Walls Have Ears: Traffic-based Side-channel Attack in Video Streaming

Jiaxi Gu*, Jiliang Wang†, Zhiwen Yu*, Kele Shen†
*School of Computer Science, Northwestern Polytechnical University, P.R. China
†School of Software, Tsinghua University, P.R. China
gujiaxi@mail.nwpu.edu.cn, jiliangwang@tsinghua.edu.cn, zhiwenyu@nwpu.edu.cn, skl16@mail.tsinghua.edu.cn

***Abstract*—Video streaming takes up an increasing proportion of network traffic nowadays. Dynamic Adaptive Streaming over HTTP (DASH) becomes the de facto standard of video streaming and it is adopted by Youtube, Netflix, etc. Despite of the popularity, network traffic during video streaming shows identifiable pattern which brings threat to user privacy. In this paper, we propose a video identification method using network traffic while streaming. Though there is bitrate adaptation in DASH streaming, we observe that the video bitrate trend remains relatively stable because of the widely used Variable Bit-Rate (VBR) encoding. Accordingly, we design a robust video feature extraction method for eavesdropped video streaming traffic. Meanwhile, we design a VBR based video fingerprinting method for candidate video set which can be built using downloaded video files. Finally, we propose an efficient partial matching method for computing similarities between video fingerprints and streaming traces to derive video identities. We evaluate our attack method in different scenarios for various video content, segment lengths and quality levels. The experimental results show that the identification accuracy can reach up to 90% using only three-minute continuous network traffic eavesdropping.**

***Index Terms*—video streaming, network traffic, privacy, side-channel attack**

I. Introduction

Nowadays, online video streaming gets more and more popular. Cisco report [1] shows that video streaming takes up a great proportion of Internet traffic and it is also in a rapid growth. The report predicts it will take up 82% of all consumer Internet traffic by 2021. Adaptive Bitrate Streaming (ABS) based on HTTP gradually becomes the major market of video streaming due to its advantages of flexibility and infrastructure-friendly property. By splitting videos into segments of multiple quality levels (bitrates), ABS enables a smart client-driven bitrate adaptation. Dynamic Adaptive Streaming over HTTP (DASH) is a representative implementation of ABS. It has been an international standard since 2011 and widely used by leading companies of video streaming, e.g., Youtube and Netflix.

Despite of DASH's popularity, we find that its segment-based data transmission brings a risk of side-channel attack based on network traffic. Typically, a video in DASH is first encoded into multiple copies of different quality levels using Variable Bit-Rate (VBR) encoding. More specifically, different average bitrates, which indicate different quality levels, are used for each video copy, leading to different video size for different bitrates. Each video copy is then split into video segments of a fixed length of playback time. Due to video complexity variation along time, the segment size also varies along time for a video copy. Each time while streaming, a client requests a video segment in a certain quality level for playback. We find that such a mechanism in DASH results in distinct traffic pattern due to segment-based transmission and segment size variation of VBR. This can be used to identify videos while streaming, which we call side-channel video identification attack. As shown in Figure 1, by eavesdropping network traffic during video streaming, attackers can recognize certain pattern of the traffic. Meanwhile, a dataset of video fingerprints can be built using downloaded video files. Attackers can then infer what video is currently streaming by comparing the traffic pattern and video fingerprints. Such traffic-based information leakage is quite serious due to the popularity of video streaming.

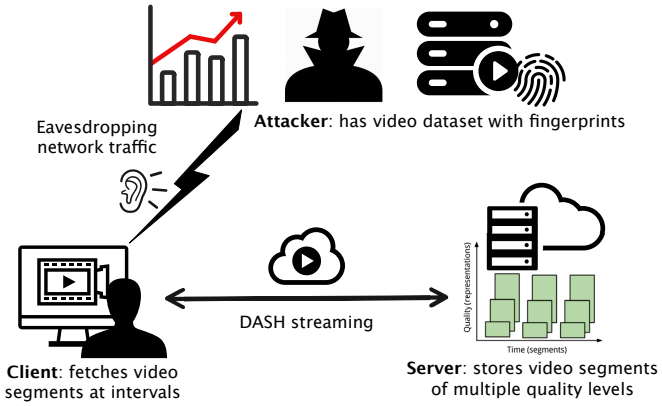


Fig. 1. DASH streaming shows distinct network traffic pattern owing to its segment-based data transmission and VBR encoding. This can be used for video identification by attackers.

Traffic-based side-channel attack has been brought into focus for years. Related literature involves various scenarios including website browsing [2], instant messaging [3], etc [4], [5]. There are also different approaches targeting video streaming [6], [7], [8]. Those approaches either need to retrieve remote metadata [6], [7] or re-stream the same video by attackers [8]. Thus, they are difficult to conduct in practical environments. Different from those methods, we aim to propose a seamless side-channel attack method for DASH video streaming. We do not require any metadata or content data from video streaming servers. Our main idea is extracting

video fingerprints merely from video files themselves, and directly computing trace pattern from network traffic of video streaming. Then we achieve seamless and efficient video identification by combining the eavesdropped traffic and video fingerprints.

The design of such a traffic-based attack method faces the following challenges in practice. First, videos are encoded into multiple quality levels in DASH while these quality levels are neither prior known nor fixed. This brings challenges for generating stable and representative video fingerprints. Second, during video streaming, quality level is adaptively selected each time according to network conditions, e.g., bandwidth. Thus traffic traces of streaming the same video exhibit uncertain patterns. Last but not least, the eavesdropped traffic may not correspond to exactly a complete video, e.g., a user only watch part of the video thus the eavesdropped traffic only contain part of the video. Even the user watches the entire video, it is time-consuming to eavesdrop the entire video traffic for video identification.

