I Know What You Saw Last Minute - Encrypted HTTP Adaptive Video Streaming Title Classification

Ran Dubin[‡], Amit Dvir[*], Ofir Pele[*†], Ofer Hadar[‡]
[*] Center for Cyber Technologies, Department of Computer Science, Ariel University, Israel
[†] Department of Electrical and Electronics Engineering, Ariel University, Israel
[‡] Department of Communication Systems Engineering, Ben-Gurion University of the Negev, Israel

***Abstract*—Desktops can be exploited to violate privacy. There are two main types of attack scenarios: active and passive. We consider the passive scenario where the adversary does not interact actively with the device, but is able to eavesdrop on the network traffic of the device from the network side. In the near future, most Internet traffic will be encrypted and thus passive attacks are challenging. Previous research has shown that information can be extracted from encrypted multimedia streams. This includes video title classification of non HTTP adaptive streams. This paper presents algorithms for encrypted HTTP adaptive video streaming title classification. We show that an external attacker can identify the video title from video HTTP adaptive streams sites such as YouTube. To the best of our knowledge, this is the first work that shows this. We provide a large dataset of 15,000 YouTube video streams of 2,100 popular video titles that was collected under real-world network conditions. We present several machine learning algorithms for the task and run a thorough set of experiments, which shows that our classification accuracy is higher than 95%. We also show that our algorithms are able to classify video titles that are not in the training set as unknown and some of the algorithms are also able to eliminate false prediction of video titles and instead report unknown. Finally, we evaluate our algorithm robustness to delays and packet losses at test time and show that our solution is robust to these changes.**

***Index Terms*—HTTP Adaptive Video Streaming, HTTP2, Encrypted Traffic, Classification, YouTube**

I. Introduction

Every day, hundreds of millions of Internet users view videos online, and these numbers are clearly going to increase [1], [2]. By 2020, the share of video traffic is expected to increase to 82% of the total IP traffic, up from 70% in 2015. Google's streaming service, YouTube, now occupies a market share of over 17% of the total mobile network bandwidth in North America [2], [3].

Currently, most of the video streaming web sites including YouTube are using HTTP Adaptive Streaming (HAS). Dynamic Adaptive Streaming over HTTP (DASH) [4] is the *de facto* standard method for HAS. DASH is a Multi Bit Rate (MBR) streaming method that was designed to improve viewer Quality of Experience (QoE) [5]. In DASH, each video is divided into short segments, typically a few seconds long ($2-16$ seconds), and each segment is encoded several times, each time with a different quality representation. The user (player) Adaptation Logic (AL) algorithm is responsible for the automatic selection of the most suitable quality representation for each segment, based on parameters such as client playout buffer and network conditions. As a result, the quality representation in DASH can change between segments.

In DASH, each quality representation is encoded in variable bit rates (VBRs). VBR varies the amount of output data per time segment and does not attempt to control the output bit rate of the encoder, so that the distortion will not vary significantly [6]. DASH often uses HTTP byte range mode. In this mode, the byte range of each segment request can be different. This depends on the client's network conditions and playout buffer levels. Fig. 1 shows an example for three downloads of the same video title over different Wi-Fi networks, all with the same quality representation. From the figure we can notice that due to network conditions variability, there are differences between the downloads.

In the near future, most Internet network traffic will be encrypted [7]. Moreover, recently, YouTube has started to encrypt their video services [8]. As a result, traditional Deep Packet Inspection (DPI) methods for information retrieval, in general, and video title classification, in particular, are not viable.

Video title classification of YouTube streams can be used by an intelligence agency with access to ISP networks to gather open source intelligence (OSINT). By knowing the political agenda of the suspects, the

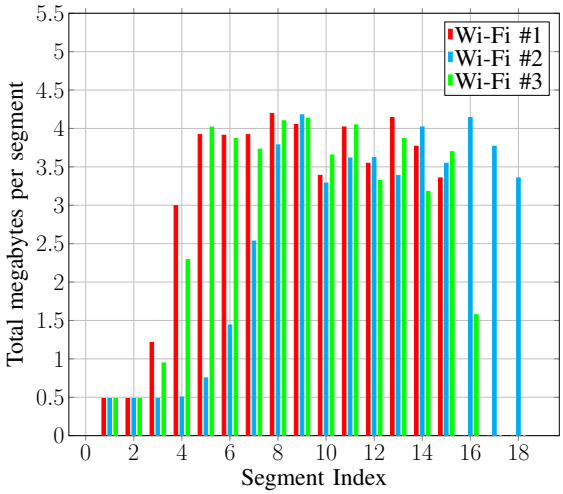


Fig. 1: Total megabytes per segment of three downloads over different Wi-Fi networks of the same video title, all with the same quality representation.

agency may classify group viewing habits over the ISP connection. Another usage might be to identify which individuals are watching a specific type of movie. These actions may violate personal privacy as they assume that the connection is secure. On the other hand, these techniques may be useful for the war against terror.

The remainder of this paper is organized as follows. In Section II we review related work. In Section III we present our framework and our suggested algorithms. In Section IV we evaluate the performance of all algorithms also under severe network conditions of long delays and high packet losses at testing times. In Section V we discuss limitations and possible countermeasures, and, finally, we conclude in Section VI.

II. Related Work

YouTube analysis was conducted on many aspects such as YouTube server location [9], [10], comparison between YouTube and other video sharing services [9], PC *vs.* mobile user access patterns [11], QoE [12], traffic characterization and its DASH implementation [13] and network analysis [14]–[18].

Many recent works have suggested methods for encrypted traffic classification and several surveys have presented detailed descriptions of the state of the art methods [19]–[33]. Several works have examined different features such as statistical flow features [19], Dynamic Time Warping (DTW) [20], session duration [34]–[36], number of packets in a session [35], [37], [38], different variance calculations of the minimum, maximum and average values of inter-arrival packet time [35], [37], payload size information [37], [39], bit rate [39], [40], Round-Trip Time (RTT) [40], packet direction [41] and server sent bit rate [42]. All these features are not suitable for video streaming classification as the payload size in video streaming is often maximum size, delays in the network are varied, and re-transmissions cause false packet counts. Our previous work on encrypted network traffic quality [43] evaluated those parameters and found that they are not effective for the HAS problem domain.

Recent works showed that video title classification of encrypted video streams is possible [29]–[31]. These works use features such as packet size and the application layer information. Saponas et al. [29] uncovered security issues with consumer electronic devices that enable information retrieval such as video title classification. Liu et al. [30] presented a method for video title classification of RTP/UDP Internet traffic. In [31] Liu et al. presented an improved algorithm which is more efficient and demonstrated excellent results on a bigger dataset with real network conditions. They used the wavelet transform for constructing unique and robust video signatures with different compactnesses.

Since these works [29]–[31] were conducted, there have been several changes in video traffic over the Internet:

(i) Adaptive byte range selection over HTTP;
(ii) MBR adaptive streaming;
(iii) HTTP version 2 [44].

As previous solutions operated on a time series with the granularity of single video frames, they do not fit modern streaming traffic. That is, DFTs and wavelet transforms capture short-term variations caused by changes of picture and long-term variations caused by changes of scene. In modern streaming these cannot be captured and indeed preliminary results showed that performance of previous methods [29]–[31] on this new domain are very poor.

