Source identification of encrypted video traffic in the presence of heterogeneous network traffic

Yan Shi, Arun Ross, Subir Biswas

Abstract

This paper uses Traffic Analysis (TA) for identifying sources of tunneled video streaming traffic. The key idea is to examine encrypted and tunneled video streaming traffic at a Soft-Margin Firewall (SMFW) that is located near the streaming client in order to identify undesirable traffic sources and to block or throttle traffic from such sources. The key contribution of the paper is the design and experimental evaluation of a novel two-stage classifier for identifying specific video sources from heterogeneous background traffic within an encrypted tunnel. Being able to classify video sources in the presence of such traffic mixture can help the SMFW to successfully obfuscate or block undesired video browsing while allowing a user to receive traffic from legitimate applications running over the same encrypted tunnel. Using OpenVPN servers for creating encryption tunnels, experiments were conducted on a large number of popular video streaming sources with various combinations of feature extraction and data processing techniques to verify the effectiveness of the two-stage classifier. It was experimentally demonstrated that by using the proposed two-stage classifier, it is indeed possible to identify video streaming sources with high accuracy and low false-positive rates in the presence of non-video background traffic within an encrypted tunnel.

1. Introduction

With increasing bandwidth availability, streaming video has become one of the major services of the Internet. The top 2 most heavily accessed video streaming sites, Netflix and YouTube, constitute roughly 50% of the North American Internet traffic according to a 2014 report [1]. As the popularity of video streaming sites grows, their usage starts to spread into private enterprises, where watching certain videos may be undesirable. For example, employers may not want certain video clips containing political, sexual, or violence related messages to be watched by their employees. Often the administrator of an enterprise network may want to block users from watching videos in order to conserve bandwidth and/or to maintain enterprise productivity. At the same time, an administrator may wish to allow users to access some of the video streaming sites due to other business reasons.

Streaming video from specific sources can be detected and subsequently blocked by adding appropriate firewall filters based on the IP packet header information. However, the problem gets complicated when the usage of encrypted tunnels, such as Virtual Private Networks (VPNs) [2] or proxies, are factored in. In this case, the information present in packet headers does not represent their actual destinations. This is because the actual headers are encrypted and replaced by the headers of the tunneling protocol instead. An example of avoiding traffic classification by going through a VPN tunnel is shown in Fig. 1. In such a case, the source, the destination, and the port numbers in a packet are encrypted, and therefore inaccessible to the firewall. As a result, such traffic cannot be blocked by the firewall using packet inspection.

This specific problem is addressed in this work by identifying video packet sources using traffic analysis as opposed to relying on packet inspection. To that end, there is a need to not only classify traffic based on the underlying video transport protocols, but also based on its source servers.

Traffic Analysis (TA) [3] is a commonly used method for retrieving information from traffic flow when the traffic itself is encrypted. The basic assumption is that even if the traffic is encrypted, the underlying protocol it uses might still leave a distinctive signature in the traffic flow. By applying machine learning methods, a classifier can be trained using existing traffic flow, and new encrypted traffic can be classified using that training model. TA is effective for tunneled traffic since its classification is based on the statistical signature in traffic metadata including, packet size, timing, and direction.

* Corresponding author.
E-mail address: shiyan3@msu.edu (Y. Shi).

https://doi.org/10.1016/j.comcom.2018.07.019
Received 2 January 2018; Received in revised form 31 May 2018; Accepted 16 July 2018
Available online 26 July 2018
0140-3664/ © 2018 Published by Elsevier B.V.
The focus of TA research so far is on its usage (and negative impact on privacy) as an attack vector, while in this work the authors propose a beneficial use of the technology as a way to allow network traffic management on encrypted traffic with minimal amount of information.

The proposed system is to build a novel firewall system based on the TA framework proposed in this paper. The proposed firewall is unique in that its aim is to disrupt the service of certain streaming providers instead of blocking all packets from their servers. We give the name “Soft-Margin Firewall” (SMFW) to such a system. As shown in Fig. 2, an SMFW will collect knowledge by training a machine-learning classifier with known data, and the trained classifier would be used by the SMFW to make decisions about whether to block or allow the traffic. In this work, we target video streaming traffic as a subject because of its importance in today’s network traffic.

The major challenge of this approach is for TA based application to be able to recognize the streaming providers of the traffic, yet with sufficient generalizability to recognize unknown traffic from the same provider. For the sake of practicality, the system also must be resistant to heterogeneous traffic streams, which are defined as traffic streams that are comprised of traffic generated by different application layer protocols.

The core contributions of this paper are: 1) a TA based firewall framework, and 2) a two-stage classification scheme as a solution to the mixed-flow problem with heterogeneous traffic streams. A review of TA as well as other related literature is provided in section II. The design of the two-stage classifier is presented in section III. Details of the design of the test setup are elaborated in section IV. Experimental details and results are provided in section V. The experiments include both classification of video streaming data with web background traffic and an additional case in which BitTorrent background traffic is involved.

2. Related works

2.1. Traffic analysis

Traffic Analysis (TA) is a machine-learning based technique to inspect network traffic and determine its nature, such as application layer protocol, traffic source, and destination, among other properties. TA works with metadata such as packet direction, packet size, packet count, and timing. Interesting pieces of information are then extracted from the traffic and combined into a feature vector in the feature space. A classifier is then trained on this feature space to distinguish between various classes. The classifier can then classify unknown traffic trace and assign it to one of the classes.