Though there are different quality levels in DASH, we find their bitrate variation trend is relatively stable for a given video. Thus we propose a differential bitrate based method to generate stable video fingerprints. For eavesdropped traffic of video streaming, we first propose a method to partition and aggregate it into segments. Then we generate effective traffic pattern by differentiating every two consecutive segments. Finally, we propose a Partial Dynamic Time Wrapping (P-DTW) method to calculate the similarity distance between the processed traffic pattern and video fingerprints. P-DTW can achieve video identification even only using a portion of traffic in video streaming.

We implement the side-channel attack method for video identification in DASH streaming. Our method requires no modification to video client and server. Our video fingerprinting process uses video files available on most video providers such as Youtube. Further, we evaluate performance of our method for different videos and the evaluation results show that identification accuracy can reach up to 90% with three-minute eavesdropping.

In summary, we have the following contributions:

1) We propose a novel attack method to derive video identity from eavesdropped traffic during video streaming without any modification to video client and server.
2) We propose a differential bitrate based feature extraction method for generating stable video features which works for practical adaptive streaming. We design an efficient partial matching method to combine video fingerprints and video stream features to derive video identity even when the eavesdropped traffic is incomplete.
3) We implement our attack method and evaluate it for real video streaming. The evaluation results show that the identification accuracy can reach up to 90% using only three-minute continuous eavesdropping.

The rest of this paper is organized as follows: Section II introduces the background of DASH and tells the motivation of our work. Section III introduces the process of video fingerprinting and pattern extraction from traffic traces in DASH. Section IV shows our method of video identification based on similarities of sequences. In Section V, we use a dataset consisting of videos of various genres and evaluate our method. Section VI introduces the related work and Section VII concludes this paper.

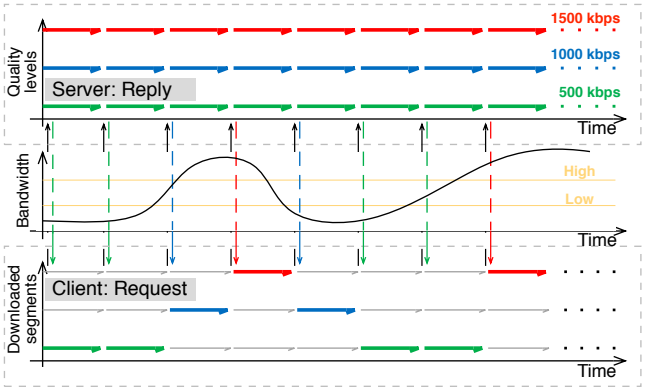


Fig. 2. In the process of DASH streaming, server is responsible for providing video segments of multiple quality levels. At intervals, client sends requests and receives video segments of specific quality levels on demand.

II. Background and Motivation

A. Basics of DASH

The whole process of DASH is described in Figure 2. On the server side, videos are pre-processed including encoding to multiple quality levels and splitting into segments. Colored lines in the figure represent available video segments of specific quality levels, i.e., 500, 1000 and 1500 kbps. Different from classic video streaming protocols, HTTP-based DASH streaming can be regarded as a series of file transmission and it follows classic request-and-reply model. Initially, video client sends a request to server for a description file which contains the metadata describing all the segments and their quality levels. After receiving and parsing the description file, video client requests for video segments on demand. Bitrate adaptation is client-driven and achieved in video segment. For example, in Figure 2, a higher quality level is chosen when available bandwidth increases otherwise a lower level is chosen.

B. VBR Encoding

On server side, videos are encoded into multiple quality levels. There are two common encoding schemes called Constant Bit-Rate (CBR) and Variable Bit-Rate (VBR). As the name suggests, CBR means that the rate at which the output data is consumed is constant. As opposed to CBR, VBR specifies an average bitrate constraint and varies the output data amount of video per time slot according to media complexity. Usually, the output of CBR has larger size than that of VBR. Considering streaming efficiency, VBR is adopted in most practical streaming services. To explain the effect of VBR encoding, we use a one-minute video clip as an example. It is encoded using H.264 and generates three

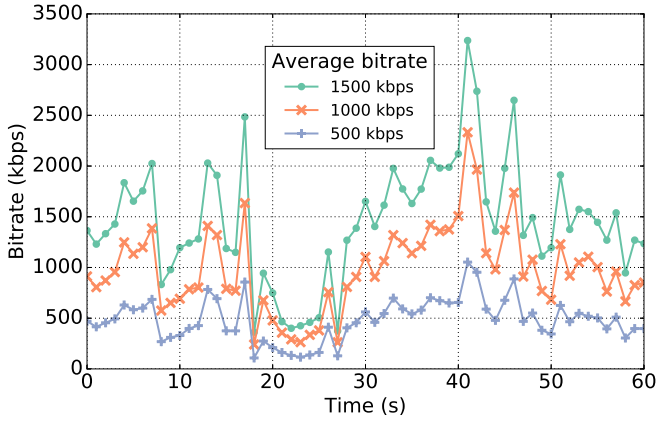


Fig. 3. Video bitrate varies though VBR specifies the quality level. For a video in different quality levels, bitrate trends show similar patterns.

different quality levels, i.e., 500, 1000 and 1500 kbps. Figure 3 shows their respective data amount per second. Although the average bitrate is fixed, the data amount per second varies as a result of VBR. Furthermore, even in different quality levels, bitrate trend follows a specific pattern which implicitly indicates the video identity. This phenomenon inspires our idea of video fingerprinting.