Our preliminary research [45] was the first to identify video titles of encrypted YouTube streams. We proposed a nearest neighbor algorithm for this task that performed well on a small scale dataset. However, when we increased the size of the dataset both in the number of streams (10,000 *vs.* 300) and number of video titles (100 *vs.* 30) we encountered a decrease in the accuracy especially under long delays and high packet loss rate. Additionally, having a large number of streams of titles that are not in the training dataset (5,000 streams *vs.* 200 streams) decreased the accuracy results. Therefore, we developed three additional algorithms. The new algorithms improved robustness to noise. In fact, they are

eager learning algorithms specifically designed for this task. An eager learning algorithm carries out a learning process before receiving the test samples whereas a lazy learning algorithm such as the nearest neighbor algorithm just stores the training data. It is noteworthy that in the previous work, we also used a regular eager learning algorithm (a Support Vector Machine with a Radial Basis Function kernel) in our comparison which yielded very poor results as it was not adapted to the task of YouTube video title classification . Therefore, we developed algorithms that are both eager and adapted to the task. We replaced the Radial Basis Function kernel to our similarity measure. Additionally, we also developed algorithms that better cope with identifying video titles that are not in the training dataset as unknowns. These new algorithms improved results significantly.

The paper's main contributions are:

- This is the first work that shows that a passive attacker, sniffing ISP or Wi-Fi open-system network traffic, can identify video titles of encrypted YouTube video streams over DASH. Inspired by other works, we exploit traffic patterns and Variable Bit-Rate (VBR) encoding. We present new methods that are applicable also to current standards of video streaming.
- We run thorough a set of experiments which shows that our classification accuracy is more than 95%.
- We provide a comprehensive dataset that contains 15,000 labeled YouTube streams of 2,100 video titles. The streams were downloaded from the Internet under real-world network conditions. The dataset [46] and crawler [47] are available for download.

III. Video Title Classification

The proposed solution architecture goes as follows. The first module (Section III-A) removes non-YouTube packets and optionally audio packets. The next module combines several YouTube packets into a *peak*. A peak is defined as a section of traffic where there is silence before and after. Features are extracted from these peaks (Section III-B) and passed into a classification algorithm (Section III-C). It is noteworthy that the input to all of our classification algorithms is only encrypted HTTP adaptive video streaming traffic.

A. Preprocessing

First, we divide the traffic into flows based on a five-tuple representation: {protocol (TCP/UDP), src IP, dst IP, src port, dst port}. Then, we decide for each flow whether it is a YouTube flow. This is done based on the Service Name Indication (SNI) field in the *Client Hello* message. If the "*googlevideos.com*" string is found in the SNI, the flow is passed to the next module. Note that the YouTube flows identification can also be carried out using machine learning techniques [48], [49].

Second, we optionally remove audio packets. In all our training data, audio bursts were smaller than 400kB, while video traffic bursts were much larger. The audio data and the video data can be found in the same 5-tuple flow and in some cases we cannot distinguish between them.

Finally, we remove TCP re-transmissions using a TCP stack [50] as re-transmissions are caused mostly by network conditions.

B. Feature Extraction

Using raw data in a machine learning method is problematic as it is non-structured and contains redundant information (even after a preprocessing step). Thus, there is a need to build a structured representation of the raw data that is informative to the specific problem domain. Building this representation is called feature extraction [51, Chapter 5.3].

In our solution, the feature extraction is done on the preprocessed traffic, where non-YouTube flows, audio packets, and TCP re-transmissions have been removed. To better understand encrypted YouTube streaming traffic properties, we examined YouTube traffic under different browsers.

Fig. 2 depicts traffic download patterns of auto quality representation using different browsers (captured by Wireshark). In the figure we can see periods of data transmissions with silence (zero bits transmission), before and after. A period of data transmission is called a peak. Rao et al. [52] and Ameigeiraset al. [53] showed the same characteristics (coined in [52], [53] as "On/Off").

Therefore we decided to encode every peak of the stream to a feature. This feature is the Bit-Per-Peak (BPP); that is, total number of bits in a peak. It is noteworthy that this feature is different from bit rate.

C. Machine Learning

Supervised classification learning methods learn a classification function from a set of pre-labeled examples. The classification function is then used for classifying unseen test examples. There are two types of supervised classification learning methods. Lazy learning algorithms store the training data as is and then apply a classification function on a new test example where

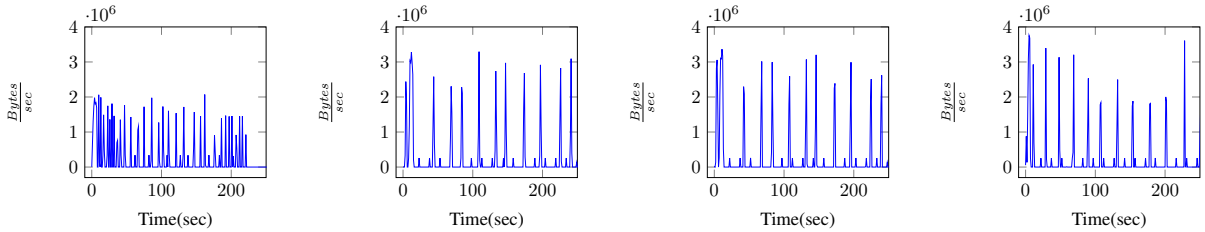


Fig. 2: Typical examples of traffic flows of auto mode downloads of the same movie using various browsers (Safari, Chrome, Firefox and Explorer) under various network conditions. The two first figures (from left) flows were downloaded via ADSL connection while the two at the right were downloaded via a WiFi connection. Note that differences between flows may be caused by auto mode, network conditions, player algorithm, etc.

the classification function is parameterized with the pre-labeled training examples. Eager learning algorithms, on the other hand, carry out a learning process on the pre-labeled training examples. Eager learning algorithms often perform better as the offline learning stage increases robustness to noise.

In this work, we adapt one lazy machine learning algorithm and two eager machine learning algorithms for the specific task of YouTube streams title classification.

The lazy machine learning algorithm we adapt is the nearest neighbor algorithm [54]. In this algorithm, the classification function computes similarities between a new test sample to all pre-labeled examples. The test sample is then assigned to the class of the most similar example from the training data.

Our adaption of the nearest neighbor algorithm uses a similarity function which fits to YouTube streams classification. Additionally, we add a classification rule that assigns unknown to test samples if they are not similar at all to any training example.

The first eager machine learning algorithm we adapt is the nearest neighbor to class algorithm [55]. This algorithm offline learning stage computes a single representation to all samples of the same class. That is, instead of storing all the training examples as is, all training examples of the same class are transformed into one sample.

We suggest two adaptions of this algorithm that differ in the transformation that is applied to training examples of the same class. The similarity function and unknown mechanism are the same as in our nearest neighbor algorithm adaptation.

The second eager machine learning algorithm we adapt is the Support Vector Machine (SVM) [56]. A binary linear SVM models training examples as points in space, and then divides the space using a hyperplane to give the best separation among the two classes. A non-linear SVM first projects the data into a different feature space. We use the similarity as features [57] projection where each sample is mapped into a vector of similarities to the training examples. For multiclass classification problems, that is, when there are more than two classes there are several extensions of the binary SVM, we use the one-vs-all adaption which has excellent performance [58]. This method trains a separate hyperplane for all classes. A test sample is then assigned to the class that won in most of the classifiers.

Our adaptions of this algorithm are the usage of the nearest neighbor to class transformation of samples, the similarity function and finally the unknown mechanism. We show that this algorithm strictly generalizes our adaption of the nearest neighbor to class algorithm. The ability of the learner to adjust weights of each title in the learning process increases its robustness to noise.

