MCM1 Practicum Report

Video Identification Based on Encrypted Network Traces and Attack Mitigation

1st Deborah Djon; 22261972
School of Computing
Dublin City University
Collins Ave Ext, Whitehall, Dublin 9
deborah.djon3@mail.dcu.ie

2nd Darragh Connaughton; 13469978
School of Computing
Dublin City University
Collins Ave Ext, Whitehall, Dublin 9
darragh.connaughton5@mail.dcu.ie

Supervisor: Geoff Hamilton
School of Computing
Dublin City University
Collins Ave Ext, Whitehall, Dublin 9
geoff.hamilton@dcu.ie

***Abstract*—In an era of increasing digital privacy concerns, side-channel attacks pose a risk to the privacy of many streaming service clients. With the attack outlined in this study, it is possible to detect what YouTube video a user is watching despite network traffic encryption. An attacker can achieve this by performing a side-channel attack that targets the byte sizes of the Hypertext Transfer Protocol (HTTP) requests transmitted between the user's browser and the YouTube server. This study investigates the feasibility of detecting YouTube videos using high-level network data captured from application logs. Different machine learning models and statistical methods were created for identifying fingerprinted YouTube videos with side-channel data in an open-world scenario, where most of the examples the model is presented with are unknown. Our research shows that it is possible to identify YouTube videos with an F1-Score of 80% in an open-world scenario. We further show that it is possible to distinguish known and unknown videos with an recall of 92% in the same open-world scenario.**

***Index Terms*—sca, machine learning, cyber security, cnn, timeseries**

I. Introduction

The popularity of video streaming is undeniably on the rise, evident in the unprecedented growth of platforms like YouTube, Netflix, and Disney+. To ensure users' privacy and protect sensitive content, network traffic encryption technologies such as Hypertext Transfer Protocol Secure (HTTPS), Virtual Private Networks (VPN), and The Onion Router (TOR) have been widely employed, rendering data unreadable to unauthorized entities. Despite these protective measures, stakeholders like Internet Service Providers (ISP) or universities may still seek to restrict access to specific content within their networks. Research highlights the feasibility of a Side-Channel Attack (SCA) on network traffic traces, which enables the identification of user content, and underscores the urgent need for enhanced security measures. This research delves into the feasibility of the SCA on data at the application layer as per the Open Systems Interconnection (OSI) model. Although network encryption renders video identification through Deep Packet Inspection (DPI) ineffective, videos can still be identified by extracting patterns in the sizes of packages traversing the network. Major streaming providers use the streaming content delivery technique Dynamic Adaptive Streaming over HTTP (DASH). The use of DASH improves a user's Quality of Experience (QoE) by reducing stalls at video play time [1]. Several studies show that using DASH for streaming content delivery leaves a unique pattern in streaming traffic that can be extracted and used to identify specific videos [2], [3], [4], [5], [6].

The primary contributions of this study are as follows:

- An extensive dataset with 6289 records that includes 50 traces of 50 videos each for fingerprinting
- A data collector that can automatically and robustly gather bitrate information from a given set of YouTube videos
- A comparison of different models that can detect fingerprinted YouTube videos based on bitrate information in an open-world scenario
- An implementation of a timing attack as a proof of concept that verifies the feasibility of our attack scenario

In this study, we analyse the feasibility of performing a side-channel and timing attack using the sizes of the requests sent between the user's browser and the YouTube server. Although there is no strict rule that defines the number of required samples per class for a Convolutional Neural Network (CNN), a rule of thumb for neural networks is to have several thousand samples per class for a neural network to perform well [7]. Additionally authors like [3] achieve high performance using a distance-measure approach for the task of video identification. Because model complexity can lead to overfitting and thus lower performance on different datasets, we analyse the use of a CNN and the less complex statistical methods logistic regression and Dynamic Time Warping (DTW) [8]. We aim to answer the following questions:

1) To what extent is there an identifiable pattern in application layer data from YouTube streams?
2) How well does a CNN perform at predicting YouTube videos in an open-world scenario using bitrate information compared to the less complex statistical methods of logistic regression and a DTW-based model?
3) How well can timing-attack data be correlated to application layer data in YouTube video streams?

The remainder of the paper is structured as follows. We first provide relevant background information in Section II. Section III outlines our research design and is followed by descriptions of our work and results in Section IV. Subsequently, we critically evaluate and relate our work with similar works in the field in Section V. Finally, Section VI summarises our work and it's future impact on the field of network security in streaming content delivery.

II. Background

The following sections outline the relevant background information required for undertaking the study. Detailed are the proposed threat model, the streaming technology which introduces the vulnerability, video fingerprinting and identification techniques.

A. Threat Model

The proposed threat model is similar a threat model to the author's in [9], in which a router is the shared resource between the attacker and the victim. Introducing contention at the shared resource, through the use of rapid probing at steady intervals, exposes the victim's network traffic data through a timing side-channel attack. The attacker's probes will be forced into the routing queue whenever the victim's network data is traversing the router at the network perimeter. Assuming, that the length of the delay is correlated with the bytes per second of the victim's current communication, the attacker seeks to predict the YouTube video the victim is watching. For prediction a model previously trained with the collected side-channel data is used. In this threat model, we assume the attacker and victim are operating from two disjoint networks, but both have access to the victim's public router.

