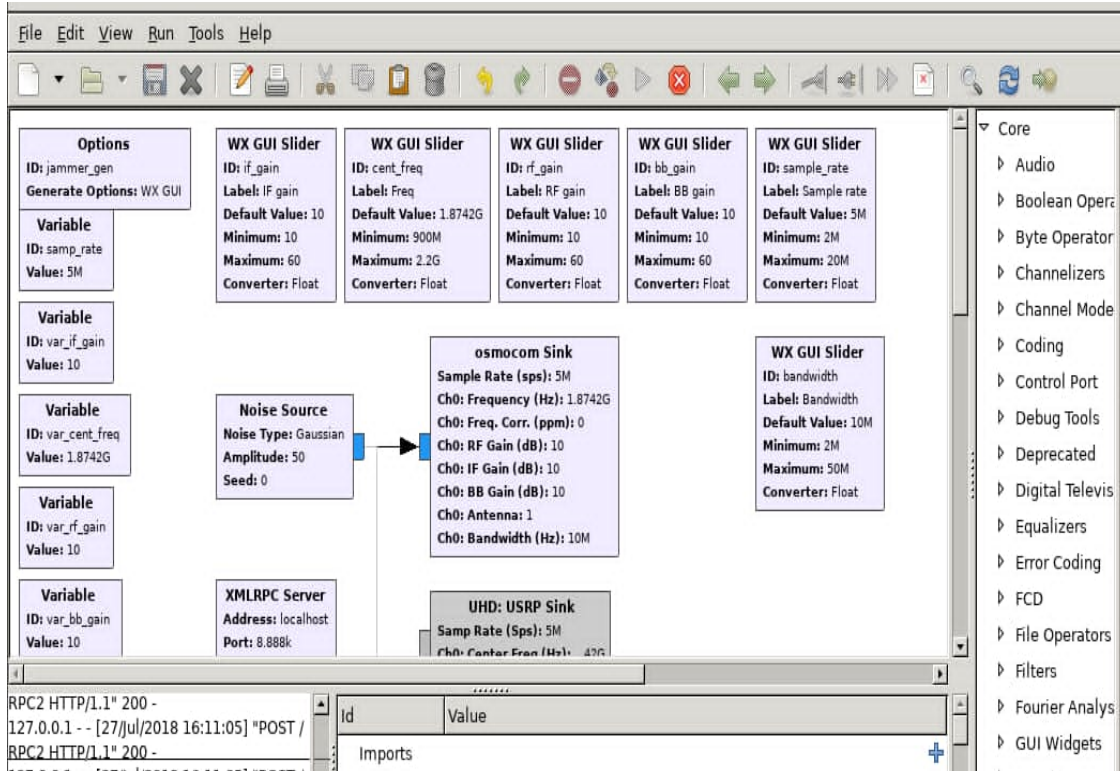


Figure 3: Hack-RF Jammer

1.4.1 Osmocom Sink:

The Osmocom sink works by taking the raw I/Q samples from the SDR hardware and converting them into a digital signal that can be decoded using various algorithms. The Osmocom sink is essentially a demodulator that can decode the physical layer of a GSM signal, which includes the modulation, timing, and frequency parameters.

The Osmocom sink supports a wide range of SDR hardware, including the popular RTL-SDR, HackRF, and USRP devices. It also supports multiple modulation schemes, including GMSK, 8PSK, and QPSK, which are used by GSM, GPRS, and EDGE respectively.

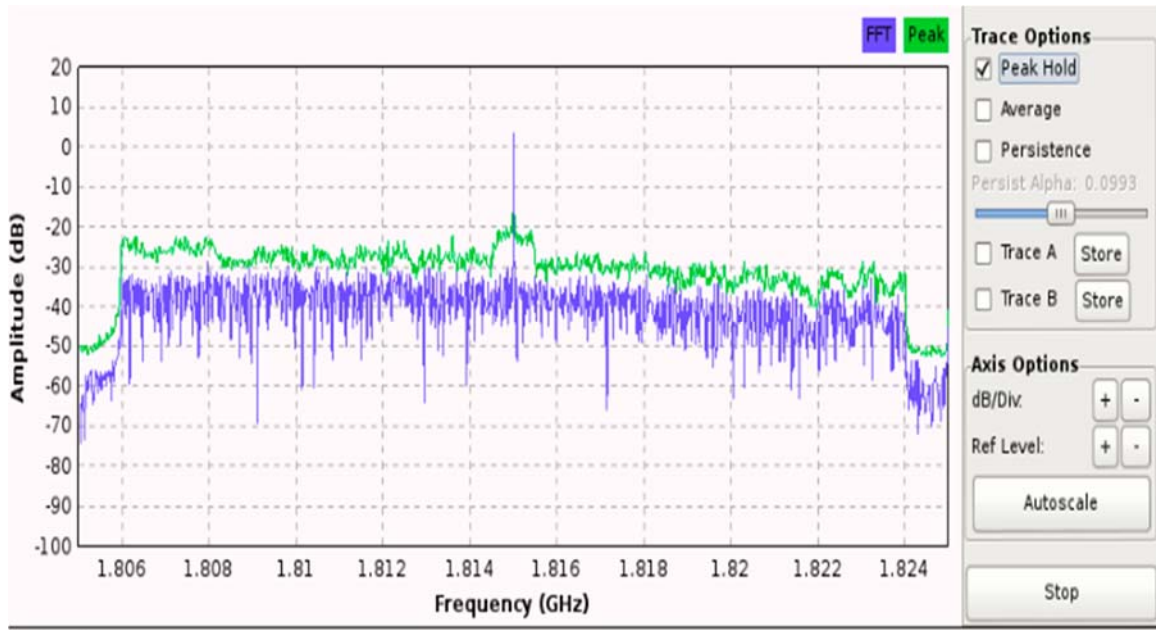


Figure 4: Example of Osmocom Sink

1.4.2 Noise Source:

A noise source is an electronic component used in radio frequency (RF) systems, including Hack-RF jammers, to introduce random noise into the RF signal. The purpose of a noise source is to increase the signal-to-noise ratio (SNR) of the system, which can improve the overall performance of the system.

In a Hack-RF jammer, a noise source is used to add random noise to the jamming signal. The jamming signal is a radio signal that is transmitted on the same frequency as the target signal, which can disrupt or block the target signal. The noise added to the jamming signal by the noise source can make the jamming signal more effective by increasing its power and reducing the signal quality of the target signal.

1.4.3 WX GUI Slider:

WX GUI Slider is a graphical user interface (GUI) component that can be

used in Hack-RF jammer software to adjust the power level of the jamming signal. The slider provides a visual representation of the power level, and users can adjust the power level by dragging the slider handle to the desired position.

The WX GUI Slider component is based on the wxPython toolkit, which is a popular GUI toolkit for Python. It allows developers to create a wide range of GUI components that are compatible with Hack-RF software.

In a Hack-RF jammer, the WX GUI Slider is typically used to adjust the power level of the jamming signal. The power level is a critical parameter in jamming, as it determines the strength of the jamming signal and how effectively it can disrupt or block the target signal.

The WX GUI Slider component can be customized to suit the needs of the user. This includes changing the range of the slider, adjusting the minimum and maximum values, and setting the initial value. The component can also be styled to match the overall design of the jammer software.

In addition to adjusting the power level of the jamming signal, the WX GUI Slider component can also be used to display other parameters of the jammer, such as frequency and modulation. This allows users to have a more comprehensive view of the jamming signal and adjust it accordingly.

1.6.4 USRP Sink:

USRP Sink is a component of the Universal Software Radio Peripheral (USRP) hardware and software system. The USRP is a software-defined radio (SDR) platform that is widely used for research, education, and experimentation in wireless communication.

The USRP Sink component is used to transmit RF signals from a computer to an attached USRP hardware device. The USRP Sink takes a stream of digital samples and converts them into an analog signal that can be transmitted over the air by the

USRP hardware.

The USRP Sink component can transmit a wide range of signals, including radio signals, audio signals, and digital data. The component supports multiple modulation schemes, including amplitude modulation (AM), frequency modulation (FM), and phase-shift keying (PSK), among others.

The USRP Sink component is typically used in research and experimentation in wireless communication systems. Researchers and students can use the USRP Sink component to transmit and receive signals and analyze the performance of wireless communication systems.

One of the key features of the USRP Sink component is its ability to transmit signals in real-time. This allows researchers to test and evaluate wireless communication systems in real-world scenarios, without the need for expensive equipment or specialized facilities.

The USRP Sink component is compatible with various software tools, including GNU Radio, which is an open-source software toolkit for SDR. GNU Radio provides a graphical interface for designing and testing SDR systems and can be used to create custom signal processing blocks and implement complex signal processing algorithms.

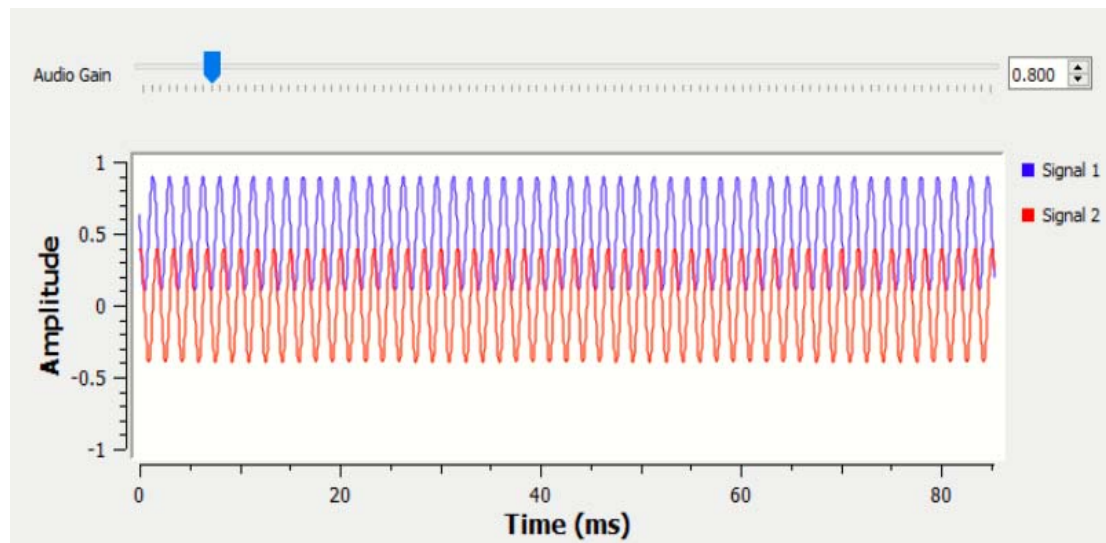


Figure 5: Example of USRP Sink

1.4 Waveform Generators

Constant Source: A constant source of signal (DC).

Noise Source: Can produce Active White Gaussian Noise or random noise.

Signal Source (e.g. Sine, Square, Saw Tooth): These generate signals of our choice (cosine, square, Saw Tooth etc.) and desired frequency.

1.5 Modulators

AM Mod/Demod: This block can perform Amplitude Modulation and demodulation with provided sample rate.

Continuous Phase Modulation: This block will take data in complex form and modulate this data with a sine wave of provided frequency and mentioned sample rate

PSK Mod / Demod: This block will perform phase shift keying modulation and demodulation on provided digital data.

GFSK Mod / Demod: Gaussian Frequency Shift Keying is a type of Frequency Shift Keying modulation where signal is passed through a gaussian filter to shape the pulses before modulating which greatly reduces spectral bandwidth and out-of-band spectrum. This is helpful when adjacent channel has high power and there is a chance of interference between channels. This block modulates data into GFSK and demodulates GFSK signal into data.

GMSK Mod / Demod: Gaussian Mean Shift Keying Modulation/ Demodulation.

QAM Mod / Demod: Quadrature Amplitude Modulation/ Demodulation.

WBFM Receive: Wide Band Frequency Modulation / Demodulation.

NBFM Receive: Narrow Band Frequency Modulation / Demodulation.

1.7 Filters:

Filters are basic components of DSP and their use is almost compulsory in every signalflowgraph. Few filter blocks provided by SDR are:

- Band Pass / Reject Filter
- Low / High Pass Filter
- IIR Filter
- Generic Filter bank
- Hilbert

Tasks as signal normalization, sync and visualization can be done by making a suitable signal processing flow graph. Just by connecting appropriate block one after other a complete system is made. There is also option of writing own block in python if some logic is not already present is available blocks.

It is important to consider GNU Radio is primarily a structure for the development of signal processing blocks and their interaction. It has inbuilt extensive library of blocks. But it should be borne in mind that GNU Radio itself is not a software that is ready to do something specific. it is the operator's task to make something useful out of it.