In what follows, we give detailed explanations of the four adapted algorithms. We recall that after the preprocessing and feature extraction, each video stream (number j of video title i) is represented by $\mathcal{S}_{ij}$, a set of Bit-Per-Peak (BPP) features (no duplicates). It is noteworthy that each BPP-set may have different cardinality. Table I summarizes symbols used in the algorithm explanations.

1) Nearest Neighbor (NN) Algorithm: The nearest neighbor similarity score between two BPP-sets, $\mathcal{S}$ and $\mathcal{S}'$, is the cardinality of the intersection set:

$$\text{sim}(\mathcal{S}, \mathcal{S}') = |\mathcal{S} \cap \mathcal{S}'| \tag{1}$$

At test time, each video stream BPP-set, $\mathcal{S}_{\text{test}}$, is classified as the video title i, that has the maximum similarity score to one of the title training stream BPP-sets or as unknown if all similarities are zero:

TABLE I: List of Abbreviations

BPP	Bit-Per-Peak
i	Video title number
j	Stream number
n	Number of video titles in the training dataset
m_i	Number of stream samples per title i in the training dataset
$\mathcal{S}, \mathcal{S}^{'}$	BPP-sets
$\mathcal{S}_{ij}$	A BPP-set of stream number j of title i
$\mathcal{S}_{\text{test}}$	A test BPP-set
$\mathcal{T}_i$	A BPP-set which is a union of all training streams of video i (Eq. 4)
$\mathcal{U}_i$	A BPP-set which is a union of all training streams of video i minus BPPs of other video titles (Eq. 7)

$$\forall \;\; 1 \leq i \leq n, \;\; s_i = \max_{j=1}^{m_i} \text{sim}(\mathcal{S}_{\text{test}}, \mathcal{S}_{ij}) \tag{2}$$

$$y(\mathcal{S}_{\text{test}}) = \begin{cases} \underset{i=1}{\overset{n}{\text{argmax}}}\, s_i & \text{if } \left(\max_{i=1}^{n} s_i\right) > 0 \\ \text{unknown} & \text{otherwise} \end{cases} \tag{3}$$

2) Nearest Neighbor to Class (NNC) Algorithm: In the nearest neighbor to the class algorithm, each video title i in the training is represented by a single BPP-set, $\mathcal{T}_i$, which is a union of all its m_i video stream BPP-sets (no duplicates):

$$\mathcal{T}_i = \cup_{j=1}^{m_i} \mathcal{S}_{ij} \tag{4}$$

As in the nearest neighbor algorithm, the similarity score is the cardinality of the intersection set. In this case, the similarity is between a BPP-set of a single stream and the BPP-set of all streams of a title:

$$\text{sim}(\mathcal{S}, \mathcal{T}_i) = |\mathcal{S} \cap \mathcal{T}_i| \tag{5}$$

At test time, each video stream set, $\mathcal{S}_{\text{test}}$, is classified as the video title i, that has the maximum similarity score to one of the n video title BPP-sets or as unknown if all similarities are zero:

$$y(\mathcal{S}_{\text{test}}) = \begin{cases} \underset{i=1}{\overset{n}{\text{argmax}}}\, \text{sim}(\mathcal{S}_{\text{test}}, \mathcal{T}_i) & \text{if } \left(\max_{i=1}^{n} \text{sim}(\mathcal{S}_{\text{test}}, \mathcal{T}_i)\right) > 0 \\ \text{unknown} & \text{otherwise} \end{cases} \tag{6}$$

3) Nearest Neighbor to Class Unique (NNCU) Algorithm: As in the nearest neighbor to class algorithm, in the nearest neighbor to class unique algorithm, each video title i in the training is represented by a single BPP-set. In the nearest neighbor to class unique algorithm, the set is a union of all its m_i video stream BPP-sets (no duplicates) *minus* BPP values that appear in sets of other video titles:

$$\mathcal{U}_i = \mathcal{T}_i \setminus \{\cup_{i'=1, i' \neq i}^{n} \mathcal{T}_{i'}\} \tag{7}$$

As in the nearest neighbor to class algorithm, the similarity score is the cardinality of the intersection set:

$$\text{sim}(\mathcal{S}, \mathcal{U}_i) = |\mathcal{S} \cap \mathcal{U}_i| \tag{8}$$

At test time, each video stream set, $\mathcal{S}_{\text{test}}$, is classified as the video title i, that has the maximum similarity score to one of the n video title BPP-sets or as unknown if all similarities are zero:

$$y(\mathcal{S}_{\text{test}}) = \begin{cases} \underset{i=1}{\overset{n}{\text{argmax}}}\, \text{sim}(\mathcal{S}_{\text{test}}, \mathcal{U}_i) & \text{if } \left(\max_{i=1}^{n} \text{sim}(\mathcal{S}_{\text{test}}, \mathcal{U}_i)\right) > 0 \\ \text{unknown} & \text{otherwise} \end{cases} \tag{9}$$

4) Similarities as Features Support Vector Machine (SFSVM) Algorithm: In this algorithm, each video stream is represented by a feature vector which is the video stream similarity to all n video title sets (thus it is an n-dimensional vector). Where the similarity is the same as in the nearest neighbor to class algorithm, Eq. 5:

$$\vec{x}(\mathcal{S}) = [\text{sim}(\mathcal{S}, \mathcal{T}_1), \ldots, \text{sim}(\mathcal{S}, \mathcal{T}_n)] \tag{10}$$

Thus, the training set is an $(\sum_{i=1}^{n} m_i) \times n$ matrix of all training stream feature vectors:

$$\begin{bmatrix} \text{sim}(\mathcal{S}_{11}, \mathcal{T}_1) & \ldots & \text{sim}(\mathcal{S}_{11}, \mathcal{T}_n) \\ \vdots & \ddots & \vdots \\ \text{sim}(\mathcal{S}_{1m_1}, \mathcal{T}_1) & \ldots & \text{sim}(\mathcal{S}_{1m_1}, \mathcal{T}_n) \\ \vdots & \ddots & \vdots \\ \text{sim}(\mathcal{S}_{n1}, \mathcal{T}_1) & \ldots & \text{sim}(\mathcal{S}_{n1}, \mathcal{T}_n) \\ \vdots & \ddots & \vdots \\ \text{sim}(\mathcal{S}_{nm_n}, \mathcal{T}_1) & \ldots & \text{sim}(\mathcal{S}_{nm_n}, \mathcal{T}_n) \end{bmatrix} \tag{11}$$

We learn one *vs.* all Support Vector Machines (SVMs) [56], [58]. That is, we learn a classifier for each video title i that classifies whether it is this title or any of the other titles. The classifiers are n-dimensional weight vectors and at test time, each video stream set, $\mathcal{S}_{\text{test}}$, is classified as the video title i, which maximizes the dot

