CHALMERS
UNIVERSITY OF TECHNOLOGY

UNIVERSITY OF GOTHENBURG

# Traffic Classification of 5G Packet Traces

Master's thesis in Computer science and engineering

JOSE ARMANDO TESEN MARAÑON

Department of Computer Science and Engineering
CHALMERS UNIVERSITY OF TECHNOLOGY
UNIVERSITY OF GOTHENBURG
Gothenburg, Sweden 2023

Master's thesis 2023

# Traffic Classification of 5G Packet Traces

JOSE ARMANDO TESEN MARAÑON

UNIVERSITY OF
GOTHENBURG

CHALMERS
UNIVERSITY OF TECHNOLOGY

Department of Computer Science and Engineering
Chalmers University of Technology
University of Gothenburg
Gothenburg, Sweden 2023

Traffic Classification of 5G Packet Traces
JOSE ARMANDO TESEN MARAÑON



Supervisor: Romaric Duvignau, Department of Computer Science and Engineering
Examiner: Risat Pathan, Department of Computer Science and Engineering

Master's Thesis 2023
Department of Computer Science and Engineering
Chalmers University of Technology and University of Gothenburg
SE-412 96 Gothenburg
Telephone +46 31 772 1000

Typeset in LaTeX
Gothenburg, Sweden 2023

Traffic Classification of 5G Packet Traces
José Armando Tesén Marañón
Department of Computer Science and Engineering
Chalmers University of Technology and University of Gothenburg

# Abstract

With the usage of mobile phones, privacy concerns have been a long-standing issue, and the recent advancement of 5G technology has only amplified these concerns. While encryption is a crucial method for protecting user privacy, studies indicate that machine learning techniques can identify web and mobile applications even though the traffic is encrypted.

To explore this problem, this project aims to investigate the potential for identifying mobile applications during encrypted communication on a 5G network. The project utilizes three machine learning models, namely k-Nearest Neighbors (k-NN), Random Forest, and Long Short-Term Memory (LSTM). To achieve this goal, various factors are analyzed, including the type of traffic, packet size, and timing information, to identify specific mobile applications.

This project's results show that it is possible to identify an app over a 5G network with an accuracy of 85% approximately, raising privacy concerns on communications over a 5G network. Under this context, this job updates the current State of Art regarding the private communications over an encrypted network, showing how privacy is vulnerable in 5G networks.

Keywords: 5G, Communication encryption, Mobile apps, Privacy, Security.

# Acknowledgements

I would like to thank my thesis supervisor Romaric Duvignau for his continuous support and guidance throughout my research.

Finally, I would like to thank my family for all the support during my studies.

José Armando Tesén Marañón, Gothenburg, 2023-08-01

# Contents

# List of Figures

# List of Tables

# 1

# Introduction

Navigating over the Internet has always been a common cyber security concern. With the rising usage of 5G, privacy is becoming an increasingly relevant topic. Historically, traffic encryption has helped solve this problem in a wired or wireless connection; however, big data and machine learning techniques have played an important role in solving different types of problems that were hard to solve. This brings the question of whether machine learning techniques can infer the type of application sending traffic in an encrypted channel.

Traffic classification over an encrypted is a challenging task because, in an encrypted channel, there is no clear-text information that can allow an easy way of filtering data. Many researchers have focused on analyzing security issues in 5G at the physical layer; meanwhile, others have mainly targeted their research on finding vulnerabilities at the encrypted Network Layer. This project's objective is to perform an analysis without considering any Layer 3 or higher data. Therefore, it would not be possible to deduce any information from the IP addresses or TCP/UDP port of the services, similar to passively eavesdropping on the link channel.

An important point to consider is that performing flow classification in an encrypted channel is more difficult without considering any Network Layer parameters. Because when analyzing packets using data from Layer 3 or higher, the packets still contain information about the IP addresses, and sometimes TCP/UDP ports in clear text. However, in an encrypted channel at the Link Layer, only the MAC addresses are in clear text, making it a challenge to recognize this type of traffic.

In this project, the focus is on communication over a 5G network. It is known that the communication between the user (endpoint) and the remote server is typically encrypted, so this project aims to investigate if it is possible to recognize this communication using machine learning techniques and evaluate the type of service based on its patterns.

This project aims to investigate possible privacy issues in communication protocols so that they can be improved. Also, it aims to design a traffic classification technique and evaluate the type of service based on its patterns, so it does not depend on whether it is working at the Link Layer or is encrypted.

The analysis is made on popular mobile applications such as Youtube, Facebook, Instagram, Twitter, LinkedIn, Spotify, TikTok, and Wikipedia because these services tend to have diverse network flow patterns.

## 1.1 Context

Currently, almost every mobile application is connected one way or another to the internet, using a Wi-Fi connection or a mobile network, such as 4G or 5G devices. Developers use encrypted channels for these communication standards to protect the end-user's privacy, so it can be assumed that nobody can identify what the user is doing over the network.

However, there have been some studies where this assumption has been proven wrong. Some researchers have focused their efforts on identifying video/streaming services over a Wi-Fi connection or a conventional internet connection, as shown by Reed et al. [1], or identifying mobile applications, as shown by Wang et al. [2]. While others have focused on revealing the weaknesses of the 5G network at their physical level [3].

Researchers have used different techniques to find their results, such as statistical analysis or machine learning techniques. Machine learning has demonstrated its effectiveness, with reported accuracies exceeding 99%, as illustrated by the work of Taylor et al. [4].

This project aims to develop a machine-learning model for identifying mobile applications on an encrypted 5G network without relying on any information about the IP addresses of the communicating parties. This analysis can also be observed in VPN or ToR connections, which have been examined in previous studies, such as the work of Ren et al. [5].

## 1.2 Problem description

As new technologies emerge, so do research threats; this time, on 5G networks, privacy is the concern to be evaluated. Several studies show that even though the communication between the end user and the application server is mostly encrypted by using HTTPS, such as YouTube, Facebook, etc., this service can still be inferred by finding its fingerprint using Machine Learning techniques; however, this analysis has mainly been done at in different types of networks, such as over Wi-Fi or a typical wired network connection; like the studies made by Wang et al. [2], Ren et al. [5], or Dimopoulos et al. [6].

At the Link Layer, the 5G communication between an end user and a base station is encrypted, so the source and destination of IP addresses are encrypted along with the data transferred over the network. This encryption makes the data traffic analysis even harder from an attacker's point of view, which is good for the end user; however, 5G still has some existing problems from its predecessor, 4G. For example, 5G is vulnerable to jamming, spoofing, and sniffing attacks, as pointed out by Lichtman et al. [3].

The lack of studies combining these two research domains, encrypted traffic analysis and avoiding the Layer 3 information on a 5G network, motivates this project to explore the possibilities of finding new vulnerabilities in modern 5G communications.

## 1.3 Goals and Challenges

The first goal is to investigate the possibility of recognizing mobile phone applications from encrypted traffic over a 5G communication network; without the need for Layer 3 data or higher, using data analysis and Machine Learning techniques. For instance, find out if it is possible to recognize different mobile application services such as YouTube, Facebook, or similar; and measure the accuracy of these machine learning models.

To initiate the analysis process, acquiring data is crucial. Therefore, this project aims to implement an environment to gather packets of 5G communication between an end user and a remote server. This project uses a 5G link, a firewall, and a sniffer. Thus, just the desired service is captured using existing tools, similar to the experimental setup made by Ajaeiya et al. [7].

## 1.4 Limitations

This project is limited to filtering or classifying mobile applications over a 5G network, not identifying any password or id account, or decrypting any data type. Also, this project is limited to the datasets that are identified as relevant to the current project.

Additionally, the goal is limited to finding the possibility to identify mobile applications over an encrypted channel, not the classification algorithm's efficiency. Hence, the project uses existing techniques and tools and adapts them to pursue the present goal, and not on developing new tools.

Finally, this project does not cover any analysis of application communication protocol; however, it must be mentioned that it could be useful since analyzing the protocols could reveal important details about how data is transmitted, processed, and protected.

## 1.5 Ethical Considerations

Regarding the project's scope, it is important to clarify that it is only focused on academic purposes, even though it may also be used for malicious ends, such as privacy invasion. So, it is recommended that the reader evaluates this project and its limitations before making any conclusion or decision based on the information that is presented. The objective is to show possible vulnerabilities in encrypted communication over a 5G network and develop awareness to improve the communications protocols.

## 1.6 Report Structure

This project is organized into five chapters:

In **Chapter 2**, the background is presented where is shown the definitions and related work of communication encryption identification.

**Chapter 3** presents the methods used in the project. This chapter describes the environment setup, the data gathering and processing, and the machine learning modeling used to develop the project.

The results are presented in **Chapter 4**, where a summary of the data collected and processed, and the accuracy of classification are shown.

The final section, **Chapter 5**, presents the discussion and conclusions of the thesis.

# 2

# Background

This chapter offers a foundation of basic definitions necessary for understanding the steps involved in the implementation process. While the level of detail provided may be limited, it is essential to be familiar with the terminology presented in this project as it facilitates the comprehension of subsequent sections.

## 2.1 5G

5G is the fifth generation of wireless network technology, succeeding the current 4G LTE standard. It is designed to offer faster internet speeds, lower latency, and greater connectivity than previous generations of wireless technology. With 5G, users can expect faster download and upload speeds, more stable connections, and the ability to connect more devices simultaneously [8].

One of the key features of 5G is its use of higher frequency radio waves than previous wireless technologies. This allows for greater capacity and faster data transfer rates, but it also requires more network infrastructure to be deployed to ensure reliable coverage. 5G networks use a combination of low-band, mid-band, and high-band frequencies to provide coverage over different distances and areas.

In addition to faster speeds and lower latency, 5G promises to support new applications and use cases, such as smart cities, autonomous vehicles, and virtual and augmented reality. This is possible by using advanced technologies such as beamforming and massive MIMO, which allow more precise and efficient use of the available spectrum.

5G also includes enhanced security features compared to previous wireless technologies. For example, 5G networks use more robust encryption and authentication methods, including features such as secure boot and remote stations to ensure the integrity of devices connecting to the network. Additionally, 5G networks support network-based security, which allows for more efficient and effective monitoring and management of security threats [9].

Overall, while 5G promises to offer significant improvements in network speed and connectivity, operators and organizations must take steps to ensure the security of these networks. This includes implementing strong encryption and authentication measures, monitoring potential security threats, and educating users about best practices for staying safe on 5G networks.

5G technology operates on radio frequency (RF) ranges between 24 and 86 GHz. The higher frequency ranges of 5G have been designated millimeter-wave (mmWave) frequencies. These frequencies can transmit data at extremely high speeds of up to 10 Gbps compared to 4G speeds, with a maximum speed of 1 Gbps.

The 5G frequency bands are divided into three main bands: low-band, mid-band, and high-band. Low-band ranges from 600 MHz to 700 MHz. Mid-band ranges from 2.5 GHz to 3.7 GHz. High-band ranges from 24 GHz to 40 GHz and in some regions, reaching up to 86 GHz.

The low-band range has a wider coverage area but lower speeds. Mid-band range provides a balance between coverage and speed, while the high-band range offers the highest speeds but has limited coverage distance as the signal has difficulty penetrating obstacles such as walls and buildings. This project is centered on the usage of mid-band range, since it deals with some video-related mobile applications.

5G technology's frequency range enables various applications such as autonomous vehicles, remote surgery, and virtual reality. Nevertheless, 5G technology is expected to continue expanding and evolving as more applications require higher speeds and lower latency.

## 2.2 Fingerprinting

An important part of the implementation is packet trace analysis and fingerprinting, which are used to analyze the encrypted network traffic. First, packet trace analysis is the process of examining the data packets that are transmitted over a network. This process can be done using a packet sniffer, such as Wireshark or tcpdump. Once packets are captured, they can be classified by size, source or destination address, service, delays, etc.

Fingerprinting is a technique that can be used to identify unique characteristics in encrypted communication. This information can be used to identify unencrypted messages or network communications. It can be used either for attackers or security purposes, such as detecting malicious software [10].

Fingerprinting can be achieved through various methods, such as port scanning, banner grabbing, protocol analysis, DNS interrogation, and web application fingerprinting. Active and passive are the two main ways to conduct fingerprinting. In active fingerprinting, custom packets are sent to the target network, and the responses are analyzed to determine their characteristics. On the other hand, passive fingerprinting involves sniffing network traffic to create a digital fingerprint of the network without sending any packets. Both methods have advantages and disadvantages, and the choice of which one to use depends on the specific circumstances and goals of the fingerprinting process [11].

### 2.2.1 Flow

In order to gain a comprehensive understanding of the network fingerprinting process, it is essential to establish well-defined descriptions of flows and bursts. This enables a more precise interpretation and analysis of the network behavior.

In the literature, Taylor et al. [4] define a flow as a unidirectional sequence of packets that are transmitted by a source to a destination over a communication network and that share the same characteristics, such as the same protocol, source IP address, and destination IP address. The analysis of data flow plays a vital role in identifying fingerprints within network traffic.

Link layer traffic flow is essential for understanding how data is moving through their network, the source of the traffic, the destination of the traffic, and how much traffic is being exchanged between devices. This information can be used to manage network performance, optimize bandwidth usage, and improve security.

In order to capture and analyze link layer traffic flows, specialized tools are used, such as flow analyzers or flow collectors. These tools collect data about each flow, including its start and end times, the amount of data transmitted, the packet sizes, and other metadata. This information is then used to identify traffic patterns, detect anomalies, and optimize network performance.