LITERATURE REVIEW

IMSI catchers, also known as "Stingrays," are tools that can be used to intercept and track mobile phone communications. These devices work by simulating a cell phone tower, tricking nearby phones into connecting to the IMSI catcher instead of a legitimate tower. Once connected, the IMSI catcher can intercept and monitor the phone's communications, including voice calls, text messages, and data transmissions.

The use of IMSI catchers by law enforcement agencies and government entities has been a subject of controversy in recent years, as their use raises concerns about privacy and civil liberties. While some jurisdictions have placed restrictions on their use, IMSI catchers remain a valuable tool for law enforcement agencies in certain situations.

In recent years, there have been a number of open-source projects aimed at developing IMSI catchers using software-defined radios and other off-the-shelf components. This IMSI Catcher is a Python-based IMSI catcher.

It is based on a software-defined radio (SDR) device, which allows for flexible, programmable reception of wireless signals. The software is designed to run on a Linux-based operating system, such as Kali Linux or Ubuntu.

The Python code used in the IMSI catcher is designed to perform a number of different functions, including scanning for nearby cell towers, identifying the International Mobile Subscriber Identity (IMSI) numbers of nearby phones, and intercepting phone calls and text messages. The software is also capable of spoofing cell tower signals, allowing it to intercept communications even if the phone is not in range of a legitimate tower.

One of the key advantages of this IMSI catcher is that it is open-source, allowing other developers to modify and improve upon the code as needed. This has led to the development of a number of additional tools and modules that can be used in conjunction with the IMSI catcher, such as modules for GPS tracking and geolocation.

While the use of IMSI catchers is controversial, there are legitimate use cases for these devices, such as in the tracking of suspected criminals or terrorists. This IMSI catcher, with its flexible and customizable software, provides a powerful tool for law enforcement agencies and other organizations that need to track and monitor mobile communications.

However, it is important to note that the use of IMSI catchers can also have unintended consequences. For example, innocent bystanders could have their communications intercepted, and the use of IMSI catchers could potentially interfere with legitimate cellular network operations. Therefore, it is important to use these devices responsibly and within the bounds of the law.

In conclusion, this Python-based IMSI catcher developed by provides a powerful tool for the interception and monitoring of mobile phone communications. While controversial, the use of IMSI catchers can have legitimate use cases in law enforcement and other areas. As with any powerful technology, it is important to use these devices responsibly and with a full understanding of their capabilities and limitations.

2.1 Limitations and Areas of Improvement

One major limitation of IMSI catchers is their reliance on software-defined radio (SDR) technology. While SDR devices are highly flexible and programmable, they are also subject to a number of technical limitations, including limited bandwidth and sensitivity. This can make it difficult to detect and intercept mobile phone communications in areas with high levels of interference or signal noise.

Another limitation of IMSI catchers is their potential impact on legitimate cellular network operations. By spoofing cell tower signals and intercepting phone communications, IMSI catchers can potentially cause interference and disrupt network operations. This can be especially problematic in areas with high levels of cellular traffic, such as busy urban areas or large events.

IMSI catchers are also limited by the accuracy of their tracking and geolocation capabilities. While these devices can be used to track the movements of mobile phones, the accuracy of this tracking is dependent on a number of factors, including the strength of the cell signal and the location of nearby cell towers. This can make it difficult to track mobile phones in certain areas or under certain conditions.

In terms of areas for improvement, there are several potential avenues for future development. One area is the development of more sophisticated and powerful SDR devices, which could improve the accuracy and sensitivity of IMSI catchers. Additionally, improvements in machine learning and artificial intelligence could be used to improve the accuracy of tracking and geolocation capabilities.

Another area for improvement is the development of more advanced and sophisticated signal processing algorithms. By improving the ability of IMSI catchers to filter out noise and interference, these devices could become more effective at intercepting and monitoring mobile phone communications.

Finally, there is a need for greater transparency and accountability in the use of IMSI catchers by law enforcement agencies and other organizations. As the use of these devices becomes more widespread, it is important to ensure that they are being used in accordance with the law and that the privacy rights of individuals are being protected.

In conclusion, while IMSI catchers are a powerful tool for intercepting and monitoring mobile phone communications, there are also limitations and areas for

improvement that must be considered. By addressing these limitations and focusing on areas for improvement, it may be possible to develop more effective and accurate IMSI catchers in the future.

2.2 Existing solutions and their drawbacks

There are a number of existing solutions for intercepting and monitoring mobile phone communications, including IMSI catchers, cell site simulators, and mobile network analysis tools. While each of these solutions has its own advantages and drawbacks, there are a number of common limitations and challenges that must be considered.

One of the most commonly used solutions for intercepting mobile phone communications is the IMSI catcher, also known as a "Stingray." These devices work by simulating a cell phone tower, tricking nearby phones into connecting to the IMSI catcher instead of a legitimate tower. Once connected, the IMSI catcher can intercept and monitor the phone's communications.

While IMSI catchers can be effective at intercepting mobile phone communications, they are also subject to a number of technical limitations. For example, they are often limited by the range and sensitivity of the software-defined radio (SDR) device used to power the IMSI catcher. Additionally, they can potentially interfere with legitimate cellular network operations, causing disruption and interference.

Another solution for intercepting mobile phone communications is the cell site simulator, also known as a "Hailstorm" or "DRTbox." These devices work by broadcasting a signal that is stronger than that of nearby cell towers, causing mobile phones to connect to the simulator instead. Once connected, the simulator can intercept and monitor the phone's communications. This concept we will use for jamming the 3G & 4G signals in our jammer.

While cell site simulators can be effective at intercepting mobile phone communications, they are also subject to a number of limitations and drawbacks. For example, they are often expensive and difficult to use, requiring specialized training and technical expertise. Additionally, they can potentially interfere with legitimate cellular network operations, causing disruption and interference.

Finally, there are a number of mobile network analysis tools that can be used to monitor and intercept mobile phone communications. These tools work by analyzing network traffic and identifying patterns and anomalies that may indicate suspicious activity.

While mobile network analysis tools can be effective at identifying and intercepting mobile phone communications, they are also subject to a number of limitations and drawbacks. For example, they are often expensive and require specialized technical expertise to use effectively. Additionally, they may be subject to false positives and false negatives, leading to inaccurate results and potentially putting innocent individuals at risk.

In conclusion, there are a number of existing solutions for intercepting and monitoring mobile phone communications, including IMSI catchers, cell site simulators, and mobile network analysis tools. While each of these solutions has its own advantages and drawbacks, they are all subject to common limitations and challenges, including technical limitations, the potential for interference with legitimate network operations, and the risk of false positives and false negatives. As with any powerful technology, it is important to use these solutions responsibly and with a full understanding of their capabilities and limitations.

Chapter 3

PROBLEM DEFINITION

The aim of our project report is to present the design, development, and evaluation of a cutting-edge technology named IMSI Catcher. IMSI Catcher is a highly specialized and sophisticated device that is used for surveillance purposes. It provides a comprehensive solution for monitoring and tracking mobile phones in a specific area. The project report will highlight the technical aspects of the IMSI Catcher, its design and development methodology, the outcome and results, and future recommendations.

3.1 Working Principle

The IMSI Catcher works by first scanning all the GSM communication frequencies in the respective cell and then tune into one of them. Once the packets starts getting captured, the IMSI Catcher can then intercept the mobile phone signals and retrieve the IMSI number of each mobile phone, providing real-time monitoring of communication activities.

The working principle of an IMSI catcher involves mimicking a legitimate cell tower and tricking nearby mobile phones into connecting to it. Once a phone is connected to the IMSI catcher, it can intercept and record all the communications between the phone and the legitimate network.

Here are the steps involved in the working principle of an IMSI catcher:

- The IMSI catcher is set up and activated in an area where the target phones are expected to be present.
- The IMSI catcher will tuned in to the frequency of the network available in the are and will start intercepting its traffic.
- The IMSI catcher collects the IMSI numbers of all the connected phones. These IMSI numbers are used to identify the phone users and track their movements.
- The IMSI catcher then will store all the IMSI numbers in the database.
- Once the phones are connected to the IMSI catcher, it can intercept and record all the communications between the phone and the network, including voice calls, text

messages, and data transmissions.

- The IMSI catcher can also spoof the location of the phones by altering the signal strength and timing of the transmissions, making it difficult to track the actual location of the phones.

IMSI catchers are often used by law enforcement agencies and intelligence services for surveillance and intelligence gathering purposes. However, they can also be used by criminals and malicious actors for illegal activities such as identity theft and cyber espionage.

3.2 Components and Functionality:

The IMSI Catcher consists of various components that perform specific functions. Some of the critical components are:

Antenna: The antenna is used to emit the signal that tricks the mobile phones to connect to the fake tower.

Radio Frequency (RF) Transmitter: The RF transmitter is responsible for transmitting and receiving signal.

Signal Processing Unit: The signal processing unit is used to process the intercepted mobile phone signals and retrieve the IMSI number.

Control Unit: The control unit manages the overall functionality of the IMSI Catcher and provides a user interface for monitoring and tracking the mobile phones.



Figure 6: Components of desired IMSI Catcher

3.3 Project Procedure:

We completed our project in milestones using the following listed steps:

3.3.1 Design and Planning

The design and planning phase involved the detailed study and analysis of the requirements and specifications of the IMSI Catcher. The design was based on the principles of effective signal transmission and processing to ensure the device provides accurate results. The project plan was also developed to ensure the timely delivery of the IMSI Catcher.

3.3.2 Development and Implementation

The development and implementation phase involved the actual construction of the IMSI Catcher, including the assembly of all the components and the integration of software systems. This phase required close collaboration between the technical team and the project stakeholders to ensure that the device met the required specifications and standards. The software systems were tested and refined to ensure they provided real-time monitoring and tracking capabilities.

3.3.3 Testing and Evaluation

The testing and evaluation phase involved a thorough assessment of the IMSI Catcher's performance and capabilities. The device was subjected to various tests, including signal strength, accuracy, and stability, to ensure it provided reliable and consistent results. The testing phase also included a comprehensive evaluation of the software systems, including their user-friendliness and functionality.

METHODOLOGY

4.1 Project Steps:

In this chapter, we will discuss the methodology used in the development of this IMSI Catcher to catch 3G & 4G network signals by downgrading them into 2G using a Hack-RF Jammer and RTL-SDR.

4.1.1 Hardware Setup:

The first step in developing an IMSI Catcher is to set up the required hardware. For this project, we will need a Hack-RF ONE to create a Jammer and an RTL-SDR for interception. The Hack-RF Jammer will be used to downgrade the 4G & 3G signal to 2G, and the RTL-SDR will be used to capture the downgraded signal.

4.1.2 Software Installation

Once the hardware setup is complete, the next step is to install the required software. We will be using Python as the programming language for this project. We will also need to install various Python libraries such as Scapy, PyUSB, and RTL-SDR.

4.1.3 Creating a Downgrade Script

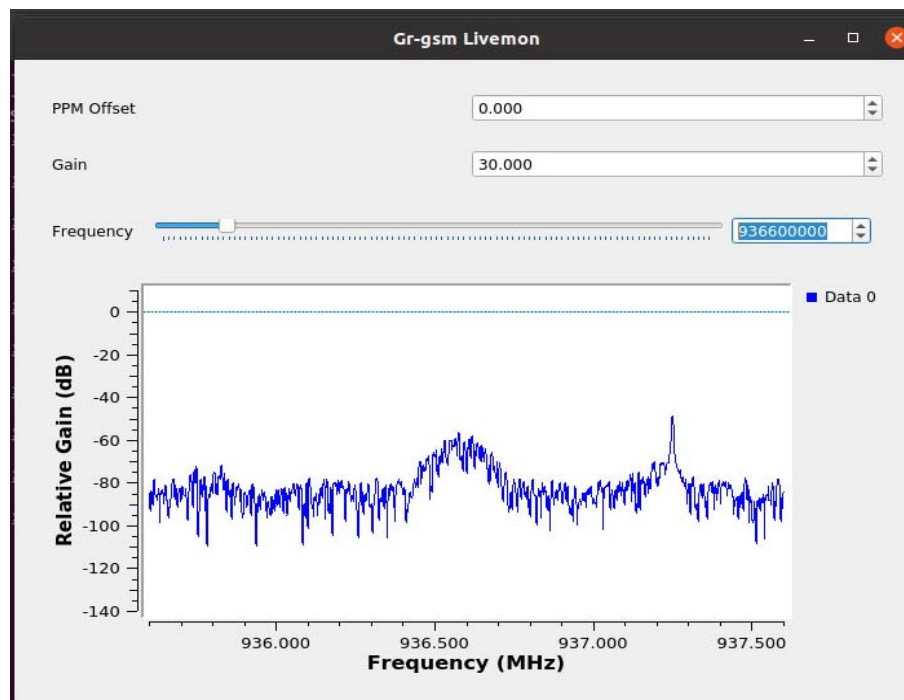
The next step is to create a flowgraph in GNU RADIO that will downgrades the 4G & 3G signal to 2G. We will be using the Hack-RF Jammer for this purpose. The Hack-RF Jammer can be controlled using Python, and we will be sending commands to the Jammer from our Python script to downgrade the signal.