TABLE II: Algorithm training and testing samples

Algorithm	Training Sample	Testing Sample
Nearest Neighbor	$\mathcal{S}_{ij}$	$\mathcal{S}_{\text{test}}$
Nearest Neighbor to Class	$\mathcal{T}_i$	$\mathcal{S}_{\text{test}}$
Nearest Neighbor to Class Unique	$\mathcal{U}_i$	$\mathcal{S}_{\text{test}}$
Similarities as Features Support Vector Machine	$\vec{x}(\mathcal{S}_{ij}) = [\text{sim}(\mathcal{S}_{ij}, \mathcal{T}_1), \ldots, \text{sim}(\mathcal{S}_{ij}, \mathcal{T}_n)]$	$\vec{x}(\mathcal{S}_{\text{test}}) = [\text{sim}(\mathcal{S}_{\text{test}}, \mathcal{T}_1), \ldots, \text{sim}(\mathcal{S}_{\text{test}}, \mathcal{T}_n)]$

product between the class weight vector and the features vector or as unknown if all similarities are zero:

$$\vec{x}(\mathcal{S}_{\text{test}}) = [\text{sim}(\mathcal{S}_{\text{test}}, \mathcal{T}_1), \ldots, \text{sim}(\mathcal{S}_{\text{test}}, \mathcal{T}_n)] \tag{12}$$

$$y(\mathcal{S}_{\text{test}}) = \begin{cases} \underset{i=1}{\overset{n}{\text{argmax}}} \left(\vec{w_i} \cdot \vec{x}(\mathcal{S}_{\text{test}})\right) & \text{if } \left(\underset{i=1}{\overset{n}{\text{max}}} \text{sim}(\mathcal{S}_{\text{test}}, \mathcal{T}_i)\right) > 0 \\ \text{unknown} & \text{otherwise} \end{cases} \tag{13}$$

The SVM regularization parameter C and γ were found with 5-fold cross-validation on the training dataset over the following sets accordingly: $\{2^{-5}, 2^{-3}, ..., 2^{15}\}$ and $\{2^{-15}, 2^{-13}, ..., 2^{3}\}$.

It is noteworthy that if we learn the following weight vectors:

$$\forall \ 1 \leq i \leq n, \ \ \vec{w_i} = [0, \ldots, \overset{i}{\overset{\smile}{1}}, \ldots, 0] \tag{14}$$

This exactly models the nearest neighbor to the class algorithm. Thus, this algorithm generalizes the nearest neighbor to the class algorithm.

A summary of algorithm training and testing samples is in Table II.

IV. Performance Evaluation

In this section, we evaluate the proposed encrypted HTTP adaptive video streaming title classification algorithms. First, we describe the dataset in IV-A. Then we report experimental results in Section IV-B.

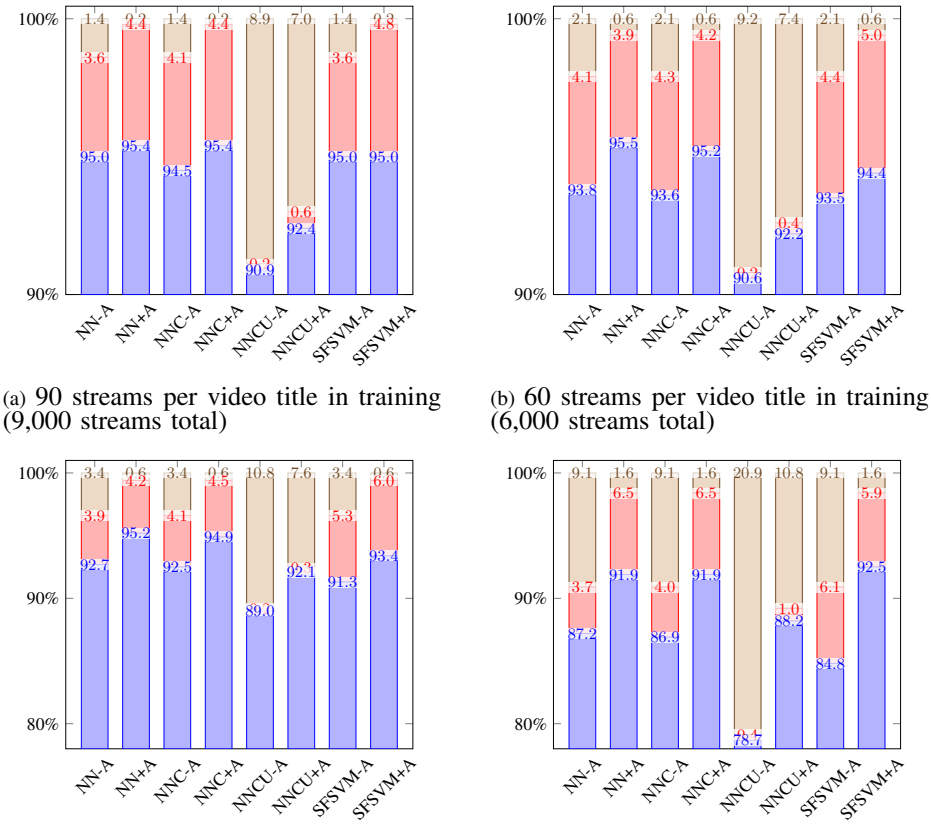
A. Dataset

We recall that we want to identify and differentiate between a set of known video titles, whereas in testing we might observe a much larger group of video titles. For this, we collected a dataset that contains 10,000 YouTube streams (that were downloaded via a real-world Internet connection during a period of a month over different real-world network conditions) of 100 video titles (100 streams per video title). As it is not reasonable that a group of people will watch only the above "target" video titles, we added 5,000 streams of 2,000 other video titles which are *not* included in the "target" video titles set. These titles are used as the *unknown* titles in the evaluation.

The video titles used in this study are popular YouTube videos from different categories such as news, sports, nature, video action trailers, and GoPro videos. In this study we decided to use the Chrome browser since it is the most popular browser on the market and its popularity is growing [59]. It is noteworthy that also in other browsers, YouTube streams have peaks or "on/off" traffic patterns [43], [52], [53]. Therefore using other browsers is also possible.

We used the default auto mode of the YouTube player (the player decides which quality representation to download based on estimation of the client network conditions). We used the Selenium web automation tool [60] with ChromeDriver [61] for the crawler, so it will simulate a user video download in exactly the same manner a normal user behaves. We used Ad-block Plus [62] to eliminate advertisements *only* in the training datasets. This work assumes that video advertisement exists in the network in real scenarios. We do not assume that the advertisement can be distinguished from the viewed video on the encrypted network traffic level. From our preliminary work [45] we found that the video advertisement is distributed in different network flows and the video title download is distributed by several parallel network flows. Therefore, when testing unknown titles (titles that are not in the training titles) we do not filter out the video advertisement. Our algorithms classify each flow separately and we do not assume any prior knowledge about how many different flows exist per download. As a video advertisement is a different flow we can classify it as unknown video title (thus, effectively, the number of unknown is higher than 2,000).

We offer the following training/testing datasets:(1) *TrainKnown* - The dataset contains 9,000 video streams of 100 video titles for training the different classifiers; (2) *TestKnown* - The dataset contains 1,000 video streams (different also from the training streams) of the same 100 video titles from the *TrainKnown* data set; (3) *TestUnknown* - The dataset contains 5,000 (unknown)



(a) 90 streams per video title in training (9,000 streams total)

(b) 60 streams per video title in training (6,000 streams total)

(c) 30 streams per video title in training (3,000 streams total)

(d) 5 streams per video title in training (500 streams total)

Title-Unknown-Error: predict video title unknown while it is video title i from the training data
Title-Title-Error: predict video title i while it is video title $i' \neq i$ from the training data
Title-Accuracy: predict the right video title

Fig. 3: *Title-Accuracy*, *Title-Title-Error* and *Title-Unknown-Error* results for different training data set sizes and different learning algorithms. We can see that all algorithms (see Table III for algorithm acronyms) were able to identify the video title of an encrypted HTTP adaptive stream (HAS) with very high *Title-Accuracy*.

video streams of 2,000 video titles. The video titles are not from the *TrainKnown* data set. The dataset was recorded without ad-block and thus the streams also contain advertisements;(4) *TestKnownDelay* - The dataset contains 400 video streams of 10 video titles (titles taken from the *TrainKnown* data set); each was downloaded with an additional added drop percentage from the following list: $\{100, 300, 600\}[ms]$;(5) *TestKnownDrop* - The dataset contains 400 video streams of 10 video titles (titles taken from the *TrainKnown* data set), each was downloaded with an additional added drop percentage from the following list: $\{1\%, 3\%, 6\%\}$. The dataset [46] and crawler [47] are available for download.