Since masking traffic entirely is not practical, TA is effective in situations where extracting information from encrypted and obfuscated traffic is the goal. TA is proven to be effective in identifying security-enhancing protocols such as the Onion Router (Tor) protocol [4,5,6] and other commonplace protocols including FTP, IMAP and HTTP [7]. The work of Hermann et al. [8] is an early piece that established that good recognition can be achieved when using a conventional VPN system. Panchenko et al. [4] target the Tor protocol with a combination of features, including traffic amount, packet count, packet size among others, and achieved a significantly higher recognition rate (55%) compared to previous works. This is considered a breakthrough and helped popularize Tor as a target for further TA based researches, including [5,9,6]. At the same time, other researchers expressed concern on the practicality of TA attacks, since most proponents use idealized scenarios that are not representative of the real-world traffic [10]. All of the above works assume the role of an attacker when performing TA, and the targets are usually specific web pages, since revealing the browsing history of the target is the attacker’s goal.

On the other hand, there are relatively few researches on streaming video source identification. In [11], the authors present a TA based framework for recognizing video content from streaming traffic. Other works from the same researchers [12] have used a wavelet-based feature to identify video content. The application of both methods is limited because the video content needs to be known, which is an unrealistic scenario for an SMFW configuration. In [13], the authors present the detection results on the traffic data for several popular video streaming sites with various combinations of classifier and features. In [14], several different aspects of the feature set are explored. It identifies Packet Size Distribution as one of the effective classification features for identifying protocols used by tunneled video streaming traffic. It also shows that for the same streaming protocol in a secure tunnel, there exists sufficient difference between various video streaming sites that permit a classifier to separate them. The work in [15] explores packet arrival interval (PAI) as a feature for video streaming source identification. A detailed analysis of the PAI feature in [15] also revealed that the network path for traffic flow could leave a signature in PAI. The results in [14,15] pave the way for site-based TA applications, as attempted in this work. The classifier in this paper is designed based on the previous results described in [13,14,15]. It uses the classification method that was proven to work well in this context, and the feature set will include the features that were proven to work well (packet size and PAI).

Another issue with previous works on TA is that they assume homogeneity in network traffic. In other words, the traffic flow in the tunnel is assumed to contain traffic of only one protocol type from one source to one destination. The reality of tunneled traffic (e.g., through encrypted VPN or Proxy server) is that it often contains traffic of different protocol types from different sources, and to different destinations. The mixture of traffic can alter features significantly and is identified as a limiting factor in the real-world applicability of TA [10]. This mixed-flow problem, which to best of our knowledge has not been addressed by other works, is the target of this work. Mixed (heterogeneous) traffic in a tunnel is defined as traffic flow in which there is a
video streaming flow coupled with an unspecified volume of non-video traffic in the tunnel. Pure (homogeneous) traffic flow in a tunnel is defined as traffic flow in which there is just one video streaming traffic flow. Traffic that contains more than one video streams is excluded from the scope of this work, but will be addressed in future works.

This work is based on our preliminary research in [16] which developed and evaluated the performance a two-stage classifier in the presence of BitTorrent Traffic and Video Streaming Traffic. It was concluded in [16] that a two-stage classifier can successfully boost the performance of a source-identification solution when there is heterogeneous traffic within an encrypted tunnel. Building on the preliminary concepts from [16], in this paper, we develop an architecture and the related methods with the following major goals. First, the preliminary design of the two-stage classifier is significantly augmented with new methods for feature selection (i.e., Correlation Feature Selection (CFS)), and various forms of mix-and-match training of both the classifier stages. Second, a new, much larger dataset that consists of web traffic and BitTorrent traffic is included to evaluate the performance of the classifier architecture. Third, new architecture level performance indices are developed for generalized evaluation of the two-stage classifier architecture. Finally, Packet Size Interval (PAI) is analyzed in detail to reveal its contribution in the feature-set for enhancing the performance of source identification.

2.2. Video streaming technology

Streaming protocols: Streaming protocols are designed to ensure good streaming video quality in the presence of limited and often fluctuating network bandwidth. A list of the video streaming providers and the associated streaming protocol is provided in Table 1. Real-time Messaging Protocol (RTMP) [17] is used by most of the providers, while Smooth Streaming [18] is used by only 2 of them (Amazon and Netflix). Those two adopters of Smooth Streaming, however, have a significant market share in the online video rental business. Both protocols are targeted in this work.

2.3. Virtual private network

A VPN is a means to connect isolated private networks through public Internet [19]. VPN keeps sensitive information private by transmitting the network traffic through a secure tunnel. The tunnel ensures that the message body and the real source/destination information are encrypted between the 2 ends of the tunnel. The tunnel can be created on top of public networks, where it is possible for adversaries to tap into the traffic yet unable to decrypt the information or reveal the actual source and destination.

There are many variations of VPN in terms of the tunneling protocol a VPN uses to create a secure tunnel. Two of the popular protocols in use are the Internet Protocol Security (IPsec) [20], and the Secure Socket Layer/Transport Layer Security (SSL/TLS) [21], OpenVPN [22], the VPN solution used in this work, supports an SSL-based tunneling protocol.

Video streaming traffic that is being tunneled through VPN services is difficult to identify because of the added obfuscation, padding, encryption, packet fragmentation, etc. Different VPN solutions may have different ways of obfuscating traffic. Combined with the diverse types of video streaming protocol, a significant variation could be introduced into the traffic stream by these operations. In [23], the authors survey several obfuscation techniques, and categorize them into four groups: encryption, randomization (make traffic behave like random data), mimicry (masquerading the traffic to look like another protocol) and tunneling (embedding traffic as the payload of another protocol). VPNs belong to the tunneling group, but they can also employ techniques from other groups to further obfuscate traffic. An example of obfuscation technology is Obfsproxy [23], which is a pluggable obfuscation layer that is developed by the same team that develops Tor. Obfsproxy extends the traffic mimicry functionality of Tor, which makes its traffic look like SSL traffic, to be able to masquerade Tor traffic as a wide range of protocols. Obfsproxy makes the identification of traffic more difficult by padding packets with misleading information, changing delays between packets, and inserting dummy packets into the traffic flow. To make the study more focused, we choose OpenVPN, which is widely used by inexpensive personal VPN services as well as corporate VPN services, as the target platform [2]. OpenVPN does not have additional protocol obfuscation besides tunneling. The concepts proposed in this paper can scale to other streaming protocols and VPN/Proxy mechanisms.