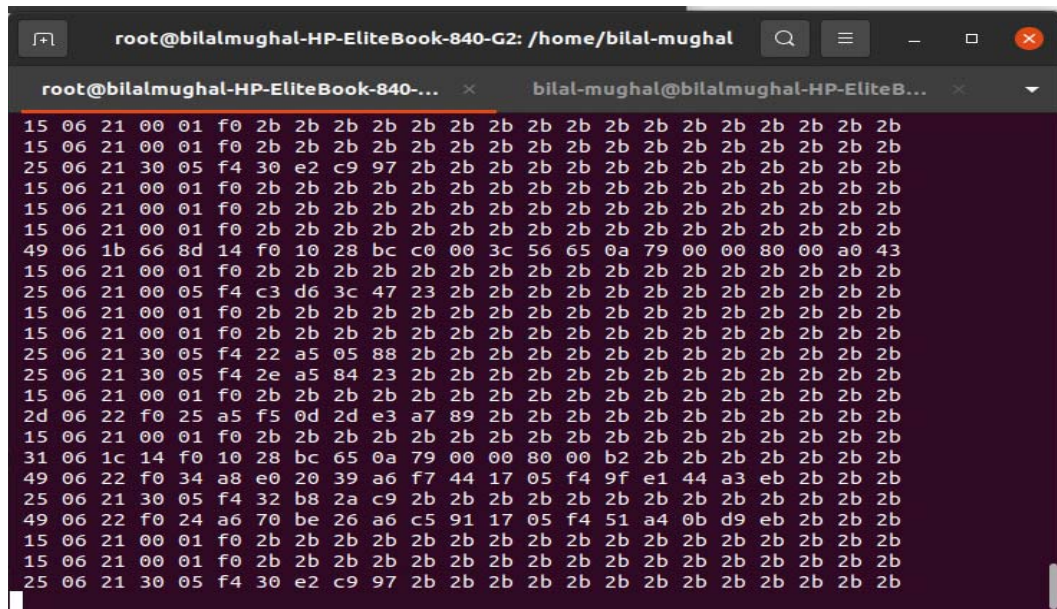
4.1.4 Capturing the Downgraded Signal

Once the signal is downgraded to 2G, we will use the RTL-SDR to capture the signal. The RTL-SDR is a software-defined radio that can be controlled using Python. We will use the Python RTL-SDR library to capture the downgraded signal.

4.1.5 Extracting IMSI Information

The captured signal will contain information about the IMSI of the devices connected to the network. We will use the Python Scapy library to extract the IMSI information from the captured signal. Scapy is a powerful packet manipulation library that can be used to analyze network traffic.





```
root@bilalmughal-HP-EliteBook-840-G2: /home/bilal-mughal
root@bilalmughal-HP-EliteBook-840-... x bilal-mughal@bilalmughal-HP-EliteB... x
15 06 21 00 01 f0 2b 2b 2b 2b 2b 2b 2b 2b 2b 2b 2b 2b 2b 2b
15 06 21 00 01 f0 2b 2b 2b 2b 2b 2b 2b 2b 2b 2b 2b 2b 2b 2b
25 06 21 30 05 f4 30 e2 c9 97 2b 2b 2b 2b 2b 2b 2b 2b 2b 2b
15 06 21 00 01 f0 2b 2b 2b 2b 2b 2b 2b 2b 2b 2b 2b 2b 2b 2b
15 06 21 00 01 f0 2b 2b 2b 2b 2b 2b 2b 2b 2b 2b 2b 2b 2b 2b
15 06 21 00 01 f0 2b 2b 2b 2b 2b 2b 2b 2b 2b 2b 2b 2b 2b 2b
49 06 1b 66 8d 14 f0 10 28 bc c0 00 3c 56 65 0a 79 00 00 80 00 a0 43
15 06 21 00 01 f0 2b 2b 2b 2b 2b 2b 2b 2b 2b 2b 2b 2b 2b 2b
25 06 21 00 05 f4 c3 d6 3c 47 23 2b 2b 2b 2b 2b 2b 2b 2b 2b 2b
15 06 21 00 01 f0 2b 2b 2b 2b 2b 2b 2b 2b 2b 2b 2b 2b 2b 2b
15 06 21 00 01 f0 2b 2b 2b 2b 2b 2b 2b 2b 2b 2b 2b 2b 2b 2b
25 06 21 30 05 f4 22 a5 05 88 2b 2b 2b 2b 2b 2b 2b 2b 2b 2b
25 06 21 30 05 f4 2e a5 84 23 2b 2b 2b 2b 2b 2b 2b 2b 2b 2b
15 06 21 00 01 f0 2b 2b 2b 2b 2b 2b 2b 2b 2b 2b 2b 2b 2b 2b
2d 06 22 f0 25 a5 f5 0d 2d e3 a7 89 2b 2b 2b 2b 2b 2b 2b 2b
15 06 21 00 01 f0 2b 2b 2b 2b 2b 2b 2b 2b 2b 2b 2b 2b 2b 2b
31 06 1c 14 f0 10 28 bc 65 0a 79 00 00 80 00 b2 2b 2b 2b 2b 2b
49 06 22 f0 34 a8 e0 20 39 a6 f7 44 17 05 f4 9f e1 44 a3 eb 2b 2b
25 06 21 30 05 f4 32 b8 2a c9 2b 2b 2b 2b 2b 2b 2b 2b 2b 2b
49 06 22 f0 24 a6 70 be 26 a6 c5 91 17 05 f4 51 a4 0b d9 eb 2b 2b
15 06 21 00 01 f0 2b 2b 2b 2b 2b 2b 2b 2b 2b 2b 2b 2b 2b 2b
15 06 21 00 01 f0 2b 2b 2b 2b 2b 2b 2b 2b 2b 2b 2b 2b 2b 2b
25 06 21 30 05 f4 30 e2 c9 97 2b 2b 2b 2b 2b 2b 2b 2b 2b 2b
```

Figure 7 A and B: Extracting the IMSI Information

4.1.6 Storing IMSI Information

Finally, we will store the extracted IMSI information in a database. We will be using SQLite for this purpose, as it is a lightweight database that can be easily integrated with Python.

4.1.7 Creating a User Interface

To make it easier to use the IMSI Catcher, we will create a user interface using Python's Tkinter library. The user interface will allow the user to start and stop the IMSI Catcher and view the captured IMSI information.

4.1.8 Testing

Once the IMSI Catcher is complete, we will test it in a controlled environment to ensure that it is working as expected. We will also perform various tests to ensure that the IMSI information is being captured correctly and stored in the database.

4.2 Working Principle

An IMSI Catcher is a device that can intercept and collect the International Mobile Subscriber Identity (IMSI) of mobile devices connected to a cellular network. This project aims to create an IMSI Catcher using Python, a Hack-RF Jammer, and an RTL-SDR to capture 4G internet signals by downgrading them into 2G.

4.2.1 The Hack-RF Jammer

The Hack-RF Jammer is a software-defined radio that can be controlled using Python. It can be programmed to transmit and receive radio signals of various frequencies and modulation schemes. In this project, we will be using the Hack-RF Jammer to downgrade the 4G signal to 2G.

4.2.2 Downgrading the 4G Signal

The 4G signal is downgraded by sending a command to the Hack-RF Jammer to transmit a signal that is at a lower frequency and modulation scheme than the original 4G signal. This causes the mobile device to switch to 2G mode, and the IMSI Catcher can then capture the downgraded signal.

4.2.3 The RTL-SDR

The RTL-SDR is a software-defined radio that can be used to capture and analyze radio signals. It can be controlled using Python, and it is capable of capturing signals in a wide range of frequencies. In this project, we will use the RTL-SDR to capture the downgraded 2G signal.

4.2.4 Capturing the Downgraded Signal

The downgraded 2G signal can be captured using the RTL-SDR and the Python RTL-SDR library. The captured signal will contain information about the IMSI of the mobile devices connected to the network.

4.2.5 Extracting IMSI Information

To extract the IMSI information from the captured signal, we will use the Python Scapy library. Scapy is a powerful packet manipulation library that can be used to analyze network traffic. It can extract the IMSI information from the captured signal and provide it in a usable format.

4.2.6 Storing IMSI Information

The extracted IMSI information will be stored in a database using SQLite. SQLite is a lightweight database that can be easily integrated with Python. It can be used to store and retrieve IMSI information for further analysis.

4.2.7 User Interface

To make the IMSI Catcher easier to use, we will create a user interface using REACT APP and FLASK. The user interface will allow the user to start and stop the IMSI Catcher and view the captured IMSI information and can also search for specific IMSI in the database.

DETAILED DESIGN AND ARCHITECTURE

This section will provide a design detail of our application including high level system as per the project and its system model.

5.1 System Architecture:

A high-level overview of the system architecture, including the different components and how they interact with each other.

A diagram of the system architecture, showing the relationships between the components and their roles.

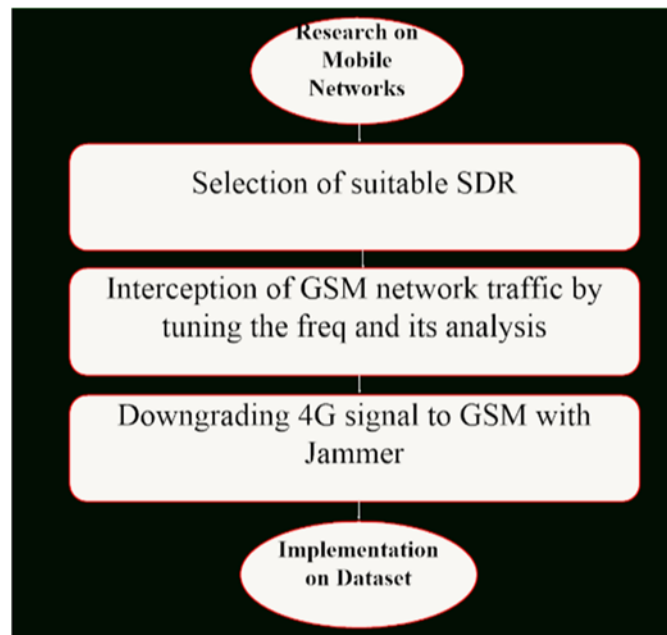


Figure 8: Flowchart of Project

5.2 Hardware and Software Components:

A description of the hardware and software components used in the project, including the HackRF jammer, RTL-SDR, and any other components.

Detailed specifications for each component, including their capabilities and limitations.



Figure 9: HackRF one & RTL SDR

5.3 Signal Processing

A detailed explanation of how the signal processing works in the IMSI catcher, including how the HackRF jammer is used to downgrade the 4G signal to 2G, and how the RTL-SDR is used to capture the downgraded signal.

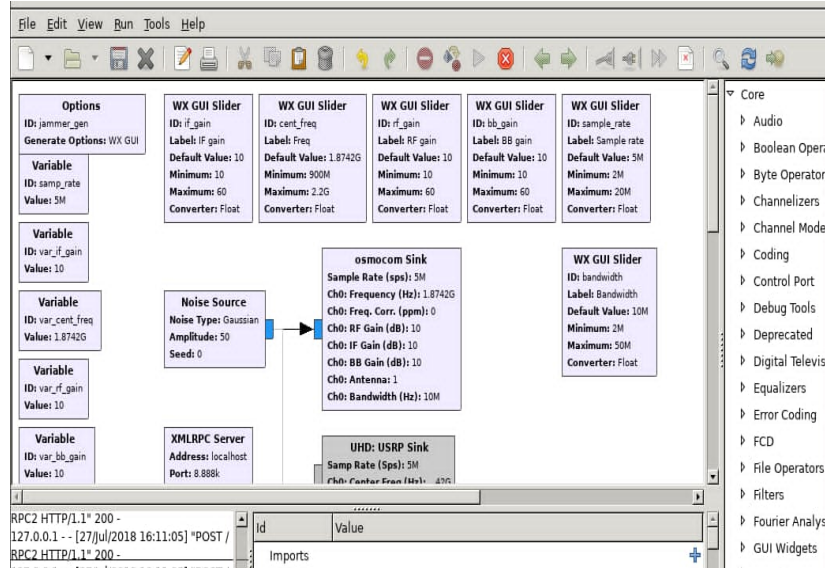


Figure 10: Flowgraph of Jammer

5.4 User Interface

A description of the user interface used to control the IMSI catcher, including any buttons, sliders, or other controls used to adjust the jamming signal.

