

Side-Channel Attacks in Video Identification via Encrypted Traffic Analysis

Brian Ho
2025bho@tjhsst.edu

December 18, 2024

Abstract

This research explores the feasibility of identifying 360-degree video titles through encrypted traffic analysis. Building on prior work in video fingerprinting for 2D streams, this project applies machine learning techniques to detect distinct traffic patterns in 360-degree video streams. By examining potential burst patterns and segment sizes in network data, the study aims to assess whether encrypted 360-degree video streams can be accurately identified, without direct access to the video content. This work could expand privacy-focused video identification techniques to a new class of media.

Contents

1	Introduction	3
2	Background / Literature Review	3
2.1	Encrypted Traffic Analysis	3
2.1.1	Encrypted Traffic Analysis Fundamentals	3
2.1.2	Prior Research on Video Fingerprinting	4
2.1.3	Threat Model	4
2.2	Wireshark Overview	5
2.3	Overview of Video Streaming	5
2.4	Machine Learning in Video Fingerprinting	5
3	Methods	6
3.1	2D Video Capture and Dataset Construction	6
3.2	360-Degree Video Capture (Planned)	7
4	Results	7
5	Discussion	9
5.1	Challenges and Implementation Effort	9
5.2	Informal User Interface Evaluation	10
5.3	Planned Extensions Before tjSTAR	10
6	Conclusions	11
7	References	11

1 Introduction

Encrypted traffic analysis has emerged as a field of research focused on identifying patterns in encrypted data streams to infer information about the transmitted content. Even though encryption protects the content of the data, certain patterns such as packet size, frequency of data bursts, and transmission times remain visible. Prior research has demonstrated that these metadata elements can be used to identify 2D video titles. For instance, studies have shown that distinct traffic patterns allow researchers to successfully fingerprint 2D video streams based on burst sizes unique to individual videos [1]. The core premise is that while encryption obfuscates content, it does not protect the traffic’s observable characteristics, which is a side channel vulnerability that can be exploited by an attacker. 360-degree videos present a unique challenge for encrypted traffic analysis due to their immersive nature and higher data transmission demands. Unlike traditional 2D videos, 360-degree videos capture a full spherical view, allowing users to look in any direction, which requires streaming significantly more data to accommodate different viewports. The continuous transmission of data related to different angles and scenes results in more complex traffic patterns. These complexities make identifying 360-degree videos from encrypted traffic more difficult than identifying 2D videos. However, we predict that analyzing the data bursts and transmission behaviors may still provide unique patterns that can be exploited for fingerprinting.

2 Background / Literature Review

2.1 Encrypted Traffic Analysis

2.1.1 Encrypted Traffic Analysis Fundamentals

Encrypted traffic analysis is the practice of inferring information from data that is protected by encryption protocols such as HTTPS, TLS, or VPN tunnels. Although the content of the communication is encrypted and cannot be viewed directly, metadata, such as packet sizes, frequency, direction, and duration, remain visible to observers. These metadata elements can be leveraged to uncover information about user activity or the nature of the content being transmitted. For example, the periodic burst patterns of streaming ser-

vices or the signature of specific applications can all persist in the encrypted traffic flow.

2.1.2 Prior Research on Video Fingerprinting

A key study in encrypted video fingerprinting is Schuster et al.’s *Beauty and the Burst* [1], which demonstrated that video streams leak identifiable information through their burst patterns. The paper identifies how modern streaming protocols, specifically MPEG-DASH, lead to content-dependent packet bursts due to variable bitrate encoding and segment-based delivery. These burst patterns, consisting of alternating “on” (download) and “off” (idle) periods, create consistent, repeatable traffic signatures for a given video.

In their experiments, Schuster et al. trained Convolutional Neural Networks (CNNs) on time series data such as packets per second and burst sizes, achieving up to 99% accuracy when classifying YouTube videos from encrypted traffic. They further proved that their method is viable even in “open-world” settings, where the video may not belong to a small known set.

Importantly, the paper also introduced various attacker models: not only on-path attackers like ISPs or compromised routers, but also off-path attackers capable of measuring network congestion remotely via JavaScript-based side channels. These scenarios emphasize the real-world feasibility and privacy implications of traffic fingerprinting, motivating further exploration.

2.1.3 Threat Model

The threat model for this research assumes a passive adversary with the ability to monitor encrypted network traffic. A common example of such an adversary is an Internet Service Provider (ISP), which has full visibility into the metadata of user traffic.