3. Classifier design

The key part of the architecture in Fig. 2 is a novel classifier, which is shown in Fig. 3.

3.1. Two-Stage classifier

Since the composition of a network traffic flow could change over time, the classifier evaluates traffic in small time slices, which are termed as samples. Each sample is a temporal portion of traffic flow recorded within a certain period. The proposed classifier is split into two stages. In the first stage, a classifier is fed a combination of features extracted from the sampled traffic flow, and it determines if the sample corresponds to a mixed traffic flow or a pure flow. For a pure flow case, the second stage determines the sample’s video source. Meaning, only those samples that are classified as pure in stage-1 are passed on to stage-2.

Two conditions are assumed for the classifier to work: 1) the target video streaming traffic forms the majority of the traffic in a tunnel, and 2) non-video streaming traffic only occurs sporadically. These assumptions are based on the observation that video consumption demands high level of user attention. Therefore, the web traffic will be sporadic. A sample may be mixed or pure depending on the

![Fig. 3. Two-stage traffic source classification scheme.](image-url)
composition of traffic in the tunnel during that period. We assume 50% of the video watching sessions are accompanied by web browsing activities. We believe these assumptions to be reasonably close to realistic human behavior.

One goal of the two-stage classifier is to maximally reduce the chance of false positives. False positives are costlier since they disrupt normal services going through an SMFW. Following the example of [24], it is done on stage-1 classifier by adjusting the cost of false positives to be higher than false negatives, while on the stage-2 classifier, it is done by first converting the n-class classifier to binary classifiers and then assign a higher cost for the false positive detection (detection of a certain video streaming class when its traffic is not present in data) a higher cost than the false negative detection. Since there are 7 classes in the experiment, the stage-2 classifier is actually an ensemble of 7 binary classifiers. The ensemble is based on a max-confidence rank algorithm, which takes the classifier output that has the highest confidence rank as the final output. For a RandomForest Classifier, that means the output that has the most popular votes wins. The classifier is designed to detect traffic samples from the targeted video streaming providers with a minimal amount of false positive.

3.2. Feature set

For each packet within an encrypted tunnel, the SMFW can observe a 3-tuple: \((time, direction, size)\). \(time\) is the observation timestamp, the \(direction\) is a binary value indicating the direction of that packet (either from inside the SMFW to the outside, or vice versa), and \(size\) is the number of bytes in that packet. Converting the raw data to this tuple series is the first step in feature extraction. Although the tuple series is already a lot smaller than the raw traffic data, it can still be quite large, and the raw parameters may lack robustness to compensate for the pseudo-randomness of network traffic. Thus, practical features are usually combinations of one or more statistical characteristics of the tuple series. Criteria for a good feature set include: 1) distinctiveness: the feature should be able to capture the difference between the various traffic classes, 2) robustness: the feature set should minimize the influence of the inherent randomness in traffic, and 3) ease of computation: it should fit in a resource-limited environment such as a router. From the collected \((time, direction, size)\) information, the following features are computed.

Packet count: The number of packets in each sample is treated as one of the classification features. Data collected from the experiment shows the distribution of packet count to be appreciably correlated to the presence of mixture in video streaming traffic. A few examples of the difference in packet count distribution between mixed traffic and pure traffic are shown in Fig. 4. Amazon prime video traffic and BlinkX video traffic collected with or without mixed-in video traffic are shown. While it is more evident in the case of Amazon that pure traffic has a higher packet count, BlinkX traffic also shows the same tendency. This trend has been generally observed for other video sources and different traffic mixes. The difference between pure and mixed traffic is due to rate throttling when a client initiates multiple download sessions. The video streaming client can detect that the network interface is crowded and choose to play a lower bit-rate version of the video instead, which results in less video streaming traffic and in turn, less overall packet count [25]. The traffic in Fig. 4 is taken from without the tunnel, and it reflects the difference in traffic clearly.

Packet size distribution: Distribution of packet size is obtained by counting the number of packets in a sample as they fall into different bins according to their direction and size. This results in a histogram. In this work 30 bins that uniformly divide the range \([-1500, +1500]\) bytes are used to compute the feature. The packet size has a sign associated with it to indicate the direction of this packet (minus is upstream, and plus is downstream). In order to remove bias, before feature computation, all TCP ACK packets are removed from the stream based on the packet size below a threshold of 53 bytes [4]. The feature histogram is then calculated and normalized, i.e. the value of each bin is replaced by its value divided by the sum of all bins. This mitigates the influence of traffic load on the feature.

A few sample features that are collected from the experimental environment are shown in Fig. 5. Samples from 3 different sites both with and without the background traffic (labeled as Mixed and Pure respectively) are shown. The variation of the feature between different sites is clearly visible. Background traffic has an observable influence on the feature.

Packet arrival interval (PAI): PAI is defined as the distribution of inter-packet time. In what follows we show that PAI contains useful timing information, which can be helpful in source server classification. The following is an analysis of the PAI feature that is originally presented in [15]. Fig. 6 depicts two consecutive packet transmissions from the server to the client. The packets are sent from the server with an interval \(\Delta\), and they reach the client-side router at time instants \(T_1\) and \(T_2\). The PAI is defined as \(\Delta T = T_2 - T_1\). The other timing parameter of interest is the Inter Packet Delay Variation (IPDV) [26], represented by \(\Delta D = D_2 - D_1\). The parameter \(\Delta\) is termed as Inter Packet Generation Delay (IPGD), which depends on specific video content and coding methods. Path delays \(D_1\) and \(D_2\) and \(\Delta D\) represent the properties of the specific route including the congestion situations in the intermediate routers. The quantity \(\Delta T\) depends on both \(\Delta\) and \(\Delta D\). From Fig. 6,
\[ \delta T = \delta D + \Delta \] when the packets arrive in-order (i.e. \( T_1 < T_2 \)). Considering both in-order and out-of-order cases, PAI can be generalized as:

\[ \delta T = [\delta D + \Delta] \]  