Screenshots of the user interface, showing the different controls and their functions.



Figure 11: GUI

5.5 Security and Privacy

A discussion of the security and privacy implications of using an IMSI catcher, including the potential risks and ways to mitigate them.

An explanation of any security or privacy features built into the IMSI catcher, such as encryption or anonymization.

These are just some of the elements that are included in a detailed design and architecture section for a project involving an IMSI catcher using a HackRF jammer and RTL-SDR.

IMPLEMENTATION AND TESTING

In this chapter we discuss the implementation and programming techniques that we use, we discuss how we convert the system from analysis state into real system that can be used by android users.

6.1 IMSI Catcher:

The code utilizes the Flask framework for building a web application and SocketIO for real-time communication between the server and clients. The code establishes a connection to a MySQL database to store IMSI records.

```
import ctypes
import json
from optparse import OptionParser
import datetime
import io
import socket

imsitracker = None

class tracker:
    imsistate = {}

    imsis = []
    tmsis = {}
    nb_IMSI = 0

    mcc = ""
    mnc = ""
    lac = ""
    cell = ""
```



```

country = ""
brand = ""
operator = ""

purgeTimer = 10

show_all_tmsi = False
mcc_codes = None
sqlite_con = None
mysql_con = None
mysql_cur = None
textfilePath = None
output_function = None

def __init__(self):
    self.load_mcc_codes()
    self.track_this_imsi("")
    self.output_function = self.output

def set_output_function(self, new_output_function):
    self.output_function = new_output_function

def track_this_imsi(self, imsi_to_track):
    self.imsi_to_track = imsi_to_track
    self.imsi_to_track_len = len(imsi_to_track)

def str_tmsi(self, tmsi):
    if tmsi != "":
        new_tmsi = "0x"
        for a in tmsi:
            c = hex(a)
            if len(c) == 4:

```

```

        new_tmsi += str(c[2]) + str(c[3])
    else:
        new_tmsi += "0" + str(c[2])
    return new_tmsi
else:
    return ""

def decode_imsi(self, imsi):
    new_imsi = ""
    for a in imsi:
        c = hex(a)
        if len(c) == 4:
            new_imsi += str(c[3]) + str(c[2])
        else:
            new_imsi += str(c[2]) + "0"

    mcc = new_imsi[1:4]
    mnc = new_imsi[4:6]
    return new_imsi, mcc, mnc

def str_imsi(self, imsi, packet=""):
    new_imsi, mcc, mnc = self.decode_imsi(imsi)
    country = ""
    brand = ""
    operator = ""

    if mcc in self.mcc_codes:
        if mnc in self.mcc_codes[mcc]:
            brand, operator, country, _ = self.mcc_codes[mcc][mnc]
            new_imsi = f"{mcc} {mnc} {new_imsi[6:]}"
        elif mnc + new_imsi[6:7] in self.mcc_codes[mcc]:
            mnc += new_imsi[6:7]
            brand, operator, country, _ = self.mcc_codes[mcc][mnc]
            new_imsi = f"{mcc} {mnc} {new_imsi[7:]}"

```

```

else:
    country = f"Unknown MCC {mcc}"
    brand = f"Unknown MNC {mnc}"
    operator = f"Unknown MNC {mnc}"
    new_imsi = f"{mcc} {mnc} {new_imsi[6:]}"

try:
    return new_imsi, country, brand, operator
except Exception:

    print("Error", packet, new_imsi, country, brand, operator)
    return "", "", "", ""

def load_mcc_codes(self):

    with io.open('mcc-mnc/mcc_codes.json', 'r', encoding='utf8') as file:
        self.mcc_codes = json.load(file)

def current_cell(self, mcc, mnc, lac, cell):
    brand = ""
    operator = ""
    country = ""
    if mcc in self.mcc_codes and mnc in self.mcc_codes[mcc]:
        brand, operator, country, _ = self.mcc_codes[mcc][mnc]
    else:
        country = f"Unknown MCC {mcc}"
        brand = f"Unknown MNC {mnc}"
        operator = f"Unknown MNC {mnc}"
    self.mcc = str(mcc)
    self.mnc = str(mnc)
    self.lac = str(lac)
    self.cell = str(cell)
    self.country = country

```

```

self.brand = brand
self.operator = operator

def sqlite_file(self, filename):
    import sqlite3
    print("Saving to SQLite database in %s" % filename)
    self.sqlite_con = sqlite3.connect(filename)
    self.sqlite_con.text_factory = str
    # FIXME
    self.sqlite_con.execute("CREATE TABLE IF NOT EXISTS
observations(stamp datetime, tmsi1 text, tmsi2 text, imsi text, imsicountry text,
imsibrand text, imsioperator text, mcc integer, mnc integer, lac integer, cell
integer);")

def text_file(self, filename):
    txt = open(filename, "w")
    txt.write("stamp, tmsi1, tmsi2, imsi, imsicountry, imsibrand, imsioperator,
mcc, mnc, lac, cell\n")
    txt.close()
    self.textfilePath = filename

def mysql_file(self):
    import os.path
    if os.path.isfile('.env'):
        import MySQLdb as mdb
        from decouple import config
        self.mysql_con = mdb.connect(config("MYSQL_HOST"),
config("MYSQL_USER"), config("MYSQL_PASSWORD"),
config("MYSQL_DB"))
        self.mysql_cur = self.mysql_con.cursor()

    if self.mysql_cur:
        print("mysql connection is success :)")
    else:

```

```

        print("mysql connection is failed!")
        exit()
    else:
        print("create file .env first")
        exit()

    def output(self, cpt, tmsi1, tmsi2, imsi, imsicountry, imsibrand, imsioperator,
    mcc, mnc, lac, cell, now, packet=None):
        print(f"{str(cpt):7s} ; {tmsi1:10s} ; {tmsi2:10s} ; {imsi:17s} ;
    {imsicountry:16s} ; {imsibrand:14s} ; {imsioperator:21s} ; {str(mcc):4s} ;
    {str(mnc):5s} ; {str(lac):6s} ; {str(cell):6s} ; {now.isoformat():s}")

    def pfields(self, cpt, tmsi1, tmsi2, imsi, mcc, mnc, lac, cell, packet=None):
        imsicountry = ""
        imsibrand = ""
        imsioperator = ""
        if imsi:
            imsi, imsicountry, imsibrand, imsioperator = self.str_imsi(imsi, packet)
        else:
            imsi = ""
            now = datetime.datetime.now()
            self.output_function(cpt, tmsi1, tmsi2, imsi, imsicountry, imsibrand,
            imsioperator, mcc, mnc, lac, cell, now, packet)

        if self.textfilePath:
            now = datetime.datetime.now()
            txt = open(self.textfilePath, "a")
            txt.write(f"{str(now)}, {tmsi1}, {tmsi2}, {imsi}, {imsicountry}, {imsibrand},
    {imsioperator}, {mcc}, {mnc}, {lac}, {cell}\n")
            txt.close()

        if tmsi1 == "":
            tmsi1 = None
        if tmsi2 == "":

```

```

        tmsi2 = None

    if self.sqlite_con:
        self.sqlite_con.execute(
            u"INSERT INTO observations (stamp, tmsi1, tmsi2, imsi, imsicountry,
            imsibrand, imsioperator, mcc, mnc, lac, cell) " + "VALUES (?, ?, ?, ?, ?, ?, ?, ?,
            ?, ?, ?);",
            (now, tmsi1, tmsi2, imsi, imsicountry, imsibrand, imsioperator, mcc,
            mnc, lac, cell)
        )
        self.sqlite_con.commit()
        pass

    if self.mysql_cur:
        print("saving data to db...")

        query = ("INSERT INTO `imsi` (`tmsi1`, `tmsi2`, `imsi`, `mcc`, `mnc`,
        `lac`, `cell_id`, `stamp`, `deviceid`) VALUES (%s, %s, %s, %s, %s, %s, %s, %s,
        %s)")

        arg = (tmsi1, tmsi2, imsi, mcc, mnc, lac, cell, now, "rtl")
        self.mysql_cur.execute(query, arg)
        self.mysql_con.commit()

    def header(self):
        print(f'{"Nb IMSI":7s} ; {"TMSI-1":10s} ; {"TMSI-2":10s} ; {"IMSI":17s} ;
        {"country":16s} ; {"brand":14s} ; {"operator":21s} ; {"MCC":4s} ; {"MNC":5s} ;
        {"LAC":6s} ; {"CellId":6s} ; {"Timestamp":s}')

    def register_imsi(self, arfcn, imsi1="", imsi2="", tmsi1="", tmsi2="", p=""):
        do_print = False
        n = ""
        tmsi1 = self.str_tmsi(tmsi1)
        tmsi2 = self.str_tmsi(tmsi2)
        if imsi1:

```

```

        self.imsi_seen(imsi1, arfcn)
    if imsi2:
        self.imsi_seen(imsi2, arfcn)
    if imsi1 and (not self.imsi_to_track or imsi1[:self.imsi_to_track_len] ==
self.imsi_to_track):
        if imsi1 not in self.imsis:

            do_print = True
            self.imsis.append(imsi1)
            self.nb_IMSI += 1
            n = self.nb_IMSI
            if self.tmsis and tmsi1 and (tmsi1 not in self.tmsis or self.tmsis[tmsi1] !=
imsi1):

                do_print = True
                self.tmsis[tmsi1] = imsi1
            if self.tmsis and tmsi2 and (tmsi2 not in self.tmsis or self.tmsis[tmsi2] !=
imsi1):

                do_print = True
                self.tmsis[tmsi2] = imsi1

            if imsi2 and (not self.imsi_to_track or imsi2[:self.imsi_to_track_len] ==
self.imsi_to_track):
                if imsi2 not in self.imsis:

                    do_print = True
                    self.imsis.append(imsi2)
                    self.nb_IMSI += 1
                    n = self.nb_IMSI
                    if self.tmsis and tmsi1 and (tmsi1 not in self.tmsis or self.tmsis[tmsi1] !=
imsi2):

                        do_print = True

```

```

        self.tmsis[tmsi1] = imsi2
    if self.tmsis and tmsi2 and (tmsi2 not in self.tmsis or self.tmsis[tmsi2] !=
    imsi2):

        do_print = True
        self.tmsis[tmsi2] = imsi2

    if not imsi1 and not imsi2 and tmsi1 and tmsi2:
        if self.tmsis and tmsi2 in self.tmsis:

            do_print = True
            imsi1 = self.tmsis[tmsi2]
            self.tmsis[tmsi1] = imsi1
            del self.tmsis[tmsi2]

    if do_print:
        if imsi1:
            self.pfields(str(n), tmsi1, tmsi2, imsi1, str(self.mcc), str(self.mnc),
            str(self.lac), str(self.cell), p)
        if imsi2:
            self.pfields(str(n), tmsi1, tmsi2, imsi2, str(self.mcc), str(self.mnc),
            str(self.lac), str(self.cell), p)

    if not imsi1 and not imsi2:

        if self.tmsis and tmsi1 and tmsi1 in self.tmsis and "" != self.tmsis[tmsi1]:
            self.imsi_seen(self.tmsis[tmsi1], arfcn)
        if self.show_all_tmsi:
            do_print = False
            if tmsi1 and tmsi1 not in self.tmsis:
                do_print = True
                self.tmsis[tmsi1] = ""
            if tmsi1 and tmsi1 not in self.tmsis:

```



```

        do_print = True
        self.tmsis[tmsi2] = ""
    if do_print:
        self.pfields(str(n), tmsi1, tmsi2, None, str(self.mcc), str(self.mnc),
str(self.lac), str(self.cell), p)

```

```

def imsi_seen(self, imsi, arfcn):
    now = datetime.datetime.utcnow().replace(microsecond=0)
    imsi, mcc, mnc = self.decode_imsi(imsi)
    if imsi in self.imsistate:
        self.imsistate[imsi]["lastseen"] = now
    else:
        self.imsistate[imsi] = {
            "firstseen": now,
            "lastseen": now,
            "imsi": imsi,
            "arfcn": arfcn,
        }
    self.imsi_purge_old()

```

```

def imsi_purge_old(self):
    now = datetime.datetime.utcnow().replace(microsecond=0)
    maxage = datetime.timedelta(minutes=self.purgeTimer)
    limit = now - maxage
    remove = [imsi for imsi in self.imsistate if limit >
self.imsistate[imsi]["lastseen"]]
    for k in remove:
        del self.imsistate[k]