B. Experimental Results

We recall that our classifiers have two types of predictions: a video title $1 \leq i \leq n$ and unknown. Unknown means that the classifier predicts that the given stream video title is not in the training set. We use the following evaluation metrics:

Title-Accuracy Number of times that the classifier predicted video title i and it was true, divided by the total number of streams in the test set that their title is in the training set.

Title-Title-Error Number of times that the classifier predicted video title i and it was false, divided by the total number of streams in the test set that their title is in the training set.

Title-Unknown-Error Number of times that the classifier predicted video title unknown while it was a video title from the training set, divided by the total number of streams in the test set that their title is in the training set.

Unknown-Accuracy Number of times that the classifier predicted video title unknown and it is indeed not a video title from the training set, divided by the total number of unknown streams in the testing set.

Unknown-Title-Error Number of times that the classifier predicted video title unknown while it was a video title from the training set, divided by the total number of unknown streams in the testing set.

It should be noted that we do not have Unknown-Unkown-Error as we do not try to differentiate between unknown video titles.

We first report results using variable training dataset sizes. For all following experiments, we used the *Test-Known* data set. For training, we used the other 9,000 streams (90 streams per video title from the dataset), 6,000 streams (60 streams per video title from the dataset), 3,000 streams (30 streams per video title from the data set) and 500 streams (5 streams per video title from the dataset). All the test video streams were different from the ones that were used in the training phase, because of network conditions while streaming video from YouTube. In these experiments, the testing set did not contain streams of video titles that were not in the training data. We compared all algorithms with our features and also experimented with and without the removal of audio features.

The results of these experiments are depicted in Fig. 3. Algorithm acronyms can be found in Table III. There are several observations. First, all algorithms were able to accurately identify the video title of an encrypted HTTP adaptive stream (HAS). *Title-Accuracy* was higher than 90% using 60 or more streams per video title (as there are 100 classes a chance classifier *Title-Accuracy* is only 1% for this task). Even using only 5 streams per video title, NN+A, NNC+A and SFSVM+A *Title-Accuracy* was larger than 90%. Using also BPPs of audio peaks the *Title-Accuracy* was higher than 95% using 90 streams per video title, but the *Title-Title-Error* which is a more severe error than the *Unknown-Title-Error* was also higher. The NNCU algorithm *Title-Accuracy* was lower in comparison to the other algorithms and moreover without the audio features. But, the *Title-Accuracy* is still high (89% or higher for 30 streams per video title or more, and 78.7% for 5 streams per video title) and the *Title-Title-Error* were almost eliminated.

We also experimented with the classifiers (NN, NNC and SFSVM) with 2,000 video titles that were not in the training set. The *Unknown-Accuracy* rate (predicting unknown video title when it is not in the training dataset) can be seen in Fig. 4. We can see that NN+A NNC+A, SFSVM+A, had the lowest *Unknown-Accuracy*, although it is still much higher than chance. We can also see that removing the audio features improves *Unknown-Accuracy* considerably to 97.8%. Finally, NNCU, with and without audio, identified all unknowns streams.

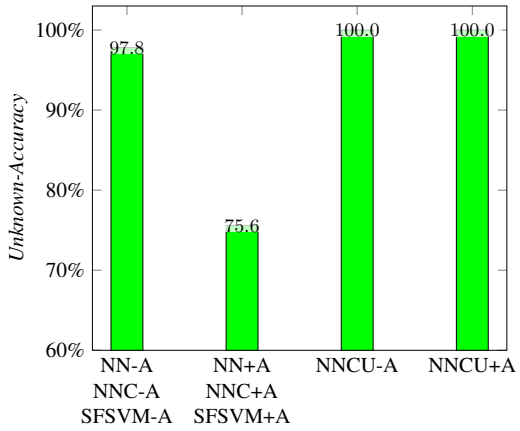


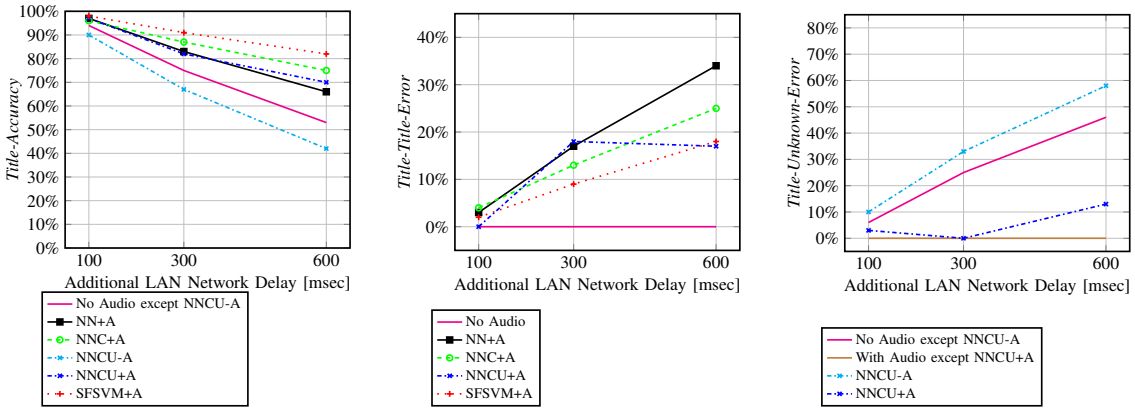
Fig. 4: The *Unknown-Accuracy* for all algorithms. NN, NNC and SFSVM classifiers are equivalent for the task of unknown prediction.

As network conditions vary, we also tested our algorithms with additional LAN network delay and packet loss on test time. That is, only the testing data is changed and not training data. The additional delay and drop affects the client player and causes it to select different (lower) representations. The delays and packet loss were added using the clumsy application [63].

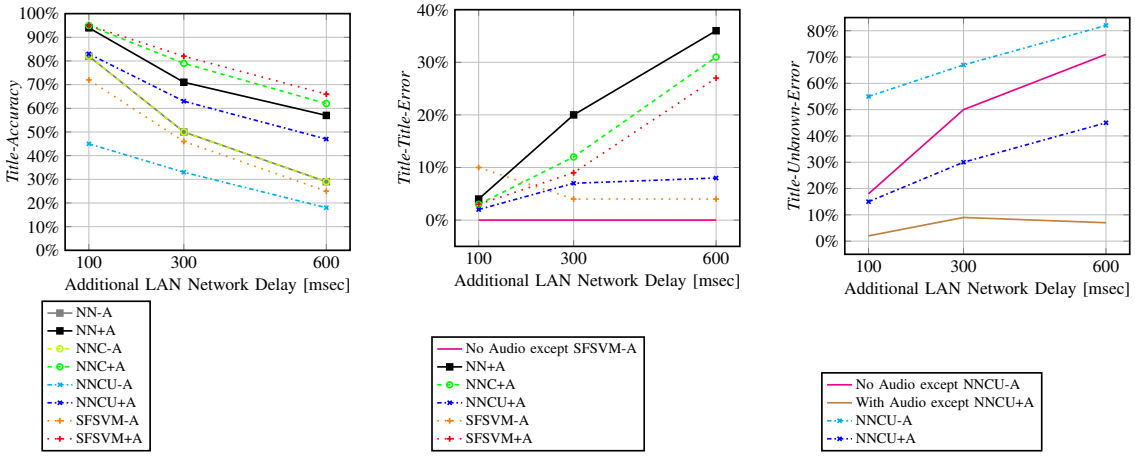
The results of additional LAN network delay are depicted in Fig. 5. We can see that using the largest training dataset the SFSVM+A method outperformed all other methods and achieved *Title-Accuracy* of more than 80% even under severe network delays of 600msec. We conjecture that using an eager algorithm with a learning phase and the usage of distance to class made this algorithm robust to changes. We can see that NNC+A *Title-Accuracy* was also high. Training with less streams, the