There are several types of traffic flows, including unidirectional, bidirectional, and multicast flows. Unidirectional flows move data packets in one direction from the source device to the destination device. Bidirectional flows move data packets in both directions, while multicast flows are used to send data packets to multiple devices simultaneously.

In this project, traffic flow is referred to as the volume of downloaded and uploaded bits (downlink and uplink) at the Link Layer from the end user's (mobile phone) point of view. A flow is a sequence of packets sharing the same pair of MAC addresses (bidirectional flow). Packets are represented as 4-tuples, the source and destination MAC address, the size of the packet, and the arrival time. By examining the flow of downlink and uplink traffic, we can find fingerprint patterns and identify mobile apps. And by analyzing the data link layer traffic flow, we can also gain insights into the behavior of network applications.

### 2.2.2 Burst

As stated by Taylor et al. [4], the burst is a way of analyzing network traffic by grouping packets that are sent in the same direction with a short time interval between them. This can help identify patterns and features of different applications or protocols that generate traffic.

Bursts can occur at different levels within a network, including at the application level, transport level, and network level. At the application level, bursts can occur due to spikes in user activity or increased demand for a particular service or application. At the transport level, bursts can be caused by the use of larger data packets or by the transmission of multiple packets at the same time. At the network level, bursts

can occur due to congestion, high bandwidth usage, or even denial-of-service (DDoS) attacks.

Peaks in Figure 2.1 correspond to sudden increases in activity, or bursts, in the packet trace extracted from Tiktok usage for 30 seconds approximately. These bursts can be seen as spikes in the graph, indicating periods of high link layer traffic. This figure represents the data rate of the downloaded (green) and uploaded (red) packets; this way of representing the link layer traffic flow is used across the rest of the document.

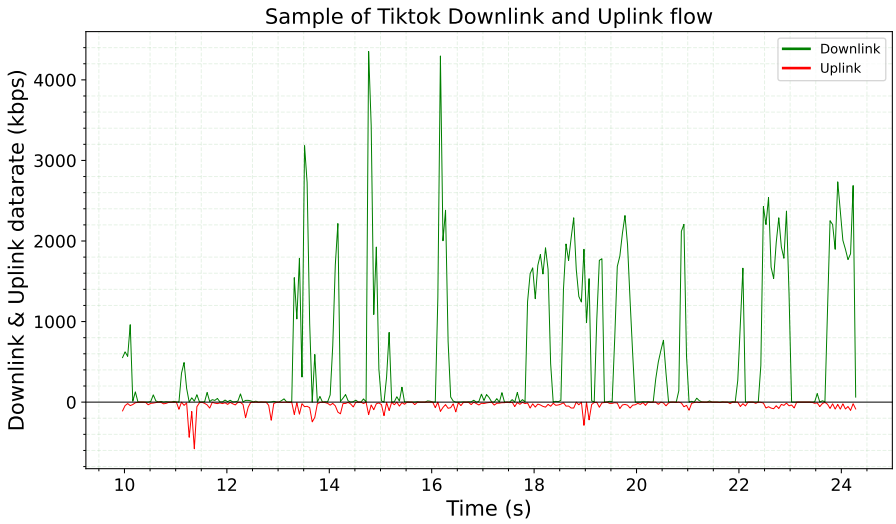


Figure 2.1: Example of burst Network Traffic.

Dividing the network traffic into bursts and analyzing only the sections containing genuine communication can significantly improve the efficiency of traffic analysis. Furthermore, analyzing burst peaks separately helps to have a more comprehensive understanding of the link layer traffic and usage trends. This can help identify improvement areas and optimize the machine learning model performance.

### 2.2.3 Bursts Detection

Burst detection is the process of identifying and analyzing bursts of traffic within a network. This process involves the use of specialized tools and techniques to monitor network traffic and detect sudden increases in data volume. The purpose of burst detection is to identify anomalies or abnormal behavior within network traffic, which can be an indicator of network problems, security threats, or extract fingerprints.

There are several techniques used to detect bursts in network traffic, including threshold-based detection, statistical analysis, and machine learning. Threshold-based detection involves setting a specific threshold value for network traffic and triggering an alert if the traffic volume exceeds that value. Statistical analysis involves analyzing the distribution of network traffic over time and identifying sudden changes in that distribution. Machine learning algorithms can also detect bursts by analyzing patterns and trends within network traffic.

Kleinberg's algorithm is used in this project. Kleinberg's algorithm is a burst detection algorithm that identifies periods in which a target event is uncharacteristically frequent or "bursty". It can be used to detect bursts in a continuous stream of events (i.e. video streaming) or in discrete batches of events (i.e. web surfing). It was introduced by Jon Kleinberg in 2002 [12].

The algorithm is based on the idea that a bursty event is likely to be followed by other bursty events and that they are likely to be related to each other. The algorithm uses a probabilistic model to identify these bursts and to distinguish them from random fluctuations in the data.

When the algorithm detects a deviation in the flow that exceeds a certain threshold, it signals a burst. The algorithm then looks for the end of the burst by searching for a point where the activity level returns to the background level.

## 2.3 Encryption

Encryption is the process of converting data into an unreadable or encrypted format to protect its confidentiality, integrity, and authenticity during transmission or storage. This is achieved by using an algorithm that converts the plaintext message into an encrypted ciphertext message, which can only be decrypted using a specific key or password.

There are two types of encryption: symmetric encryption and asymmetric encryption. In symmetric encryption, the same key is used for encryption and decryption, while in asymmetric encryption, different keys are used for encryption and decryption. Asymmetric encryption is often used for secure communication between two parties, while symmetric encryption is used for protecting data at rest [13].

Some popular encryption algorithms include:

- Advanced Encryption Standard (AES): This symmetric encryption algorithm is widely used for protecting data at rest. It is a block cipher that uses a fixed block size of 128 bits and key sizes of 128, 192, or 256 bits.
- RSA: This asymmetric encryption algorithm is widely used for secure communication between two parties. It uses a public key for encryption and a private key for decryption.
- Triple DES: This symmetric encryption algorithm is used for protecting data at rest. It is a block cipher that uses a key size of 168 bits.

Encryption is not invulnerable, and it can be vulnerable to attacks. Some common attacks include brute force attacks, in which an attacker tries to guess the encryption key by trying all possible combinations, and side-channel attacks, a topic that is expanded in section 2.7.

## 2.4 TLS and HTTPS

Before introducing the characteristics of mobile app network communication, it is important to establish the concepts of Transport Layer Security (TLS) and Hypertext Transfer Protocol Secure (HTTPS).

TLS is a cryptographic protocol that provides secure communication over the Internet. It was formerly known as SSL (Secure Socket Layer) until SSL 3.0, after which it was renamed TLS. TLS is used to establish a secure communication channel between two parties over the internet, such as a web browser and a web server. The most common application of TLS is HTTPS, the secure version of HTTP [14].

HTTPS is a combination of HTTP and TLS. It provides encrypted communication between a web browser and a web server. When a user accesses a website using HTTPS, the web browser first establishes a TLS connection with the server. The TLS protocol provides authentication, confidentiality, and integrity to the communication channel. This means that the data exchanged between the browser and the server is encrypted and cannot be intercepted or modified by any third party [15].

The TLS protocol works by using a combination of symmetric and asymmetric encryption techniques. When a TLS connection is established, the server and the client negotiate a shared secret key that is used to encrypt and decrypt the data exchanged between them. The shared secret key is generated using asymmetric encryption, which is a technique that uses a pair of keys, a public key, and a private key. The server sends its public key to the client, and the client uses it to encrypt a random number, which is sent back to the server. The server uses its private key to decrypt the random number, and both parties use the random number to generate the shared secret key.

TLS also provides authentication and integrity to the communication channel. The server sends its digital certificate to the client during the TLS handshake process. The digital certificate contains the server's public key and some other information, such as the server's domain name and the name of the organization that issued the certificate. The client verifies the authenticity of the digital certificate by checking whether it was issued by a trusted Certificate Authority (CA) and whether the domain name in the certificate matches the domain name of the server.

TLS handshake is an exchange of information between a client and a server that establishes a secure connection using cryptographic algorithms. The handshake involves negotiating the protocol version, selecting cryptographic algorithms, authenticating each other with digital certificates, and generating a shared secret key for encrypting subsequent messages [16]. Figure 2.2 shows a representation of the TLS handshake.

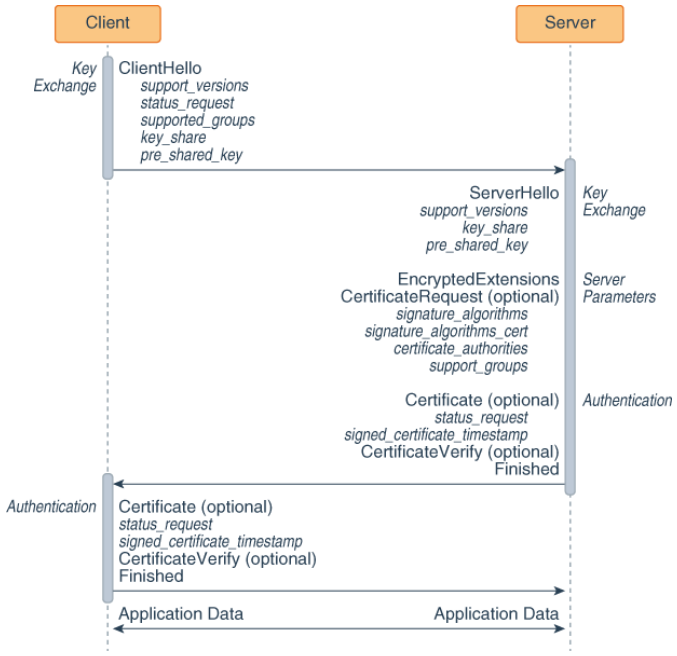


Figure 2.2: TLS 1.3 Handshake [16].

In summary, TLS and HTTPS provide a secure communication channel over the internet by using encryption, authentication, and integrity mechanisms. These mechanisms ensure that the data exchanged between the client and the server is protected from eavesdropping, tampering, and forgery.

## 2.5 Mobile App Network Communication

Even though the benefits of mobile apps are undeniable, there are several concerns surrounding data privacy and security in mobile app network communications. First, sensitive user information, such as login credentials, browsing history, location data, and contact lists, are often exchanged between the app and servers. This data interchange is necessary for delivering the app's services, but it also raises concerns about data privacy and the potential for this information to be at risk if it is not adequately protected.

Unsecured networks are a risk for privacy breaches as mobile app network communications might be intercepted on them, or in this project, encrypted communications, disclosing sensitive user data to malicious actors.

Mobile applications are different from typical web applications in how they download data from the servers, a web application downloads more data to provide the correct information to the end user, for instance, the HTML, CSS, and Javascript libraries. Instead, mobile applications have all the basic visual structures built-in into the app, so it only needs to download a fraction of that data, such as updated user posts as described by Al-Naami et al. [17], since mobile applications interact with the server by the use of APIs.

HTTPS is essential for securing data transmitted over the internet, including data exchanged between mobile apps and servers. Mobile apps communicate with servers using Application Programming Interfaces (API), which often rely on HTTPS to ensure the confidentiality and integrity of the data transmitted between the app and the server.

Mobile apps that communicate with servers using HTTP instead of HTTPS are vulnerable to various attacks, such as man-in-the-middle attacks, in which an attacker intercepts and modifies the data exchanged between the app and the server. Attackers can also intercept the app's authentication credentials, such as usernames and passwords, and use them to gain unauthorized access to the server.

Libraries and frameworks, such as OpenSSL, can be utilized by mobile app developers to incorporate HTTPS into their applications. OpenSSL, an open-source software library, offers various cryptographic functions and tools and implements the SSL and TLS protocols that enable secure communication on the Internet. With OpenSSL, data can be encrypted and decrypted, digital signatures can be created, and the validity of digital certificates can be verified [18].

## 2.6 Machine Learning

Machine learning is a subfield of artificial intelligence that focuses on developing algorithms and statistical models that can analyze and interpret large sets of data and make predictions or decisions based on that analysis without being explicitly programmed [19].

Machine learning aims to enable computers to learn from data, adapt to new situations, and improve performance on a specific task. To achieve this, machine learning algorithms are designed to automatically identify patterns and relationships within data and use that knowledge to make predictions or take actions [20].

There are three main types of machine learning: supervised learning, unsupervised learning, and reinforcement learning. In supervised learning, the algorithm is trained on labeled data, where the correct outputs are already known. In unsupervised learning, the algorithm is given unlabeled data and must find patterns and relationships on its own. In reinforcement learning, the algorithm learns by receiving feedback in the form of rewards or penalties for its actions [21].

Although this project employs machine learning models as tools, they are not the central focus. This section only briefly defines each technique used during the implementation stage. These definitions help readers understand the models used and their role in the project's methodology.

Data classification can be made manually so that the raw data can be classified by statistical mechanisms using network communication characteristics such as delay, size, or other network parameters. However, data classification can also be done using machine learning techniques that are precise and relatively easy to implement. For instance, k-Nearest Neighbors (k-NN) is a simple but powerful algorithm that can be used in several applications; on this occasion, it could be used to classify the

data collected for this project. Some other algorithms, such as Random Forest, Long short-term memory (LSTM) are also considered.