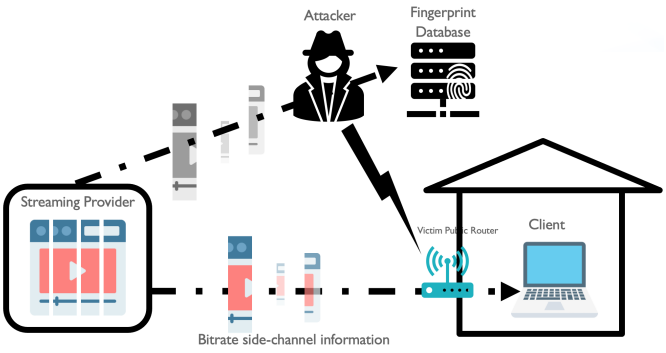

Fig. 1. Threat model

B. Dynamic Adaptive Streaming over HTTP

DASH is an implementation of Adaptive Bitrate Streaming (ABS), which seeks to maintain the user's QoE. With ABS videos are encoded in multiple distinct representations of varying quality levels with each representation segmented across time [10], [3], as shown in Figure 2. A streaming session is a series of sequential client-initiated HTTP requests, for individual DASH segments, in which the client can adapt the quality level of the next requested segment in response to changing network conditions. The streaming session can be divided into an initial playback buffer hydration stage followed by a steady-state stage where data is transferred at regular intervals [11].

DASH uses the H.264 industry standard for video compression. H.264 encoding reduces the size of the digital video through the removal of redundancy within and between

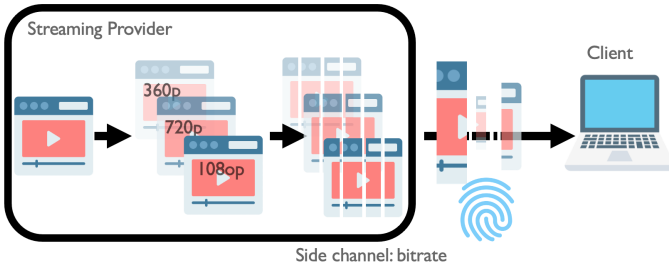

Fig. 2. DASH illustration

frames among other techniques [12]. H.264 encoding divides frames into Macroblocks of N x N pixels. Prediction is used to find similarities between Macroblocks within and across adjacent frames [13]. The encoding process results in a unique representation across videos based on video-specific characteristics. The correlation between encoded on-disk segments and network traffic patterns, coupled with the sequential nature of DASH, leaks a video-specific identifiable pattern even in encrypted traffic for certain videos [3].

C. Video Fingerprinting

DASH's adaptive bitrate strategy leads to variances in traffic measures of the same video captured at different points in time. Figure 3 is exemplary of the identifiable pattern, displaying three traffic traces of the same video. However, it is evident that these traces have variances in the size of the packages and shifts in time. A video fingerprint must be able to capture a unique pattern despite these variations.

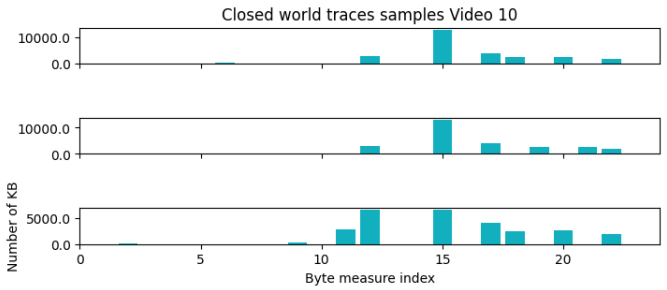

Fig. 3. Example traces of the same video

We define a video fingerprint f as the pattern that characterises the sequence of package sizes $s = (s_1, s_2, ..., s_n)$ associated with streaming a specific YouTube video. Several methods can be implemented to fingerprint a YouTube video.

One method to create such a fingerprint is grouping the package sizes into periods, where each period contains all measures assumed to belong to one dash segment [4], [3]. Figure 4 illustrates this process. This can mitigate differences in time [14] [15].

Variations in package sizes of different traces of the same video can be mitigated by regarding the relative differences

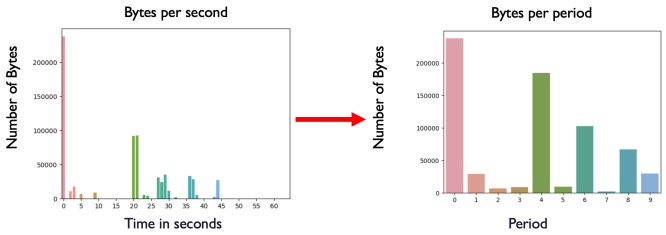

Fig. 4. Traffic trace aggregation into bytes per period

between bitrate measurements. One such method is the Differential Fingerprint (DF) described in Equation 1. The fingerprint is a sequence of fingerprint measures f_i that represent the difference between two adjacent sequence measures s_i and s_{i+1} in proportion to the first measure [4] and [3].

Other possible fingerprints include the simple difference fingerprint in Equation 2, and the absolute difference fingerprint in Equation 3 [4]. In addition to previous authors we define the magnitude preserving fingerprint in Equation 4, which aims to capture positive and negative differences better than the regular differential fingerprint, by normalising two adjacent measures by the larger value of the two values. Lastly, we define normalised differential fingerprint in Equation 5 which produces with fingerprinting values between 0 and 1.

$$f_i = F_{df}(s_i, s_{i-1}) = \frac{s_i - s_{i-1}}{s_{i-1}} \tag{1}$$

$$f_i = F_{sdf}(s_i, s_{i-1}) = s_i - s_{i-1} \tag{2}$$

$$f_i = F_{adf}(s_i, s_{i-1}) = |s_i - s_{i-1}| \tag{3}$$

$$f_i = F_{mpdf}(s_i, s_{i-1}) = \frac{s_i - s_{i-1}}{max(s_i, s_{i-1})} \tag{4}$$

$$f_i = F_{ndf}(s_i, s_{i-1}) = \frac{s_i - s_{i-1}}{(s_i + s_{i-1})} \tag{5}$$

The traffic traces are time series by nature. Thus, the fingerprint can also consist of extracted time-series features such as the Fourier entropy, the number of duplicate values, or variance.