```

```

class gsmtap_hdr(ctypes.BigEndianStructure):
    _pack_ = 1

```

```

_fields_ = [
    ("version", ctypes.c_ubyte),
    ("hdr_len", ctypes.c_ubyte),
    ("type", ctypes.c_ubyte),
    ("timeslot", ctypes.c_ubyte),
    ("arfcn", ctypes.c_uint16),
    ("signal_dbm", ctypes.c_ubyte),
    ("snr_db", ctypes.c_ubyte),
    ("frame_number", ctypes.c_uint32),
    ("sub_type", ctypes.c_ubyte),
    ("antenna_nr", ctypes.c_ubyte),
    ("sub_slot", ctypes.c_ubyte),
    ("res", ctypes.c_ubyte),
]

def __repr__(self):
    return "%s(version=%d, hdr_len=%d, type=%d, timeslot=%d, arfcn=%d,
signal_dbm=%d, snr_db=%d, frame_number=%d, sub_type=%d,
antenna_nr=%d, sub_slot=%d, res=%d)" % (
        self.__class__, self.version, self.hdr_len, self.type,
        self.timeslot, self.arfcn, self.signal_dbm, self.snr_db,
        self.frame_number, self.sub_type, self.antenna_nr, self.sub_slot,
        self.res,
    )

def find_cell(gsm, udpdata, t=None):

    global mcc
    global mnc
    global lac
    global cell
    global country

```

```

global brand
global operator

if gsm.sub_type == 0x01:
    p = bytearray(udpdata)
    if p[0x12] == 0x1b:
        # FIXME
        m = hex(p[0x15])
        if len(m) < 4:
            mcc = m[2] + '0'
        else:
            mcc = m[3] + m[2]
        mcc += str(p[0x16] & 0x0f)

        m = hex(p[0x17])
        if len(m) < 4:
            mnc = m[2] + '0'
        else:
            mnc = m[3] + m[2]

        lac = p[0x18] * 256 + p[0x19]
        cell = p[0x13] * 256 + p[0x14]
        t.current_cell(mcc, mnc, lac, cell)

```

```

def find_imsi(udpdata, t=None):
    if t is None:
        t = imstracker

    gsm = gsmtap_hdr.from_buffer_copy(udpdata)

    if gsm.sub_type == 0x1:

```

```

    find_cell(gsm, udpdata, t=t)
else:
    p = bytearray(udpdata)
    tmsi1 = ""
    tmsi2 = ""
    imsi1 = ""
    imsi2 = ""
    if p[0x12] == 0x21:
        if p[0x14] == 0x08 and (p[0x15] & 0x1) == 0x1:

            imsi1 = p[0x15:][:8]

            if p[0x10] == 0x59 and p[0x1E] == 0x08 and (p[0x1F] & 0x1) ==
0x1:

                imsi2 = p[0x1F:][:8]
            elif p[0x10] == 0x4d and p[0x1E] == 0x05 and p[0x1F] == 0xf4:

                tmsi1 = p[0x20:][:4]

            t.register_imsi(gsm.arfcn, imsi1, imsi2, tmsi1, tmsi2, p)

        elif p[0x1B] == 0x08 and (p[0x1C] & 0x1) == 0x1:

            tmsi1 = p[0x16:][:4]
            imsi2 = p[0x1C:][:8]
            t.register_imsi(gsm.arfcn, imsi1, imsi2, tmsi1, tmsi2, p)

        elif p[0x14] == 0x05 and (p[0x15] & 0x07) == 4:

            tmsi1 = p[0x16:][:4]
            if p[0x1B] == 0x05 and (p[0x1C] & 0x07) == 4:
                tmsi2 = p[0x1D:][:4]
            else:

```

```

        tmsi2 = ""

        t.register_imsi(gsm.arfcn, imsi1, imsi2, tmsi1, tmsi2, p)

    elif p[0x12] == 0x22:
        if p[0x1D] == 0x08 and (p[0x1E] & 0x1) == 0x1:

            tmsi1 = p[0x14:][:4]
            tmsi2 = p[0x18:][:4]
            imsi2 = p[0x1E:][:8]
            t.register_imsi(gsm.arfcn, imsi1, imsi2, tmsi1, tmsi2, p)

def udpserver(port, prn):
    sock = socket.socket(socket.AF_INET, socket.SOCK_DGRAM)
    server_address = ('localhost', port)
    sock.bind(server_address)
    while True:
        udpdata, address = sock.recvfrom(4096)
        if prn:
            prn(udpdata)

def find_imsi_from_pkt(p):
    udpdata = bytes(p[UDP].payload)
    find_imsi(udpdata)

if __name__ == "__main__":
    imstracker = tracker()
    parser = OptionParser(usage="%prog: [options]")
    parser.add_option("-a", "--alltmsi", action="store_true",
dest="show_all_tmsi", help="Show TMSI who haven't got IMSI (default : false)")
    parser.add_option("-i", "--iface", dest="iface", default="lo", help="Interface

```

```

(default : lo)")
    parser.add_option("-m", "--imsi", dest="imsi", default="", type="string",
help='IMSI to track (default : None, Example: 123456789101112 or "123 45
6789101112)")')
    parser.add_option("-p", "--port", dest="port", default="4729", type="int",
help="Port (default : 4729)")
    parser.add_option("-s", "--sniff", action="store_true", dest="sniff", help="sniff
on interface instead of listening on port (require root/suid access)")
    parser.add_option("-w", "--sqlite", dest="sqlite", default=None, type="string",
help="Save observed IMSI values to specified SQLite file")
    parser.add_option("-t", "--txt", dest="txt", default=None, type="string",
help="Save observed IMSI values to specified TXT file")
    parser.add_option("-z", "--mysql", action="store_true", dest="mysql",
help="Save observed IMSI values to specified MYSQL DB (copy .env.dist to .env
and edit it)")

(options, args) = parser.parse_args()

if options.sqlite:
    imsitracker.sqlite_file(options.sqlite)

if options.txt:
    imsitracker.text_file(options.txt)

if options.mysql:
    imsitracker.mysql_file()

imsitracker.show_all_tmsi = options.show_all_tmsi
imsi_to_track = ""
if options.imsi:
    imsi = "9" + options.imsi.replace(" ", "")
    imsi_to_track_len = len(imsi)
    if imsi_to_track_len % 2 == 0 and imsi_to_track_len > 0 and
imsi_to_track_len < 17:
        for i in range(0, imsi_to_track_len - 1, 2):

```

```

        imsi_to_track += chr(int(imsi[i + 1]) * 16 + int(imsi[i]))
    imsi_to_track_len = len(imsi_to_track)
    else:
        print("Wrong size for the IMSI to track!")
        print("Valid sizes :")
        print("123456789101112")
        print("1234567891011")
        print("12345678910")
        print("123456789")
        print("1234567")
        print("12345")
        print("123")
        exit(1)
    imsitracker.track_this_imsi(imsi_to_track)
    if options.sniff:
        from scapy.all import sniff, UDP
        imsitracker.header()
        sniff(iface=options.iface, filter=f"port {options.port} and not icmp and udp",
        prn=find_imsi_from_pkt, store=0)
    else:
        imsitracker.header()
        udpserver(port=options.port, prn=find_imsi)

```

6.1.1 API Code: This code will connect frontend of the project with its backend. It will tell the front end that IMSI CATCHER has started and storing the IMSI number in its database and will give it a signal to start showing the output on GUI.

```

import json

import flask as fl;

from flask_socketio import SocketIO;

from flask_cors import CORS, cross_origin;

import mysql.connector

```

```

async_mode = None

app = flask.Flask(_name_)

socket_ = SocketIO(app, async_mode='eventlet',cors_allowed_origins="*")

cors = CORS(app, resources={r"/api/": {"Access-Control-Allow-Origin": ""}})

myDb      =      mysql.connector.connect(host='localhost',      user='root',
password='P@$$word01',database="imsi_catcher")

dbController = myDb.cursor();

# dbController.execute("DROP TABLE imsi_record")

dbController.execute("SHOW TABLES")

isTableExist = False;

for x in dbController:

    if x[0] != 'imsi_record' and isTableExist != True:

        isTableExist = False

    else:

        isTableExist = True

# tmsi1, tmsi2, imsi, imsicountry, imsioperator, mcc, mnc, lac, cell, now

if isTableExist == False:

    print("creating table")

    dbController.execute("CREATE TABLE imsi_record (id INT
AUTO_INCREMENT PRIMARY KEY, tmsi1 VARCHAR(255), tmsi2
VARCHAR(255), imsi VARCHAR(255), imsicountry VARCHAR(255), imsioperator

```



```
VARCHAR(255), imsioperator VARCHAR(255), mcc VARCHAR(255), mnc  
VARCHAR(255), lac VARCHAR(255), cell VARCHAR(255), now  
VARCHAR(255))")
```

```
@app.route('/get-records', methods=['Get'])
```

```
@cross_origin()
```

```
def check():
```

```
    sql = "SELECT * FROM imsi_record ORDER BY id DESC"
```

```
    dbController.execute(sql)
```

```
    data = dbController.fetchall()
```

```
    return fl.jsonify({
```

```
        "code": 1,
```

```
        "msg": 2,
```

```
        "data": data
```

```
    })
```

```
@app.route('/getbyimsi', methods=['POST'])
```

```
@cross_origin()
```

```
def getDataByIMSINumber():
```

```
    body = fl.request.get_data().decode('utf-8')
```

```
    body = json.loads(body)
```

```
    searchStr = body['search'];
```

```
    sql = "SELECT * FROM imsi_record WHERE imsi LIKE %s"
```

```
    val = (searchStr,)
```

```
    dbController.execute(sql, val)
```

```

    data = dbController.fetchall()

    print(data);

    return fl.jsonify({

        'code': 1,

        'data': data,

        'msg': 'data found!'

    })

# tmsi1, tmsi2, imsi, imsicountry, imsibrand, imsioperator, mcc, mnc, lac, cell, now

@app.route('/save', methods=['POST'])

@cross_origin()

def save():

    body = {

        'tmsi1': fl.request.values.get('tmsi1'),

        'tmsi2': fl.request.values.get('tmsi2'),

        'imsi': fl.request.values.get('imsi'),

        'imsicountry': fl.request.values.get('imsicountry'),

        'imsibrand': fl.request.values.get('imsibrand'),

        'imsioperator': fl.request.values.get('imsioperator'),

        'mcc': fl.request.values.get('mcc'),

        'mnc': fl.request.values.get('mnc'),

        'lac': fl.request.values.get('lac'),

        'cell': fl.request.values.get('cell'),

        'now': fl.request.values.get('now'),

```

```

}

print("saving recode", body)

sql = "INSERT INTO imsi_record (tmsi1, tmsi2, imsi, imsicountry, imsibrand,
imsioperator, mcc, mnc, lac, cell, now) VALUES (%s, %s, %s, %s, %s, %s, %s, %s, %s,
%s, %s, %s)"

val = (body['tmsi1'], body['tmsi2'], body['imsi'], body['imsicountry'],
body['imsibrand'], body['imsioperator'], body['mcc'], body['mnc'], body['lac'],
body['cell'], body['now'])

dbController.execute(sql, val)

myDb.commit()

socket_.emit('hello', 'refresh')

return fl.jsonify({"success": True, 'code': 1, "msg": 'data-saved'})

@socket_.on('message')

def handle_message(data):

    print(data)

    socket_.emit("hello", "handshak too")

if(__name__ == '__main__'):

    socket_.run(app, debug=True)

```

6.2 Using GSMTAP Library:

Intercepting GSM traffic using a GSMTAP library involves the following steps:

Connect to the SDR device: The first step is to connect to the Software-Defined Radio (SDR) device. This is usually done using a library like PyBombs or GNU

Radio Companion.

Configure the SDR device: Once connected to the SDR device, you need to configure it to capture GSM traffic. This is done by setting the appropriate frequency, gain, and other parameters.

Capture the GSM traffic: Once the SDR device is configured, you can start capturing GSM traffic. This is usually done by setting up a flow graph that captures the traffic and then processing it using a signal processing library like GNURadio.

Decode the GSM traffic: Once the GSM traffic is captured, you need to decode it to extract useful information. This can be done using a library like libosmocore or pySim.

Analyze the decoded traffic: Finally, you can analyze the decoded traffic to extract information like IMSI numbers, SMS messages, and voice calls.

6.2.1 Setup and Initialization:

First, you will need to initialize the HackRF jammer and RTL-SDR components and set up the required parameters. For example, you can use the pyrtlsdr and hackrf libraries to initialize the components.

6.2.2 Signal Processing and Capture:

Next, you can use the GSM library to capture and decode the downgraded signal. For example, you can use the gsmtap() function to capture and decode the signal.