### 2.6.1 K-Nearest Neighbors (k-NN)

K-Nearest Neighbors (k-NN) is a machine learning algorithm that is used for classification or regression tasks. The k-NN algorithm works by finding the k closest data points in a training dataset to a new data point and then predicting the label or value of the new data point based on the labels or values of its nearest neighbors.

The "k" in k-NN represents the number of considered neighbors. For example, if k is equal to 3, then the algorithm will look for the 3 closest neighbors to the new data point in the feature space. Once the k neighbors are identified, the predicted label or value for the new data point is based on the majority class or average value of the K neighbors. The algorithm assigns the most common class among its k-nearest neighbors for classification and the average of the k-nearest neighbors for regression [22].

To find the k-Nearest Neighbors to a new data point, the algorithm needs a way to measure the distance between data points in the feature space to find the K-nearest neighbors to a new data point. The most commonly used distance metric is Euclidean distance, which is simply the straight-line distance between two points in space, which could be calculated as the equation 2.1.

$$\begin{aligned} d(x_i, x_j) &= \sqrt{\sum_{k=1}^{n} (x_{ik} - x_{jk})^2} \\ &= \sqrt{(x_{i1} - x_{j1})^2 + (x_{i2} - x_{j2})^2 + \cdots + (x_{in} - x_{jn})^2} \end{aligned} \tag{2.1}$$

One of the strengths of k-NN is that it is a non-parametric algorithm, which means that it does not make any assumptions about the underlying distribution of the data. This makes it more flexible than other machine learning algorithms that make assumptions about the data, such as linear or logistic regression. However, there are also some limitations of the k-NN algorithm. One of the main limitations is that it can be computationally expensive, especially for large datasets or high-dimensional feature spaces. This is because the algorithm has to calculate the distance between the new data point and every point in the training set to find the k-Nearest Neighbors. Figure 2.3 shows a simple representation of classification based on distances, where it shows how a sample is classified in a cluster.

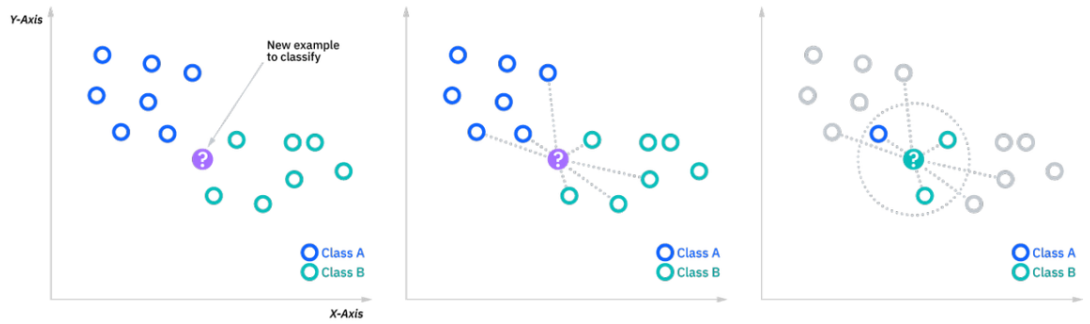


Figure 2.3: K-Nearest Neighbors model [23].

In the k-Nearest Neighbors model, the tuning parameter is typically the value of k, which represents the number of nearest neighbors to consider when making a prediction. Choosing the optimal value for k is critical in achieving good performance with k-NN.

The value of k can have a significant impact on the performance of the algorithm. If k is too small, the algorithm may be sensitive to noise or outliers in the data. The model may overfit the training data and fail to generalize well to new data. While if k is too large, the algorithm may over-generalize and miss essential patterns in the data, resulting in poor performance.

Other parameters that can be tuned in a k-NN model include:

The **distance metric** determines the distance between two data points in a dataset. It is a critical parameter that helps the algorithm identify the nearest neighbors of a given data point. Different distance metrics can be used, depending on the nature of the data and the problem being solved. For instance:

- Euclidean distance. Calculated as the square root of the sum of the squared differences between the corresponding features of two data points, as shown in the equation 2.1.
- Manhattan distance. Calculated as the sum of the absolute differences between the corresponding features of two data points, as shown in equation 2.2.

$$\begin{aligned} d(x_i, x_j) &= \sum_{k=1}^{n} (|x_{ik} - x_{jk}|) \\ &= |x_{i1} - x_{j1}| + |x_{i2} - x_{j2}| + \cdots + |x_{in} - x_{jn}| \end{aligned} \tag{2.2}$$

- Minkowski distance. A generalization of Euclidean distance and Manhattan distance. It takes a parameter "p" which determines the degree of the distance metric. When p=2, it is equivalent to Euclidean distance, and when p=1, it is equivalent to Manhattan distance, as shown in equation 2.3.

$$d(x_i, x_j) = (\sum_{k=1}^{n}(|x_{ik} - x_{jk}|^p)^{\frac{1}{p}}$$
$$= (|x_{i1} - x_{j1}|^p + |x_{i2} - x_{j2}|^p + \cdots + |x_{in} - x_{jn}|^p)^{\frac{1}{p}} \quad (2.3)$$

The **weighting** scheme parameter is the method used to assign weights to the neighboring data points during the classification or regression process. When making a prediction, the algorithm considers the distance (or similarity) between the new data point and the neighboring points. The weighting scheme can be used to adjust the importance of nearby data points based on their distance from the new point. There are several common weighting schemes used in k-NN; for example:

- Uniform weighting. All of the neighboring data points are given equal weight in the classification or regression process.
- Distance-weighted. Weights to neighboring points are based inversely on their distance from the new data point, with closer points being given more weight.
- Kernel-weighted. Weights are based on a kernel function, which can take into account the distance or other properties of the neighboring points.

Finally, **leaf size** determines the number of training samples assigned to a leaf node in the data structure. The leaf size parameter controls the granularity of the partitioning process. Smaller values of leaf size lead to finer partitions, which can capture more local variations in the data and increase the computation time and memory requirements. Larger values of leaf size, on the other hand, result in coarser partitions that are more efficient but may miss some important details. In general, the optimal leaf size depends on various factors, such as the size and dimensionality of the dataset, the required precision of the predictions, and the computational resources available.

### 2.6.2 Random Forest

Random Forest is an ensemble learning method for classification and regression that constructs a multitude of decision trees at training time and outputs the class that is the mode of the classes (classification) or the mean prediction (regression) of the individual trees. The key idea behind Random Forest is to combine multiple decision trees, each trained on a different subset of the training data and a random subset of the features, to reduce the model's variance and improve its generalization performance [24].

Random forests start by randomly selecting a subset of the data. This subset will be used to train the first decision tree. The tree is trained using a splitting algorithm, such as Classification and Regression Trees (CART). The splitting algorithm chooses the best feature to split the data on at each node in the tree.

Once the first decision tree is trained, it is used to make predictions on the remaining data. These predictions are then used to train the second decision tree. This process

is repeated for a large number of decision trees. The final prediction is the average of the predictions of all the trees.

Random forests have some advantages over other machine learning algorithms. They are relatively easy to implement and can be used on various data sets. They are also robust algorithms that can handle noise and outliers in the data, and they can be used for both classification and regression tasks. Figure 2.4 shows a representation of decision-making based on trees, where it is represented an input produces different responses on a tree.

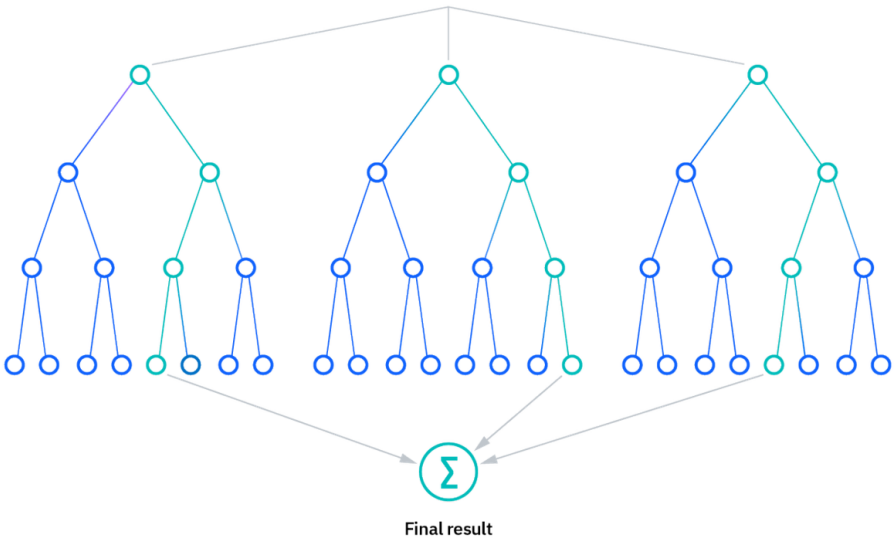


Figure 2.4: Random Forest model [25].

In a Random Forest model, several tuning parameters can be adjusted to optimize the model's performance. Some of the most important tuning parameters include:

**Number of estimators**: This parameter determines the number of trees to be built in the forest. Increasing this parameter can improve the model's performance but also increase the computational cost.

**Maximum depth**: This parameter determines the maximum depth of each tree in the forest. Increasing this parameter can improve the model's performance but also increase the risk of overfitting.

By tuning these parameters, we can adjust the model's balance and achieve better performance.

### 2.6.3 Recurrent Networks

Recurrent neural networks (RNNs) are a type of neural network that are designed to process sequential data, such as time series or natural language. Unlike feedforward neural networks, which process inputs in a fixed order, RNNs maintain an internal state that allows them to process sequences of variable length and to use past inputs to inform their processing of future inputs.

The key feature of RNNs is the use of recurrent connections, which allow information to be passed from one-time step to the next. This allows the network to maintain a memory of past inputs and to use this information to make predictions about future inputs [26]. Figure 2.5 shows a representation of a recurrent neural network.

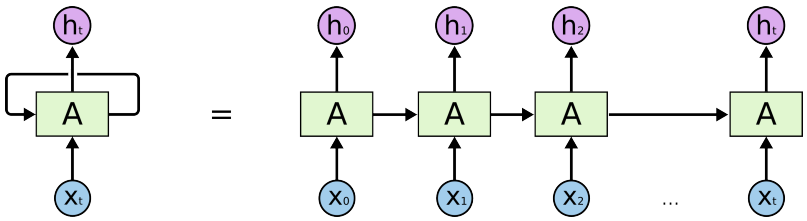


Figure 2.5: Recurrent Network model [26].

### 2.6.4 Long short-Term Memory

Long Short-Term Memory (LSTM) is a recurrent neural network (RNN) architecture designed to overcome the vanishing gradient problem in traditional RNNs. LSTMs use memory cells and gates to selectively retain and forget information over long periods, allowing them to capture long-term dependencies in sequential data [27].

LSTMs have become a popular architecture for various sequence prediction tasks such as natural language processing, speech recognition, and time series analysis.

An LSTM cell has three main components: a cell state, an input gate, an output gate, and a forget gate. The cell state is a long-term memory unit that can store information from previous time steps, while the gates control the flow of data into and out of the cell state.

The input gate determines which information from the current time step should be allowed into the cell state. It takes the current input and the previous hidden state as inputs and applies an activation function. This allows the network to store or discard information based on its relevance selectively.

The forget gate determines which information from the previous time step should be forgotten. It takes the current input and the previous hidden state as inputs and applies a sigmoid activation function, which produces a value between 0 and 1 for each element of the previous cell state. This value represents how much of the corresponding element should be retained or forgotten, and is multiplied by the previous cell state to determine the updated cell state.

The output gate determines which information from the cell state should be used to predict the current time step. It takes the current input and the previous hidden state as inputs and applies an activation function. This value represents how much of the corresponding element should be used for the output. The resulting value is the hidden state for the current time step, which is used to make the prediction.

These gates are learned during training and adaptively regulate the memory cell's contents, enabling the LSTM to selectively remember or forget information based on the input sequence.

LSTMs have proven effective in capturing long-term dependencies in sequential data, making them a valuable tool for many real-world applications. They are often combined with other deep learning models, such as convolutional neural networks (CNNs), for tasks such as image captioning and video analysis [28]. Figure 2.6 represents the LSTM cell architecture, where it is shown how the cells are connected between them.

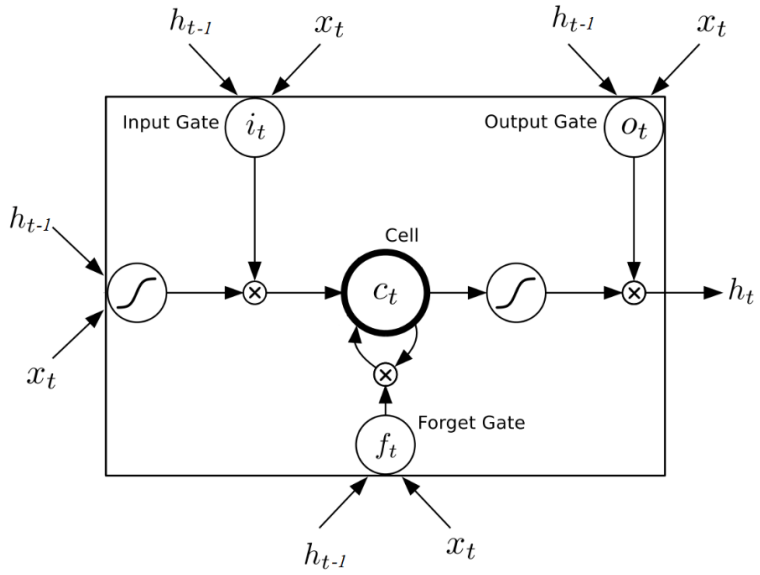


Figure 2.6: LSTM cell architecture [29].