C. Traffic of DASH

To investigate the traffic of DASH, we separately stream three different videos named *Big Buck Bunny*, *Hockey Prodigy* and *Tears of Steal* using DASH. For consistency, all of these three videos are encoded in an average bitrate of 1000 kbps and chunked in 6-second segment. Their respective traffic traces are shown in Figure 4. As explained previously, network traffic of DASH actually indicates periodic segment downloads. Thus, traffic traces of different videos show distinct patterns as figure shows.

Further, DASH video streaming has three main characteristics which make it more vulnerable. First of all, streaming process shows traffic peaks because of its segment-based transmission. Thus, the traffic trace pattern is more distinct than continuous data transmission. Second, transmitted video segments are strictly in order. In other words, video segments arrive in a fixed sequence corresponding to the playback order. Third, different from webpages, video streaming normally has longer life cycle and there is sufficient traffic data for completing attack during one single session.

In summary: On one hand, though one video can be encoded in different quality levels, its bitrate trend is invariant due to the mechanism of VBR. On the other hand, DASH streaming follows a periodic segment-based data transmission. The resulting network traffic shows distinct patterns while streaming different videos. It is possible to identify streaming videos by eavesdropping network traffic. However, it is quite challenging to finding a good match between video bitrate trend and the traffic pattern. Moreover, due to bitrate adaptation of DASH, transmitted video segments vary in quality level which makes traffic pattern unstable.

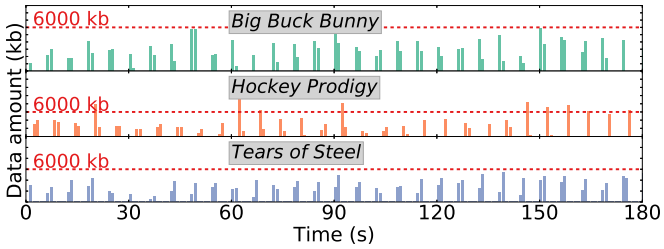


Fig. 4. DASH's segment-based transmission produces traffic peaks while streaming. The resulting traffic traces show distinct patterns while streaming different videos.

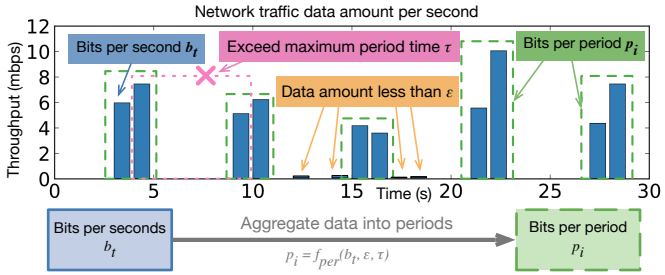


Fig. 5. Network traffic of streaming is measured in every time unit. A threshold is set to filter out too low traffic amount. Video data amount is aggregated into periods in accordance with video segments.

III. Video Fingerprints and Traffic Patterns

As DASH adopts fixed length of video segment, we fingerprint videos based on segmentation rules. Given a video of n seconds, we calculate the data amount per second and get a sequence denoted as $\boldsymbol{a} = (a_1, a_2, \ldots, a_i, \ldots)$. However, by this naive means, different quality levels result in different fingerprints. For this problem, we propose a differential-based method. For any adjacent data amount a_i and a_{i-1}, we use Equation 1 to calculate the differential between them. For consistency without loss of generality, we set $r_1 = 0$ to represent no differential at the beginning. Thus, video fingerprints can be represented by $\boldsymbol{r} = (r_1, r_2, \ldots, r_i, \ldots)$. It eliminates the influence of multiple quality levels and emphasizes the bitrate trend.

$$r_i = f_{diff}(a_i, a_{i-1}) = (a_i - a_{i-1})/a_{i-1} \tag{1}$$

We denote transmitted data amount every second as b_t at time t. Here, t is constrained to $1 \leq t \leq T$ and T is the whole eavesdropping period. In DASH, network traffic mostly consists of these two parts: client's request for new segments and server's reply with video segment data. The former as HTTP request is negligible because it is very small in comparison with the latter replied video data. In addition, DASH's specific MPD file also need to be transmitted before streaming. It also can be ignored because its amount is only several kilobytes. Thus, network traffic can be regarded as video data amount transmitted from server to client. In the

Algorithm 1: $f_{per}(b_t, \epsilon, \tau)$ aggregates network traffic.

Input : $b_t (1 \leq t \leq T)$: Throughput per second.
ϵ: Throughput threshold.
τ: Maximum time interval of one period.
Output: $\boldsymbol{p}$: Data amount per period.

```
Init(p_i) = 0 ;          // Empty each period.
i = 1 ;      // Start period index from 1.
t' = 0 ;     // Start time of each period.
for t ← 1 to T do
    // There is little data amount.
    if b(t) < ε then continue ;
    // Interval exceeds the maximum.
    if t − t' > τ then
        i ++ ; // Skip to the next period.
        t' = t ;         // Update start time.
    end
    p_i += b_t;     // Add to current period.
end
return p = (p_1, p_2, ..., p_i, ...)
```

histogram in Figure 5 for example, the traffic peaks in the figure are mostly caused by video data transmission.