Another option for fingerprinting is automatic pattern extraction through a CNN as in [5]. A CNN has the capability to recognise more complex features not evident to humans or outside of well-known time-series features.

D. Video Identification

Video identification involves comparing a sample of a victim's traffic to all fingerprints in the attacker's fingerprint database and producing a classification.

Video fingerprints in the form of a fingerprinting sequence f can be compared using distance measures such as DTW. DTW is a similarity measure often used for time series. The algorithm can compare sequences with differing speeds and time delays. The algorithm computes an optimal path between two series that minimises the distances between them. Figure 5 illustrates the DTW alignment of two normalised traffic traces of the same video. On the left-hand side and bottom of the first graphic are the sequences that are compared. In the centre of the left graphic is the path that minimises the distance between the two sequences. The right graphic shows the two different traces (blue and black) and the points in each trace the algorithm identified as equivalent. These points are connected through dotted lines. DTW enables alignments that are non-linear and thus makes the distance measures preferable over linear distance measures such as cross-correlation [16]. [3] demonstrates the feasibility of using Partial DTW (P-DTW), which has the potential to align sequences of differing lengths and find partial alignments.

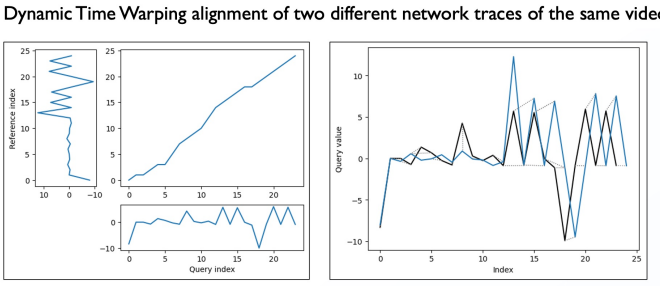

Fig. 5. Dynamic Time Warping Alignment

Another option for fingerprinting is employing an automatic approach using deep neural networks. [5] and [9] demonstrate the use of CNNs for automatic video identification. Although the CNN can be used for both fingerprint extraction and video identification [5], previous authors have also achieved successful classifications using hand-crafted features like a Simple Difference Fingerprint (SDF) as input to their CNN [4], [9].

III. Methodology

A. Attack Scenario

The attack scenario consists of a victim watching YouTube via an internet-connected router which is accessible to the Attacker. Two Raspberry Pi model 4 devices were used to simulate the router and victim devices. The victim router connects to the available internet-connected router using the Raspberry Pi's inbuilt wireless network adapter on interface wlan0. An external USB wireless Ethernet adapter is used to create a second private Wireless Local Access Network (WLAN), wlan1. Access Point (AP) capabilities are handled by the Linux utility hostapd [17] and was configured to use Ethernet 802.11g and set ignore_broadcast_ssid to false to allow our victim to identify the AP. The dhcpd [18] utility was used to convert the router into a Dynamic Host Configuration Protocol (DHCP) server listening on the wlan1 interface, equipping the router with the ability to handle incoming DHCP Discovery requests coming from the victim machine and any

other host which joins the private WLAN. Finally, to prevent the server from dropping traffic intended for another server, IPv4 forwarding was enabled.

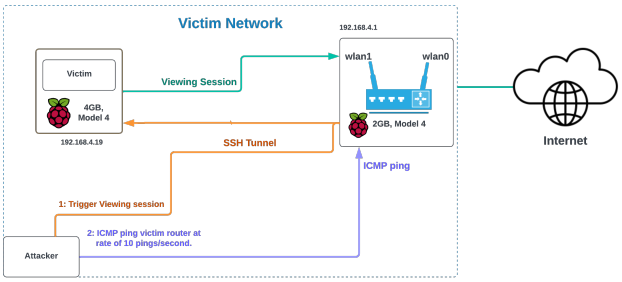

Fig. 6. Attack Scenario

B. Data Collection

We collected a dataset of 6289 YouTube video bitrate measure traces, including 50 traces of 50 videos for fingerprinting. Additionally, one or more traces of 3665 videos each are captured for the open-world scenario, described in Section III.C.1. For each video, we capture the number of bytes transmitted at a given timestamp between the user's browser and the YouTube server for an interval of two minutes.

The data capture sessions are performed using Selenium [19], powered by the Firefox geckodriver [20]. Selenium provides a means of simulating browser sessions programmatically. The geckodriver was chosen to avoid CAPTCHA, a prevalent occurrence when interacting with YouTube using the chromium driver [19]. To enforce a blank slate across viewing sessions, browser caching functionality was disabled. Disabling the cache ensures the browser retrieves the entire contents of the video for each viewing. Finally, a Firefox profile with AdBlocker installed seeded each browser session to ensure the network traffic data pertained solely to the contents of the requested video.