Several tuning parameters can be adjusted for an LSTM model. Some of the most important ones include the following:

**Number of LSTM layers**: This parameter determines how many layers of LSTMs are stacked on top of each other. Adding more layers can improve the model's ability to capture complex relationships in the data and increase training time and potential overfitting.

**Number of nodes in each LSTM layer**: This parameter determines the number of LSTM nodes in each layer. The more nodes, the more complex the LSTM layer and the better it can capture patterns in the data. However, this can also lead to overfitting.

**Dropout Rate**: Dropout can be used to regularize the model and prevent overfitting. It involves randomly dropping out nodes from the LSTM layer during training, which makes the model more robust to different inputs. The dropout rate determines the probability of dropping out of each node.

**Learning rate**: This parameter determines the step size used in optimizing the neural network. A higher learning rate can lead to faster convergence during training but may result in overshooting the optimal weights for the model.

**Batch size**: This parameter determines the number of training samples used in each

step of the training. A smaller batch size may increase the amount of noise in the gradient calculation but can lead to faster convergence and more accurate results.

**Sequence length**: This parameter determines the size of the input sequences fed into the LSTM model. Longer sequences can capture more temporal dependencies in the data but can also lead to memory issues and slow training times.

**Activation function**: The activation function determines how the LSTM model processes input and generates output. The most common activation functions are sigmoid, tanh, and ReLU, and different activation functions are better suited for different types of data and models.

## 2.7 Security and Privacy

Several researchers have used this technique to extract information from encrypted communication traffic (not decrypt); for instance, Wang et al. [2] use side-channel information leaks to identify patterns in packets and detect smartphone applications over Wi-fi channels.

Finally, privacy refers to the ability of an individual or group to keep certain information, actions, or aspects of their lives out of public view and control. It is the right to be free from unwanted or unwarranted intrusions, surveillance, or interference in one's personal affairs.

In the context of cybersecurity, privacy refers to the protection of personal and sensitive information that is transmitted or stored electronically. This includes the confidentiality, integrity, and availability of data and protecting user identities and online activities.

Cybersecurity measures aim to prevent unauthorized access, disclosure, or misuse of personal information, such as financial data, medical records, and login credentials. Privacy protection in cybersecurity also includes using encryption, firewalls, and other security technologies to safeguard data and prevent hacking, phishing, and other cyber attacks.

In addition, privacy regulations and standards, such as the General Data Protection Regulation (GDPR) and the California Consumer Privacy Act (CCPA), provide legal protections for individuals' data and establish guidelines for organizations to follow in the collection, use, and disclosure of such data [30, 31].

Privacy in the context of cybersecurity refers to protecting personal information and safeguarding individuals' rights to control their own data and online activities.

Under this context, Side-channel attacks exploit the physical properties of cryptographic devices to circumvent traditional security measures and recover secret information. Side-channel attacks involve measuring and analyzing the physical information leaked during cryptographic computations to deduce intermediate values [32].

## 2.8 Related Work

Currently, there are some studies about analyzing encrypted traffic, especially at the Network Layer. These studies focus on finding fingerprints based on bandwidth consumption using machine learning techniques, showing exciting results, such as finding fingerprints of Video/Streaming usage and inferring its usage with very good accuracy, as shown by Papadogiannaki et al. [33], and Wang et al. [2]. Wang et al. focuses on the fact that despite the widespread use of encryption in mobile apps, it is still possible to make inferences about the apps being used based on patterns and characteristics of the encrypted data traffic. By analyzing the size, timing, and meta-data of the encrypted data traffic, the researchers could accurately identify the apps being used on the phone. They applied their method to 9 popular Android apps and achieved an average accuracy of 86%. The implications of this study highlight the importance of privacy concerns about app usage on mobile devices, even when encryption is used.

Additionally, some studies identified applications over an encrypted channel over a 3G and 4G channel based on fingerprints for each application, such as the one made by Ren et al. [5]. This paper proposes a novel method to identify apps from encrypted traffic using frequency distribution fingerprints, which can capture the statistical characteristics of traffic streams more effectively than previous methods. The paper also evaluates the system's performance on real-world datasets and shows that it can achieve high accuracy and efficiency even with multi-smartphone sources.

There are also some traffic analyses at the Link Layer, especially on Wi-Fi connections, as demonstrated by Reed et al. [1]. Reed et al.'s work proposes a novel method to identify DASH videos (Netflix) streamed over encrypted Wi-Fi connections by analyzing traffic patterns. The method can achieve high accuracy and a low false positive rate without decrypting or modifying packets. Still, it was only tested on a small dataset of 25 samples, and it is unknown if it can scale to a large dataset. This research is expanded in the work of Björklund et al. [34] and Duvignau [35], who were able to demonstrate that their methods can be applied to larger datasets, such as the Netflix dataset and the SVT Play dataset, with high accuracy.

A very important paper for this project is "AppScanner: Automatic Fingerprinting of Smartphone Apps From Encrypted Network Traffic" by Taylor et al. [4]. The paper proposes a tool called AppScanner that can automatically identify smartphone apps based on their network traffic, even when the traffic is encrypted. The tool uses a combination of machine learning and signature-based approaches to analyze network traffic and identify which app is responsible for it. Overall, the paper demonstrates the potential of machine learning and signature-based approaches for identifying apps based on their network traffic. This project takes this analysis as a reference, given that Wi-Fi connections are similar to 5G connections in some ways, such as the connection between the end user and the access point being encrypted.

Furthermore, Kamal et al. [36] have shown that it is possible to classify network traffic even if it is contained inside a VPN channel, so this gives the intuition that it could be possible in a 5G network. This is a challenging problem because most

Internet traffic is encrypted, and traditional methods of video identification rely on inspecting packet headers or payloads, which are not feasible for encrypted traffic. The paper uses a CNN to classify the traffic pattern plots generated from sniffing network traffic.

Finally, other studies have shown that 5G is vulnerable at the physical layer as it still carries some issues from 4G LTE that could be exploited and break the security or privacy of the end user. For instance, Piqueras et al. [37] analyze the physical downlink and uplink control channels and signals and identify the weakest links in 5G.

# 3

# Methods

This chapter discusses the methodology of the proposed solution to address the problem statement. This phase is a crucial stage in the project cycle as it involves the actual execution of the solution. The focus is on the implementation plan, which provides the roadmap for the solution in an organized and sequential manner. It also discusses the resources needed for successful implementation, including the technical considerations.

## 3.1 Methodology overview

The first step is to set up a lab environment. For this project, the data is collected from a mobile Android phone using a 5G router over a 5G network. Once the lab environment is set up, the next step is to gather and process the data for the machine learning training stage. Then, the processing involves cleaning, filtering, and detecting bursts to ensure that the data is of good quality and can be used for training the machine learning model. This project chooses three machine learning models, k-NN, Random Forest, and LSTM. Implementing the machine learning model involves setting up the model architecture and its parameters. The next step is to train the model using the processed data, this involves providing the model with input data and the corresponding output or label data, allowing it to learn from the examples and refine its internal parameters. Finally, the model is evaluated and compared to determine its performance. The evaluation results are then used to determine whether the model is effective and to identify the mobile apps.

As summary, the implementation process of this project can be represented as shown in Figure 3.1.

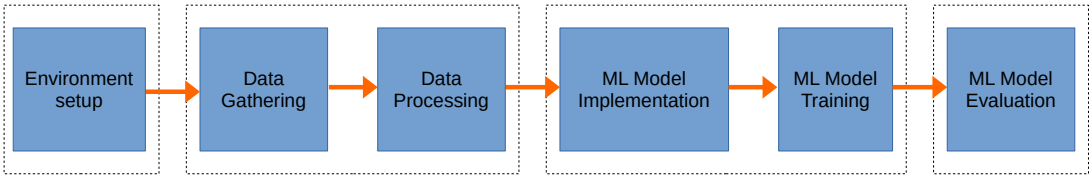


Figure 3.1: Project Process.

## 3.2 Intuition behind burst based traffic detection

Even though communication between a mobile phone and an application server is encrypted in several layers, such as HTTPS or VPNs, there are characteristics in the link layer traffic flow that could help infer which application a person is using.

For example, Figure 3.2 shows a link layer traffic flow sample of a user interacting with the Wikipedia mobile app. It is easy to notice that when interacting with this app, the traffic generated consists of small spikes of link layer traffic (around 1Mbps) because the nature of the application is mainly based on text.

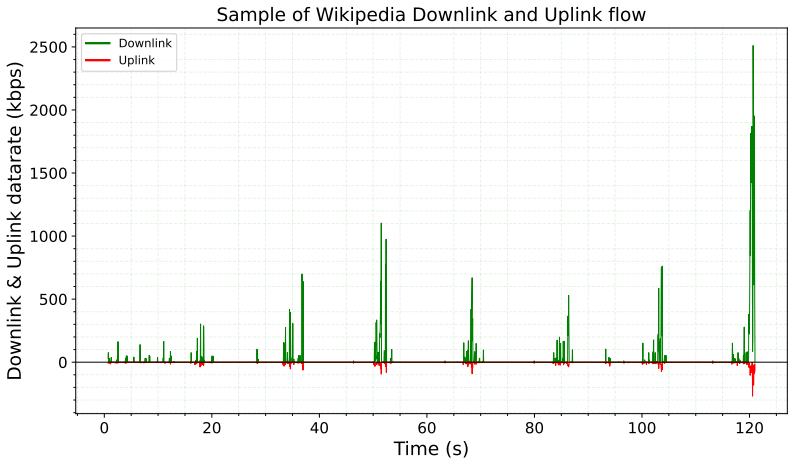


Figure 3.2: Sample of Wikipedia link layer traffic flow.

Meanwhile, the Tiktok mobile app is predominantly video-based, as is evidenced by the spikes in link layer traffic seen in Figure 3.3, which shows a sample link layer traffic flow of a user interacting with the app. In the example, these spikes reach a maximum of 6Mbps to 8Mbps, and more packets are transmitted than in the previous example.

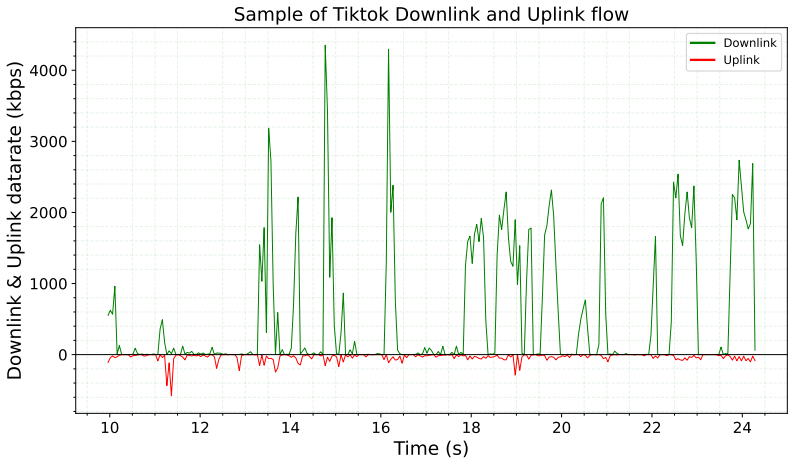


Figure 3.3: Sample of Tiktok link layer traffic flow.

Comparing these two figures, it is notable that there is more link layer traffic when a user interacts with the Tiktok application. For example, some characteristics that are evident for the human eye are the average, the maximum, and the total amount of transmitted bits.

Following this reasoning, it could be possible to identify which application is being used, even if the network traffic is encrypted. In the example, the difference between the network traffic from Tiktok and Wikipedia is easy to identify. However, it would be hard for a human to spot the difference between other applications like Facebook and Instagram, which could be less evident to the human eye.

Thus, the approach is to implement a machine learning model to identify mobile applications using statistical characteristics such as Standard Deviation, Mean, Median, Kurtosis, etc.

## 3.3 Environment Setup

The implementation involves setting up a test environment to collect network information from mobile apps. To accomplish this, a compact network circuit is created to facilitate gathering this data, as described in Section 3.4.

Three main devices are utilized during this stage, a mobile phone, a sniffer, and a 5G router or gateway. A mobile phone generates network traffic using well-known applications. This mobile phone is a regular Android phone, and the app interaction is made by human behavior and automated script-based movements.

In addition, the sniffer is strategically positioned to intercept and analyze all the data transmitted between the mobile phone and the 5G router. This is achieved by utilizing software that enables the sniffer to capture and store all the network traffic generated from the mobile phone in real-time, including programs like tcpdump, and scripts in Python. Additionally, the sniffer also performs other functions, such as blocking undesirable packets, such as ARP requests. To facilitate the setup process, the mobile phone's default gateway is the sniffer, ensuring that every bit of link layer traffic flows through it.

And last, a 5G Router enables communication to the Internet over a 5G network. This link is directly connected to the sniffer, and to avoid undesirable network traffic, the sniffer has a firewall policy for blocking. This 5G Router is an E-Lins H685 Series, as shown in Figure 3.4. It is a compact 3G, 4G, and 5G cellular gateway that connects a local area network, either wired or wireless (Wi-Fi) connection to a cellular network[1].