In DASH, segment length refers to the playback time for player consuming the segment which is normally fixed. Video segment size does not strictly correspond to the traffic peaks because there are no segment boundaries in traffic. When network quality gets poor, transmission of one segment may result in several continuous traffic peaks. Fortunately, there are two observations we can rely on. First, the data amount of one video segment is limited by global segment length. Normally, in order to ensure a smooth video streaming experience, segment size is not too large so the data amount of one video segment does not need very long transmitting time. Second, DASH's buffer-based strategy determines that video player keeps downloading video segments until the buffer is full. During the stable video streaming process, the interval between two downloads of video segments is normally stable. Based on these two observations, we design a function $f_{per}(b_t, \epsilon, \tau)$ described in Algorithm 1 to aggregate traffic data into periods. Here, we use the term *period* to represent the time range in which one segment is completely transmitted. In this function, it takes three arguments. The first b_t is the sequential measures of traffic data per second. The second argument ϵ is responsible for denoising by filtering out very low data amount. The last τ means the maximum length of time one period covers. By clustering the measures close to each other in time, we can get a sequence of data amount in periods denoted as $\boldsymbol{p} = (p_1, p_2, \ldots, p_i, \ldots)$ returned by f_{per}. It is illustrated using green dashed boxes in Figure 5.

After traffic aggregation, the absolute time information of transmission is dropped. It is reasonable because absolute time of transmitting each video segment can be greatly influenced by network quality and adaptation strategies. Thus, it is neither meaningful nor stable as a feature for further video identification. In contrast, transmission order of segments remains invariant.

IV. Video identification

A. Video Fingerprinting

By definition in Section III, one r_i actually corresponds to one time unit, that is, one second. However, video data transmission is in segment which normally lasts several seconds. Thus, fingerprints have to be aggregated into segments. we denote segment length as a constant value L. The essential of video fingerprinting is to split the sequence $\boldsymbol{r}$ into chunks each of which contains L measures.

Based on Equation 1, we have an equivalent form of it to represent data amount a_i:

$$a_i = \prod_{j=1}^{j=i} (r_j + 1) a_1$$

Then, we sum up every L seconds of data amount as Equation 2 shows. We use s_i to represent the total data amount in i-th segment. We use $\mathcal{R}(r_i, L)$ to represent the part related to video fingerprints and segment length.

$$\begin{aligned} s_i &= a_{(i-1)L+1} + a_{(i-1)L+2} + \ldots + a_{(i-1)L+L} \\ &= \sum_{j=1}^{j=L} a_{(i-1)L+j} \\ &= \sum_{j=1}^{j=L} \prod_{k=1}^{k=(i-1)L+j} (r_k + 1)\, a_1 \\ &= \mathcal{R}(r_i, L)\, a_1 \end{aligned} \tag{2}$$

Finally, we use f_{diff} to calculate differential between segments denoted as d_i^{fp} in Equation 3. We set $d_1^{fp} = 0$ to show zero initial differential.

$$d_i^{fp} = \begin{cases} 0 & \text{if } i = 0 \\ f_{diff}(s_i, s_{i-1}) = \frac{\mathcal{R}(r_i, L) - \mathcal{R}(r_{i-1}, L)}{\mathcal{R}(r_{i-1}, L)} & \text{if } i > 1 \end{cases} \tag{3}$$

Now, given a segment length L and the bitrate trend $\boldsymbol{r}$, we can compute effective video fingerprints $\boldsymbol{d}^{fp} = (d_1^{fp}, d_2^{fp}, \ldots, d_i^{fp}, \ldots)$ accordingly.

B. Traffic Pattern Extraction

In Section III, we compute $\boldsymbol{p} = (p_1, p_2, \ldots, p_i, \ldots)$ in accordance with video segments using the measures of traffic per second. Here, p_i represents data amount transmitted in i-th period. Though, video data is transmitted in segment order according to DASH protocol. Bitrate adaptation as the most significant feature of DASH results in multiple-level traffic while transmitting video segments in different quality levels. Therefore, it is inappropriate to use the absolute data amount within each period p_i. Instead, we use differential function f_{diff} in Equation 1 to calculate the traffic differential between periods. Therefore, we have Equation 4 in which d_i^{tr} represents the i-th differential data amount. Similarly, we set $d_1^{tr} = 0$ to indicate no traffic differential at the beginning. It is an effective

and stable traffic pattern which can be further used for video identification.

$$d_i^{tr} = \begin{cases} 0 & \text{if } i = 0 \\ f_{diff}(p_i, p_{i-1}) = \frac{p_i - p_{i-1}}{p_{i-1}} & \text{if } i > 0 \end{cases} \quad (4)$$

C. Similarity Measurement

After eavesdropping network traffic while streaming for some time, we can work out its traffic pattern $\boldsymbol{d}^{tr} = (d_1^{tr}, d_2^{tr}, \ldots, d_i^{tr}, \ldots)$. Meanwhile, for each video in dataset, we calculate video fingerprints $\boldsymbol{d}^{fp} = (d_1^{fp}, d_2^{fp}, \ldots, d_i^{fp}, \ldots)$ according to Equation 3. Video identification is to find out which video is most likely to generate the traffic pattern. For this purpose, similarities between $\boldsymbol{d}^{tr}$ and each $\boldsymbol{d}^{fp}$ of different videos need to be computed and then compared. This problem can be regarded as a classic temporal series matching problem. Given a traffic pattern, the video fingerprints which has the highest similarity to it is chosen as its matched one.