(1)

Both \( \delta D \) and \( \Delta \) can be modeled as stochastic processes. Fig. 7 depicts example \( \delta T \) distributions for video being downloaded from Amazon, YouTube, and Netflix. The traffic samples presented in the figure are taken from 1-minute slices of actual traffic data from the mentioned sites. Then the distribution of PAI is extracted for all consecutive packet pairs. Note that the distributions are clearly bi-modal (having 2 peaks), with a stable saddle point at approximately 0.66 ms between the 2 modes. The first peak is at 10 µs, while the second peak is at 1 ms-10 ms depending on the site. The general observation here is that PAI, as observed at the router near the client, generally show a bi-modal distribution irrespective of the content and the streaming protocol and video coding combinations used by the server.

To understand the distribution, we start by examining shifted gamma distribution [27], which is an accepted realistic model for expressing end-to-end route delay. This model assumes that the path delays (i.e., \( D_1 \) and \( D_2 \)) have a constant component \( t_0 \), and another component that follows a gamma distribution. It works well when all links in a path have similar delays. The probability density function of this delay model is shown in Eq. 2. Here \( t_0 \) is the shift, while \( \alpha \) and \( \beta \) are the respective shape and rate factors of the gamma distribution.

\[ f_{\delta D}(t) = \frac{\beta^\alpha}{\Gamma(\alpha)} (t - t_0)^{\alpha-1} e^{-\beta(t-t_0)} \]  

(2)

With the assumption that end-to-end delays are independent the probability density function of IPDV can be derived from Eq. 2 as its autocorrelation function.

\[ f_{\delta D}(t) = \int_{-\infty}^{\infty} f_{\delta D}(t) f_{\delta D}(t-\tau) d\tau \]  

(3)

Because of the symmetrical nature of IPDV, its PDF when \( \tau < 0 \) is a mirror of the part when \( \tau \geq 0 \). The shape of \( f_{\delta D}(t) \) is shown in Fig. 8. It is Gaussian-like because it is the sum of a number of independent, identically distributed random variables and therefore is subject to the central limit theorem. For comparison, a second dotted line shows a Gaussian density function with identical parameters.

Since \( \delta D \) depends on the property of an end-to-end route, it can generally be assumed to be stationary as long as the network conditions change with a time constant that is larger than the classifier-training interval. The author of [28] points out that the stationary assumption can be effective for up to an hour, which is sufficient for the classifier model to be updated with new training done based on data collected within the hour.

Observe that the distribution \( f_{\delta D}(t) \) in Fig. 8 has only one mode, while the distribution in Fig. 7 for \( \delta T \) has two modes (2 major peak concentrations). There is one concentration of PAI from 1us to 1 ms, which corresponds to the mode shown in Fig. 8. Another concentration happens between 10 ms to 10 s, which cannot be explained by IPDV. This other mode in Fig. 7 is contributed by Inter Packet Generation Delay (IPGD), which is present due to the activity of application layer protocols. It can also generally be assumed to be stationary when the coded video generation is modeled as an on-off process as shown below.

To investigate this further, we examine probed (i.e., at the \( r \) in Fig. 2) traffic samples from various video-streaming services as depicted in Fig. 9. A common visual pattern across all the samples is that the traffic switches between two distinct states. One with short periods of high intensity, and the other with longer duration lower intensity. This alternating behavior is because data chunks in video streaming protocols are downloaded in bursts. Fig. 9 generally confirms this.

This general observation can be modeled using an on-off model [29]. Under such a model, the packet generation process \( \Delta \) can be
the shape is very similar to tf t f t. We can use this value as an estimation (i.e., 1500 bytes) is about 10 us is because, for a 1 Gbps node, the source delay of a typical IP packet streaming protocol has either because the packets are being queued but not sent yet, or the in Fig. 7exists because the source is switched to o

assumed to be stationary when the system is in one of the two states. The process can be modeled as:

$$\Delta = \begin{cases} \Delta_{on} & P(on) \\ \Delta_{off} & P(off) \end{cases}$$

$$\Delta_{on} \sim \Gamma(\alpha, \beta_{on}), \ \Delta_{off} \sim \Gamma(\alpha, \beta_{off})$$

Note that the position of the first peak in Fig. 7 is close to 10us. This is because, for a 1 Gbps node, the source delay of a typical IP packet (i.e., 1500 bytes) is about 10us. We can use this value as an estimation of E(\Delta_{on}). The shape of this peak carries IPDV information with little distortion because the highly concentrated distribution of \( \Delta \) acts essentially as a Dirac impulse function. And since the distribution of \( f_{IPV}(t) = f_{IS}(t) f_{IPG}(t) \), the shape is very similar to \( f_{IS}(t) \). The second peak in Fig. 7 exists because the source is switched to off state occasionally, either because the packets are being queued but not sent yet, or the streaming protocol has finished transmitting a data block and waiting for the time to transmit the next block.

To summarize, the above analysis confirms the statement in Eq. 1, which claims that the feature Packet Arrival Interval (PAI) does capture information about both Inter-packet Delay Variation (IPDV) and the Inter Packer Generation Delay (IPGD). While IPDV reflects network conditions, IPGD indicates application-layer properties. The hypothesis here is that with such information diversity about the underlying streams, PAI would be able to classify video sources with high enough accuracy for our application.

3.3. Automated feature selection

We use an automated process by which the most relevant features in a feature set can be selected by a feature selector algorithm for a particular classification problem. Due to the high number of features involved in Traffic Analysis, and the low signal-to-noise ratio of some features, feature selection is conducted automatically rather than manually. A naïve approach would be to generate different subsets of the original feature set using all possible combination of attributes, train classifiers based on them and measure the performance of those classifiers against a common test set. The subset that performs best would win the selection process.