Previous authors trained models on network layer data, captured by reading device's network interface. Obtaining data at this level of granularity requires read permission to the device's network interface, which is most often restricted to the root user [21]. In this paper, application-level data is captured from the Open Source proxy.py's [22] standard output. A successful Software Supply Chain (SSC), in which adversaries seek to inject malicious code third-party proprietary and open-source software used by downstream developers, could allow an adversary to exfiltrate the aforementioned metadata without requiring a foothold on the victim's network or elevated privileges [23].

1) URL Validation: Validation was required to extract the set of valid URLs, which were said to be videos 2 minutes or longer, out of the superset of 9000 URLs gathered. Selenium was used to extract the source of each URL, which was subsequently parsed to retrieve the video length, subscriber count and view count. The validation component condensed

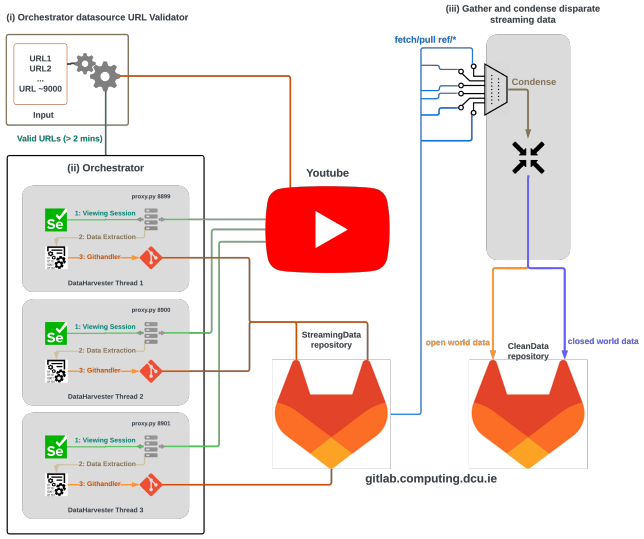

Fig. 7. DataHarvester Architecture

the approximately 9000 URLs to a set of approximately 6000 valid URLs.

2) Data Collection Orchestration: The functionality necessary to retrieve N view traces of a specific URL was encapsulated in the DataHarvester Thread (DHT) class. The orchestrator acts as a main program which converts the set of valid URLs into a pool of DHT tasks which are subsequently processed by the ThreadPoolExecutor class. An upper bound of three concurrent workers was chosen after witnessing a noticeable, predictable degradation of video quality when this limit was breached.

The DHT begins by initialising the Context Managed Githandler class. The Context Manager design pattern in Python allows the developer to define default behaviour upon entering and exiting the object. Upon entry, Githandler clones the master branch of the streamingdatarepository repository to a DHT specific location on disk and creates a remote branch whose name is a Unique Identifier (UID) consisting of an amalgam of the URL, port and timestamp. The decision to use a unique branch for each DHT instance, while greatly bloating the number of branches, removed the potential for fast-forward errors and merge conflicts. The DHT finishes by initialising the Harvester class. The Harvester is responsible for gathering N traces of the specified URL via the Viewer and Proxy classes. The Selenium viewer session is proxied through a DHT-specific proxy instance to enforce network isolation and to capture application layer traffic. The Open Source proxy.py [22] library was used for this purpose. The Viewer class handles the entire interaction with YouTube via Selenium, including navigation through the various popup menus, clicking the play button and watching the video for the specified duration. Upon terminating the viewing session, a kill signal is sent to the various proxy processes, the standard

output is captured and subsequently fed into an extractor which retrieves the relevant metadata. The data is converted to a pandas DataFrame, written to disk added, then finally added, committed and pushed to GitLab.

3) Gather and Condense Streaming Data: The final stage of the data collection effort gathers the streaming data from the disparate Git branches created by each respective DHT instance and condenses them into a large DataFrame ready for analysis. To efficiently iterate through each branch reference, a PullFetchThread class was created which is responsible for fetching the data from a specific remote branch. Once gathered, the data is condensed into an open-world and closed-world pandas DataFrame and stored in the cleandata repository.

4) Trigger Viewing Session: The YouTube viewing functionality from the aforementioned DHT class was deployed on the victim's device which allows us to simulate the victim passively viewing YouTube. To trigger an attack, an Secure Shell (SSH) tunnel was created through the victim's router to the victim's device. A command traverses the tunnel from the attacker to the victim host to initiate the viewing session. Upon exiting, the start time of the viewing session is returned to the victim which is used to trim the irrelevant data from the side channel data.

5) Internet Control Messaging Protocol Fingerprint Attack: A separate process was launched prior to triggering the remote viewing session executing a basic script which will send Internet Control Messaging Protocol (ICMP) Ping requests to the IP address of the router's external wireless Universal Serial Bus (USB) adapter at a rate of 10 pings/second. The script prints the timestamp and the Round Trip Time (RTT) of each PING request in the standard output, which is subsequently captured and parsed upon terminating the attack process. The start time returned from step one is used to trim the irrelevant attack data occurring prior to the start of the attack. We collected 3 traces for each of the closed world videos.

C. Video Detection Models

We employ and compare three models for video identification in our study, a CNN, a logistic regression and a Partial DTW-based model. We use these models, to validate and improve upon previously deployed models. We also test their performance in a closed and open-world setting. We compare the models by means of accuracy and F1 score and recall on the "unknown" class in the open-world scenario.

1) Open-World Scenario: We define an open-world scenario for our attack where most traffic traces encountered are unknown to the model. Therefore, the aim is to create a model that correctly rejects the large number of unknown traces with high recall, yet confidently identifies known traces with high precision and recall. Therefore, we choose the macro-averaged F1 metric to evaluate our models, as it balances the recall and precision and is not biased towards a majority class.