Methods of series matching problem customizing for various applications have been proposed, e.g., Discrete Wavelet Transform (DWT), Discrete Fourier Transform (DFT), Dynamic Time Warping (DTW) and Symbolic Aggregate approXimation (SAX). However, in our problem, there are two important characteristics which cannot be neglected. The first one is that the number of traffic measures may be quite few when eavesdropping lasts only a short time. This makes statistics-based methods unfeasible. Second, the traffic pattern may only correspond to a portion of the whole video. Thus, partial matching has to be considered in similarity calculation.

Before measuring similarity, we need to make sequences normalized to the same scale. Z-normalization, also known as "Normalization to Zero Mean and Unit of Energy", is a well-known transformation of sequences. Given a sequence, the formulation of Z-normalization is:

$$Z(x_i) = \frac{x_i - \mu}{\sigma}$$

where μ is the mean of sequence and σ is the standard deviation. By applying Z-normalization, elements of sequences can be scaled into a fixed range. Original sequence features which help to focus on the structure are preserved. Nevertheless, it does not work for our problem because of the two characteristics we mention before. When the length of $\boldsymbol{d}^{tr}$ is much smaller than that of $\boldsymbol{d}^{fp}$, it is unreasonable to use global mean and variance on the short sequence, i.e., $\boldsymbol{d}^{tr}$. Further, it is sensitive to outliers because global statistics are used. Thus, a more robust and stable normalization scheme is needed here. Sigmoid function

$$S(x_i) = \frac{1}{1 + e^{-x_i}}$$

meets our requirements because it is independent from global statistics. Also, it preserve the trend of sequences. After applying sigmoid transformation, sequences are scaled into $(0, 1)$. Therefore, we have:

$$D_i^{fp} = S(d_i^{fp}) \in (0, 1)$$
$$D_i^{tr} = S(d_i^{tr}) \in (0, 1)$$

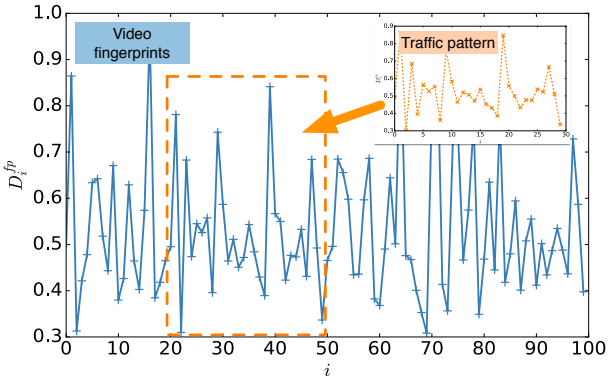
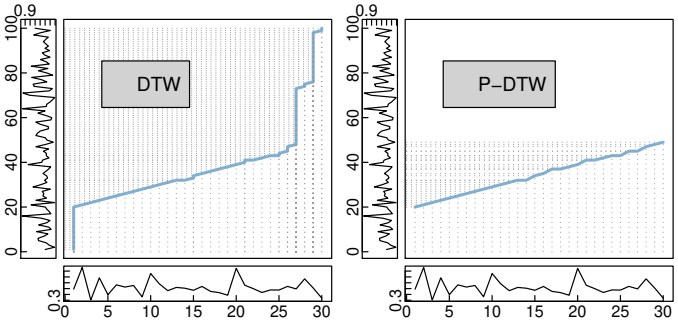


Fig. 6. Similarity between the traffic pattern and video fingerprints is computed using temporal sequence analysis. Traffic pattern may correspond to a part of whole video fingerprints because of short time of eavesdropping.



(a) Classic DTW tries to align each element of sequences and leads to unreasonable results in sub-sequence matching.

(b) P-DTW allows to relax both the start-point and end-point and makes partial matching possible.

Fig. 7. P-DTW relaxes the constraint of matching every element to support partial matching between sequences.

Figure 6 illustrates the sequences of video fingerprints and traffic pattern to be matched. They are both normalized to $(0, 1)$. We use DTW for calculating similarity. Its essential idea is aligning two sequences by warping the time axis iteractively to find the optimal match between them. However, classic DTW takes full length of sequences into calculation of cumulative cost. Thus, inspired by the open-end DTW algorithm [9], we propose a variant of DTW named P-DTW for partial matching. The difference between classic DTW and P-DTW is illustrated in Figure 7. In classic DTW whose warping curve is shown in Figure 7a, every single element of both sequences are tried to be matched. In other words, it means series heads and tails are constrained to match each other. While in Figure 7b, this constraint is relaxed for achieving partial matching. Obviously, this is more reasonable in our problem since traffic pattern correspond to a part of a video in most cases. In P-DTW, we regard the sequence of video fingerprints $\boldsymbol{D}^{fp}$ as a template sequence and traffic pattern $\boldsymbol{D}^{tr}$ is a query sequence. The principle of P-DTW is to find a proper sub-sequence on the template to minimize its distance to the query sequence. For this purpose, we use a naive method which iterates on all the sub-sequences of template. The subsequence which reaches the smallest distance

Algorithm 2: f_{P-DTW}: P-DTW for Partial matching.

Input : s^{tem}: Template sequence.
s^{que}: Query sequence.
Output: $dist$: Similarity distance between sequences.