However, the cost of such a search would be too high if the original feature set contains many attributes. Therefore, a more practical approach [30] to feature selection involves the use of a suboptimal search of the feature space. Suboptimal search makes the process faster. The search can also be further optimized with a proxy measure chosen as the optimization goal rather than using the classification performance as a goal.

In this work, we use the Correlation Feature Selection (CFS) [30] measure to select the optimal features. CFS is only used to on stage-1 classifier because stage-1 needs to run quickly, while stage-2 classifier uses the full feature since it needs more resolution to classify traffic source. The stages of our classification architecture were summarized in Fig. 3. CFS constructs subsets based on the assumptions that: 1) a good subset should contain mostly uncorrelated features, and that a feature should provide independent information to the subset instead of merely being a combination of other features in the subset, and 2) that features could be weighed individually against the classification problem, and a combination of features that perform well individually indicates a subset that could perform well. We use CFS and the “BestFirst” condition [31] from Weka [32] library for this purpose. Weka is a tool for performing machine-learning tasks, which is developed by The University of Waikato. It provides common tools for machine learning applications, which includes the CFS algorithm used in this work. With the “BestFirst” condition, the CFS selector will greedily search the attribute space for the next attribute that has the best evaluation to the selected subset of attributes. The algorithm also backtracks to try a different path when there is no improvement for several iterations, until a stop condition is reached.

4. Experimental setup

To test the performance of the proposed classification scheme, a series of experiments are set up as follows. An OpenVPN server [22] was installed on the campus of Michigan State University. The detail of the setup is shown in Fig. 10. The videos are streamed from multiple streaming providers to a client location that is set off campus (so that the client is in a different domain than the VPN server). In OpenVPN version 1.5, a tunnel can be either UDP or TCP based. We use the UDP tunnel because it is more efficient and therefore used by more service providers. An encrypted OpenVPN tunnel is created from the client to the VPN server, and then the videos are streamed from commercial streaming servers such as YouTube, Netflix etc. (see Table 1) to the client via the OpenVPN tunnel.

A router with Wireshark [33] probe is installed at client’s home. The traffic is collected at this router. The setup emulates a real-world operational scenario of the proposed SMFW. Numerous video streaming servers are chosen for data collection during the experiment. A complete list of the servers is given in Table 1.

Most of the video streaming servers use content distribution network in order to speed up service delivery and save cost. To enable that, the actual server machine that a client downloads from is a function of the client’s location, network load, and various other factors. Servers from our experiments are resolved and matched in an IP geolocation dataset to reveal their actual location. The result is also shown in Table 2. The number of samples collected during the test period is shown in Table 3. The difference in sample size from one site to another is a result of the length of the chosen video stream (each sample is 20 seconds long) and the condition by which samples are filtered (samples

![Fig. 9. On-off patterns in video streaming traffic samples.](image)

![Fig. 10. Experimental setup with OpenVPN.](image)
with less than 1500 packets/sec are discarded). The dataset is collected over a 6-week period at the client location. The set S contains 147,563 samples, each 20 s long. Normal Internet usage by human participants within the same period is sampled as “Profile” traffic, which provides a background for the samples to be compared against. The data is collected using automated script that generates random web page views during video playbacks. The average rate is set to be 1 view/minute, and 50% of the video streams are generated random web page views during video playbacks. The average rate is set to be 1 view/minute, and 50% of the video streams are randomly chosen to have mixed-in web traffic.

Data processing: All traffic streams collected from the experimental setup are cut into 20-second slices and a feature vector is extracted from each slice. The feature vector contains packet size distribution (30 attributes), packet count (1 attribute) and PAI (40 attributes) of each sample. The samples are collected either with or without background traffic. The resulting dataset is tagged with 2 labels, one is whether the dataset is collected with or without background traffic present, and another is the video streaming site the traffic originates from. Packet count, packet size distribution, and PAI features are extracted from every sample and the features are concatenated into a feature vector the format of which is shown in Fig. 11.

For a fair comparison, all classification models in this experiment are trained with RandomForest classifier from Weka machine learning library. A RandomForest classifier [34] is an aggregation of tree classifiers that form a “forest”. Each tree in the forest is trained on a random subset of the attributes. The classifier outputs the result of a popular vote of all the tree classifiers. A tree classifier is a type of classifier which memorizes the training set as a hierarchy of conditions. A sample is classified by starting at the “root” of the tree, which contains the broadest condition, then following the conditional branches that match the sample, until the evaluation reaches one of the leaves of the tree. The label associated with that leaf is the output for the sample. The dataset is separated into 2 different partitions. One partition, which contains only “pure” samples and “profile” samples, is for the training of the stage-2 classifiers, while the other, which contains “pure” samples, “mixed” samples, as well as “profile” samples, is used for training stage-2 classifiers.

Stage-1 classifier solves a binary classification problem of distinguishing traffic samples with background traffic from those samples that do not have background traffic. Before training, the partition for stage-1 is further split into 3 sub-partitions: one for CFS feature selection, one for training, and one for testing. A feature selector is first trained using the CFS partition. The 2 candidates of the stage-1 classifier are trained using different subsets of attributes chosen from the training set. The first candidate is trained using the packet count feature alone, and the 1st label (video streaming site) is erased. The second candidate is trained using an optimal subset of attributes selected by the feature selector, with the 1st label also erased. To decrease the chance of misclassifying mixed samples as pure samples, the cost for misclassifying mixed samples as pure is set to be 1.5 times the cost of misclassifying a pure sample as a mixed one. The data flow for training stage-1 classifier is shown in Fig. 12. In the following text, the 1st candidate classifier will be referred to as Stage1_PacketCount, and the 2nd candidate will be referred to as Stage1_CFS. The test set will be used in the testing procedure, described in Fig. 14.

