Inferring Connections between Telegram Users Using Traffic Data Analysis

Gilad Fisher, Chen Dahan

October 2024

1 Abstract

Telegram is a cloud-based messaging app known for its focus on speed, security and privacy. Telegram allows users to send text messages, voice messages, photos, videos and files of various formats. It supports large group chats, public channels and end-to-end encrypted communication through the app. Telegram, with over 950 million monthly users, has become very popular in recent years. Users around the world use the app to communicate through private chats, groups, and channels, with all messages encrypted to ensure privacy. This encryption guarantees that the content of the conversations remains secure and inaccessible to unauthorized parties, allowing users to communicate without concern. Additionally, the encrypted data makes it difficult to determine if two users are even communicating with each other. In this project, we focus on identifying when two users are engaged in a conversation. We analyze the traffic patterns of the data to determine how they can be utilized for our objectives. Our analysis is demonstrated through a evaluation of the data's quality, using machine learning models for training and testing.

2 Introduction

2.1 Background

Because of Telegrams strict privacy policy and encryption, users often assume their interactions are completely hidden from 3rd parties. In this project we exploit a fundamental vulnerability in encrypted instant message platforms like Telegram to find out if two users are conversing using a traffic analysis attack. Several traffic analysis attacks were explored in recent years:

1. **Flow correlation:** This method of traffic analysis attack focuses on finding and correlating signal patterns between users. It has been studied on various anonymity providing platforms such as Telegram, Tor, Signal, etc. [6]

2. **Intersection attacks:** Here, the attacker will observes users' activity periods and identifies intersections between those periods. While low in resolution, this method is computationally simple and can be applied on a large scale. [3]

3. **Flow watermarking:** This technique involves intentionally delaying signal patterns from the sender in a unique way, then detecting that delay in the suspected receiver's traffic [5]

Our project will implement the **Flow correlation** attack to analyze traffic patterns and infer potential communication between users. We chose this method because it offers an optimal balance between resolution and simplicity compared to other techniques. Additionally, we developed a time-series data pipeline designed for use by machine learning models. To ensure the quality of our pipeline, we employed machine learning models for validation and testing.

2.2 Objectives

The goal of this project is to infer whether two users are communicating with each other by analyzing encrypted traffic patterns. To achieve this, we aim to build a streamlined data pipeline that automates the analysis process and supports the deployment of machine learning models. This pipeline will facilitate efficient data handling and model training, allowing us to demonstrate results from multiple applications of the analyzed data.

A primary advantage of our model is its scalability, enabling effective analysis and recreation of large-scale social networks from network traffic. This capability not only facilitates the examination of user interactions within extensive systems but also allows researchers and practitioners to uncover insights into communication patterns, user behavior, and network dynamics. By leveraging this scalable approach, we can adapt the model to accommodate varying data volumes and complexities, making it applicable in diverse scenarios, ranging from academic research to real-world applications in cybersecurity and social media analysis.

2.3 Scope

This project operates under the assumption that both users are online during the data recording process. Additionally, we have determined that only media files—such as photos and videos—are suitable for analysis using our proposed model, meaning that text messages remain inaccessible to this method.

3 Related Work

Numerous studies have addressed traffic analysis in encrypted messaging platforms over the years. One significant contribution comes from Afzal et al., who analyzed traffic patterns in the Signal messaging app [1]. The researchers identified distinct traffic patterns associated with various user actions, such as initiating calls, typing, sending media, and coming online. Their insights into media message traffic closely align with our own findings, particularly in distinguishing between different types of user activities based on packet size and frequency. However, unlike our method, their analysis was not automated.

Another relevant study by Erdenebaatar et al. explored identifying specific apps being used by users through traffic analysis. By applying machine learning techniques, they successfully classified the app in use based on network traffic data, achieving high accuracy [4]. While this demonstrates the potential of using machine learning for automated traffic analysis, their work differs from ours in that it focuses on app classification rather than inferring communication between two users.

4 Methodology

4.1 Data Generation

Initially, we developed a file generator and automated a conversation between two users using Python and Selenium. The generator creates random messages with different content, sizes and types. It illustrates a real conversation between the users. Network traffic for both users was recorded using Wireshark.