$\mathcal{D} = \{\}$; // A set of distances.
foreach *sub-sequence* s' *in* s^{tem} **do**
  // Index starts from 1.
  $n = length(s^{que})$; $m = length(s')$;
  // Initialize a $n \times m$ matrix.
  $M = array[0:n, 0:m]$;
  $M[0,0] = 0$;
  for $i = 1$ **to** n **do** $M[i,0] = +\infty$;
  for $i = 1$ **to** m **do** $M[0,i] = +\infty$;
  for $i = 1$ **to** n **do**
    for $j = 1$ **to** m **do**
      // Euclidean distance.
      $cost = d(s^{que}[i], s'[j])$;
      // Asymmetric step pattern.
      $M[i,j] = cost + \min(M[i-1,j], M[i-1,j-1], M[i-1,j-2])$;
    end
  end
  $\mathcal{D}.append(M[n,m]/n)$;
end
return $dist = \min(\mathcal{D})$;

is chosen as the final similarity. The whole process of P-DTW is shown in Algorithm 2.

In the idea of DTW, step pattern is critical for cumulative cost computation and distance normalization as well. Various step patterns have been proposed for solving different problems. The most common step pattern is the symmetric pattern described as $\min(M[i-1,j], 2*M[i-1,j-1], M[i,j-1])$. Its corresponding normalization denominator is the sum of sequence lengths which is stable because of global alignments. While in P-DTW, the length of sub-sequence is unstable. Thus, in P-DTW, we use an asymmetric step pattern, i.e., $\min(M[i-1,j], M[i-1,j-1], M[i-1,j-2])$. By this means, we can use the length of query sequence n to get a reasonable normalization.

Given a traffic pattern $\boldsymbol{D}^{tr}$ and video fingerprints $\boldsymbol{D}^{fp}$, their similarity distance can be measured by:

$$dist = f_{P-DTW}(\boldsymbol{D}^{tr}, \boldsymbol{D}^{fp}) \in [0,1) \tag{5}$$

A smaller $dist$ indicates a higher probability for the given traffic pattern matching the fingerprints.

D. Video Identification

Given a traffic pattern and a dataset containing n videos, there are n distances generated. Empirically, a threshold of distance can be set for identifying the target video. Setting such threshold requires distances between matched pairs and unmatched pairs are distinguishable enough and keep stable to multiple variables. For evaluating this, three factors including eavesdropping time, segment length and video content are considered and the results are illustrated in Figure 8. First, we use 10-minute traffic trace of streaming a certain video in a fixed 6-second segment length. We randomly truncate sub-traces representing different eavesdropping time for calculating $dist$ to different video fingerprints. The results are shown in Figure 8a. Red boxes show the distances between traffic traces and their matched video fingerprints. These distances are significantly below those unmatched ones. Besides, as eavesdropping time varies, distances of matched pairs keep stable. Second, we keep eavesdropping network traffic for 2 minutes in different segment lengths. Figure 8b shows the results in this case. We can see that the distances of matched pairs are stable and all below the unmatched ones when segment length changes. Finally, by controlling video content, we keep eavesdropping time 2 minutes and segment length 6 seconds. Eight different video clips are streamed respectively and the results are shown in Figure 8c. Though discriminability between matched and unmatched pairs slightly vary in different videos, similarity distances mostly keep stable and discriminative. Thus, it is reasonable to set a threshold of similarity distance for identifying videos.

V. Implementation and Evaluation

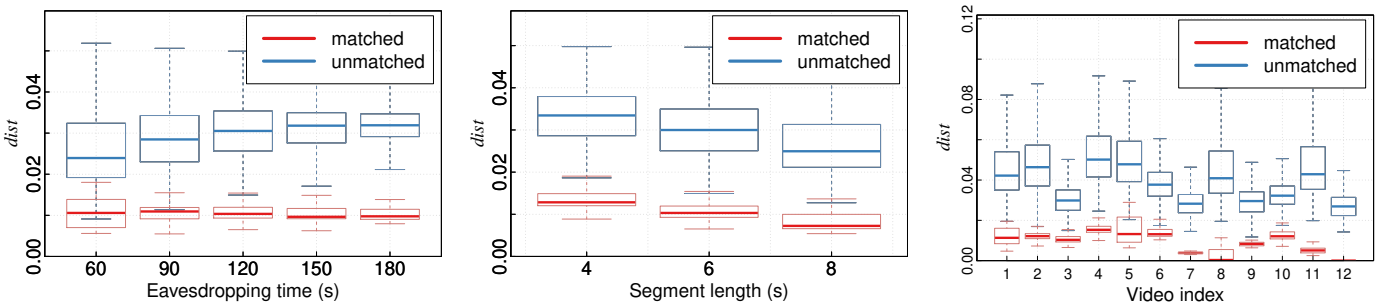
A. Experimental Settings

We implement a typical DASH workflow using FFmpeg and GPAC toolkit. FFmpeg is used for video encoding and MP4Box provided by GPAC is used for splitting videos into segments. A remote HTTP server is used for hosting video data. The client is a PC and we use Osmo4 included in GPAC as a video player. Video client is connected to a "hacked" router through ethernet. In order to simulate a case of eavesdropping network traffic, we use Ettercap which is widely used for capturing network traffic.

We collect 200 videos as a dataset and the average playback length is 726 seconds. It contains various content including animation, sports, action, etc. Each video is encoded in three different quality levels, i.e., 800, 1200 and 1600 kbps. As quality levels used by streaming servers are unknown in reality, these three levels are different from those used in DASH streaming in our experiments. Actually, video fingerprints keep stable as bitrate trend is invariant in different quality levels. To illustrate this, Figure 9 shows the normalized bitrate traces of video copies in different quality levels. They are very similar due to Equation 1 which adopts a bitrate differential instead of absolute bitrate. From our dataset, twelve videos are randomly selected for DASH streaming. Segment length varies in 4, 6 and 8 seconds. Quality levels are adapted in 500, 1000, 1500 and 2000 kbps.