To reflect the above-described open-world scenario we create training and test datasets with an 80/20 train test split of our 2500 fingerprinting traces. The training and test sets are stratified, i.e. the proportion of samples per class is equal. For the open-world models, we add 3000 open-world traces to the closed-world test dataset and 662 open-world traces to the training dataset, to reflect the dominance of unknown traces in an open-world scenario. Depending on the model, we sample differing amounts of "unknown" samples from the open-world traces for the training set. All models are created with the Python Libraries TensorFlow [24], scikit-learn [25] and TSFRESH [26].

2) Data Analysis: A thorough analysis of the dataset is performed before the models are created. The following sections describe key-findings relevant to modelling.

Fingerprint Uniqueness

Visual inspection and hypothesis testing reveal significant similarities between application layer traffic traces of the same video. We use DTW-barycenter averaging to analyse patterns in traces of different videos. DTW-barycenter averaging is a method for averaging time series, that synthesises a new series that minimises the DTW-distances to each element in a set of time series [16]. Figure 8 exemplary shows the DTW-barycenter averages of three videos in our dataset (blue). All of the original traces belonging to the respective video are displayed in grey.

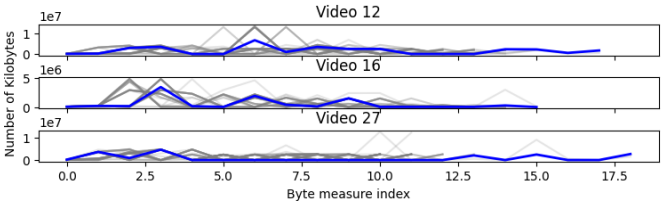

Fig. 8. Averaging fingerprints

The similarity of video traces stemming from one video (within-group similarity) and the dissimilarity of traces stemming from different videos (between-group similarity) is examined by means of hypothesis testing. The following hypotheses are analysed.

1) Traces belonging to one video are highly correlated with a maximal cross-correlation of above 0.5.
2) Traces belonging to different videos are not highly cross-correlated, having a maximal cross-correlation of below 0.5.

Cross-correlation is a similarity measure, often used to compare time series. It measures the correlation between two-time series at different offsets and does thus consist of multiple correlation measures. From the cross-correlation, the highest value is chosen for testing the hypothesis.

For the within-group similarity, we calculate the cross-correlation between samples of the same video. For the between-group similarity, we calculate the cross-correlation between traces of different videos. Due to the exponential complexity of this problem, we sample a big enough sample size from our dataset and perform t-tests.

For the within-group similarity, we sample six traffic traces and another six different traffic traces of each of the 50 classes. We then cross-correlate the 1800 pairs and use a t-test to analyse, if the mean of the sample is significantly above 0.5. Our test concludes, that at a 0.5% significance level, the samples are significantly highly correlated. The mean correlation of the samples is 0.77.

Similarly, the same tests for analysing the between-group similarity are performed. For this, we choose six traces for each video and cross-correlate them each with another 6 traces from different videos. We could not conclude that the between-group differences are significantly below 0.5. The mean is 0.57. Although the traces of different videos are cross-correlated, the within-group cross-correlations are significantly higher than the between-group correlations, which leaves enough evidence to conclude the traffic traces belonging to one video are distinguishable from traces of other videos.

Binning Intervals for the Traffic Measures

The traffic measures are not taken in steady time intervals. Similar to previous authors we analyse the effect of different discretisation binning intervals [3] on the within and between-group cross-correlations. Figure 9 shows that, unlike the previous authors, the bin size has no apparent effect on the group coherence. This can be explained by our data being less granular and already aggregated as we are capturing the package sizes on the application level, rather than the network level.

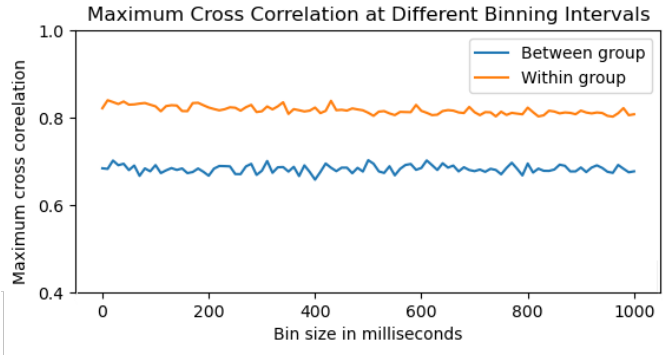

Fig. 9. Effect of bin-size on cluster coherence

Fingerprinting Method

Lastly, we analyse the effect of the different fingerprinting methods on the fingerprint group cohesion. As shown in Figure 10 the Differential Fingerprint (DF), leads to the highest within-group cross-correlation. However, it also produces a high between-group cross-correlation. This is not ideal as it reduces group cohesion. The simple difference fingerprint and the Magnitude-Preserving Differential Fingerprints (DF) are chosen for subsequent analysis and modelling as they lead to high group cohesion.