To enhance the authenticity of our simulation, we supplemented this generated data with real network traffic data collected by Bahramali et al. [2]. This combination allowed us to more accurately reflect real-life conversations, providing a robust dataset for our analysis.

4.2 Exploratory Data Analysis

The recorded traffic was imported into a Jupyter notebook file for further processing and analysis.

Through exploration, we identified several important patterns. Notably, we observed periodic groups of packets responsible for maintaining the server-to-user connection, which made it difficult to distinguish between regular traffic and packets carrying text messages. As a result, we excluded text messages from our analysis.

Additionally, we discovered that packets related to media messages (e.g., images or videos) exhibit distinct, fixed sizes, as noted by Afzal et al. [1]. This distinction allowed us to reliably identify media messages and focus our analysis on them.

This discovery allows us to very accurately tell if a user is sending or receiving a message of this sort. We categorized the packets into two main types:

1. **Type A:** Small packets used for text messages and to maintain the server connection.

2. **Type B:** Large packets with a fixed size, specifically associated with sending or receiving media messages.

4.3 Pipeline

To focus our analysis, we filter the data to include only the **Type B** packets. Next, we created a time-series representation where each row corresponds to a window of t seconds of the recording and sums the number of **Type B** packets in that window of time. The data was further divided into sections of T hours, with each section representing a unique recording of two users' communications.

Now each section is represented by two vectors: one for each user $\vec{T} = [t_1, t_2, \ldots, t_n]$ and $\vec{T'} = [t'_1, t'_2, \ldots, t'_n]$ where for every $i \in n$, t_i is the number of **Type B** packets observed for each user during the i-th window of time.

We then extracted key metrics describing the relationship between these two vectors to condense the recording into a single row of features for further analysis.

To condense the recording into a single row of features for further analysis, we extracted key metrics that describe the relationship between the two vectors. The dataset was enhanced by adding several features: the sum of the vector for user 1, the sum for user 2, and the correlation between the two vectors. Additionally, we calculated the dot product between the two vectors, $\vec{T} = [t_1, t_2, \ldots, t_n]$ and $\vec{T'} = [t'_1, t'_2, \ldots, t'_n]$

$$\vec{T} \cdot \vec{T'} = t_1 t'_1 + t_2 t'_2 + \cdots + t_n t'_n = \sum_{i=1}^{n} t_i t'_i \tag{1}$$

by multiplying corresponding elements of the vectors and then summing the results, distance between the two vectors

$$d(\vec{T}, \vec{T'}) = \sqrt{(t_1 - t'_1)^2 + (t_2 - t'_2)^2 + \cdots + (t_n - t'_n)^2} \tag{2}$$

and cosine similarity by calculating the cosine of the angle between them

$$\text{Cosine Similarity} = \cos(\theta) = \frac{\vec{T} \cdot \vec{T'}}{|\vec{T}||\vec{T'}|} \tag{3}$$

At this stage, we have generated a streamlined dataset, ready for training machine learning models.

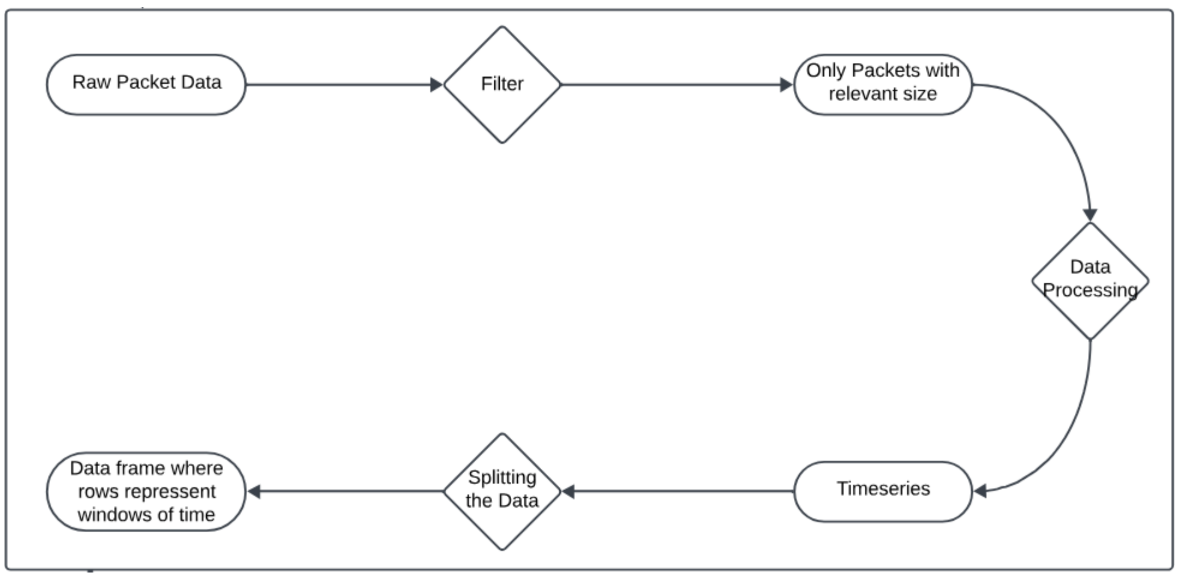
Raw Packet Data
Filter
Only Packets with relevant size
Data Processing
Timeseries
Splitting the Data
Data frame where rows repressent windows of time