After passing the first stage classifier, samples are either labeled as “mixed” or “pure”. The stage-2 classifier will only examine the samples labeled as pure by the stage-1 classifier. Two candidates for the stage-2 classifier are trained. The first candidate is a classifier that is only trained with the subset of all pure samples from the stage-2 partition, and the second classifier is trained with the entire stage-2 partition. Both candidates are comprised of 7 RandomForest classifiers as specified in section III. Each of the RandomForest classifiers is cost sensitive. The cost of making a false positive prediction is 10 times that of making a false negative prediction. Finally, the prediction of a sample is determined by the “winning” classifier, which is the classifier that 1) made a positive prediction, and 2) made the prediction with the highest confidence among the classifiers. Also, since there is a default class in the training set (class 8 which are comprised of “profile” samples) every sample that is rejected by all other classifiers is regarded as class 8. Henceforth, the candidates will be called Stage2_Pure and Stage2_Mixed respectively. The data flow for training the stage-2 classifier is shown in Fig. 13.

The experiments are run from a personal computer with an AMD 7700 K CPU. The Java heap size for running the weka classifiers is 1GB. The run time for training stage-1 classifier is about 165 seconds for the CFS dataset, and 22 seconds for the packet count dataset, while the stage-2 classifiers range from the shortest 75 seconds (BlinkX) to the longest 367 seconds (YouTube) depending on the size of the dataset. Once trained, the evaluation time of both stage-1 classifier and the stage-2 classifier on the test set, which contains 29,512 samples, are

<table>
<thead>
<tr>
<th>Service</th>
<th>Server location</th>
<th>Service</th>
<th>Server location</th>
</tr>
</thead>
<tbody>
<tr>
<td>Amazon</td>
<td>Seaford, Delaware</td>
<td>CNN News</td>
<td>Ann Arbor, MI</td>
</tr>
<tr>
<td>BlinkX</td>
<td>Seattle, Washington</td>
<td>YouTube</td>
<td>Mountain View, CA</td>
</tr>
<tr>
<td>DailyMotion</td>
<td>New York, NY</td>
<td>Netflix</td>
<td>Southfield, MI</td>
</tr>
</tbody>
</table>

Table 3
Sample counts.

<table>
<thead>
<tr>
<th>Service</th>
<th>Pure</th>
<th>Mixed</th>
<th>Service</th>
<th>Pure</th>
<th>Mixed</th>
</tr>
</thead>
<tbody>
<tr>
<td>Amazon</td>
<td>13,019</td>
<td>14,002</td>
<td>CNN News</td>
<td>17,977</td>
<td>16,777</td>
</tr>
<tr>
<td>BlinkX</td>
<td>2942</td>
<td>2846</td>
<td>YouTube</td>
<td>5616</td>
<td>6041</td>
</tr>
<tr>
<td>CNN News</td>
<td>10,999</td>
<td>10,145</td>
<td>Netflix</td>
<td>9866</td>
<td>10,524</td>
</tr>
<tr>
<td>DailyMotion</td>
<td>9501</td>
<td>10,026</td>
<td>Profile</td>
<td></td>
<td>5989</td>
</tr>
</tbody>
</table>

Fig. 12. Training the stage-1 classifier.
within a few seconds.

Because there are two classification stages, the performance evaluation is more complex compared to a usual single-stage classification scheme. First, the same test set is used in testing both stages. The stage-1 classifier will receive a selected subset of the attributes (Packet Count or CFS chosen) while the stage-2 classifier will receive the full range of attributes. Second, the result from stage-1 classifier will have an impact on the stage-2 classifier, so the test set has to be separated into subsets based on predictions made by stage-1 classifier. From here on, $T$ is defined as the entire test set, $T_{pure}$ is defined as the part of the test set that is pure, according to the ground truth, while $T_{pure}$ is the part of the test set that is pure according to the prediction of the stage-1 classifier. Similarly, $T_{mixed}$ and $T_{mixed}$ are the mixed part of test set according to ground truth and prediction of stage-1 classifier respectively.

The performance of the stage-2 classifier has to be gauged in a way that respects the output of the stage-1 classifier. The overall process is shown in Fig. 14. After the samples are tagged by the stage-1 classifier, both $T_{pure}$ and $T_{mixed}$ the stage-2 classifier will be tested twice, first time with $T_{pure}$ as input and second time with $T_{pure}T_{mixed}$ in the test separately. Note it that the design goal of the two-stage classifier is to filter out unqualified samples and as such, only the result of stage-2 classifier on $T_{pure}$ is the actually intended input for the stage-2 classifier. However, evaluating the performance of stage-2 classifier on $T$ allows us to measure its performance gain with respect to the scenario where there is no stage-1 classifier. The output of the stage-2 classifier is the estimated label of the sample (possible values are listed in Table 2). We define the performance indicators in (Eqs. 5, 6, 7, 8).

**Filtered Accuracy**: defined as the accuracy of stage-2 classifier on $T_{pure}$. (Eq. 5)

$$\frac{|x \in T \cap x \text{ predicted correctly}|}{T_{pure}}$$

**Unfiltered Accuracy**: defined as the accuracy of the stage-2 classifier on $T$. (Eq. 6).

$$\frac{|x \in T \cap x \text{ predicted correctly}|}{T}$$

**Filtered False Positive Rate (FPR)**: defined as the weighted average of the FPR of the 7 classes on $T_{pure}$. (Eq. 7). Here $T_{pure}$ in the equation stands for the subset of samples that belong to class $i$ in $T_{pure}$.

$$\frac{1}{n} \sum_{i} |x \in T_{pure} - (T_{pure}) \cap x \text{ predicted as class } i|$$

**Unfiltered False Positive Rate**: defined as the weighted average of the FPR of the 7 classes on $T$. (Eq. 8). Here $T_{i}$ in the equation stands for the subset of samples that belong to class $i$ in $T$.

$$\frac{1}{n} \sum_{i} |x \in T - T_{i} \cap x \text{ predicted as class } i|$$

The definition of these 4 performance indices is intended to reveal a full picture of the effect of the two-stage classification process. It is known [13] that mixture leads to lower recognition rate in stage-2 classifier, but it may also result in lower false positive because of removal of potentially noisy (therefore lower in quality) samples early on. It is important to evaluate the performance gain from introducing the stage-1 classifier, which is defined in (Eq. 9):

$$\eta = \frac{\text{Filtered False Positive Rate}}{\text{Unfiltered False Positive Rate}}$$

This index shows the improvement introduced by the two-stage classifier, in terms of the reduction of false positive detection rate. The smaller the ratio, the more the stage-1 classifier has contributed to the reduction of false positives.