[1]https://www.e-lins.com/H685-4G-Router.html

Figure 3.4: E-Lins H685 Router[1].

The technical specifications of this device are flexible, as it can communicate over Frequency Ranges 1 and 2. Frequency Range 1 (FR1) covers the frequency range from 450 MHz to 6 GHz, while Frequency Range 2 (FR2) covers the millimeter-wave frequency range from 24.25 GHz to 52.6 GHz. This device can provide reliable and high-speed connectivity for a wide range of applications by operating on both frequency ranges. One of the key advantages of using millimeter-wave frequencies is their ability to provide a high data rate throughput over a 5G link. Finally, the circuit can be represented as shown in Figure 3.5.

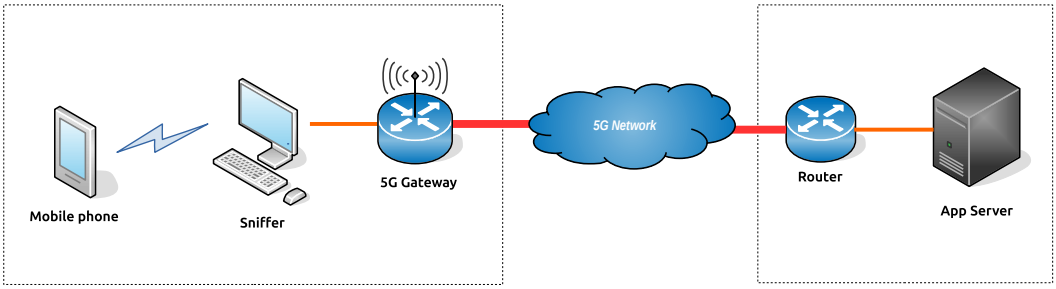


Figure 3.5: Traffic sniffing & Data Gathering.

The sniffer contains two virtual machines with an encrypted link between them as shown in Figure 3.6. This link uses an OpenVPN link that adds a second layer of encryption to the typical HTTPS/TLS communication. This encrypted link helps the data gathering by avoiding any other network traffic generated by the host PC and avoiding any information regarding the network traffic generated by the mobile phone.

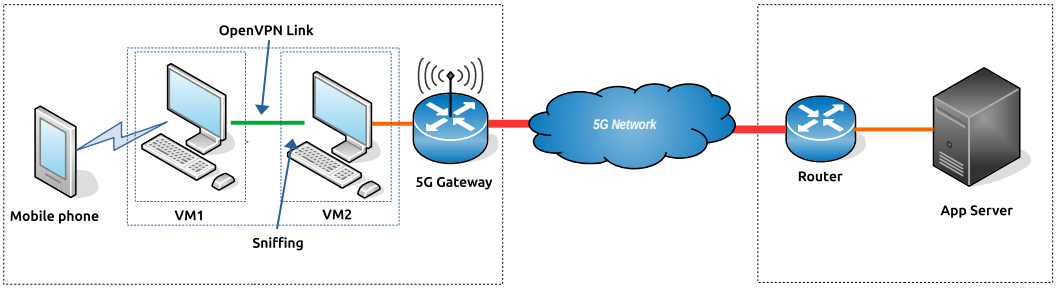


Figure 3.6: Traffic sniffing & Data Gathering (internal view).

## 3.4 Data Gathering

Once the lab environment is set up, the next step is to collect the network traffic. The process of data gathering and processing consists of the steps shown in Figure 3.7.

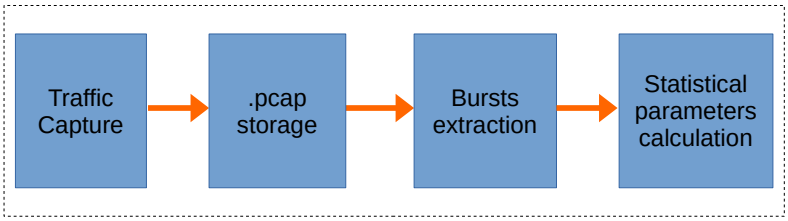


Figure 3.7: Data Gathering Process.

Before capturing the traffic network, it is important to choose mobile applications. The mobile applications used in this project are popular mobile applications from the Android Play Store, and they are chosen to have different or similar characteristics. For example, Facebook and Instagram are applications that are similar to one another. In contrast, applications like Youtube and Wikipedia are very different. This way, it is more likely to find different types of fingerprints in this stage of the process. Table 3.1 summarizes the mobile applications used in this project.

| **Mobile Application** | **Description** | **App Type** |
|---|---|---|
| Youtube | A video-sharing platform where users can upload, view, and share videos. | video-based |
| Instagram | A social media platform where users can share photos and videos, follow other users, and engage with content through likes and comments. | video/picture-based. |
| Facebook | A social networking platform where users can connect with friends and family, join groups, and share updates and media. | video/picture-based. |
| Twitter | A microblogging platform where users can post short messages (tweets) of up to 280 characters and follow other users for real-time updates. | mainly picture-based. |
| LinkedIn | A professional networking platform where users can connect with colleagues, showcase their skills and experience, and search for job opportunities. | mainly picture-based. |
| Tiktok | A social media platform where users can create and share short-form videos set to music or other audio. | video-based. |
| Spotify | A music streaming service that provides access to a vast library of songs and podcasts for on-demand listening. | music stream-based. |
| Wikipedia | A free, online encyclopedia that allows users to create, edit, and contribute to articles on a wide range of topics. | text-based. |

Table 3.1: List of tested mobile applications.

This project uses a Sony Xperia Z3 and a Motorola G7 (Android phones) to generate network traffic from the applications. The traffic is generated by scrolling and tapping arbitrary content from each application for approximately 60 seconds, using the build-in random search function for content of the application if possible. This scrolling and tapping generate network traffic from video and image downloads. For example, on Youtube, the user scrolls over the application, randomly selects a video, and starts watching it for approximately 60 seconds. Table 3.2 shows a summary of the application link layer traffic flow generation while capturing the network traffic.

| **Application** | **link layer traffic flow Generation** |
|---|---|
| Youtube | Scroll and play a random video. |
| Instagram | Scroll over the "Home" tab or "Search" tab. |
| Facebook | Scroll over the "News Feed" tab or the "Watch" tab (Reels or Videos). |
| Twitter | Scroll over the "For You" tab. |
| LinkedIn | Scroll over the "Home" tab. |
| Tiktok | Scroll over the "For you" tab and start watching any random video. |
| Spotify | Start playing a random song. |
| Wikipedia | Open a random Wiki from the random link generator: "https://en.wikipedia.org/wiki/Special:Random". |

Table 3.2: Application link layer traffic flow generation.

During this stage, the sniffer recollects all the traffic from the mobile phone, which is encrypted, so there is no information related to IP addresses or TCP ports. At this point, the user can only run one application at a time; thus, traffic collected only belongs to one application.

Then, all the network data is stored in a .pcap file, where the traffic collected is saved. A .pcap file is a data file containing the link layer traffic flow transmitted and received within a network.

In this scenario, a .pcap file contains only encrypted network traffic, so the information that can be extracted is the size of each transmitted packet, and the timing between packets. For example, Figure 3.8 shows a sample of packets transmitted and received using the Youtube application. Every dot represents the length of a packet downloaded (green) or uploaded (red). Even though this data could be used for the analysis, it is still hard to distinguish between different statistical features, such as mean, median, or maximum value. Therefore, the next step is to process this raw network traffic and present in a more usable format.

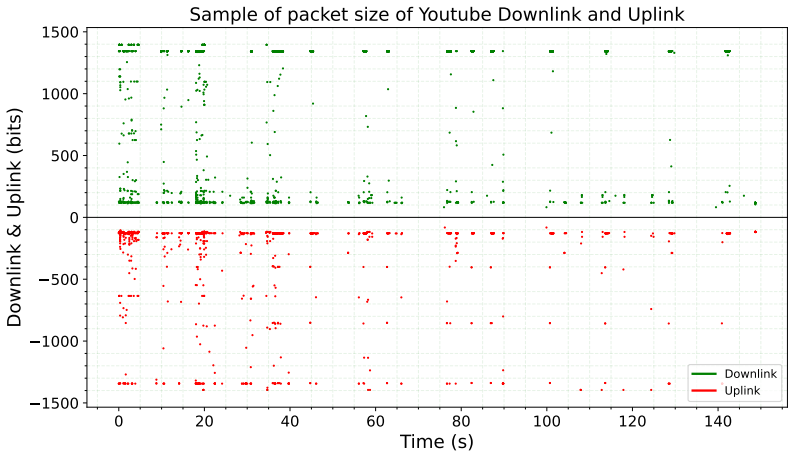


Figure 3.8: Network Traffic Gathering.

## 3.5 Data Processing

Following the initial data collection, the next step is to transform the .pcap link layer traffic flow based on packets into a flow based on total transmitted bits per second. For example, Figure 3.9 shows the same packet link layer traffic flow of Figure 3.8 in a way that it accumulates the number of bits transmitted every 50ms.

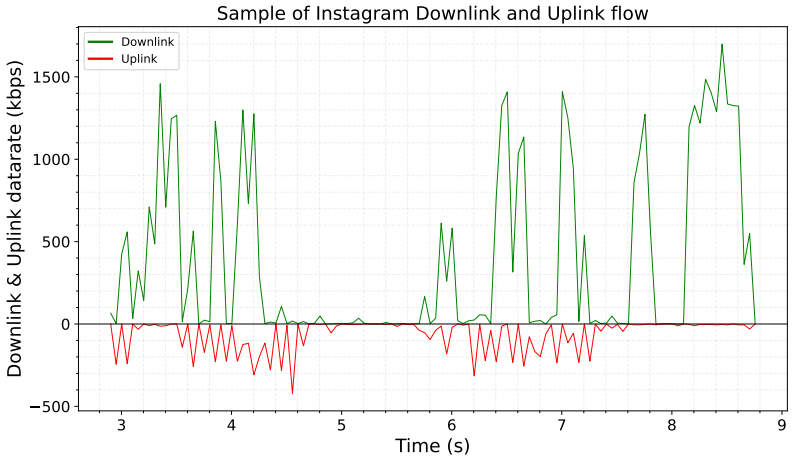


Figure 3.9: Representation of link layer traffic flow in kbits per second.

After that, the next step in analyzing the traffic data is to identify and extract traffic bursts for a more detailed analysis. This process involves using the Python library, bursts_detection[2], designed to detect bursts in large datasets. This package utilizes Kleinberg's burst detection algorithm, explained in subsection 2.2.2.

Additionally, during the burst detection process, any empty gaps (zero value gaps) between bursts in the link layer traffic flow are eliminated. Each burst time is subsequently followed by the next detected burst. This is done to prevent moments where the user is not interacting with the app from being included in the analysis. However, it is important to note that the original flow is not discarded, and both flows are considered during the final analysis. Figure 3.10 shows an example of a link layer traffic flow without zero values.

[2]http://github.com/nmarinsek/burst_detection

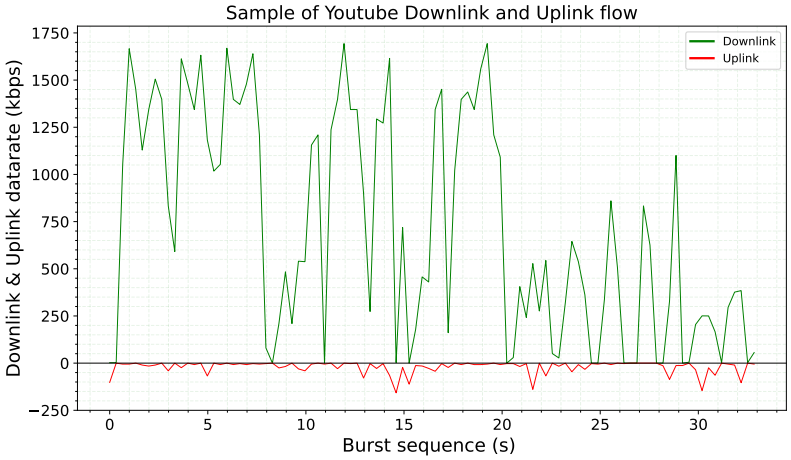


Figure 3.10: Link layer traffic flow without zero values.

For instance, Figure 3.11 presents a comparison of the link layer traffic flow and the detected bursts. The two graphs correspond to the same link layer traffic flow, only that the second one only contains the detected bursts. Each detected burst is surrounded by a rectangle with a different color, and after detection, each burst is put one after the next. Finally, each arrow connects each burst detected in the first plot to its corresponding position in the second.