6.2.3 User Interface

A user interface can be implemented using a REACT APP and FLASK.

6.2.3.1 App.css Code:

```
.App {
```

```
text-align: center;
background: url(../public/ISMI.jpg);
background-repeat: no-repeat;
}
```

```
.wrapper {
display: flex;
flex-direction: column;
}
```

```
.page-header {
display: flex;
flex-direction: column;
flex-grow: 1;
}
```

```
.title {
width: 100%;
color: #c6e5ff;
font-size: 55px;
font-family: sansrif;
font-weight: bolder;
letter-spacing: 10px;
}
```

```
.searcher-container{
display: flex;
justify-content: end;
align-items: center;
}
```

```
.search-inp{
width: 400px;
margin-right: 10px;
}
```

```
.search-btn{
width: 100px;
}
```

```
.table-container{
margin: 10px;
height: 80vh;
overflow-y: scroll;
color: #ffff !important;
}
```

```
th{
color: #ffff !important;
}
```

```
.not-found-container{
width: 100%;
```

```

    display: flex;
    justify-content: center;
  }

.App-logo {
  height: 40vmin;
  pointer-events: none;
}

@media (prefers-reduced-motion: no-preference) {
  .App-logo {
    animation: App-logo-spin infinite 20s linear;
  }
}

.App-header {
  background-color: #282c34;
  min-height: 100vh;
  display: flex;
  flex-direction: column;
  align-items: center;
  justify-content: center;
  font-size: calc(10px + 2vmin);
  color: white;
}

.App-link {
  color: #61dafb;
}

@keyframes App-logo-spin {
  from {
    transform: rotate(0deg);
  }

  to {
    transform: rotate(360deg);
  }
}

```

6.2.3.2 App.js Code:

```

import './App.css';
import 'bootstrap/dist/css/bootstrap.min.css';
import Button from 'react-bootstrap/Button';
import Form from 'react-bootstrap/Form';
import Table from 'react-bootstrap/Table';
import { useEffect, useState } from 'react';

```

```

import axios from 'axios';
import io from 'socket.io-client';

const socket = io('http://127.0.0.1:5000');

function App() {
  let search = sessionStorage.getItem('search') !== " ?
  sessionStorage.getItem('search') : "
  let [record , setRecord] = useState([]);
  useEffect(()=>{
    if(!record.length){
      getAllRecords();
    }
    socket.on('connect', () => {
      console.log("connected");
    });
    socket.on('hello', (d) => {
      if(!search || search.trim() == ""){
        console.log("current v==>",search)
        getAllRecords()}
      // setRecord(()=> )
    });
  },[])

  useEffect(()=>{},{[search]})

  const getAllRecords = () =>{
    let data = [];
    axios.get('http://127.0.0.1:5000/get-records').then((res)=>{
      data = res.data.data;
      setRecord(()=> data);
    })
  }
}

```

```

const onSearchChange = (event) =>{
  search = event.target.value;
  // setSearch(()=> _s)
  sessionStorage.setItem('search', search);
  console.log(search);
  if(!search || search.trim() == ""){
    console.log("ppp")
    getAllRecords();
  }else{
    axios.post('http://127.0.0.1:5000/getbyimsi',{
      search: search
    }).then((res)=>{
      let data = res.data.data;
      setRecord(()=> data);
    })
  }
}

return (
  <div className="App">
    <div className="wrapper">
      <div className='page-header'>
        <div className='title'>IMSI-Catcher</div>
        <div className='searcher-container'>
          <div className='search-inp'>
            <Form>
              <Form.Control type="text" placeholder="Search IMSI number here..."
onChange={onSearchChange} />
            </Form>
          </div>
          {/* <div className='search-btn'>
            <Button variant="outline-primary" >Search</Button>
          </div> */}

```



```

</div>
</div>
<div className='table-container'>
  {
    record.length > 0 && <Table striped bordered hover>
    <thead>
      <tr>
        <th>#</th>
        <th>Tmsi1</th>
        <th>Tmsi2</th>
        <th>Imsi</th>
        <th>Imsi Country</th>
        <th>Imsi Brand</th>
        <th>Imsi Operator</th>
        <th>mcc</th>
        <th>mnc</th>
        <th>lac</th>
        <th>cell</th>
        <th>TimeStamp</th>
      </tr>
    </thead>
    <tbody>
      {
        record.map((ele, indR)=>{
          return (
            <tr>
              {
                ele.map((_e)=>{
                  return(
                    <th>{_e}</th>
                  )
                })
              }
            </tr>
          )
        })
      }
    </tbody>
  }

```

```

        )
    })
}
</tbody>
</Table>
}
{
    record.length <= 0 && <div className='not-found-container'>
        <h1>Data not found</h1>
    </div>
}

</div>
</div>
</div>
);
}

```

6.2.4 Security and Privacy

You can implement security and privacy features using encryption and anonymization techniques.

6.3 HackRF Customized Jammer:

The flowgraph defines a GNU Radio that generates a jamming signal using a HackRF software-defined radio. The signal is generated with the following parameters that can be adjusted using sliders in the graphical user interface (GUI):

- RF gain
- IF gain
- Center frequency

- Baseband gain
- Bandwidth

The code sets up the GUI using the PyQt5 library and defines the slider widgets using the RangeWidget class. It also creates an instance of the osmosdr.sink block, which is a GNU Radio block for transmitting data through an SDR. The osmosdr.sink is configured with the parameters set by the sliders in the GUI, and it is used to transmit the jamming signal generated by the flowgraph.

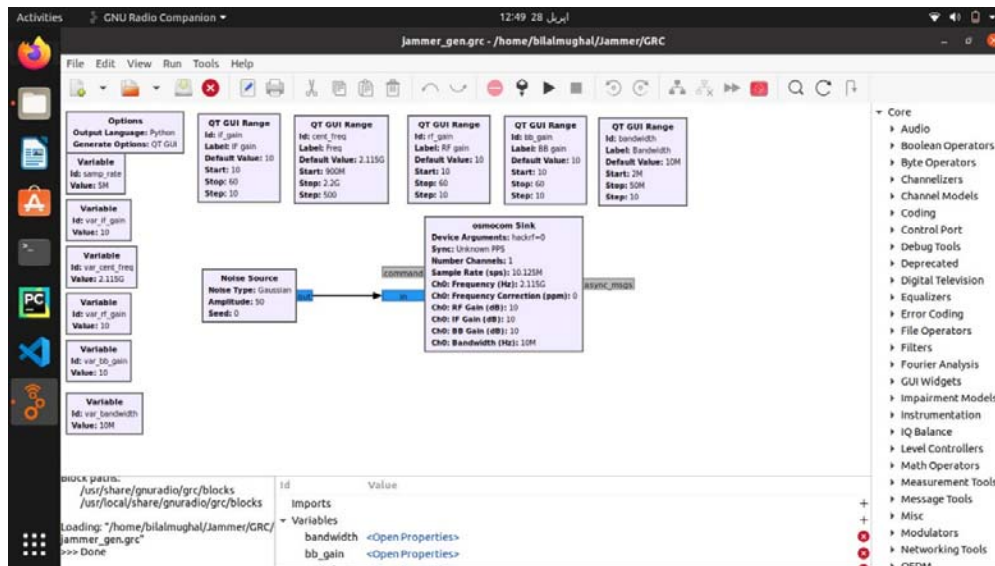


Figure 12: Flowgraph of Jammer

6.3.1 Code:

```
from distutils.version import StrictVersion
```

```
if __name__ == '__main__':
```

```
    import ctypes
```

```
    import sys
```

```

if sys.platform.startswith('linux'):
    try:
        x11 = ctypes.cdll.LoadLibrary('libX11.so')
        x11.XInitThreads()
    except:
        print("Warning: failed to XInitThreads()")

from gnuradio import analog
from gnuradio import gr
from gnuradio.filter import firdes
import sys
import signal
from PyQt5 import Qt
from argparse import ArgumentParser
from gnuradio.eng_arg import eng_float, intx
from gnuradio import eng_notation
from gnuradio.qtgui import Range, RangeWidget
import osmosdr
import time
from gnuradio import qtgui

class jammer_gen(gr.top_block, Qt.QWidget):

    def _init_(self):
        gr.top_block._init_(self, "Jammer Gen")
        Qt.QWidget._init_(self)
        self.setWindowTitle("Jammer Gen")
        qtgui.util.check_set_qss()
        try:
            self.setWindowIcon(Qt.QIcon.fromTheme('gnuradio-grc'))
        except:
            pass
        self.top_scroll_layout = Qt.QVBoxLayout()
        self.setLayout(self.top_scroll_layout)

```

```

self.top_scroll = Qt.QScrollArea()
self.top_scroll.setFrameStyle(Qt.QFrame.NoFrame)
self.top_scroll_layout.addWidget(self.top_scroll)
self.top_scroll.setWidgetResizable(True)
self.top_widget = Qt.QWidget()
self.top_scroll.setWidget(self.top_widget)
self.top_layout = Qt.QVBoxLayout(self.top_widget)
self.top_grid_layout = Qt.QGridLayout()
self.top_layout.addLayout(self.top_grid_layout)

self.settings = Qt.QSettings("GNU Radio", "jammer_gen")

try:
    if StrictVersion(Qt.qVersion()) < StrictVersion("5.0.0"):
        self.restoreGeometry(self.settings.value("geometry").toByteArray())
    else:
        self.restoreGeometry(self.settings.value("geometry"))
except:
    pass

#####
# Variables
#####
self.var_rf_gain = var_rf_gain = 10
self.var_if_gain = var_if_gain = 10
self.var_cent_freq = var_cent_freq = 2115000000
self.var_bb_gain = var_bb_gain = 10
self.var_bandwidth = var_bandwidth = 10e6
self.samp_rate = samp_rate = 5e6
self.rf_gain = rf_gain = var_rf_gain
self.if_gain = if_gain = var_if_gain
self.cent_freq = cent_freq = var_cent_freq
self.bb_gain = bb_gain = var_bb_gain
self.bandwidth = bandwidth = var_bandwidth

```

```

#####
# Blocks
#####
self._rf_gain_range = Range(10, 60, 10, var_rf_gain, 200)
self._rf_gain_win = RangeWidget(self._rf_gain_range, self.set_rf_gain, 'RF
gain', "counter_slider", float)
self.top_grid_layout.addWidget(self._rf_gain_win)
self._if_gain_range = Range(10, 60, 10, var_if_gain, 200)
self._if_gain_win = RangeWidget(self._if_gain_range, self.set_if_gain, 'IF
gain', "counter_slider", float)
self.top_grid_layout.addWidget(self._if_gain_win)
self._cent_freq_range = Range(900e6, 2200e6, 500, var_cent_freq, 200)
self._cent_freq_win = RangeWidget(self._cent_freq_range,
self.set_cent_freq, 'Freq', "counter_slider", float)
self.top_grid_layout.addWidget(self._cent_freq_win)
self._bb_gain_range = Range(10, 60, 10, var_bb_gain, 200)
self._bb_gain_win = RangeWidget(self._bb_gain_range, self.set_bb_gain,
'BB gain', "counter_slider", float)
self.top_grid_layout.addWidget(self._bb_gain_win)
self._bandwidth_range = Range(2e6, 50e6, 10, var_bandwidth, 200)
self._bandwidth_win = RangeWidget(self._bandwidth_range,
self.set_bandwidth, 'Bandwidth', "counter_slider", float)
self.top_grid_layout.addWidget(self._bandwidth_win)
self.osmosdr_sink_0 = osmosdr.sink(
    args="numchan=" + str(1) + " " + 'hackrf=0'
)
self.osmosdr_sink_0.set_time_unknown_pps(osmosdr.time_spec_t())
self.osmosdr_sink_0.set_sample_rate(bandwidth+bandwidth/80)
self.osmosdr_sink_0.set_center_freq(cent_freq, 0)
self.osmosdr_sink_0.set_freq_corr(0, 0)
self.osmosdr_sink_0.set_gain(rf_gain, 0)
self.osmosdr_sink_0.set_if_gain(if_gain, 0)
self.osmosdr_sink_0.set_bb_gain(bb_gain, 0)