5. Experimental results

5.1. Feature selection

Correlation Feature Selector (CFS) algorithm is first applied to the CFS slice in the stage-1 dataset. The results of the CFS selector is then applied to the stage-1 training set and also saved for the testing stage since the same selector has to be applied to the test set as well. Streaming-site label (see Fig. 11) is erased when invoking CFS. The attributes chosen by CFS for stage-1 is shown in Fig. 15. Each square block in the figure represents one attribute in the combined feature vector, and those marked as black are the ones chosen by the CFS algorithm. The CFS algorithm chose the features partly from the packet size distribution and partly from the packet arrival interval to form the optimal subset. Packet count is not included in the selected subset.

An interesting observation is a tendency for the CFS algorithm to focus on certain regions of the PAI feature. In both cases, the chosen subset of PAI concentrates on certain regions. Those regions correspond to the small end of the time scale (< 50μs), and the larger end of the time scale (> 0.17 s). According to the observations formulated in [15], the smaller end is where the influence of network condition is coded, while the larger end is where the source characteristic is coded. The fact that CFS algorithm picks up those regions is a corroborated of the observations in [15].

5.2. Scenario #1 (Stage1 PacketCount + stage2 pure)

Scenario #1 simulates the following operational condition in a real-world traffic classification system. When collecting training data from the SMFW, all traces are collected without any background traffic mixed in, and packet count is used as an indicator of the pure/mixed nature of traffic trace in the stage-1 of the classification process. In this case, the confusion matrix of the stage-1 classifier is shown in Table 4.
The matrix is normalized by the sum of each row, so the table actually shows true positive/false positive, true negative and false negative rates. Because the training set is generated differently between the Stage2_Pure and Stage2_Mixed classifiers, there are 2 different confusion matrices.

The accuracy of Stage1_PacketCount is 41.33% and the false positive rate is 6.97%. On top of the results of matrix of Stage2_Pure and Stage2_Mixed classification matrices.

As shown in Table 5, Samples rejected by all 7 classifiers are considered to be of class 8 (default class).

The filtered accuracy of Stage2_Pure, with the filtering of Stage1_PacketCount, is 91.30%, compared to the unfiltered 80.9%, while only 11.95% of the samples passed the filter. The \( \eta \) value in this case is 18.37. The Accuracies on different subsets are shown in Fig. 16.

We can see that while the accuracy of Stage1_PacketCount is low, the filtering still boosted the filtered accuracy of Stage2_Pure from the unfiltered 80.49% to 91.30%, at the expense of passing only 11.95% of the samples to the 2nd stage.

### Table 4

Confusion matrix (Stage1_PacketCount).

<table>
<thead>
<tr>
<th>Test passes</th>
<th>Mixed</th>
<th>Pure</th>
</tr>
</thead>
<tbody>
<tr>
<td>Stage2_Pure</td>
<td>Mixed</td>
<td>93.03%</td>
</tr>
<tr>
<td>Pure</td>
<td></td>
<td>89.47%</td>
</tr>
<tr>
<td>Stage2_Mixed</td>
<td>Mixed</td>
<td>90.03%</td>
</tr>
<tr>
<td>Pure</td>
<td></td>
<td>88.18%</td>
</tr>
</tbody>
</table>

### Table 5

Confusion matrix (Stage2_Pure).

<table>
<thead>
<tr>
<th></th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>95%</td>
<td>0%</td>
<td>3%</td>
<td>0%</td>
<td>1%</td>
<td>1%</td>
<td>449%</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>4%</td>
<td>94%</td>
<td>2%</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>100%</td>
<td>0%</td>
<td>88%</td>
<td>6%</td>
<td>1%</td>
<td>4%</td>
<td>1128%</td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>0%</td>
<td>0%</td>
<td>22%</td>
<td>74%</td>
<td>4%</td>
<td>1%</td>
<td>1482%</td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>1%</td>
<td>1%</td>
<td>1%</td>
<td>92%</td>
<td>0%</td>
<td>0%</td>
<td>416%</td>
<td></td>
</tr>
<tr>
<td>6</td>
<td>6%</td>
<td>0%</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>7</td>
<td>4%</td>
<td>33%</td>
<td>4%</td>
<td></td>
<td></td>
<td></td>
<td>2285%</td>
<td></td>
</tr>
</tbody>
</table>

5.3. Scenario #2 (Stage1_CFS + stage2_pure)

Scenario #2 simulates the following operational condition. When collecting training data from the SMFW, all traces are collected without any background traffic mixed in, and the CFS subset is used as an indicator of the pure/mixed nature of traffic trace in the stage-1 of the classification process. In this case, the confusion matrix of the stage-1 classifier is shown in Table 6.

The accuracy of Stage1_CFS is 58.05% and its false positive rate is 2.08%. The confusion matrix of Stage2_Pure, including both accepted samples and rejected samples, is shown in Table 5 since it’s the same as the Stage1_PacketCount classifier. The accuracy of Stage1_PacketCount is 32.50% and its false positive rate is 6.97%. The confusion matrix of Stage2_Mixed includes both accepted samples and rejected samples, is shown in Table 7. The performance indices are shown in Fig. 16. The filtering boosted the filtered accuracy of Stage2_Pure from the unfiltered 80.49% to 95.95%, while only 25.33% of the samples passed the filter. The \( \eta \) value in this case is 18.96.