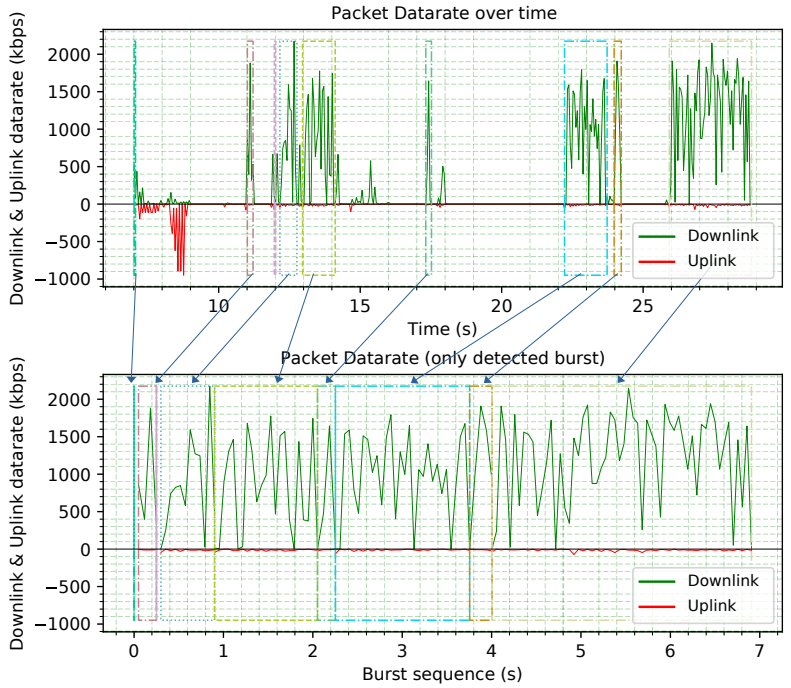


Figure 3.11: Burst detection process

This project uses k-NN, Random Forest, and LSTM to explore alternatives for traffic identification. The implementation of k-NN and Random Forest is similar to the work of Taylor et al. [4]; they extracted various statistical analysis parameters, including

Variance, Standard Deviation, Mean, and Kurtosis, before the actual classification. Table 3.3 shows the parameters considered for the implementation. These parameters are calculated from the complete flow and the filtered flow with no empty gaps.

| **Features (Downlink & Uplink)** | **Mathematical Expression** |
|---|---|
| Mean | $\mu = \frac{1}{N}\sum_{i=1}^{N} x_i$ |
| Variance | $\mathrm{Var}(X) = \sigma^2 = \frac{1}{N}\sum_{i=1}^{N}(x_i - \mu)^2$ |
| Standard Deviation | $\mathrm{SD}(X) = \sigma$ |
| Kurtosis | $\mathrm{K} = \kappa = \frac{\mu^4}{\sigma^4} - 3$ |
| Skew | $\mathrm{S} = \gamma = \frac{\mu_3}{\sigma^3}$ |
| Maximum | $\max_{1 \le i \le N}\{x_i\}$ |
| Minimum | $\min_{1 \le i \le N}\{x_i\}$ |
| 90% Percentile | $P_x = T(\frac{N * x}{100})$, where T contains all $x_i$ in ascending order. |
| 80% Percentile | |
| 70% Percentile | |
| 60% Percentile | |
| 50% Percentile | |
| 40% Percentile | |
| 30% Percentile | |
| 20% Percentile | |
| 10% Percentile | |
| Total amount of packets | $N$ |
| Total amount of transmitted bits | $\mu_N$ |

Table 3.3: Statistical Parameters.

In the end, these features are calculated for the complete link layer traffic flow and the flow without empty gaps. Then these two sets of features are concatenated for being processed in the machine learning model. This approach accomplishes better the classification results compared with a one-feature approach.

## 3.6 Model Parameters

This project tests three machine learning models, k-NN, Random Forest, and Long short-Term Memory. The goal is to compare different approaches and see if mobile app identification over a 5G network is possible.

a) k-NN

This machine learning model is simple, the main parameters are the number of neighbors, leaf size, and the type of evaluation distance, as explained in Section 2.6.1. So, the parameters used in this project are shown in Table 3.4.

| **Parameters** | **Values** |
|---|---|
| Number of nearest neighbors | 5, 25, 50, 75, 100, 150, 200, 250, 300 |
| Distance | Euclidean, Manhattan, Minkowski (p=3,4,5) |
| Leaf size | 2, 5, 10, 25, 30 |

Table 3.4: k-NN Parameters.

The design is done with the usage of the Python library Sklearn KNeighborsClassifier[3] that gives a set of features to implement the machine learning model.

b) Random Forest Classifier

This machine learning model is also simple, the main parameter is the number of trees or estimator, which means the number of possible outcomes evaluated by the model, as explored in Section 2.6.2. So, the parameters used in this project are shown in Table 3.5.

| **Parameters** | **Values** |
|---|---|
| Estimators | 5, 25, 50, 100, 150, 200, 250, 300 |
| Max Depth | 5, 10, 15, 20, 30, 50 |

Table 3.5: Random Forest Parameters.

Similar to k-NN, the machine learning model is made with the help of the Python library, Sklearn KNeighborsClassifier[4] that includes a set of customizable parameters as the number of estimators, max depth, etc.

c) LSTM

This model needs more tuning. since this model requires the design of neural network layers and the number of neurons on each layer. First, every sample is split into small chunks, so it can be fed into the model.

| **Parameters** | **Tentative model/values** | |
|---|---|---|
| Number of layer | 2 | 4 |
| 1st layer | LSTM: 64, 128, 256, 512, 1024 neurons | LSTM: 64, 128, 256, 512, 1024 neurons |
| 2nd layer | Dense (Output) | Repeat: 5 times |
| 3rd layer | | LSTM: 64, 128, 256, 512, 1024 neurons |
| 4th layer | | Dense (Output) |

Table 3.6: LSTM Parameters.

[3]https://scikit-learn.org/stable/modules/generated/sklearn.neighbors.KNeighborsClassifier.html
[4]https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.RandomForestClassifier.html

For this model, the implementation is done with the help of the Python library Tensorflow LSTM[5], along with other Tensorflow modules, it makes the design of the neural network model easier.

## 3.7 Evaluation and Testing

In order to evaluate the performance of the models, the complete dataset is divided into two distinct subsets. The first subset, which consists of 80% of the total samples, is designated as the training set and is used to develop and optimize the models. The remaining 20% of the data is reserved for testing the model's generalization ability and is referred to as the test set.

To ensure that the training process is unbiased and that the models' performance on the test set is not affected by the selection of training data, the split was performed randomly. This ensures that the samples in the test set are not included in the training set and that they are representative of the overall distribution of the data. The use of a random split also helps to reduce the risk of overfitting, whereby the model is overly specialized to the training data and performs poorly on new, unseen data.

It is important to note that the choice of the split ratio can affect the model's performance. For instance, a larger training set may provide more opportunities for the model to learn and generalize well, while a smaller training set may lead to underfitting, where the model is too simplistic and cannot capture the underlying patterns in the data. Therefore, the split ratio is chosen to balance the need for sufficient training data and the desire to have a robust evaluation of the test set.

[5]https://www.tensorflow.org/api_docs/python/tf/keras/layers/LSTM

# 4

# Results

In this chapter, the results of the evalutation are presented. The initial focus is on the characteristics of the environment setup, such as data rate, delay, signal strength, and other relevant factors outlined in this section. By examining these variables, we can gain insights into the conditions that may have influenced the project's outcomes. Next, the focus is on the data collected in the lab environments, highlighting various statistics about the gathered data. Finally, a summary of the results of the machine learning model is presented, which takes into account the data collected from both simulated and real-world scenarios.

## 4.1 Enviroment Setup

Before showing results related to the data collected or the machine learning model, some general characteristics of the environment setup are presented. Table 4.1 shows the Internet data rate:

| **Internet Access** | **Data rate(Mbps)** |
|---|---|
| Download | 162.36 |
| Upload | 30.04 |

Table 4.1: Internet data rate.

Although the Internet data rate is not constant, as it sometimes drops to 15 Mbps download and 5 Mbps upload; however, this does not influence the results, because the mobile phone uses at most 10 Mbps. Finally, Figure 4.1 shows an image of the lab setup where it is shown the connections of the 5G router, the computer, and the mobile phone that generates the network traffic.

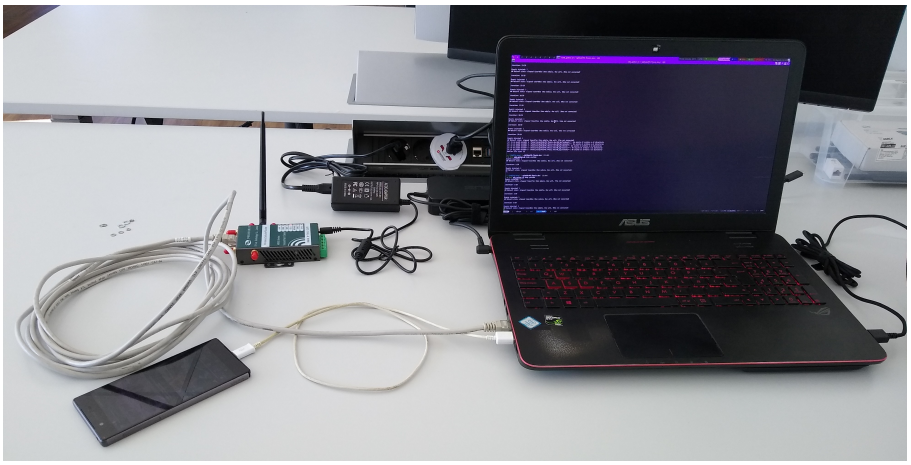

Figure 4.1: Lab Setup.

## 4.2 Data Gathering Results

The packets collected are preprocessed, so the evaluation is based on the number of bits transmitted every 50ms as described in Section 3.4 and Section 3.5. So after processing the network traffic, we obtain a flow as shown in Figure 4.2, an example of the link layer traffic flow captured during the sniffing. This flow corresponds to an Instagram traffic flow.

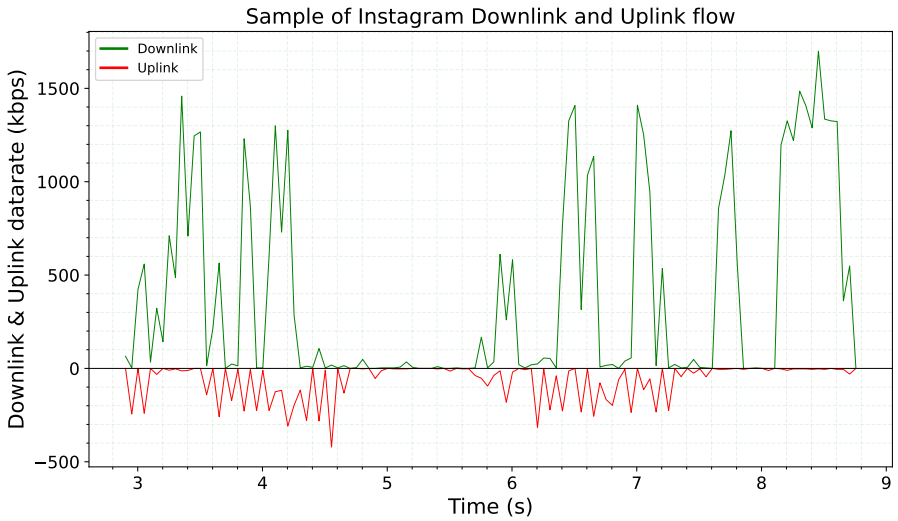


Figure 4.2: Instagram link layer traffic flow.

The total amount of samples gathered for each application is shown in Table 4.2.

| Application | Number of Samples |
|---|---|
| Facebook | 237 |
| Instagram | 250 |
| Linkedin | 231 |
| Spotify | 246 |
| Tiktok | 236 |
| Twitter | 240 |
| Wikipedia | 247 |
| Youtube | 233 |
| **Total Samples** | **1920** |

Table 4.2: Amount of samples captured.

## 4.3 Data Processing Results

After extracting the bursts of the link layer traffic flow, some statistical values are extracted as mentioned in Section 3.5. This section displays some relevant statistical values such as mean, total, and median for the mobile app samples, which were filtered through burst detection and extracted from the link layer traffic flow to improve precision in the analysis.

For a general view of the link layer traffic flow, Figure 4.3 shows the distribution of the mean of transmitted bits per second for every application. This figure is a boxplot, which is a statistical visualization tool that displays the distribution of a dataset. It consists of a box with whiskers extending from either end. The box in the plot represents the interquartile range (IQR), which is the range between the first quartile (Q1) and the third quartile (Q3) of the dataset. The median, or the middle value of the dataset, is represented by a line inside the box. The whiskers in the plot extend to the minimum and maximum values within a specified range, typically 1.5 times the IQR. Any points outside this range are considered outliers and are plotted as individual points beyond the whiskers.

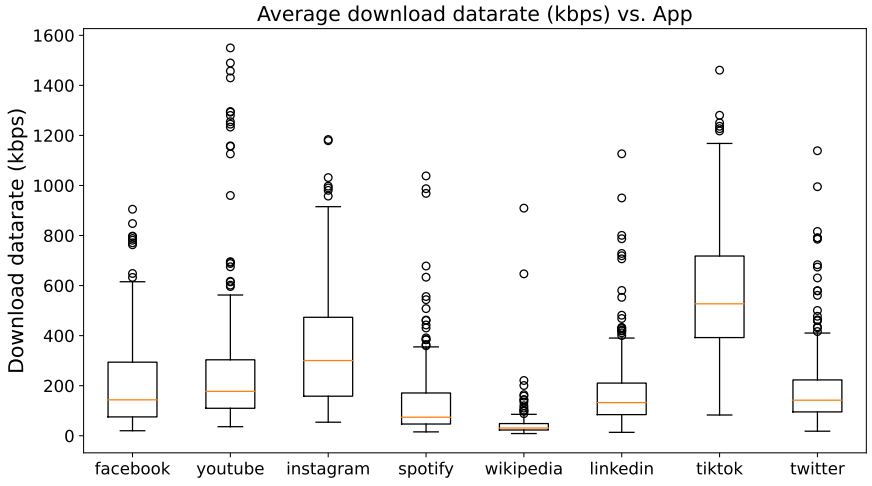


Figure 4.3: Average of transmitted kbits per second.