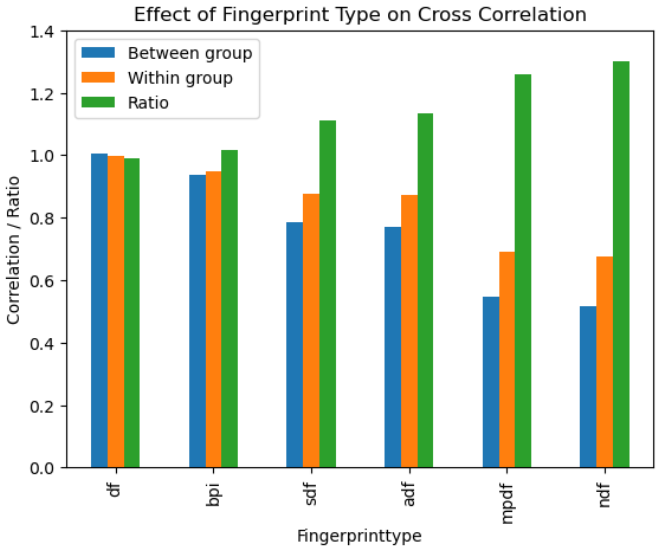

Fig. 10. Group coherence by fingerprint type

Correlating Timing Attack Data

The uniqueness of the timing data traces as well as their similarity to the application-layer traces was analysed. First, all Round Trip Times (RTT) below 0.03 seconds were filtered out. They indicate a low number of packages being transmitted between the browser and the YouTube server and are thus considered noise. Next, the RTTs are aggregated into 6-second bins according to the data aggregation proposed by [3].

The within and between-group cross-correlations and DTW distances are analysed. As the data are of different nature cross-correlation alone may not yield accurate results, as the measure is not suited for signals stretched in time. No significant difference in the within- and between-group similarities of the RTTs data could be found. Three RTT traces are compared to another 6 traces from the application layer data. Comparing the similarities of RTT and application level traces of the same and different videos showed no significant difference. This indicates a low probability of an identifiable pattern fingerprint in the RTT data and connection between the two datasets.

3) Data Pre-Processing: The byte size measures are discretised into second bins as a form of data and noise reduction. This transformation is performed confidentially as the binning interval has no significant effect on group cohesion.

Next, all fingerprints outlined in sec:videoFingerprintingare calculated from the Bytes Per Scond (BPS) measures. The logistic regression and CNN models require data input of equal length. For this two versions of the dataset are created. In the first version, all fingerprints are extended with a padding value to match the length of the longest fingerprint. In the second version, a 120-long array where each element represents one second in the two-minute data-capture interval is created. Each data point in a BPS sequence is then assigned to the corresponding index in the array.

4) Logistic Regression: For the logistic regression the Python library TS-fresh is used for automatic feature ex-

traction and selection [26]. Two options for a logistic regression were considered, a multinomial logistic regression, and a one-vs-rest logistic regression. A multinomial logistic regression assumes all possible classes are known. In a one-vs-rest logistic regression, different binary classifiers are trained for each of the N given classes. As the given problem must deal with "unknown" samples, a multinomial regression prived not ideal. Therefore, we employed a one-vs-rest logistic regression, which was tested in the open and closed-word. For the open world, we define all predictions that cannot be assigned to one of the 50 classes by any of the one-vs-rest classifiers with a probability of above 0.2 as "unknown". Among values ranging from 0 to 1, 0.2 proved the best-performing threshold. 370 features, such as the autocorrelation lags, are automatically extracted and selected from the dataset using TSFRESH. Further analysis of the features revealed high variance inflation factors indicating high multi-colinearity. This violates the assumption of independent variables of for logistic regression and reduces the model parameter's significance [27]. After removing these factors 72 features remain for modeling. Different datasets with the MPFP, as well as the SDF, are tested. The best performance is reached with an MPDF, the reduced 72 features, and an L2 regularisation inverse regularisation strength of 0.1.

5) Dynamic Time Warping Model: We follow [3] and implement their P-DTW-based model for video fingerprinting and identification. For this, we first create a fingerprint database consisting of MDFP and SDF fingerprints. Prediction is performed by selecting the closest matching fingerprint in the database to a given transformed traffic trace. The authors define one fingerprint for each of the classes they regard, by collecting one traffic trace. However, in our case, we have a number of traces from which we need to choose one fingerprint. Therefore, we investigate, the characteristics of a good fingerprint. We hypothesise that longer traces of the same classes with more variance, carry the most identifiable information and are thus best for prediction. Further, we investigate the benefit of averaging several fingerprints. To test these hypotheses we first select the ten highest variance traces for each of the 50 videos. We then create ten synthesised averages of each of the one to ten highest variance traces. Analogously, we repeat the process for the ten longest traces. The longer traces do not improve prediction, averaging the 10 highest variance traces yields the best results.

6) Convolutional Neural Network: We follow [4] and [9] and recreate a CNN for video fingerprinting and identification. Both the 120-dimensional vector as well as the padded fingerprint measures for the MPDF and SDF were tested. The best performance was reached with the SDF. To prevent overfitting, the drop-out rate of the CNN was increased to 0.9, the number of epochs was reduced to 100 and early stopping was introduced. The architecture of our model is displayed in Figure 11.