TABLE III: List of Algorithm Acronyms

NN	Nearest Neighbor
NNC	Nearest Neighbor to Class
NNCU	Nearest Neighbor to Class Unique
SFSVM	Similarities as Features Support Vector Machine
-A	without audio features
+A	with audio features



(a) *Title-Accuracy*, 90 streams per video title in training (b) *Title-Title-Error*, 90 streams per video title in training (c) *Title-Unknown-Error*, 90 streams per video title in training



(d) *Title-Accuracy*, 5 streams per video title in training (e) *Title-Title-Error*, 5 streams per video title in training (f) *Title-Unknown-Error*, 5 streams per video title in training

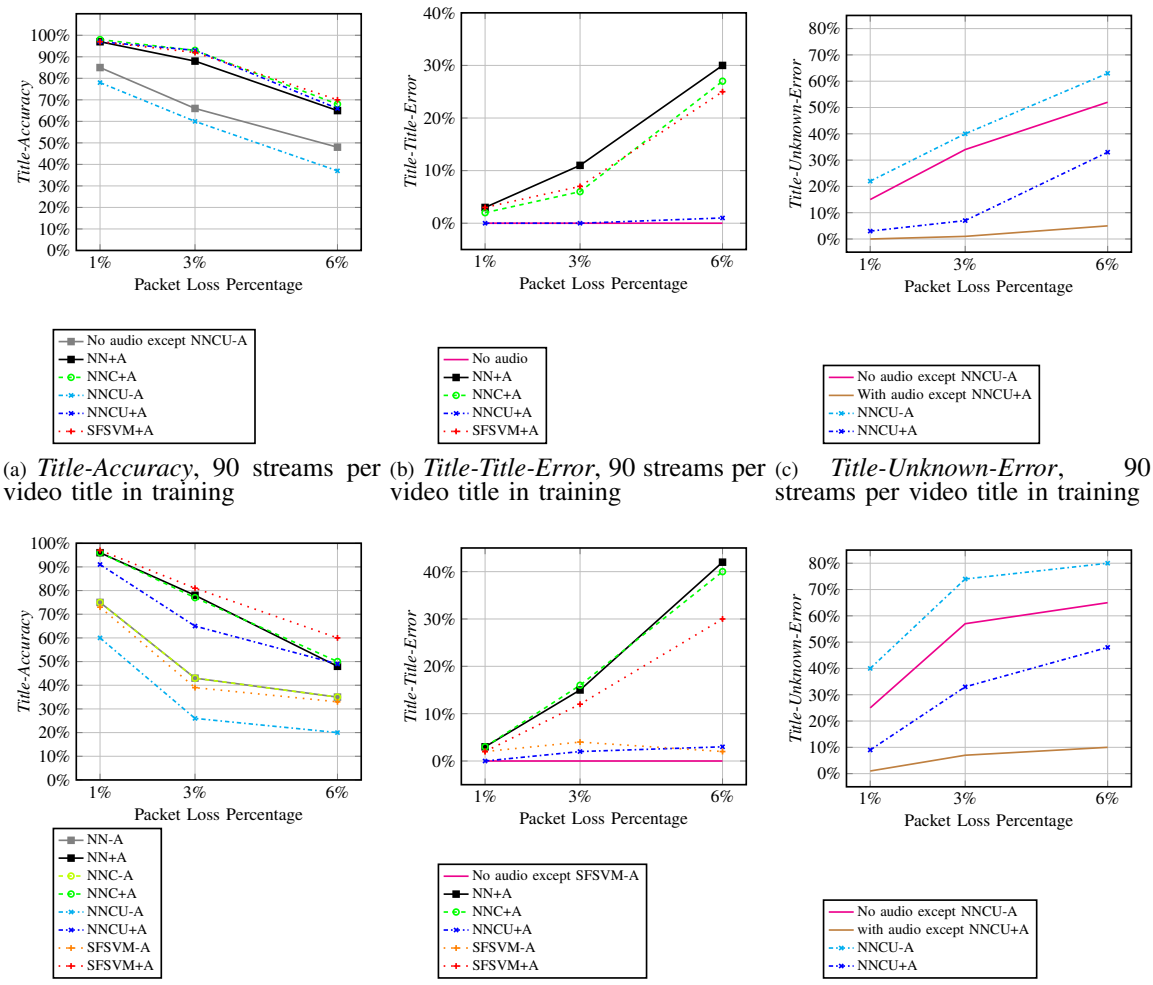
Fig. 5: *Title-Accuracy*, *Title-Title-Error* and *Title-Unknown-Error* results for different additional LAN network delay, different training data set sizes, and different learning algorithms.

algorithms NNC+A and SFSVM+A are comparable and both outperformed all other algorithms. Like previous results the NNCU algorithm and not using audio features eliminated *Title-Title-Error*. Out of all algorithms with low *Title-Title-Error*, NNCU+A *Title-Accuracy* was the best.

The results of additional packet loss are depicted in Fig. 6. We can see that using the largest training dataset the SFSVM+A method slightly outperformed all other methods and achieved *Title-Accuracy* of more than 70% even under severe packet loss of 6%. NNC+A, NNCU+A, NN+A all also exhibited good performance. Training with less streams, method *Title-Accuracy* was generally reduced. Similar to previous results the NNCU algorithm and not using audio features almost eliminated *Title-Title-Error*. Again, NNCU+A *Title-Accuracy* was the best.

V. Possible Countermeasures and Limitations

Users and service providers might believe that by using the right encryption and authentication mechanisms, their communications are secure. However, this is not always true. As presented in this paper, it is possible to develop classifiers for TLS/SSL encrypted traffic that are able to identify the video title from video HTTP adaptive streams (HAS) from websites such as YouTube. The contribution of this paper was to investigate the extent to which it is feasible to identify the video title from a set of videos while also identifying new video titles as unknown. While it is out of the scope of the paper



(a) *Title-Accuracy*, 90 streams per video title in training

(b) *Title-Title-Error*, 90 streams per video title in training

(c) *Title-Unknown-Error*, 90 streams per video title in training

(d) *Title-Accuracy*, 5 streams per video title in training

(e) *Title-Title-Error*, 5 streams per video title in training

(f) *Title-Unknown-Error*, 5 streams per video title in training

Fig. 6: *Title-Accuracy*, *Title-Title-Error* and *Title-Unknown-Error* results for different training data set sizes, different additional packet loss percentage and different learning algorithms.

to investigate possible countermeasures, we discuss it shortly following some related issues.

A possible countermeasure can be a padding technique which may be effective against traffic analysis approaches. However, it has to be considered that padding countermeasures are already standardized in TLS, explicitly to frustrate attacks on a protocol that are based on analysis of the lengths of exchanged messages [21]. The intuition is that the information is not hidden efficiently, and the analysis of these features may still allow information leakage.