Figure 1: Data Flow Diagram.
Explanation: This diagram outlines the key steps in our data processing pipeline, illustrating how raw network traffic recordings are transformed into a single row of data.

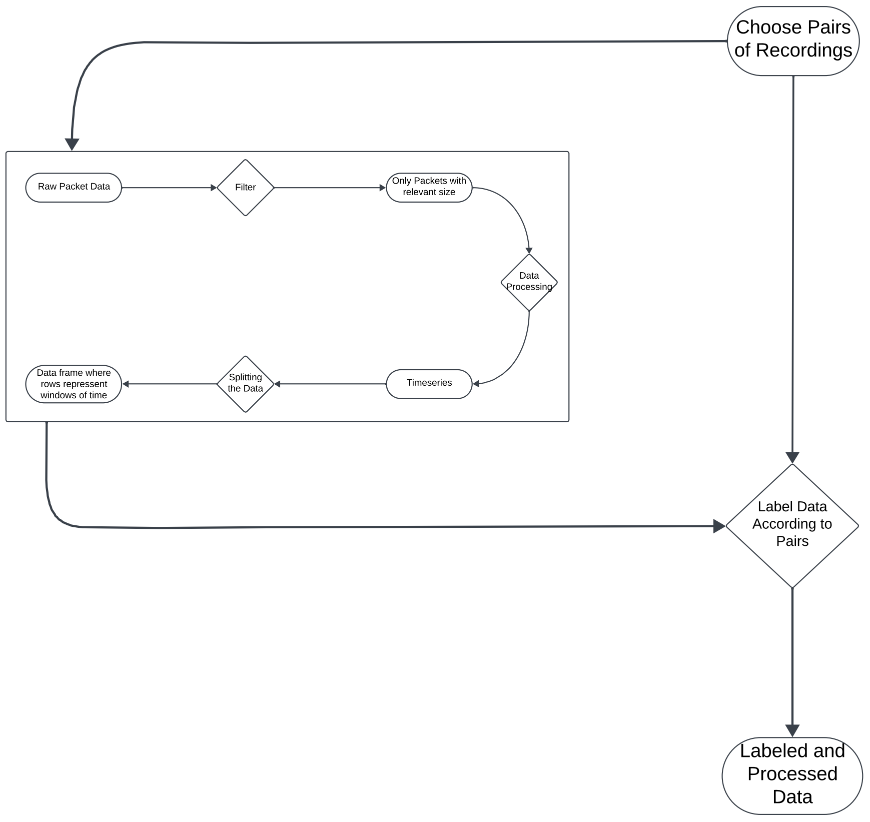
Choose Pairs of Recordings
Raw Packet Data
Filter
Only Packets with relevant size
Data Processing
Timeseries
Splitting the Data
Data frame where rows repressent windows of time
Label Data According to Pairs
Labeled and Processed Data

Figure 2: Data Flow Diagram.
Explanation: This figure illustrates the labeling process, which builds upon the steps outlined in the previous figure. If the selected pair of users comes from the same recording, it is labeled as 1 (indicating they are conversing). Otherwise, it is labeled as 0 (indicating no conversation).