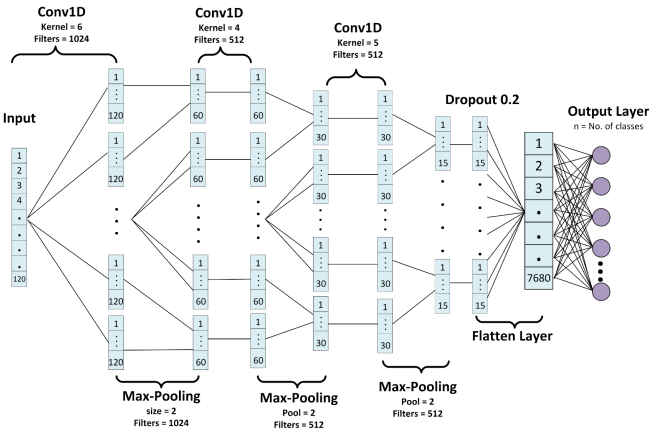

Fig. 11. CNN architecture [4]

IV. Results

As shown in table I of all tested models the CNN performs best in the open and closed-world scenario. With an F1 score of 80%, it outperforms itself in the open-world scenario compared to the closed-world scenario. In addition, it distinguishes between known and unknown traces with 92% recall. The logistic regression performs better than the DTW model, both in the open and closed-world scenario. The DTW model performs worst out of all tested models, with an F1 score of 17%.

TABLE I
Models Test Scores

Closed World			
Model	**DTW**	**LR**	**CNN**
F1 score	0.39	0.63	0.79
Accuracy overall	0.46	0.64	0.78
Open World			
Model	**DTW**	**LR**	**CNN**
F1 score	0.17	0.31	0.80
Accuracy overall	0.18	0.75	0.78
Recall negatives	0.12	0.27	0.92

V. Discussion

Our work is related to previous works in the field of video fingerprinting in several aspects. Our models work with higher-level data than previous authors like [4], [9] and [3]. We are working on data that is based on HTTP package sizes, which is layer seven data, whereas previous authors work with lower-level and thus more granular data.

We followed the CNN architectures of [4]. With minor changes to the architecture, we were able to validate their results on a new dataset and confirm their model works comparably well in an open-world scenario.

We based our DTW model on the works of [3], yet could not replicate their results in the open or closed-world scenarios. This could potentially be due to an inherent bias towards low-variance fingerprints of the model. Our model predicts

traces with a low variance significantly more often than traces with a high variance. This, however, is only a correlation, not causation and requires further research. Although the model performs worse out of all tested models, with an F1 score of 17% it outperforms the baseline of 1/51 or 2%.

To our knowledge, we are working with the largest dataset for the task of YouTube video fingerprinting in the reviewed literature. Only [9] employed an open-world scenario, and we were able to improve upon their results. They claim a false positive rate of 0 however, this only applies to 20% of their data, as they could not extract an identifiable fingerprint from the remaining data. Although we simulated our data in a more realistic scenario, the open-world, it has to yet be tested in a real-world setting.

To our knowledge, we are the first in the field to employ a logistic regression for the task of YouTube video identification. However, the model's performance was significantly reduced in the open-world scenario, despite implementing a number of regularisation and feature engineering techniques. We suspect this is due to a lack of linearity between the input features and the logit (dependent variable in logistic regression) which violates the assumptions of logistic regression. Options for improving the model could include adding higher-order terms to discover such a relationship. Perhaps a non-linear model such as a k-nearest neighbour or a decision tree could be explored in future research.

VI. Conclusion and Future Work

We have conducted extensive research that implements, compares and improves existing and new models for YouTube video fingerprinting and identification. In this work, we were able to answer all of the proposed research questions and open up new areas for future research.

We were able to extract and identify patterns from application-level data with all proposed models. All models perform better than the baseline of 2%. The CNN model performs best in the open and closed-world with an F1 score of 80 in the closed world and a recall of 92% on the "unknown" class. Contrary to our initial belief, this shows, that the CNN model can effectively extract patterns from a relatively small number of samples per class. This may be due to the lack of complexity of the data, as the data is 1-dimensional. However, this aspect requires further research.

In contrast, we were not able to reach similar performances with our statistical models. Logistic regression, as well as the DTW, fail to generalise on unrelated samples of the open-world. They are thus, not suited for a real-world implementation of the attack scenario.

We could not find a correlation between the timing data and our application-level data. This reduces the likelihood of our proposed attack working in a real-world scenario. Nonetheless, the results of this research can be used to employ similar attacks that directly target application-level data as described in Section III.B. This requires further research.

Finally, we collected an extensive dataset containing more than 6000 open and closed-world traces that we made openly available for further research.

References

[1] Michail Michalos, Stelios Kessanidis, and S.L. Nalmpantis. "Dynamic Adaptive Streaming over HTTP". In: *Journal of Engineering Science and Technology Review* 5 (June 1, 2012), pp. 30–34. DOI: 10.25103/jestr.052.06. (Visited on 04/25/2023).

[2] Vera Rimmer et al. "Automated Website Fingerprinting through Deep Learning". In: *Proceedings 2018 Network and Distributed System Security Symposium*. Feb. 17, 2018. DOI: 10.14722/ndss.2018.23105. (Visited on 10/05/2022).

[3] Jiaxi Gu et al. "Traffic-Based Side-Channel Attack in Video Streaming". In: *Ieee-Acm Transactions on Networking* 27.3 (June 1, 2019), pp. 972–985. DOI: 10.1109/TNET.2019.2906568. (Visited on 10/03/2022).

[4] Waleed Afandi et al. "Fingerprinting Technique for YouTube Videos Identification in Network Traffic". In: *IEEE Access* 10 (July 19, 2022), pp. 76731–76741. DOI: 10.1109/ACCESS.2022.3192458. (Visited on 03/04/2023).

[5] Muhammad U. S. Khan et al. "SCNN-Attack: A Side-Channel Attack to Identify YouTube Videos in a VPN and Non-VPN Network Traffic". In: *Electronics* 11.3 (3 Jan. 2022), p. 350. DOI: 10.3390/electronics11030350. (Visited on 10/02/2022).