In the example, Wikipedia has a very low average data rate during its usage; instead, video-related apps have a higher average data rate, such as Tiktok.

Another interesting statistical value is the number of bits transmitted during the capture. Figure 4.4 shows some general view of the applications. Similar to the previous example, Wikipedia has fewer bits transmitted compared to video-related apps, such as Youtube or Instagram.

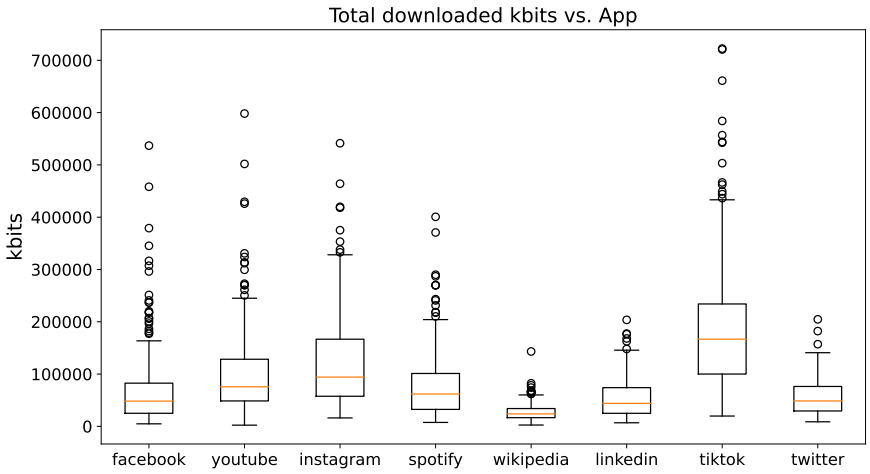


Figure 4.4: Total amount of transmitted bits.

Additionally, both figures show a considerable amount of outliers, which can have consequences on the model predictions, either positively or negatively.

## 4.4 Classification Results

This project uses three models: k-NN, Random Forest, and Long short-Term Memory. This section shows a summary of the results found for each model. First, the models are evaluated in a typical symmetric internet connection, a reference point for comparison with the 5G connection.

a) k-NN Model

Figure 4.5 illustrates both the accuracy and the confusion matrix for the k-NN model. The model's accuracy was only able to achieve a maximum of 50.65% over the entirety of the test dataset.

This suggests that the model could benefit from further refinement, and the dataset may need to be optimized to enhance its performance.

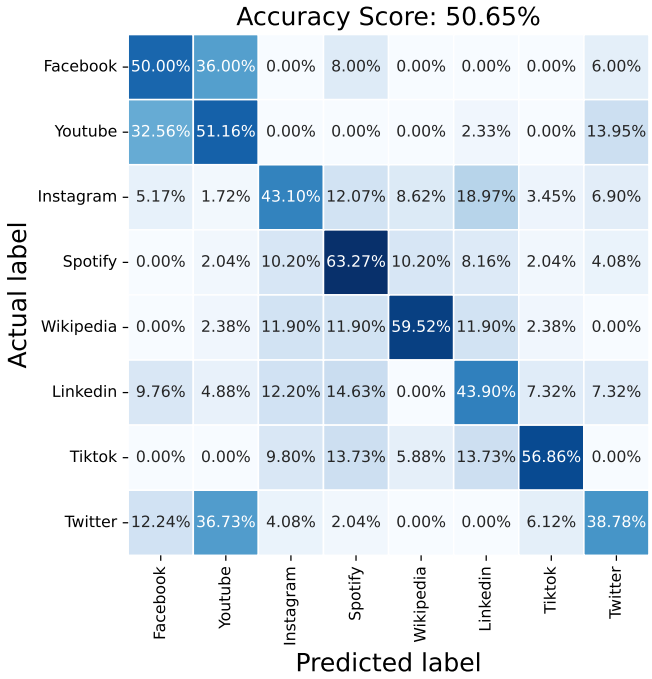


Figure 4.5: Confusion Matrix k-NN.

Some additional conclusions can be extracted from the confusion matrix. For instance, the accuracy of identifying Wikipedia is usually higher than others. This happens because of the nature of Wikipedia which is a text-based application and is very different from the others. It is worth to notice that Spotify shows good accuracy; however, that is not consistent when testing several times.

Table 4.3 shows the parameters of the best implemented model.

| **Parameters** | **Values** |
|---|---|
| Number of nearest neighbors | 10 |
| Distance | Manhattan |
| Leaf size | 30 |

Table 4.3: k-NN Parameters.

The observed low accuracy of the k-NN model could potentially be attributed to the presence of a significant number of outliers in the dataset, which may have a negative impact on the model's ability to make accurate predictions.

Figure 4.6 presents the variations in accuracy when different parameters are tested for the k-NN model. The main parameter, apart from the number of neighbors, is Minkowski distance which is a generalization of the Euclidean as mentioned in Section 2.6.1. No significant changes in the accuracy of the model are observed concerning other parameters, such as leaf size; therefore, they are not shown in the figure.

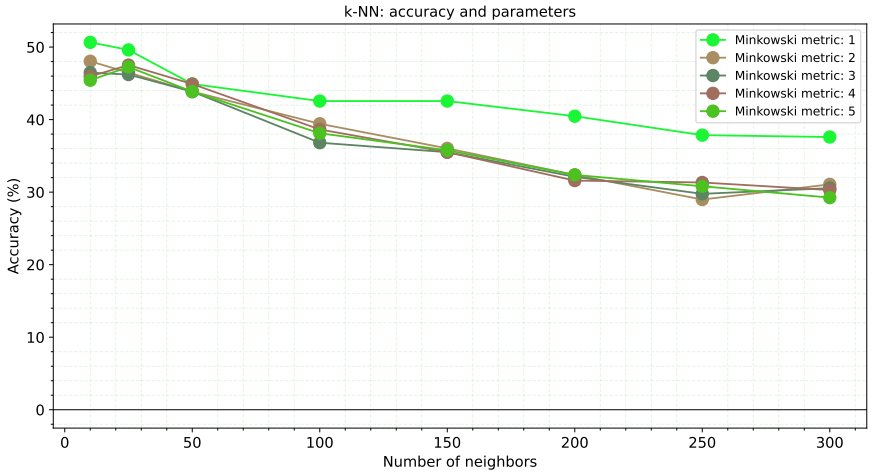


Figure 4.6: Accuracy over different parameters.

Interestingly, it is observed that as the number of neighbors increases, the accuracy diminishes, which suggests that this model is sensitive to the presence of outliers in the dataset.

b) Random Forest Model

The Random Forest model accomplishes an accuracy of 87.73%, showing that it is possible to identify mobile applications even if the communication channel on a 5G network is encrypted. Figure 4.7 shows the confusion matrix for the Random Forest Model.

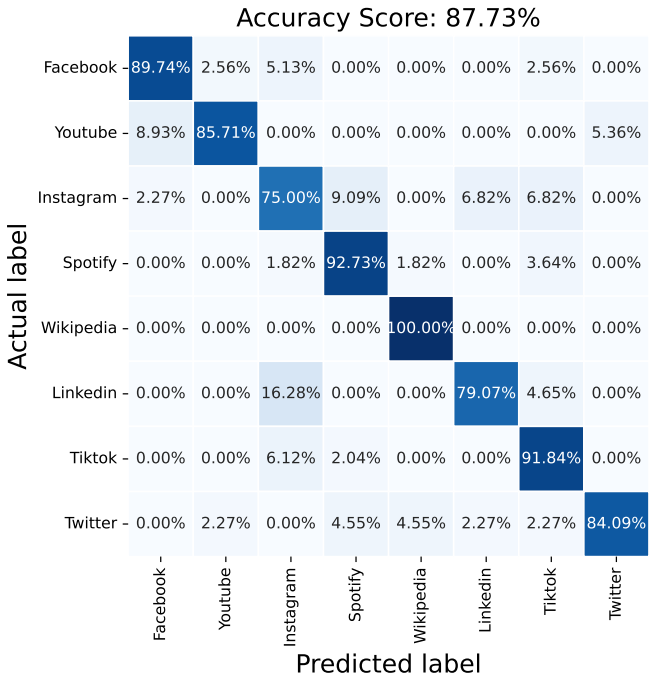


Figure 4.7: Confusion Matrix Random Forest.

Table 4.4 shows the parameters of the best Random Forest model.

| **Parameters** | **Values** |
|---|---|
| Estimators | 200 |
| Max Depth | 20 |

Table 4.4: Random Forest Parameters.

In contrast with the k-NN model, accuracy is not affected significantly as parameters change. Figure 4.8 presents how accuracy changes as other parameters vary. Notably, accuracy does not change significantly after the number of estimators reaches the value of 50. Other parameters, such as maximum depth and weight, do not have much influence on the result. Figure 4.8 only shows the accuracy when the number of estimators and the maximum depth is changing.

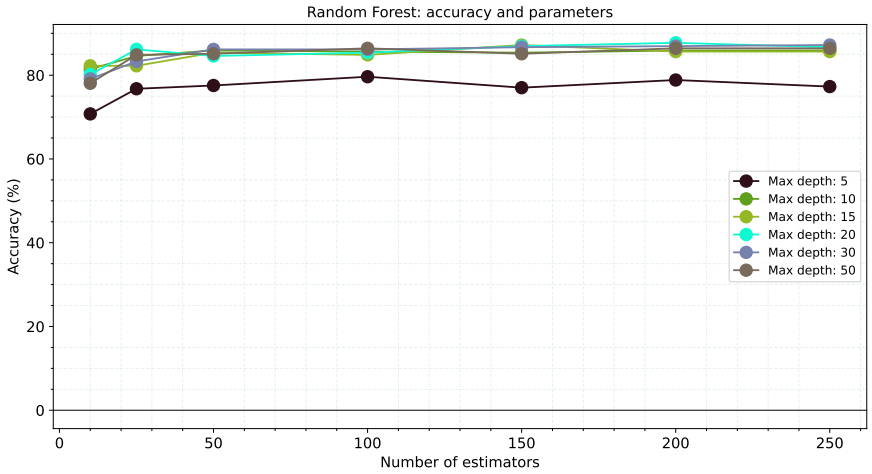


Figure 4.8: Accuracy over different parameters.

Finally, these results of the project also confirm the statement made in Section 2.6.2 that Random Forest models are less sensitive to outliers when compared with the k-NN model.

c) Long short-Term Memory

In contrast to the k-NN or Random Forest model, LSTM does not utilize the statistical data; instead, it takes a different approach by using sequential data as input, as explained in Section 3.6. Even though the accuracy is lower than the Random Forest model, there is room for improvement since the accuracy is only 41%.

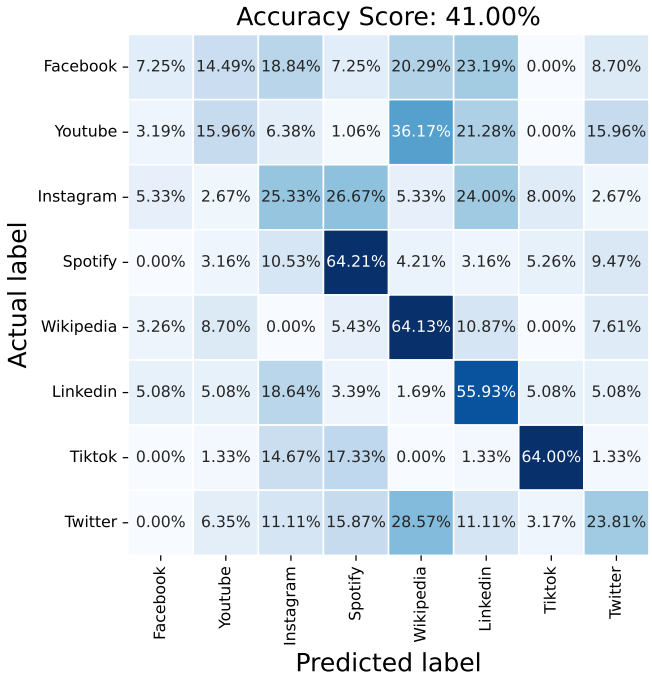


Figure 4.9: Confusion Matrix LSTM.

In contrast to the Random Forest confusion matrix, LSTM shows higher confusion for Wikipedia and others like Facebook and Youtube; but it accomplishes higher accuracy over Twitter or Instagram.

Table 4.5 shows the parameters of the LSTM model, to achieve best accuracy in the evaluation dataset.

| **Parameters** | **Values** |
|---|---|
| Number of layer | 2 |
| 1st layer | LSTM: 256 neurons |
| 4th layer | Dense (Output) |

Table 4.5: LSTM Parameters.

Two main parameters change the accuracy of this model, the number of neurons and the number of layers. Figure 4.10 shows how the accuracy changes while changing those two parameters. It is notable that not necessarily accuracy improves when increasing the number of layers, or increasing the number of neurons on the LSTM layer.