In a real-world scenario, an ISP could analyze traffic patterns to identify which videos a user is watching on platforms like YouTube. This information could then be monetized by selling inferred user preferences to advertisers. In addition to commercial motives, more sophisticated adversaries such as government surveillance agencies or compromised network infrastructure (e.g., a malicious access point) could use similar techniques to track user behavior, profile individuals, or censor content.

2.2 Wireshark Overview

Wireshark is an open-source network protocol analyzer that allows users to capture and inspect data packets as they travel across a network. It works by placing a network interface into "promiscuous mode," which enables the capture of all incoming and outgoing packets on that interface. Each packet can then be examined in detail, including its headers, source and destination IP addresses, ports, and protocol information.

Wireshark supports a wide range of protocols (eg. DNS, TLS, HTTPS) and offers extensive filtering capabilities. Although Wireshark does not decrypt encrypted content, it is still highly useful for encrypted traffic analysis. This is because metadata such as packet sizes, timing, and direction are not encrypted and can potentially reveal consistent patterns. This ability to capture observable traffic characteristics makes Wireshark a foundational tool for side-channel analysis in this study.

2.3 Overview of Video Streaming

Modern video platforms such as YouTube and Netflix rely on adaptive bitrate streaming to deliver content efficiently across a variety of network conditions. Two of the most commonly used protocols for this purpose are MPEG-DASH (Dynamic Adaptive Streaming over HTTP) and HLS (HTTP Live Streaming). In this study, we focus exclusively on YouTube, which uses MPEG-DASH. These protocols break video files into small segments, typically ranging from 2 to 10 seconds in length, and allow the client device to request segments at different quality levels depending on available bandwidth.

While this approach ensures smooth playback and reduces buffering, it also generates consistent and predictable traffic patterns. Each time a segment is requested, a burst of data is transmitted, followed by a period of inactivity as the segment is played. This repeating cycle creates a recognizable pattern of alternating "on" (download) and "off" (idle) periods in the traffic.

2.4 Machine Learning in Video Fingerprinting

Machine learning plays a helpful role in encrypted traffic analysis due to its ability to identify subtle patterns in noisy data that would be difficult to

isolate through manual inspection. In prior research, Convolutional Neural Networks (CNNs) have been widely used for video fingerprinting. For example, in Beauty and the Burst [1], Schuster et al. trained CNNs on time series data representing packets per second and burst sizes, achieving near-perfect accuracy. More recently, researchers have also begun exploring the use of Graph Neural Networks (GNNs), which model the relationships between traffic bursts as nodes and edges in a graph. This approach can capture more complex dependencies and has shown significant promise.

The methods described in the next section build upon these findings, adapting existing fingerprinting techniques towards replication and the unique structure of 360-degree video traffic.

3 Methods

3.1 2D Video Capture and Dataset Construction

Two publicly available 2D YouTube videos were selected for initial testing: a dog video (<https://www.youtube.com/watch?v=An2VTBjuAA4>) and a cat video (<https://www.youtube.com/watch?v=hmw-w4PFms8>). Each video was played in a Google Chrome browser using incognito mode to ensure that no previously cached data would interfere with the traffic. YouTube Premium was enabled to prevent advertisements from introducing noise into the network traces.

Wireshark was used to capture encrypted network traffic during each playback session. Captures began slightly before the refresh button was pressed and continued until shortly after the video ended. Ten playback sessions were recorded for each video, resulting in 20 ‘.pcapng’ files total.

A Python script named `createdataset.py` was developed to preprocess the raw capture files. Using the `pyshark` and `tqdm` libraries, each ‘.pcapng’ file was filtered by `udp.port == 443`, which corresponds to QUIC traffic typically used by YouTube’s adaptive streaming service. The script extracted two features per packet: `packet_length`, signed to indicate directionality, and `arrive_time_delta`, the inter-arrival time between packets. Each processed file was appended to a JSON list under its respective video label (e.g., `cat.json`, `dog.json`), resulting in a structured dataset.

The processed dataset was used with the open-source repository https://github.com/jmhIcoding/traffic_classification_utils, which includes

multiple encrypted traffic classification models. For this study, a one-dimensional Convolutional Neural Network (1D-CNN) model replicating the architecture used in Schuster et al.’s *Beauty and the Burst* [1] was employed. The model consists of three consecutive convolutional layers with ReLU activations (each with 32 filters and a kernel size of 5), followed by dropout and max pooling. The network concludes with a dense layer and a softmax output for classification. Input traces were standardized to a fixed length of 200 packets, and the model was trained for 200 epochs using the Adam optimizer with a learning rate of 0.0008.