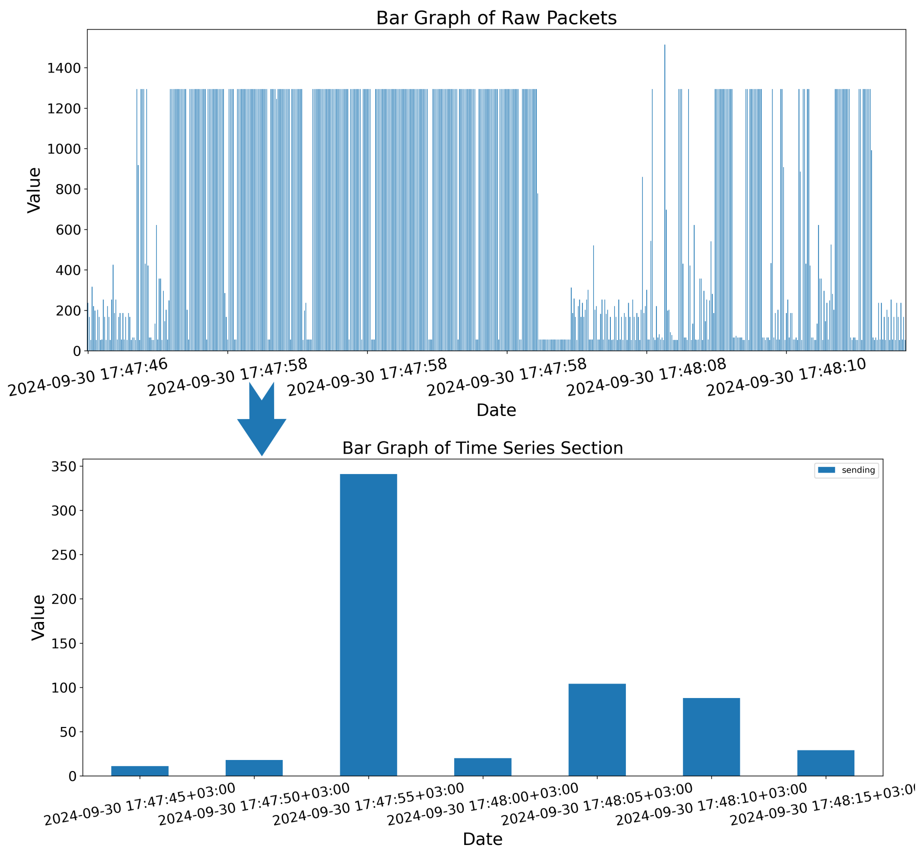

Figure 3: Simplification of Input

Explanation: This figure illustrates the simplification process of the input data used for analysis. The bars in the first graph represent individual packets exchanged between the user and the server, with the bar height corresponding to the packet length. In the second graph, the bars show the number of **Type B** packets (media-related packets) within each time window. The third window clearly indicates the transmission of a media file, as seen by the spike in **Type B** packet activity.

5 Implementation

During the implementation of our project, we encountered several significant challenges:

1. **Noisy Data**: The recorded network traffic contained a considerable amount of noise, which complicated the process of accurately identifying relevant patterns associated with user interactions. Many of the captured packets exhibited little to no correlation with the actual sending or receiving of messages, which made it difficult to discern meaningful communication activities from irrelevant traffic.

2. **Excessive Data Volume**: The sheer volume of data generated from the recordings posed difficulties in processing and analyzing, necessitating efficient data handling techniques to extract meaningful insights.

3. **Feature Extraction**: Identifying and selecting the most relevant features from the filtered data was challenging, as the success of our machine learning models heavily relied on the quality of the features used. The primary objective in this phase was to effectively represent the relationship between the two vectors as a scalar.

4. **Complex Data Structures**: The nature of the data, including the variety of packet types and their interactions, introduced additional complexity that required careful consideration during analysis. Simplifying the problem into a concise and accurate form involved considerable trial and error, as we navigated the intricacies of the data.

5. **Limited Computational Power**: Due to processing speed limitations, we had to discard certain features, such as packet payloads, which could have been useful for deeper analysis. Payload data, while potentially valuable, requires significant computational resources to process, and including it would have greatly increased the load. This omission may have reduced the accuracy of our analysis, as important communication patterns might reside in those discarded features.