[6] Luming Yang et al. "Markov Probability Fingerprints: A Method for Identifying Encrypted Video Traffic". In: *2020 16th International Conference on Mobility, Sensing and Networking (Msn 2020)*. 2020, pp. 283–290. DOI: 10.1109/MSN50589.2020.00055. (Visited on 10/21/2022).

[7] Dan C. Cireşan, Ueli Meier, and Jürgen Schmidhuber. "Transfer Learning for Latin and Chinese Characters with Deep Neural Networks". In: *The 2012 International Joint Conference on Neural Networks (IJCNN)*. 2012, pp. 1–6. DOI: 10.1109/IJCNN.2012.6252544. (Visited on 12/28/2022).

[8] Xia Hu et al. "Model Complexity of Deep Learning: A Survey". In: *Knowledge and Information Systems* 63 (Oct. 1, 2021). DOI: 10.1007/s10115-021-01605-0. (Visited on 07/26/2023).

[9] Roei Schuster, Vitaly Shmatikov, and Eran Tromer. "Beauty and the Burst: Remote Identification of Encrypted Video Streams". In: *Proceedings of the 26th USENIX Conference on Security Symposium*. 2017, pp. 1357–1374. ISBN: 978-1-931971-40-9. (Visited on 12/30/2022).

[10] Kamila Ragimova, Vyacheslav Loginov, and Evgeny Khorov. "Analysis of YouTube DASH Traffic". In: *2019 IEEE International Black Sea Conference on Communications and Networking (BlackSeaCom)*. 2019-06-03,

pp. 1–5. DOI: 10.1109/BlackSeaCom.2019.8812782. (Visited on 04/23/2023).

[11] Abdelhak Bentaleb et al. “A Survey on Bitrate Adaptation Schemes for Streaming Media Over HTTP”. In: *IEEE Communications Surveys Tutorials* 21.1 (2019), pp. 562–585. DOI: 10.1109/COMST.2018.2862938. (Visited on 02/23/2023).

[12] Humberto de Jesus Ochoa Dominguez et al. “The H.264 Video Coding Standard”. In: *IEEE Potentials* 33.2 (2014), pp. 32–38. DOI: 10.1109/MPOT.2013.2284525. (Visited on 03/22/2023).

[13] Jian-Wen Chen, Chao-Yang Kao, and Youn-Long Lin. “Introduction to H.264 advanced video coding”. In: *Asia and South Pacific Conference on Design Automation, 2006.* 2006, 6 pp. DOI: 10.1109/ASPDAC.2006.1594774.

[14] Christopher Mueller. *MPEG_DASH Dynamic Adaptive Streaming Over HTTP — Bitmovin*. URL: https://bitmovin.com/dynamic-adaptive-streaming-http-mpeg-dash/ (visited on 07/31/2023).

[15] T. Berners-Lee. *Uniform Resource Identifier (URI): Generic Syntax*. URL: https://www.ietf.org/rfc/rfc3986.txt (visited on 01/2023).

[16] François Petitjean, Alain Ketterlin, and Pierre Gançarski. “A Global Averaging Method for Dynamic Time Warping, with Applications to Clustering”. In: *Pattern Recognition* 44.3 (Mar. 2011), pp. 678–693. DOI: 10.1016/j.patcog.2010.09.013. (Visited on 05/03/2023).

[17] Theo de Raadt. *hostapd*. https://man.openbsd.org/hostapd.8. 2022. (Visited on 06/22/2023).

[18] Ted Lemon. *dhcpd*. https://man.openbsd.org/dhcpd. 2022. (Visited on 06/22/2023).

[19] Jason Huggins. *geckodriver*. https://github.com/SeleniumHQ/selenium. 2004.

[20] mozilla. *geckodriver*. https://github.com/mozilla/geckodriver. 2022.

[21] IBM. *tcpdump*. https://www.ibm.com/docs/zh/aix/7.1?topic=t-tcpdump-command. 2023. (Visited on 06/22/2023).

[22] Abhinav Singh. *proxy.py*. https://github.com/abhinavsingh/proxy.py. 2019. (Visited on 06/22/2023).

[23] Betul Gokkaya, Leonardo Aniello, and Basel Halak. *Software supply chain: review of attacks, risk assessment strategies and security controls*. 2023. arXiv: 2305.14157. (Visited on 06/22/2023).

[24] Martín Abadi et al. *TensorFlow: Large-Scale Machine Learning on Heterogeneous Systems*. Software available from tensorflow.org. 2015. URL: https://www.tensorflow.org/ (visited on 06/22/2023).

[25] Fabian Pedregosa et al. “Scikit-Learn: Machine Learning in Python”. In: *J. Mach. Learn. Res.* 12 (Nov. 2011), pp. 2825–2830. ISSN: 1532-4435. (Visited on 06/25/2023).

[26] Romain Tavenard et al. “Tslearn, A Machine Learning Toolkit for Time Series Data”. In: *Journal of Machine Learning Research* 21.118 (2020), pp. 1–6. URL: http://jmlr.org/papers/v21/20-091.html (visited on 06/12/2023).

[27] Jill C. Stoltzfus. “Logistic Regression: A Brief Primer: LOGISTIC REGRESSION: A BRIEF PRIMER”. In: *Academic Emergency Medicine* 18.10 (Oct. 2011), pp. 1099–1104. DOI: 10.1111/j.1553-2712.2011.01185.x. URL: https://onlinelibrary.wiley.com/doi/10.1111/j.1553-2712.2011.01185.x (visited on 07/29/2023).