The preprocessed JSON files were passed to the model, which internally applies burstification, padding, and segmentation. The CNN was then trained to classify samples based solely on encrypted metadata.

3.2 360-Degree Video Capture (Planned)

To evaluate the methodology’s scalability to 360-degree videos, additional captures will be conducted using a Meta Quest 2 headset. A Windows machine configured as a mobile hotspot will serve as the access point, allowing traffic from the VR device to be isolated and recorded. The YouTube VR application will be used to play two 30-second immersive videos, each for around ten trials.

Wireshark will again be used to capture network traffic, which will be filtered and processed using the same `createdataset.py` script as in the 2D case. Although 360-degree videos are expected to produce higher data volume and more complex burst structures, the same modeling and classification pipeline will be applied to assess its ability to generalize across video formats.

Metric of Success: A successful outcome for this project will be that the machine learning model successfully identifies 360-degree video titles based on the analysis of encrypted traffic patterns. By analyzing features such as burst intervals and transmission frequencies, the model should accurately differentiate between various 360-degree videos.

4 Results

Before adopting the external GitHub model, an initial experiment was conducted to assess the feasibility of building a lightweight custom classifier using features extracted from QUIC traffic. A Python pipeline was developed in

two stages: first, a script used `tshark` to filter QUIC packets from `.pcapng` files and extract relevant metadata (timestamp, packet length). These values were stored in individual CSV files corresponding to each video capture.

In the second stage, burst sequences were extracted by applying a 2-second interarrival time threshold to identify periods of concentrated activity. In other words, treating any gap of more than two seconds between consecutive packets as the start of a new burst. For each burst, the sum of packet lengths was calculated. A simple fully connected neural network (FCNN) was then trained to classify bursts as either “cat” or “dog” video samples. Each full capture file was classified based on a majority vote of burst-level predictions.

Despite tuning the model architecture and training it over 1000 epochs, this method failed to yield consistently accurate results. Classification accuracy hovered around chance level on the test set (`cat5.csv` and `dog5.csv`), indicating that this approach was insufficiently robust for generalization.

Consequently, the project transitioned to using the established model from the GitHub repository https://github.com/jmhIcoding/traffic_classification_utils. This model is based on the *Beauty and the Burst* CNN architecture [1] and was better suited to capturing complex patterns in encrypted traffic.

The GitHub repository implementation initially posed challenges due to deprecated TensorFlow APIs and undocumented preprocessing expectations. These issues were addressed and are discussed in Section 6.2.

Results

The CNN model was tested on progressively larger datasets, starting with 5 dog and 5 cat samples. The model achieved 100% training and validation accuracy by epoch 27, and 100% test accuracy on the held-out capture. Precision, recall, and F1-score were all 1.00, suggesting that the model was able to memorize the training data and generalize effectively to unseen data in this limited case.

To further evaluate the model’s robustness, the experiment was repeated with 10 dog and 10 cat samples. Training initially showed signs of instability, with test accuracy remaining at 0% for the first 20 epochs and high variability in loss values. However, the model eventually converged, reaching 100% accuracy on both validation and test sets by epoch 46. Subsequent epochs maintained this perfect performance, indicating that the model was capable

of learning robust features when provided with slightly more data.

Overall, these results demonstrate that while small datasets can lead to unstable learning dynamics, the *Beauty and the Burst* model can achieve perfect classification performance even on minimal traffic samples when tuned properly.

5 Discussion

5.1 Challenges and Implementation Effort

Substantial effort was required to adapt the open-source GitHub repository for this project’s needs. Originally developed for older environments, the repository relied heavily on deprecated TensorFlow 1.x functions and Keras models saved using outdated APIs. Key components—such as model training loops, evaluation logic, and model serialization—were not compatible with TensorFlow 2.x. As a result, the codebase had to be systematically refactored.

The transition involved rewriting legacy `tf.Session()` and placeholder-based logic into TensorFlow 2’s eager execution model, which uses dynamic computation graphs. Layers were redefined using the updated `tf.keras` syntax, and the training process was rewritten using the `Model.fit()` API for compatibility. Additionally, the original model saving methods (e.g., `tf.train.Saver()`) were replaced with `model.save()` and `model.load_model()` to support the HDF5 and SavedModel formats.