Addressing these challenges was crucial to the successful execution of our methodology and the validity of our findings. To determine whether there was communication between two users, we analyzed their Wireshark recordings. As previously described, we created time-series data representing windows of t seconds from the recordings. For each time window, we examined whether there were packets to compare between the two recordings. If so, we calculated the correlation between the corresponding rows in both recordings. This approach aimed to identify any relationship between the transmissions. The recordings were divided into one-hour chunks to allow for more precise evaluation. Each one-hour segment was treated as a vector containing the values from the time window rows, providing a more granular analysis. Additionally, we investigated correlations between rows from different time windows to account for possible packet delay over time between the users. To achieve this, we applied a shifting technique, where the rows were shifted iteratively by varying amounts.

6 Testing and Evaluation

Labeling the data points at the processing stage will help us evaluate any model's success. We can split the features and the labels to X and Y. Deploying a model on X will result in a prediction Y', which will then be compared to the true outcomes Y. This comparison allows us to assess how well the model is performing. We employ several standard evaluation metrics, such as accuracy, precision, and recall, to measure the model's success. These metrics help ensure that the model is not only making correct predictions but is also doing so consistently and in line with the problem's requirements.

Furthermore, this process helps us identify potential areas for improvement, such as whether the model is biased towards certain predictions or whether it's performing better in some cases than others. By carefully analyzing these metrics, we can fine-tune the model and improve its robustness, ensuring that it generalizes well to new, unseen data.

7 Results

To evaluate the quality of the data generated by the pipeline, we trained and tested multiple machine learning models. The dataset comprised 151 entries, summing up to 151 hours of recorded traffic, and we selected models well-suited for small datasets, including Logistic Regression, Decision Tree, Support Vector Machine, and Random Forest. We averaged the performance of all models across multiple random states to ensure consistency and robustness of the results.

Metric	**Value**
Accuracy	95.99%
Precision	96.32%
Recall	95.99%
F1 Score	95.89%

Table 1: Evaluation Metrics for Traffic Analysis Model

All the models we used performed well and provided similar outputs. We believe further data collection and testing may provide more insights on ways to improve the Pipeline.

8 Conclusion

In this project, we successfully demonstrated the potential to infer communication between Telegram users by analyzing encrypted traffic patterns through the development of a data pipeline and the application of machine learning models. We were able to effectively process network traffic and extract meaningful correlations between users' transmissions to better fit machine learning models. Despite the challenges posed by noisy data and the large volume of traffic, our approach proved viable for identifying communication activity when media files were involved.

Due to the efficiency and modularity of the pipeline, the process of inferring communication between users can be scaled for large-scale applications.

9 Future Work

This research demonstrates the feasibility of applying traffic analysis techniques on a larger scale, with the potential to generate comprehensive graphs that elucidate the intricate relationships between users and groups. Such visual representations could provide invaluable insights into real-life social structures, highlighting communication patterns and interactions that may otherwise remain hidden. By further developing these methodologies, we aim to uncover deeper social dynamics, enabling a better understanding of user behavior and the formation of communities in encrypted messaging platforms.

References

[1] Asmara Afzal, Mehdi Hussain, Shahzad Saleem, M Khuram Shahzad, Anthony TS Ho, and Ki-Hyun Jung. Encrypted network traffic analysis of secure instant messaging application: A case study of signal messenger app. *Applied Sciences*, 11(17):7789, 2021.

[2] Alireza Bahramali, Ramin Soltani, Amir Houmansadr, Dennis Goeckel, and Don Towsley. Practical traffic analysis attacks on secure messaging applications. *arXiv preprint arXiv:2005.00508*, 2020.

[3] George Danezis and Andrei Serjantov. Statistical disclosure or intersection attacks on anonymity systems. In *International Workshop on Information Hiding*, pages 293–308. Springer, 2004.

[4] Zolboo Erdenebaatar. *Generating and Analyzing Encrypted Traffic of Instant Messaging Applications: A Comprehensive Framework.* PhD thesis, 2023.

[5] Amir Houmansadr, Negar Kiyavash, and Nikita Borisov. Rainbow: A robust and invisible non-blind watermark for network flows. In *NDSS*, volume 47, pages 406–422. Citeseer, 2009.

[6] Steven J Murdoch and George Danezis. Low-cost traffic analysis of tor. In *2005 IEEE Symposium on Security and Privacy (S&P'05)*, pages 183–195. IEEE, 2005.