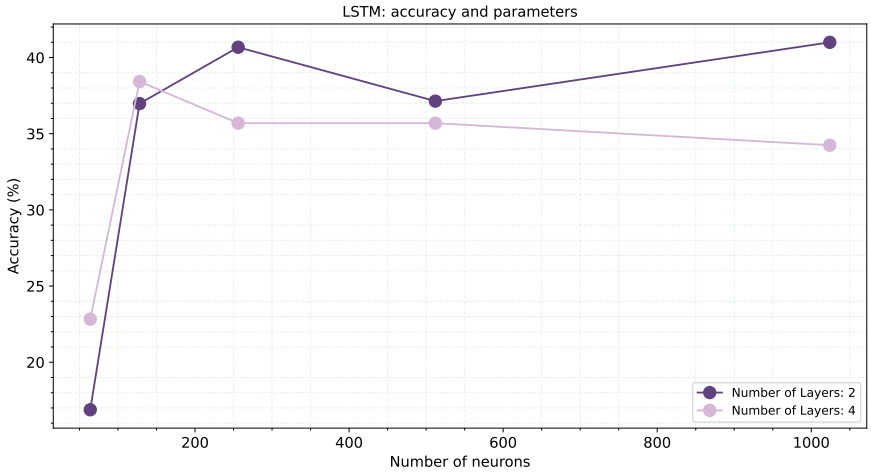


Figure 4.10: Accuracy over different parameters.

d) Summary of the results

The findings of this project indicate that it is possible to identify or classify mobile applications over a 5G network, even if the traffic flow is encrypted, especially from the results found on the Random Forest Model. Table 4.6 shows a summary of the results.

| **Model** | **Accuracy** | **Paramenters** |
|---|---|---|
| k-NN | 50.65% | Neightbors:10 |
| | | Leaf size: 30 |
| | | Distance: Manhattan |
| Random Forest | 87.73% | Estimators: 200 |
| | | Depth: 20 |
| LSTM | 41.00% | Layers: 2 |
| | | LSTM neurons: 256 |

Table 4.6: Summary of results.

Finally, Despite this project being focused on a small set of applications, these outcomes suggest that the approach could be extended to a wider range of applications.

# 5

# Conclusion

This section provides a summary of the key findings and insights from the project, as well as recommendations for future work.

## 5.1 Discussion

The present project demonstrates the feasibility of identifying mobile applications over a 5G network, but there is room for improvement in terms of accuracy. Investigating the mechanisms behind the application communications could enhance accuracy. By gaining a deeper understanding of how applications function, data could be preprocessed more effectively before training the machine models, leading to more precise identification results.

Although the scope of this project is limited to analyzing a small set of applications, it is possible to scale up the analysis to include a larger number of applications. Additionally, it may be important to consider the nature of the application being analyzed, such as whether it is a text-based or social-media application when conducting future analyses.

This project builds upon previous research by expanding the findings that 5G networks are vulnerable to side-channel attacks and that applications can be identified over the network. This is a useful contribution to the literature, given that 5G is a relatively new technology.

It is worth noting that the Random Forest model reaches accuracy in the range of 87%, which is similar to the values found in the research conducted by Taylor et al. [4]. This indicates that these results are good, but there is also room for improvement.

Also, other approaches could be useful, such as grouping the dataset by the type of app. For instance, grouping video-type apps, social media, or text-based apps in different categories, etc., so the classification could be easier, given that the features are more distinguishable from one to another.

Additionally, this project also explored the effectiveness of FFT (Fast Fourier Transform) in analyzing link layer traffic flow on the frequency domain. However, the test results were not promising. Despite the initial expectation that the technique would provide valuable insights, the preliminary analysis did not show any significant

differences in traffic classification. A similar scenario occurred when attempting to analyze the delay between the arrival of packets. This analysis did not demonstrate any improvement in the classification. Therefore, other techniques or methodologies may need to be explored to achieve the desired results in link layer traffic flow analysis.

It is important to note that while some approaches showed no success in this project, there is still potential for improvement by taking a different filtering process. Future studies could explore alternative methods for identification to maximize accuracy and improve the effectiveness of the overall approach.

## 5.2 Conclusion

The project concludes that identifying mobile applications on a 5G network, despite encrypted network traffic, is feasible. The results demonstrate that the accuracy of app identification depends on the chosen machine learning model due to the diverse nature of traffic flows. Notably, this project finds that the Random Forest machine learning model performs the best among the tested models, given its resistance to outlier values.

Furthermore, the k-NN machine learning model is susceptible to the presence of numerous outliers, as shown in Figure 4.3. Additionally, this project suggests that identifying applications with different characteristics is possible, such as text-based and video-based applications like Wikipedia and Youtube. This could also expand the application identification, allowing to perform the identification of different classes of applications like Streaming, Social, Games, News, etc.

In summary, the project confirms that identifying mobile applications on a 5G network with encrypted traffic is achievable, and the Random Forest machine learning model is the most accurate approach. The research also suggests that the identification of applications with different characteristics is feasible, providing implications for areas such as network security and mobile application management.

# Bibliography


[1] Andrew Reed and Benjamin Klimkowski. "Leaky streams: Identifying variable bitrate DASH videos streamed over encrypted 802.11n connections". In: 2016 13th IEEE Annual Consumer Communications & Networking Conference (CCNC). Jan. 2016, pp. 1107–1112. DOI: 10.1109/CCNC.2016.7444944.

[2] Qinglong Wang, Amir Yahyavi, Bettina Kemme, and Wenbo He. "I know what you did on your smartphone: Inferring app usage over encrypted data traffic". In: 2015 IEEE Conference on Communications and Network Security (CNS). Sept. 2015, pp. 433–441. DOI: 10.1109/CNS.2015.7346855.

[3] Marc Lichtman, Raghunandan Rao, Vuk Marojevic, Jeffrey Reed, and Roger Piqueras Jover. *5G NR Jamming, Spoofing, and Sniffing: Threat Assessment and Mitigation*. May 1, 2018. 1 p. DOI: 10.1109/ICCW.2018.8403769.

[4] Vincent F. Taylor, Riccardo Spolaor, Mauro Conti, and Ivan Martinovic. "AppScanner: Automatic Fingerprinting of Smartphone Apps from Encrypted Network Traffic". In: 2016 IEEE European Symposium on Security and Privacy (EuroS&P). Mar. 2016, pp. 439–454. DOI: 10.1109/EuroSP.2016.40.

[5] Qiuning Ren, Chao Yang, and Jianfeng Ma. "App identification based on encrypted multi-smartphone sources traffic fingerprints". In: *Computer Networks* 201 (Dec. 24, 2021), p. 108590. ISSN: 1389-1286. DOI: 10.1016/j.comnet.2021.108590.

[6] Yorgos Dimopoulos, Ilias Leontiadis, Pere Barlet-Ros, and Konstantina Papagiannaki. *Measuring Video QoE from Encrypted Traffic*. Nov. 14, 2016. 513 pp. DOI: 10.1145/2987443.2987459.

[7] Georgi Ajaeiya, Imad Elhajj, Ali Chehab, Ayman Kayssi, and Marc Kneppers. "Mobile Apps identification based on network flows". In: *Knowledge and Information Systems* 55 (June 1, 2018), pp. 1–26. DOI: 10.1007/s10115-017-1111-8.

[8] 3GPP. *5G System Overview*. URL: https://www.3gpp.org/technologies/5g-system-overview.

[9] ENISA. *Security in 5G Specifications - Controls in 3GPP*. URL: https://www.enisa.europa.eu/publications/security-in-5g-specifications.

[10] Guoqiang Shu and David Lee. "A Formal Methodology for Network Protocol Fingerprinting". In: *IEEE Transactions on Parallel and Distributed Systems* 22.11 (Nov. 2011), pp. 1813–1825. ISSN: 1558-2183. DOI: 10.1109/TPDS.2011.26.

[11] Timothy J. Shepard. "TCP Packet Trace Analysis". PhD thesis. Feb. 1, 1991.

[12] Jon Kleinberg. "Bursty and Hierarchical Structure in Streams". In: *Data Mining and Knowledge Discovery* 7.4 (Oct. 1, 2003), pp. 373–397. ISSN: 1573-756X. DOI: 10.1023/A:1024940629314.

[13] Vipul Goyal. "Lecture Notes on Introduction to Cryptography". In: (2020). URL: https://www.cs.cmu.edu/~goyal/15356/lecture_notes.pdf.

[14] Eric Rescorla. *The Transport Layer Security (TLS) Protocol Version 1.3*. Request for Comments RFC 8446. Internet Engineering Task Force, Aug. 2018. DOI: 10.17487/RFC8446.

[15] Eric Rescorla. *HTTP Over TLS*. Request for Comments RFC 2818. Internet Engineering Task Force, May 2000. DOI: 10.17487/RFC2818.

[16] IBM Corporation. *IBM Documentation*. Nov. 14, 2022. URL: https://www.ibm.com/docs/en/sdk-java-technology/8?topic=handshake-tls-13-protocol.

[17] Khaled Al-Naami, Swarup Chandra, Ahmad Mustafa, Latifur Khan, Zhiqiang Lin, Kevin Hamlen, and Bhavani Thuraisingham. *Adaptive encrypted traffic fingerprinting with bi-directional dependence*. Dec. 5, 2016. 177 pp. DOI: 10.1145/2991079.2991123.

[18] Ivan Ristić. *OpenSSL Cookbook*. 3rd Edition. 2021. URL: https://www.feistyduck.com/library/openssl-cookbook/online/.

[19] Ethem Alpaydin and Francis Bach. *Introduction to Machine Learning*. 3rd ed. Adaptive Computation and Machine Learning Ser. MIT Press, 2014. ISBN: 978-0-262-32574-5.

[20] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep learning*. Adaptive computation and machine learning. MIT Press, 2016. ISBN: 978-0-262-03561-3.

[21] Christopher M. Bishop. *Pattern recognition and machine learning*. Information science and statistics. Springer, 2006. ISBN: 978-0-387-31073-2.

[22] Steven S. Skiena. "Distance and Network Methods". In: *The Data Science Design Manual*. Texts in Computer Science. Cham: Springer International Publishing, 2017, pp. 303–349. ISBN: 978-3-319-55444-0. DOI: 10.1007/978-3-319-55444-0_10.

[23] IBM Corporation. *What is the k-nearest neighbors algorithm?* URL: https://www.ibm.com/se-en/topics/knn.

[24] Leo Breiman. "Random Forests". In: *Machine Learning* 45.1 (Oct. 1, 2001), pp. 5–32. ISSN: 1573-0565. DOI: 10.1023/A:1010933404324.

[25] IBM Corporation. *What is Random Forest?* URL: https://www.ibm.com/topics/random-forest.

[26] Christopher Olah. *Understanding LSTM Networks*. URL: https://colah.github.io/posts/2015-08-Understanding-LSTMs/.

[27] Sepp Hochreiter and Jürgen Schmidhuber. "Long Short-term Memory". In: *Neural computation* 9 (Dec. 1, 1997), pp. 1735–80. DOI: 10.1162/neco.1997.9.8.1735.

[28] Klaus Greff, Rupesh Srivastava, Jan Koutník, Bas Steunebrink, and Jürgen Schmidhuber. "LSTM: A search space odyssey". In: *IEEE transactions on neural networks and learning systems* 28 (Mar. 13, 2015). DOI: 10.1109/TNNLS.2016.2582924.

[29] MIT Open Learning Library. *Introduction to Machine Learning*. Dec. 18, 2019. URL: https://openlearninglibrary.mit.edu/courses/course-v1:MITx+6.036+1T2019/courseware/Week11/rnn/?activate_block_id=block-v1%3AMITx%2B6.036%2B1T2019%2Btype%40sequential%2Bblock%40rnn.

[30] *California Consumer Privacy Act (CCPA)*. State of California - Department of Justice - Office of the Attorney General. Oct. 15, 2018. URL: https://oag.ca.gov/privacy/ccpa.

[31] GDPR. *What is GDPR, the EU's new data protection law?* GDPR.eu. Section: GDPR Overview. Nov. 7, 2018. URL: https://gdpr.eu/what-is-gdpr/.

[32] Yang Li, Mengting Chen, and Jian Wang. "Introduction to side-channel attacks and fault attacks". In: 2016 Asia-Pacific International Symposium on Electromagnetic Compatibility (APEMC). Vol. 01. May 2016, pp. 573–575. DOI: 10.1109/APEMC.2016.7522801.

[33] Eva Papadogiannaki and Sotiris Ioannidis. "A Survey on Encrypted Network Traffic Analysis Applications, Techniques, and Countermeasures". In: *ACM Computing Surveys* 54.6 (July 13, 2021), 123:1–123:35. ISSN: 0360-0300. DOI: 10.1145/3457904.

[34] Martin Bjorklund, Marcus Julin, Philip Antonsson, Andreas Stenwreth, Malte Akvist, Tobias Hjalmarsson, and Romaric Duvignau. "I See What You're Watching on Your Streaming Service: Fast Identification of DASH Encrypted Network Traces". In: Nov. 2022.

[35] Romaric Duvignau. "Metainformation Extraction from Encrypted Streaming Video Packet Traces". In: 2022 International Conference on Electrical, Computer, Communications and Mechatronics Engineering (ICECCME). Nov. 2022, pp. 1–6. DOI: 10.1109/ICECCME55909.2022.9988476.

[36] Ali Kamal, Syed Muhammad Ammar Hassan Bukhari, Muhammad Usman Shahid Khan, Tahir Maqsood, and Muhammad Fayyaz. *Traffic Pattern Plot: Video Identification in Encrypted Network Traffic*. Aug. 27, 2022.

[37] Roger Piqueras Jover. *The current state of affairs in 5G security and the main remaining security challenges*. Apr. 18, 2019. DOI: 10.48550/arXiv.1904.08394. arXiv: 1904.08394[cs]. (Visited on 05/26/2023).