B. Evaluation of Similarity Measurement

There are two parameters we have to determine according to Algorithm 1. The data amount threshold ϵ and the maximum period length τ. Normally, data amount of traffic noise such as MPD transmission and HTTP requests is far smaller than that of video segment data in orders of magnitude. Therefore,



(a) Similarity distance changes with different eavesdropping time.

(b) Similarity distance changes with different segment lengths.

(c) Similarity distance changes with different video content.

Fig. 8. Similarity distance $dist$ keeps stable for measuring similarity between the traffic pattern and video fingerprints.

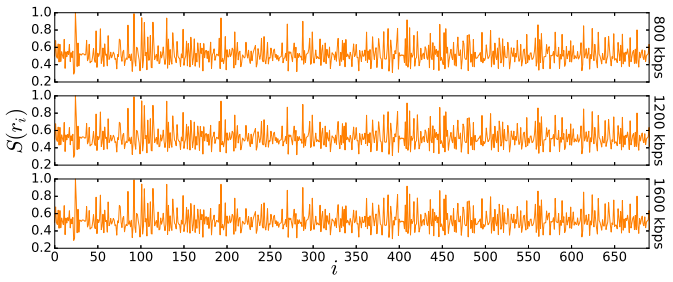
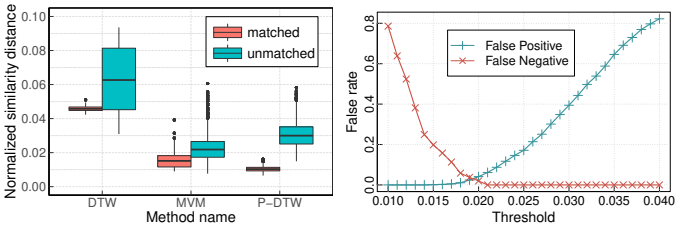


Fig. 9. This example is generated by encoding a certain raw video in four different quality levels. These four copies have very similar bitrate trend.



(a) P-DTW shows more stability and discriminability comparing with other two methods.

(b) Threshold of P-DTW distance for video identification is determined by minimizing false rates.

Fig. 10. P-DTW is more effective than other methods and a threshold of distance is further computed for video identification.

we use a small $\epsilon = 20$ kilobytes in our experiments. As for τ, it is directly related to segment length and adaptation strategy. While in stable streaming, time interval between transmissions of video segments is around one segment length. Thus, τ is theoretically between 0 to segment length L and we set $\tau = L/2$ in our experiments.

For different series matching methods including Minimal Variance Matching (MVM) [10] which allows arbitrary skips of elements in template sequence, we evaluate their performance. We use traffic data generated with 2-minute eavesdropping and calculate its similarity distances with video fingerprints. The results are shown in Figure 10a. Classic DTW unsurprisingly has poor performance because of its global alignments. MVM brings overall low distances but the discriminability is still below that of P-DTW. Then, we use P-DTW for computing similarity threshold for video identification. We use 1000 traffic traces in different conditions varying in video bitrates, segment lengths and eavesdropping time. Figure 10b shows false rates using different thresholds. We find that 0.019 is an appropriate threshold for minimizing identification false rates. In the following evaluation of our method, we define a uniform "accuracy" based on this threshold as follows. Given an eavesdropped traffic trace, we randomly truncate one thousand sub-sequences of a specific length which is to be evaluated on. Then we use these sub-sequences for actual performance evaluation. If and only if the similarity distance between the traffic pattern and its matched video fingerprints is below the threshold, it counts as a true test. So, the accuracy is defined by the ratio of number of true tests to that of total tests, i.e., one thousand.

C. Single-bitrate Streaming

To begin with, we consider single-bitrate streaming. In our experiments, two variables including video quality level and segment length are controlled respectively. Video quality level varies in 500, 1000, 1500 and 2000 kbps. Segment length varies in 4, 6 and 8 seconds. For a given quality level and segment length, traffic is eavesdropped for around 10 minutes in each DASH session while streaming twelve videos from our dataset. We evaluate the identification accuracy when eavesdropping time varies. Figure 11 shows the results in different cases. We find that identification accuracy is greatly influenced by segment length while video bitrate has little impact. It can be intuitively explained with the number of traffic measures. When eavesdropping time is fixed, smaller segment length leads to more traffic measures which reveal more clues for identification.

D. Multiple-bitrate Adaptive Streaming

In order to investigate how bitrate adaptation influences video identification accuracy, we design experiments of adaptive streaming. Video bitrate is adaptive in 500, 1000, 1500, 2000 kbps. Using Osmo4, we can manually select video quality level for video segments while streaming. In this experiment, bitrate adaptation is triggered in every one minute. Traffic sub-traces of certain length are extracted by randomly truncating from the complete trace accordingly. We compute accuracy upper bound by manually removing the related measure in traffic pattern. The results are shown in Figure 12. The actual identification accuracy without manual interference is very close to the upper bound.