```

```

self.osmosdr_sink_0.set_antenna("", 0)
self.osmosdr_sink_0.set_bandwidth(bandwidth, 0)
self.analog_noise_source_x_0 =
analog.noise_source_c(analog.GR_GAUSSIAN, 50, 0)

#####
# Connections
#####
self.connect((self.analog_noise_source_x_0, 0), (self.osmosdr_sink_0, 0))

def closeEvent(self, event):
    self.settings = Qt.QSettings("GNU Radio", "jammer_gen")
    self.settings.setValue("geometry", self.saveGeometry())
    event.accept()

def get_var_rf_gain(self):
    return self.var_rf_gain

def set_var_rf_gain(self, var_rf_gain):
    self.var_rf_gain = var_rf_gain
    self.set_rf_gain(self.var_rf_gain)

def get_var_if_gain(self):
    return self.var_if_gain

def set_var_if_gain(self, var_if_gain):
    self.var_if_gain = var_if_gain
    self.set_if_gain(self.var_if_gain)

def get_var_cent_freq(self):
    return self.var_cent_freq

def set_var_cent_freq(self, var_cent_freq):
    self.var_cent_freq = var_cent_freq

```

```

        self.set_cent_freq(self.var_cent_freq)

def get_var_bb_gain(self):
    return self.var_bb_gain

def set_var_bb_gain(self, var_bb_gain):
    self.var_bb_gain = var_bb_gain
    self.set_bb_gain(self.var_bb_gain)

def get_var_bandwidth(self):
    return self.var_bandwidth

def set_var_bandwidth(self, var_bandwidth):
    self.var_bandwidth = var_bandwidth
    self.set_bandwidth(self.var_bandwidth)

def get_samp_rate(self):
    return self.samp_rate

def set_samp_rate(self, samp_rate):
    self.samp_rate = samp_rate

def get_rf_gain(self):
    return self.rf_gain

def set_rf_gain(self, rf_gain):
    self.rf_gain = rf_gain
    self.osmosdr_sink_0.set_gain(self.rf_gain, 0)
    self.osmosdr_sink_0.set_gain(self.rf_gain, 1)
    self.osmosdr_sink_0.set_gain(self.rf_gain, 2)

def get_if_gain(self):
    return self.if_gain

```



```

def set_if_gain(self, if_gain):
    self.if_gain = if_gain
    self.osmosdr_sink_0.set_if_gain(self.if_gain, 0)
    self.osmosdr_sink_0.set_if_gain(self.if_gain, 1)
    self.osmosdr_sink_0.set_if_gain(self.if_gain, 2)

def get_cent_freq(self):
    return self.cent_freq

def set_cent_freq(self, cent_freq):
    self.cent_freq = cent_freq
    self.osmosdr_sink_0.set_center_freq(self.cent_freq, 0)

def get_bb_gain(self):
    return self.bb_gain

def set_bb_gain(self, bb_gain):
    self.bb_gain = bb_gain
    self.osmosdr_sink_0.set_bb_gain(self.bb_gain, 0)
    self.osmosdr_sink_0.set_bb_gain(self.bb_gain, 1)
    self.osmosdr_sink_0.set_bb_gain(self.bb_gain, 2)

def get_bandwidth(self):
    return self.bandwidth

def set_bandwidth(self, bandwidth):
    self.bandwidth = bandwidth
    self.osmosdr_sink_0.set_sample_rate(self.bandwidth+self.bandwidth/80)
    self.osmosdr_sink_0.set_bandwidth(self.bandwidth, 0)
    self.osmosdr_sink_0.set_bandwidth(self.bandwidth, 1)
    self.osmosdr_sink_0.set_bandwidth(self.bandwidth, 2)

def main(top_block_cls=jammer_gen, options=None):

```

```

    if StrictVersion("4.5.0") <= StrictVersion(Qt.qVersion()) <
StrictVersion("5.0.0"):
        style = gr.prefs().get_string('qtgui', 'style', 'raster')
        Qt.QApplication.setGraphicsSystem(style)
    qapp = Qt.QApplication(sys.argv)

    tb = top_block_cls()
    tb.start()
    tb.show()

    def sig_handler(sig=None, frame=None):
        Qt.QApplication.quit()

    signal.signal(signal.SIGINT, sig_handler)
    signal.signal(signal.SIGTERM, sig_handler)

    timer = Qt.QTimer()
    timer.start(500)
    timer.timeout.connect(lambda: None)

    def quitting():
        tb.stop()
        tb.wait()
        qapp.aboutToQuit.connect(quitting)
        qapp.exec_()

if __name__ == '__main__':

```

RESULTS AND DISCUSSION

7.1 Results:

A GSM IMSI catcher can be used for various purposes, both legal and illegal. In general, the results of using a GSM IMSI catcher depend on the purpose for which it is used. Here are some theoretical results of using a GSM IMSI catcher:

Intercepting and recording GSM communications: A GSM IMSI catcher can be used to intercept and record GSM communications, including voice calls and SMS messages. This can be useful for law enforcement agencies to monitor criminal activity, but it can also be used by criminals to spy on individuals.



The screenshot shows a web browser displaying a web application titled "IMSI-Catcher". The application has a search bar with the value "410 04 0722599972". Below the search bar is a table with the following columns: #, Tmsi1, Tmsi2, Imsi, Imsi Country, Imsi Brand, Imsi Operator, mcc, mnc, lac, cell, and TimeStamp. The table contains six rows of data, all from Pakistan, Zong, China Mobile, with the same MCC, MNC, LAC, and cell values.

#	Tmsi1	Tmsi2	Imsi	Imsi Country	Imsi Brand	Imsi Operator	mcc	mnc	lac	cell	TimeStamp
6			410 04 0808545224	Pakistan	Zong	China Mobile	410	04	7701	54872	2023-02-27 02:00:50.199945
5			410 04 0870058000	Pakistan	Zong	China Mobile	410	04	7701	54872	2023-02-27 02:00:16.317466
4			410 04 0784401004	Pakistan	Zong	China Mobile	410	04	7701	54872	2023-02-27 02:00:08.394636
3	0x26a5c5a0		410 04 0567490882	Pakistan	Zong	China Mobile	410	04	7701	54872	2023-02-27 02:00:01.770174
2			410 04 0722599972	Pakistan	Zong	China Mobile	410	04	7701	54872	2023-02-27 01:59:54.961086
1			410 04 0799846182	Pakistan	Zong	China Mobile	410	04	7701	54872	2023-02-27 01:59:39.244331

Figure 13: Storing IMSI Numbers to Database

Location tracking: A GSM IMSI catcher can also be used to track the location of a mobile device. This can be useful for tracking lost or stolen devices, but it can also be used by stalkers to track the location of their victims.

CellTower Locator

Track down a GSM/WCDMA/TD-LTE/FDD-LTE cell phone online using LAC (Location Area Code) and Cell ID. Track down a CDMA/CDMA2000 cell phone online using SID, NID and BID, and display its location on Maps.
* Indicates required.

MCC: [mcc list](#)

MNC: [mnc list](#)

Network: GSM

LAC*:

CellID*:

Ad served by Google

[Ad options](#) [Send feedback](#) [Why this ad?](#)

Figure 14: Locating GSM Tower

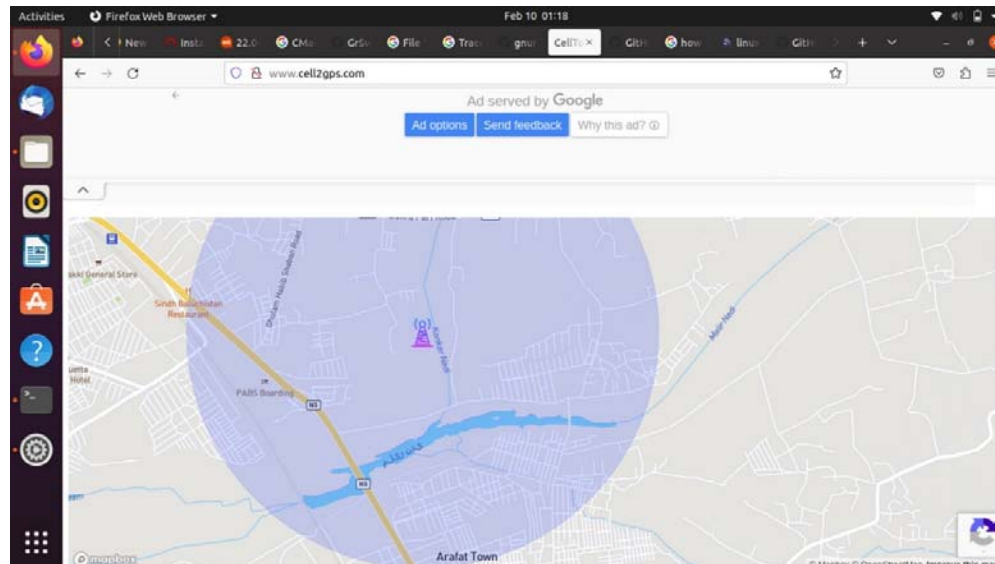


Figure 15: Cell Tower Location

Mass surveillance: A GSM IMSI catcher can be used for mass surveillance to monitor the communication of a large number of individuals in a particular area. This can be useful for intelligence agencies to gather information on potential threats, but it can also be a violation of privacy and civil liberties.

Man-in-the-middle attacks: A GSM IMSI catcher can be used to carry out man-in-the-middle attacks, where the attacker intercepts and alters the communication between two parties. This can be used to steal sensitive information, such as

passwords and banking details.

7.2 Discussion:

The IMSI catcher project that uses a special jammer to convert 4G signals into 2G and access them is an interesting and complex project that involves a deep understanding of wireless communication, signal processing, and security protocols.

In essence, an IMSI catcher is a device that can capture and intercept mobile phone signals in a particular area, allowing the operator to monitor and analyze the activity of mobile phones in that area. This project uses a HackRF jammer to downgrade 4G signals to 2G signals, making them easier to capture and access.

The HackRF jammer is a specialized device that can generate radio frequency (RF) signals in a specific frequency range, which can be used to jam or disrupt wireless signals. In this project, the HackRF jammer is used to downgrade 4G signals to 2G signals by generating a 4G signal in the same frequency range as the 4G signal. This allows the signal to be get jammed and downgrade for communication.

To capture and process the downgraded signal, the project uses RTL-SDR, a software-defined radio receiver that allows the user to capture and process radio signals in real-time. The downgraded signal is captured using RTL-SDR using the GSMTAP library, which is a Python library that provides functions for decoding GSM signals.

The project involves multiple components, including the HackRF jammer, RTL-SDR, and the GSMTAP library, as well as various signal processing and security protocols. The project requires a deep understanding of the protocols and algorithms used in wireless communication, as well as expertise in software-defined radio and Python programming.

In terms of security and privacy, the project raises concerns about the legality and ethics of intercepting mobile phone signals without the consent of the users. It is important to ensure that the project is used only for lawful purposes and that appropriate security measures are in place to protect the privacy of mobile phone users.

Overall, the IMSI catcher project that uses a special jammer to convert 4G signals into 2G and access them is a complex and challenging project that requires a deep understanding of wireless communication, signal processing, and security protocols. The project has significant implications for privacy and security and should only be used for lawful purposes.

CONCLUSIONS AND FUTURE WORK

The IMSI Catcher is a highly specialized and sophisticated device that provides a comprehensive solution for monitoring and tracking mobile phones in a specific area. The device has been designed and developed to meet the requirements and specifications of various organizations and governments, providing accurate and real-time monitoring capabilities. The project report has highlighted the technical aspects, design and development methodology, outcome and results, and future recommendations of the IMSI Catcher.

8.1 Future Work

There are several areas of future work and scope for IMSI catchers that could improve their capabilities and effectiveness. One potential area of focus is the development of more sophisticated and powerful software-defined radio (SDR) devices, which could improve the accuracy and sensitivity of IMSI catchers. Additionally, the use of machine learning and artificial intelligence could be explored to improve the accuracy of tracking and geolocation capabilities.

Another area of future work is the development of more advanced signal processing algorithms, which could help IMSI catchers filter out noise and interference more effectively, leading to more accurate intercepts of mobile phone communications. Additionally, the development of more advanced encryption and obfuscation techniques could make it more difficult for IMSI catchers to intercept and monitor mobile phone communications.

The use of IMSI catchers by law enforcement and intelligence agencies also raises important legal and ethical questions, such as concerns over privacy and the potential

for abuse. Future work in this area could focus on developing more transparent and accountable systems for the use of IMSI catchers, including greater oversight and regulation.

Finally, the ongoing evolution of mobile phone technology will continue to present new challenges and opportunities for IMSI catchers. As new wireless protocols and encryption techniques are developed, IMSI catchers will need to adapt and evolve in order to remain effective. Additionally, the ongoing proliferation of IoT devices and other connected technologies could provide new opportunities for IMSI catchers to intercept and monitor communications beyond traditional mobile phones.

In conclusion, there is significant potential for future work and scope in the development and use of IMSI catchers. By focusing on improving their technical capabilities, developing more transparent and accountable systems for their use, and adapting to ongoing changes in mobile phone technology, it may be possible to develop more effective and responsible systems for intercepting and monitoring mobile phone communications.