We strongly believe that it is not trivial to propose effective countermeasures to the attack we showed in this paper. Indeed, it is the intention of the authors to highlight a problem that is becoming even more alarming after the revelation of mass surveillance programs that are nowadays adopted by governments and nation states.

Supervised machine learning algorithms have several limitations. One of the limitations is that a labeled training dataset has to be collected. In this paper, we automate the collection of the training dataset thus somewhat mitigating this limitation.

Another limitation of regular machine learning algorithms is that they are not able to recognize classes that have not been used during the training phase. In this paper, we adapted machine learning algorithms to be able to identify unknown events as such.

Finally, in machine learning, different conditions in

testing time might hinder classification. The changes can be changes in network conditions and also in applications and protocols. In this paper we tested testing sets with different network conditions than the training ones and the classification accuracy was only slightly reduced.

VI. Conclusions

This paper showed that the video title of encrypted HTTP adaptive streams such as YouTube can be identified with high accuracy, even under severe network conditions of long delays and high packet losses at testing time. We also showed that our algorithms are able to classify video titles that are not in the training as unknown. To the best of our knowledge this is the first work to show this. A conclusion that can be drawn from this paper is that although YouTube uses HTTPS, which is assumed to be secure and modern DASH protocol, it is still not enough to protect viewing habits.

We presented several algorithms for this task and compared them on a large real-world traffic dataset. Overall, having enough training data the Similarities as Features Support Vector Machine with Audio (SFSVM+A) algorithm achieved the best accuracy even under long delays and high packet loss rate. If predicting the wrong title is an acute error we recommend to use the Nearest Neighbor to Class Unique with Audio (NNCU+A) algorithm which instead reports unknown on almost all of its errors and still has relatively high accuracy. A defense against this privacy hole would necessitate a differentiation of the peak sizes. To the best of our knowledge, this would require changes both from the client and the server in the browser and the network stack implementation, and, in particular, in the TCP and HTTP protocols.

The dataset [46] and crawler [47] are provided for future research.

References

[1] Cisco. Cisco visual networking index: global mobile data traffic forecast update, 2012-2016. http://www.cisco.com/c/en/us/solutions/collateral/service-provider/visual-networking-index-vni/mobile-white-paper-c11-520862.html, 2012.

[2] Cisco. The zettabyte era: trends and analysis. http://www.cisco.com/c/en/us/solutions/collateral/service-provider/visual-networking-index-vni/VNI_Hyperconnectivity_WP.pdf, 2015.

[3] Sandvine. Sandvine global internet phenomena report h1 , 2014. https://www.sandvine.com/downloads/general/global-internet-phenomena/2014/1h-2014-global-internet-phenomena-report.pdf, 2014.

[4] ISO/IEC. Information technology - dynamic adaptive streaming over http (dash). https://www.iso.org/standard/65274.html, May 2014.

[5] M. Seufert, S. Egger, M. Slanina, T. Zinner, T. Hofeld, and P. Tran-Gia. A survey on quality of experience of http adaptive streaming. *IEEE Communication Surveys and Tutorials*, 17(1):469–492, 2015.

[6] M. R. Izquierdo and D. S. Reeves. A survey of statistical source models for variable bit-rate compressed video. *Multimedia System*, 7(3):199–213, 1999.

[7] R. Hackett. Most internet traffic will be encrypted by year end. here's why. http://fortune.com/2015/04/30/netflix-internet-traffic-encrypted, 2015.

[8] Google. Google webmaster central blog: https as a ranking signal, august. http://googlewebmastercentral.blogspot.co.il/2014/08/https-as-ranking-signal.html, 2014.

[9] V. K. Adhikari, S. Jain, and Z. L. Zhang. Youtube traffic dynamics and its interplay with a tier-1 isp: an isp perspective. In *Special Interest Group on Data Communication (SIGCOMM)*, 2010.

[10] R. Torres, A. Finamore, J. Kim, J. Ryong, M. Mellia, M. M. Munafo, and S. Rao. Dissecting video server selection strategies in the youtube cdn. In *IEEE International Conference on Distributed Computing System (ICDCS)*, 2011.

[11] A. Finamore, M. Mellia, M. M. Munafò, R. Torres, and S. G. Rao. Youtube everywhere: impact of device and infrastructure synergies on user experience. In *Internet Measurement Conference (IMC)*, 2011.

[12] C. Sieber, T. Hossfeld, T. Zinner, P. Tran-Gia, and C. Timmerer. Implementation and user-centric comparison of a novel adaptation logic for dash with svc. In *IFIP/IEEE International Symposium on Integrated Network Management*, pages 1318–1323, 2013.

[13] J. Aorga, S. Arrizabalaga, B. Sedano, M. Alonso-Arce, and J. Mendizabal. Youtubes dash implementation analysis. In *19th International Conference on Circuits, Systems, Communications and Computers*, CSCC, pages 61–66, 2015.

[14] G. Dimopoulos. *YouTube traffic monitoring and analysis*. PhD thesis, Universitat Politècnica de Catalunya, 2012.

[15] M. Zink, K. Suh, Y. Gu, and J. Kurose. Characteristics of youtube network traffic at a campus network - measurements, models, and implications. *Computer Networks: The International Journal of Computer and Telecommunications Networking*, 2009.

[16] C. Meeyoung, K. Haewoon, R. Pablo, Y. Y. Ahn, and S. Moon. I tube, you tube, everybody tubes: analyzing the worlds largest user generated content video system. In *Internet Measurement Conference (IMC)*, 2007.

[17] C. Xianhui, B. Ip, and L. Ling. A survey of current youtube video characteristics. *IEEE MultiMedia*, 22(2):56–63, Apr 2015.

[18] S. Alcock and R. Nelson. Application flow control in youtube video streams. *ACM Sigcom Computer Communication Review*, 41(2):24–30, Apr. 2011.

[19] V. F. Taylor, R. Spolaor, M. Conti, and I. Martinovic. Appscanner: automatic fingerprinting of smartphone apps from encrypted network traffic. In *1st IEEE European Symposium on Security and Privacy*, mar 2016.

[20] M. Conti, L. V. Mancini, R. Spolaor, and N. V. Verde. Analyzing android encrypted network traffic to identify user actions. *IEEE Transactions On Information Forensics and Security*, 2016.

[21] M. Husák, M. Čermák, T. Jirsík, and P. Čeleda. Https traffic analysis and client identification using passive ssl/tls fingerprinting. *EURASIP Journal on Information Security*, 2016(1):1–14, Dec 2016.

[22] A. Dainotti, A. Pescape, and K. C. Claffy. Issues and future directions in traffic classification. *IEEE Network*, 26(1):35–40, 2012.

[23] S. Valenti, D. Rossi, A. Dainotti, A. Pescapè, A. Finamore, and M. Mellia. Reviewing traffic classification. In *Data Traffic Monitoring and Analysis*, pages 123–147. Springer, 2013.

[24] Z. Cao, G. Xiong, Y. Zhao, Z. Li, and L. Guo. A survey on encrypted traffic classification. In *Applications and Techniques in Information Security*, pages 73–81. Springer, 2014.

[25] M. Crotti, F. Gringoli, P. Pelosato, and L. Salgarelli. A statistical approach to ip level classification of network traffic. In *International Conference on Communications*, pages 170–176, June 2006.

[26] T. Okabe, T. Kitamura, and T. Shizuno. Statistical traffic identification method based on flow-level behavior for fair voip service. In *IEEE Workshop on VoIP Management and Security*, pages 35–40, April 2006.