A lack of documentation posed additional challenges. While the model code was available, expectations for preprocessing—such as how to format the JSON dataset, structure input bursts, and label samples—were not clearly explained. Understanding these requirements involved inspecting the codebase and testing different input formats to match what the model expected. Some inconsistencies, such as burst length mismatches and label alignment issues, required manual adjustments to ensure compatibility.

Despite the hurdles, these efforts resulted in a fully modernized and stable version of the original repository, now compatible with TensorFlow 2.x and suitable for ongoing experimentation. The refactored codebase preserves the core logic of the original *Beauty and the Burst* architecture while providing a much more accessible foundation for future research and replication.

5.2 Informal User Interface Evaluation

To evaluate the usability of the data preprocessing pipeline and model workflow, an informal user interface test was conducted with two peer participants. While the system lacks a graphical interface, the main user-facing components include the `createdataset.py` script, the JSON dataset format, and the classification workflow using the GitHub-based CNN model.

Each participant followed a draft `README.md` guide that outlined four key steps: (1) capturing a video stream using Wireshark, (2) saving the `.pcapng` file, (3) running the preprocessing script to convert the capture into a labeled JSON file, and (4) executing the model to classify the video and view the output.

Both participants were ultimately able to complete the process, but encountered several points of confusion along the way. One participant struggled with Python package installation due to missing dependencies, while the other was confused about where to place the capture file and how to confirm it had QUIC packets. Additionally, the classification output was initially unclear without guidance on interpretation.

Based on their feedback, the `README.md` was revised to include a clear list of required packages, step-by-step instructions with example commands, and visual examples of expected outputs. Specific tips were added on using Wireshark filters and verifying QUIC packet presence. These improvements aim to reduce confusion and make the pipeline more accessible for future researchers with minimal setup time.

5.3 Planned Extensions Before tjSTAR

Initial results demonstrate that the CNN-based approach can effectively distinguish between simple video categories (e.g., cat vs. dog) using features extracted from QUIC-encrypted traffic. However, the current evaluation uses a limited dataset of 10–20 videos. To better assess generalizability, future experiments will expand the dataset to include more video titles, more burst samples per class, and test on 3D captures. These extensions will help determine whether the model can scale to more realistic traffic classification scenarios.

6 Conclusions

This study explored the feasibility of fingerprinting video content from QUIC-encrypted traffic using burst-level features and deep learning. While an initial attempt with a lightweight neural network was unreliable, transitioning to the more sophisticated *Beauty and the Burst* CNN architecture yielded promising results. With even modest training data, the model achieved perfect accuracy on test captures, suggesting that distinctive patterns in packet bursts can serve as signatures.

The process of adapting the open-source codebase involved considerable effort. This work now provides a usable baseline for further research. Planned future work includes scaling up the dataset, conducting usability testing, and applying the pipeline to more complex video formats such as 360 degree media. Ultimately, this line of research contributes to a growing body of literature demonstrating that encrypted traffic still leaks meaningful information under certain side-channel conditions.

It reinforces growing concerns about the limitations of encryption in protecting user privacy.

7 References

- [1] Schuster, Roei, et al. "Beauty and the Burst: Remote Identification of Encrypted Video Streams." 26th USENIX Security Symposium, USENIX Association, 2017, pp. 1357-1374. <https://www.usenix.org/conference/usenixsecurity17/technical-sessions/presentation/schuster>
- [2] Cheng, Ruizhi. Guidebook of Measurement Study on Social VR Platforms. 4 Mar. 2022.
- [3] Zhao, Li, et al. "A Robust and Accurate Encrypted Video Traffic Identification Method Via Graph Neural Network." Institute of Electrical and Electronics Engineers (IEEE), 2023. ProQuest, <https://www.proquest.com/conference-papers-proceedings/robust-accurate-encrypted-video-traffic/docview/2828940864/se-2>
- [4] Li, Ying. "Classifying Encrypted WiFi Traffic using Deep Learning Methods", University of Technology Sydney, 2022. ProQuest, <https://www.proquest.com/dissertations-theses/classifying-encrypted-wifi-traffic-using-deep/docview/2877960552/se-2>

[5] JmhIcoding. (2024). *traffic_classification_utils* [GitHub repository].
https://github.com/jmhIcoding/traffic_classification_utils