5.4. Scenario #3 (Stage1_PacketCount + stage2_mixed)

Scenario #3 simulates the following operational condition. When collecting training data from the SMFW, all traces are collected with background traffic mixed in, and packet count is used as an indicator of the pure/mixed nature of traffic trace in the stage-1 of the classification process. In this case, the confusion matrix of the stage-1 classifier is shown in Table 6 since it’s the same Stage1_PacketCount classifier. The accuracy of Stage1_PacketCount is 32.50% and its false positive rate is 6.97%. The confusion matrix of Stage2_Mixed, including both accepted samples and rejected samples, is shown in Table 7. The performance indices are shown in Fig. 16. The filtering boosted the positive accuracy of Stage2_Mixed from the unfiltered 81.57% to a higher value (89.88%), while only 10.58% of the samples passed the stage-1 classifier. The \( \eta \) value in this case is 16.79.

5.5. Scenario #4 (Stage1_CFS + stage2_mixed)

Scenario #4 simulates the following operational condition. When collecting training data from the SMFW, the collection agent adds background traffic to the tunnel, and the optimal subset of feature chosen by CFS algorithm is used as an indicator of the pure/mixed nature of traffic trace in the stage-1 of the classification process. In this case, the confusion matrix of the stage-1 classifier is shown in Table 6 since the first stage classifier is also Stage1_CFS here.

### Table 6

Confusion Matrix (Stage1_CFS).

<table>
<thead>
<tr>
<th>Test passes</th>
<th>Mixed</th>
<th>Pure</th>
</tr>
</thead>
<tbody>
<tr>
<td>Stage2_Pure</td>
<td>Mixed</td>
<td>97.92%</td>
</tr>
<tr>
<td>Pure</td>
<td></td>
<td>62.62%</td>
</tr>
<tr>
<td>Stage2_Mixed</td>
<td>Mixed</td>
<td>97.92%</td>
</tr>
<tr>
<td>Pure</td>
<td></td>
<td>74.90%</td>
</tr>
</tbody>
</table>

### Table 7

Confusion matrix (Stage2_Mixed).

<table>
<thead>
<tr>
<th></th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>99%</td>
<td>0%</td>
<td>0%</td>
<td>0%</td>
<td>0%</td>
<td>0%</td>
<td>103%</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>1%</td>
<td>96%</td>
<td>2%</td>
<td>0%</td>
<td>1%</td>
<td>0%</td>
<td>444%</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>100%</td>
<td>0%</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>192%</td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>1%</td>
<td>94%</td>
<td>4%</td>
<td>0%</td>
<td>1%</td>
<td>1%</td>
<td>631%</td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>2%</td>
<td>1%</td>
<td>6%</td>
<td>90%</td>
<td>0%</td>
<td>1%</td>
<td>425%</td>
<td></td>
</tr>
<tr>
<td>6</td>
<td>3%</td>
<td>0%</td>
<td>1%</td>
<td>0%</td>
<td>94%</td>
<td>1%</td>
<td>271%</td>
<td></td>
</tr>
<tr>
<td>7</td>
<td>3%</td>
<td>5%</td>
<td>1%</td>
<td>1%</td>
<td>91%</td>
<td>952%</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
On top of the results of Stage1_CFS, the confusion matrix of Stage2_Mixed, including both accepted samples and rejected samples, is shown in Table 7. The overall accuracy of Stage2_Mixed, if without the filtering of Stage1_CFS, is 83.22%. Accuracies on different subsets are shown in Fig. 16. We see that while the filtering of Stage1_CFS only performs moderately well, the filtering boosted the filtered accuracy of Stage2_Mixed from the unfiltered 83.22% to 95.56%, while only 33.16% of the samples passed the filter.

5.6. Discussions
Comparing the 4 different scenarios, it could be observed that with appropriate tuning, the two-stage classifier configuration is capable of boosting the performance of the RandomForest classifier to have higher accuracy and significantly lowering false positive rate. As can be seen in Fig. 16, the combination that got the most boost is the Stage1_PacketCount + Stage2_Mixed pair. But this comes at the expense that Fig. 16, the combination that got the most boost is the Stage1 PacketCount passes less samples to the 2nd stage than Stage1_CFS + Stage2_Mixed pair. However, this performance is achieved by scenario #4 (Stage1_CFS + Stage2_Mixed). The conclusion is that the classifier can choose the subset of mixed-traffic data on which performance like that of pure traffic can be achieved.

6. Summary and future work
This paper proposes and analyzes a Traffic Analysis (TA) framework for identifying sources of tunnelled video streaming traffic. The main contribution of this work is a two-stage classifier that combines the power of a pre-filter classifier, which filters traffic samples according to their pureness (i.e. whether the traffic is heterogeneous or pure) with a video source classifier. Using OpenVPN servers for creating encryption tunnels, extensive experiments were conducted on a large number of popular video streaming sources with various combinations of feature extraction and data processing techniques to verify the effectiveness of the two-stage classifier. The results confirm the effectiveness of the design. It was demonstrated that the system works best when an optimal feature subset was chosen using a Correlation Feature Selector (CFS), and the training data contains mixed traffic samples. An accuracy of up to 95% is achieved using this setup. Future work includes: a) studying the model of background traffic in order to assist training noise addition, b) handling multiple video streams, probably from multiple machines in addition to the background web traffic, and c) verifying the design of the classifier in a custom-built SMFW setting that closely reflects real-world scenarios. An implementation of the proposed SMFW system that is able to detect a mixture of video streams in the traffic flow is being developed and experimented upon. Future work will present it in more detail.

References