References

David, Paul Clark, ed. (2016) Expedient of SDR Fields, Vol 2 .1st edition.

RegistryMeadow, ULC

Qing Huang, Lin Yang. Inside theRadio: Defense and Attack Guide. 1st ed.Springer:

Singapore

Kozel Derek, GIU SDR Wikipedia, 1 July 2020,

https://wikipedia.ganurdio.com/Main_Page/index.php

Macheel Usman, Great Gadgets, Scott, 2020, <https://scottgreatgadgets.org/one/hackrf/>

Martino De Andrea. edition (2021) Initials to Modern Systems EW. UK: Arztech

House

Trevor Bihl J, "Feature Classifier and Selection Development for Frequency Radios"

(2016). Thesis and Dissections. 321.

<https://articlesscholar.afit.org/edu/321>

Donald A. Raising, RD-II, Usage and Exploitation of Hack-RF for Fequency Devices



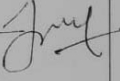
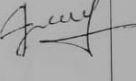

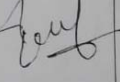
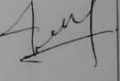
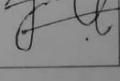
Verification and Classification Using VQIGRL Processing DS-ENG -04-12

Meeting Logs & Plagiarism Report



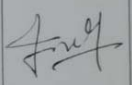
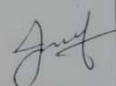
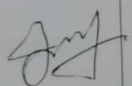

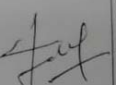

Project: IMSI Catcher FYP Meetings Log

Group Members: M Bilal Mughal, Afaq Ahmed, Faraz, Khan Nabi

Supervisor: Lt. Col. Imran Javed

S.No	Discussion Points	Date	Supervisor Signature
1	Discussing the structure, Authentication, Encryption and decryption of the GSM Network.	10/11/22	
2	Discussing the frequency bands of the 3G and 4G Network.	25/11/22	
3	Capturing the GSM Packets and analyzing them on Wireshark.	15/12/22	
4	Capturing the GSM packets using RTL-SDR and decryptions.	25/12/22	
5	Developing an algorithm of the Python for the IMSI catcher code.	11/1/23	
6	Discussion about GSMTAP library and its functionalities and Properties.	20/1/23	
7	Discussion about SQL Database and how to implement it with Project.	1/2/23	
8	Adding functionalities like saving result in TXT format and only sniffing options.	12/2/23	

FYP Meetings Log

S.No	Discussion Points	Date	Supervisor Signature
9.	Discussion about RTL SDR and Hack RF one and other properties and compatibility with project.	25/2/23	
10.	Introduction to GNU Radio and understanding the software for making the jammer.	7/3/23	
11.	Making the flowgraph of jammer and testing it by sending gaussian noise to jam the signals.	19/3/23	
12.	Discussion about React App and how to design the frontend of the project.	29/3/23	
13.	Creating an API that will connect the backend of the Project with frontend.	4/4/23	
14.	Testing the python code for GSM network and analysis of the result.	12/4/23	
15.	Testing the jammer to jam the frequency bands of 3G & 4G.	20/4/23	
16.	Final testing of the project for catching IMSI no and using jammer simultaneously.	27/4/23	

Turnitin Originality Report

Processed on: 30-Apr-2023 9:56 AM EDT

ID: 267272463

Word Count: 10668

Submitted: 3

IMSI By Bilal Janjua

Similarity Index
16%

Similarity by Source

Internet Sources: 14%
Publications: 7%
Student Papers: 11%

2% match (Internet from 01-Oct-2022)

<https://www.commons.wisc.edu/bitstream/handle/2117/323263/memoria.pdf?isAllowed=y&sequence=1>

2% match (Internet from 05-Feb-2023)

https://www.reddit.com/r/amabauradio/comments/ixegbh/i_am_a_noob_i_have_this_simple_flowgraph_that/

1% match (student papers from 30-May-2022)

[Submitted to Higher Education Commission Pakistan on 2022-05-30](#)

1% match (student papers from 07-Nov-2021)

[Submitted to Queensland University of Technology on 2021-11-07](#)

1% match (student papers from 04-Apr-2023)

[Submitted to Pontificia Universidad Catolica Madre y Maestra PUCMM on 2023-04-04](#)

1% match (student papers from 20-May-2021)

[Submitted to Bath College on 2021-05-20](#)

1% match (Internet from 15-Dec-2022)

<https://git.kent.fel.tuke.sk/vm503do/Product-Manager/commit/915f003452f9273a1880dfc29630d2ec921ef65.diff>

< 1% match (student papers from 17-May-2019)

[Submitted to Higher Education Commission Pakistan on 2019-05-17](#)

< 1% match (Internet from 30-Sep-2022)

<https://www.commons.wisc.edu/bitstream/handle/2117/330951/Thesis%20Document-approved-150.pdf?isAllowed=y&sequence=3>

< 1% match (Internet from 04-Oct-2022)

<https://github.com/Oros42/IMSI-catcher>

< 1% match (Internet from 13-Jan-2023)

https://github.com/Oros42/IMSI-catcher/blob/master/immediate_assignment_catcher.py

< 1% match (Internet from 20-Oct-2022)

<https://digital.library.txstate.edu/bitstream/handle/10877/6944/KRISHNAPPA-THESIS-2017.pdf?isAllowed=y&sequence=1>

< 1% match (Internet from 21-Oct-2022)

<https://digital.library.txstate.edu/bitstream/handle/10877/9872/SAMPATHKUMAR-THESIS-2020.pdf?isAllowed=y&sequence=1>

< 1% match (Internet from 21-Mar-2023)

<https://doi.mendelu.cz/ojs/doi/9900/02/9200.pdf>

< 1% match ()

[Laracy, Timothy Fennell. "A Software Defined Radio Interpolator for Passive Harmonic Transponders". UVM ScholarWorks. 2020](#)

< 1% match (student papers from 10-Jan-2023)

[Submitted to Kingston University on 2023-01-10](#)

< 1% match (Internet from 15-Apr-2023)

<https://lvsd.femto-st.fr/gitlab/bmarechal/gnuradio/commit/6ace6f52120aa62e8b789ea4d9f8e597a8a8672f.diff>

< 1% match (student papers from 10-Sep-2020)

[Submitted to Coventry University on 2020-09-10](#)

< 1% match (Internet from 15-Jan-2023)

<http://calderon.cud.uvigo.es/bitstream/handle/123456789/193/GarciaDominguez.pdf?isAllowed=y&sequence=1>

< 1% match (Internet from 13-Mar-2022)

<http://gitlab.slit.it/2021-210/2021-210/commit/b19f897adb740a6c05b5937963f90917e7f545a?expanded=1>

< 1% match (Internet from 21-Apr-2023)

<https://gitlab.slit.it/2021-049/2021-049/commit/5ca7d3f76c7f75603a9285a8070991c146b7478c?view=parallel>

< 1% match (Internet from 04-Jan-2023)

<https://mvsr.bc.edu/va/bitstream/handle/123456789/4923/jdeckehe.pdf?sequence=3>

< 1% match (Martin Sauter. "From GSM to LTE-Advanced Pro and 5G", Wiley, 2021)

Martin Sauter, "From GSM to LTE-Advanced Pro and 5G", Wiley, 2021
< 1% match (Internet from 08-Nov-2020) https://programtalk.com/python-examples/optparse.OptionParser/?page=2
< 1% match (student papers from 28-Nov-2021) Submitted to University of Greenwich on 2021-11-28
< 1% match ("Blockchain for Real World Applications", Wiley, 2022) "Blockchain for Real World Applications", Wiley, 2022
< 1% match (Aditya Gupta, "The IoT Hacker's Handbook", Springer Science and Business Media LLC, 2019) Aditya Gupta, "The IoT Hacker's Handbook", Springer Science and Business Media LLC, 2019
< 1% match (Internet from 05-Sep-2022) https://web.wj.edu/Pubs/E-orig/act/Available/E-orig/act-033120-032331/unrestricted/Anti-Jamming_Final_Report.pdf
< 1% match (Internet from 28-Oct-2021) https://zajuan.unizar.es/record/96736/files/TA7-TFG-2019-4169.pdf
< 1% match (student papers from 02-Apr-2023) Submitted to University of Sydney on 2023-04-02
< 1% match (Internet from 31-Jul-2015) http://business-card.me/
< 1% match () Scholar, Christopher, "RFI monitoring for the MeerKAT Radio Telescope", Department of Computer Science, 2015
< 1% match (Internet from 05-Apr-2023) http://kth.diva-portal.org/smash/get/diva2:1730799/FULLTEXT01.pdf
< 1% match (B B Harianto, M Rifai, A Irfansyah, Y Suprpto, "Design Indoor FM Communication Based on SDR and GNU Radio Using Validated Spectrum Analyzer", Journal of Physics: Conference Series, 2021) B B Harianto, M Rifai, A Irfansyah, Y Suprpto, "Design Indoor FM Communication Based on SDR and GNU Radio Using Validated Spectrum Analyzer", Journal of Physics: Conference Series, 2021
< 1% match (Michael Holloway, Chikezie Nwaoha, "Dictionary of Industrial Terms", Wiley, 2012) Michael Holloway, Chikezie Nwaoha, "Dictionary of Industrial Terms", Wiley, 2012
< 1% match (Internet from 15-Feb-2022) https://521academy.com/datascience/python/step-by-step-guide-to-setup-sql-with-python/
< 1% match (Internet from 13-Dec-2019) http://www.russells-world.com/projects/2014/10/06/sdr-2.html
< 1% match (Internet from 27-Apr-2023) https://fiosabub.org/aggregator/void%20%29%38/part-1-drunal-8-release-happy-birthday-dries-mega-episode?page=37
< 1% match (Internet from 03-Feb-2023) https://hacky.software/lima-sdr-fedora/
< 1% match (Internet from 06-Oct-2020) https://tools.eyes.uqcs.br/desastres/commit/2d849123b7e3f0b55dabae46e95d014477c8b5247w-1
< 1% match (student papers from 05-Jan-2019) Submitted to South University on 2019-01-05
< 1% match (Internet from 09-Jan-2023) https://evivankumar.home.blog/
< 1% match (student papers from 01-May-2009) Submitted to University of Huddersfield on 2009-05-01
< 1% match (Internet from 16-Jul-2021) https://gnosis.library.ucy.ac.cy/bitstream/handle/7/64251/Theocharis_Alexandros_Karathymios_2021.pdf
< 1% match (Internet from 17-Aug-2022) https://dspace.mit.edu/bitstream/handle/1721.1/144352/07991385-MIT.pdf?isAllowed=1&sequence=1
< 1% match (Internet from 11-Feb-2023) https://stackoverflow.com/questions/62879318/vuejs-error-in-render-twoerror-cannot-read-property-3-of-null
< 1% match (Internet from 22-Oct-2022) http://datapys.cs.ill.edu/publications/2015_CCPE-geb.pdf
< 1% match (Internet from 22-Jul-2012) http://www.digibay.com/vis/es/techzone/wireless/resources/glossary.html
< 1% match (publications) Arellano Delgado Alejandra Wendolína, "Desarrollo e implantación de un sitio Web de información para profesores y alumnos de la ENP No. 2 Erasmo Castellanos Quinto", TESUNAM, 2008
< 1% match (Internet from 20-Dec-2022)

6:57 PM

Turnitin - Originality Report - IMSI

https://elib.uni-stuttgart.de/handle/11682/9992/1/Osaro%20Gabriel.pdf
< 1% match (Internet from 06-Jan-2023) http://qrz.hac.gov.ck/igui/handle/123456789/9887/1/Impact%20of%20Conductivity%20Action%20Research%20by%20Secondary%20Schools
< 1% match (Internet from 12-Nov-2020) https://www.breakthroughfuel.com/blog/4-reasons-why-supply-chain-sustainability-is-profitability/
< 1% match (Internet from 18-Mar-2020) https://git.ubk.ac.at/csat2182/Web_Services_Server/commit/a2922c33be8455d297668fd6e73c483c37f21137?view=parallels&w=1
< 1% match (Internet from 22-Jul-2022) https://qibaa.osmocom.org/sdr/commit/4d79fd445259de15a165bee1d71d2338eb3b247?style=solid&whitespace=
< 1% match (Internet from 22-Feb-2023) https://repository.nyu.ac.za/handle/10294/11044/Wright_TM_of7isAllroad-v&sequence=1
< 1% match (Internet from 15-Feb-2019) http://ferendakapat634.blogspot.com/2015/
< 1% match (publications) Vázquez González Berenico, Hernández Mora Alejandro. "Adquisición de datos, procesamiento y reconocimiento de patrones para matrices de sensores SBW-UD". TESIUNAH, 2018