[27] D. X. Song, D. Wagner, and X. Tian. Timing analysis of keystrokes and timing attacks on ssh. In *Proceedings of the 10th Conference on USENIX Security Symposium - Volume 10*, pages 25–25, 2001.

[28] B. Canvel, A. Hiltgen, S. Vaudenay, and M. Vuagnoux. Password interception in a ssl/tls channel. In *Annual International Cryptology Conference*, pages 583–599. Springer, 2003.

[29] T. S. Saponas, J. Lester, C. Hartung, S. Agarwal, and T. Kohno. Devices that tell on you: privacy trends in consumer ubiquitous computing. In *Proceedings of 16th USENIX Security Symposium*, pages 5:1–5:16, 2007.

[30] L. Yali, C. Ou, L. Zhi, C. Corbett, B. Mukherjee, and D. Ghosal. Wavelet-based traffic analysis for identifying video streams over broadband networks. In *IEEE Global Telecommunications Conference*, pages 1–6, Nov 2008.

[31] Y. Liu, A. R. Sadeghi, D. Ghosal, and B. Mukherjee. Video streaming forensic content identification with traffic snooping. In *Information Security*, volume 6531 of *Lecture Notes in Computer Science*, pages 129–135. Springer, 2011.

[32] A. M. White, A. R. Matthews, K. Z. Snow, and F. Monrose. Phonotactic reconstruction of encrypted voip conversations: hookt on fon-iks. In *IEEE Symposium on Security and Privacy (SP)*, pages 3–18. IEEE, 2011.

[33] C. V. Wright, L. Ballard, F. Monrose, and G. M. Masson. Language identification of encrypted voip traffic: alejandra y roberto or alice and bob? In *Proceedings of 16th USENIX Security Symposium*, pages 1–12, 2007.

[34] V. Paxson. Empirically derived analytic models of wide-area tcp connections. *IEEE/ACM Transactions on Networking (TON)*, 2(4):316–336, 1994.

[35] R. Alshammari and A. N. Zincir-Heywood. Unveiling skype encrypted tunnels using gp. In *IEEE Congress on Evolutionary Computation (CEC)*, pages 1–8. IEEE, 2010.

[36] S. Zander, T. Nguyen, and G. Armitage. Self-learning ip traffic classification based on statistical flow characteristics. In *Passive and Active Network Measurement*, pages 325–328. Springer, 2005.

[37] D. Zhang, C. Zheng, H. Zhang, and H. Yu. Identification and analysis of skype peer-to-peer traffic. In *Internet and Web Applications and Services (ICIW), 2010 Fifth International Conference on*, pages 200–206, 2010.

[38] I. Paredes-Oliva, I. Castell-Uroz, P. Barlet-Ros, X. Dimitropoulos, and J. Sole-Pareta. Practical anomaly detection based on classifying frequent traffic patterns. In *IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS)*, pages 49–54, 2012.

[39] D. Bonfiglio, M. Mellia, M. Meo, and D. Rossi. Detailed analysis of skype traffic. *IEEE Transactions on Multimedia*, 11(1):117–127, 2009.

[40] K. T. Chen, C. Y. Huang, P. Huang, and C. L. Lei. Quantifying skype user satisfaction. In *ACM SIGCOMM Computer Communication Review*, volume 36, pages 399–410. ACM, 2006.

[41] E. Hjelmvik and W. John. Statistical protocol identification with spid: preliminary results. In *Swedish National Computer Networking Workshop*, 2009.

[42] R. Bar-Yanai, M. Langberg, D. Peleg, and L. Roditty. Realtime classification for encrypted traffic. In *Experimental Algorithms*, pages 373–385. Springer, 2010.

[43] R. Dubin, A. Dvir, O. Pele, O. Hadar, I. Raichman, and O. Trabelsi. Video quality representation classification of safari encrypted dash streams. In *IEEE Digital Media Industry and Academic Forum, Santorini, Greece*. IEEE, 2016.

[44] M. Belshe, R. Peon, and M. Thomson. Hypertext Transfer Protocol Version 2. RFC 7540, IETF, May 2015.

[45] R. Dubin, A. Dvir, O. Pele, and O. Hadar. I know what you saw last minute-encrypted http adaptive video streaming title classification- the chrome use case. In *Black Hat*, 2016.

[46] R. Dubin, O. Pele, A. Dvir, and O. Hadar. The video streams pcap files dataset. http://www.cse.bgu.ac.il/title_fingerprinting/, 2017.

[47] R. Dubin, O. Pele, A. Dvir, and O. Hadar. The research chrome youtube encrypted network traffic crawler. https://github.com/randubin/YouTube_video_title_downloader.

[48] P. Fu, L. Guo, G. Xiong, and J. Meng. Classification research on ssl encrypted application. In *Trustworthy Computing and Services*, volume 320 of *Communications in Computer and Information Science*, pages 404–411. Springer Berlin Heidelberg, 2013.

[49] G. L. Sun, Y. Xue, Y. Dong, D. Wang, and C. Li. An novel hybrid method for effectively classifying encrypted traffic. In *Proceedings of the Global Communications Conference*, pages 1–5, 2010.

[50] R. Dubin, O. Hadar, A. Noam, and R. Ohayon. Progressive download video rate traffic shaping using tcp window and deep packet inspection. In *World Congress in Computer Science, Computer Engineering and Applied Computing (WORLDCOMP)*, 2012.

[51] J. Friedman, T. Hastie, and R. Tibshirani. *The elements of statistical learning*, volume 1. Springer, Berlin, 2001.

[52] A. Rao, A. Legout, Y. S. Lim, D. Towsley, C. Barakat, and W. Dabbous. Network characteristics of video streaming traffic. In *Proceedings of the Seventh COnference on Emerging Networking EXperiments and Technologies*, CoNEXT, pages 25:1–25:12, 2011.

[53] P. Ameigeiras, J. Ramos-Muoz, J. Navarro-Ortiz, and J. M. Lpez-Soler. Analysis and modelling of youtube traffic. *Transactions on Emerging Telecommunications Technologies*, 23(4):360–377, 2012.

[54] T. M. Cover and P. E. Hart. Nearest neighbor pattern classification. *IEEE Transactions on Information Theory*, 13(1):21–27, 1967.

[55] O. Boiman, E. Shechtman, and M. Irani. In defense of nearest-neighbor based image classification. In *Computer Vision and Pattern Recognition*, 2008.

[56] C. Cortes and V. Vapnik. Support-vector networks. *Machine learning*, 20(3):273–297, 1995.

[57] Y. Chen, E. K. Garcia, M. R. Gupta, A. Rahimi, and L. Cazzanti. Similarity-based classification: concepts and algorithms. *The Journal of Machine Learning Research*, 10:747–776, 2009.

[58] R. Rifkin and A. Klautau. In defense of one-vs-all classification. *The Journal of Machine Learning Research*, 5:101–141, 2004.

[59] Youtube statistics. http://www.w3schools.com/browsers/browsers_stats.asp, 2016. Accessed: 28/4/2017.

[60] Selenium. http://www.seleniumhq.org/. Accessed: 2016-02-28.

[61] Chromedriver - webdriver for chrome. https://sites.google.com/a/chromium.org/chromedriver/. Accessed: 2016-02-28.

[62] Adblock plus. https://adblockplus.org/. Accessed: 2016-02-28.

[63] Clumsy. https://jagt.github.io/clumsy/, 2016. Accessed: 2016-02-28.