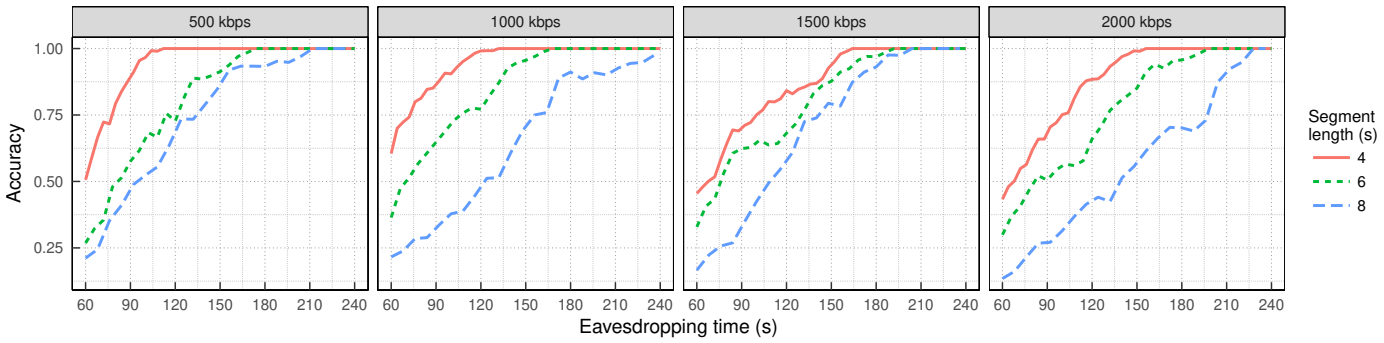


Fig. 11. Identification accuracy with different eavesdropping time in different cases.

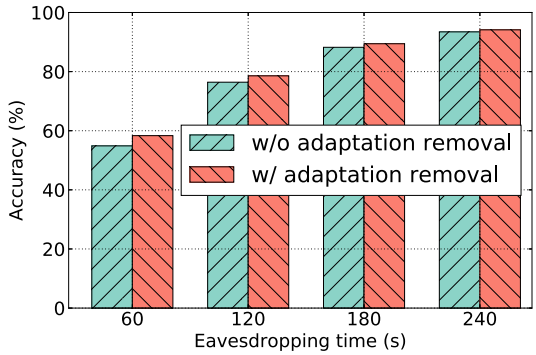


Fig. 12. The influence of extra error caused by bitrate adaptation is limited and it can be decreased with eavesdropping time getting longer.

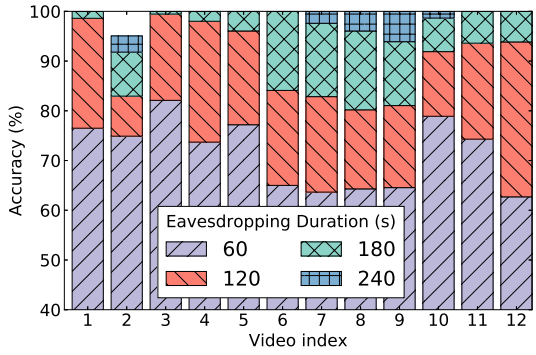


Fig. 13. Identification accuracy of streaming different videos.

Finally, we evaluate the identification performance on different video content. Twelve videos are separately streamed in adaptive bitrate with segment length of 6 seconds. Figure 13 shows the final results. The accuracy varies a little when eavesdropping time is low and it gets up to 90% when the eavesdropping time reaches 180 seconds.

VI. Related Work

Information leakage in network traffic: Side-channel attack using network traffic has raised widely concern as a serious threat to user privacy. Shuo Chen et al. show the severity and universality as such attack is based on fundamental characteristics of Internet despite encryption [11]. It is ubiquitous in comparison with other features such as hardware features [12]. Utilizing network traffic, multiple attacks covering different purposes are proposed. Skype as a service closely related to privacy is a typical case. Several works are presented for analyzing network traffic of Skype [13], [3] to detect user actions. In addition to Skype, websites also can be recognized using network traffic analysis [14], [15], [2]. User activities [16], [17], contextual localization [4] and demographics [5] are likely to be revealed in network traffic. If network traffic can be extended to general package-based communication, it will lead to critical privacy leakage, e.g., location-based applications [18], [19], [20], [21].

Network traffic eavesdropping: As network traffic contains implicit user information, traffic eavesdropping gets its popularity in computer security. Organizations such as enterprise network center or ISPs can directly monitor network traffic. In addition, local adversaries can use Wi-Fi sniffers to eavesdrop wireless traffic [16]. Routers may be hacked by attackers in the same LAN and used for eavesdropping [22]. Some public access points are probability unsafe [23]. Even, remote attackers can use network congestion to indirectly monitor the traffic pattern of victim's machine [24]. Network delay can be utilized to achieve attacks too [25]. There is also literature [26] introducing side-channel attacks by remotely sending probes and observing round trip time.

Video fingerprinting and identifying: Saponas et al. propose an early attempt of identifying videos using network traffic [27]. Aiming at video streaming in Slingbox, the authors show potential information leakage caused by VBR. Different from DASH, traffic traces in this work are directly processed into segments of 100 milliseconds. A windowed DFT is performed on the trace to achieve video identification by matching against reference traces. Also based on VBR, Yali et al. extract short and long range dependencies within video traffic to construct video signatures [28]. Reed et al. study Youtube video streaming in [6], [7]. They take advantage of DASH and VBR for fingerprinting videos on Netflix by parsing the video metadata. Another work related to ours is from Schuster et al [8]. They use network traffic bursty pattern for identifying videos but the method requires video re-streaming in the same network with victim. Besides, video fingerprinting method may inspire the identification of other things such as pedestrians [29] in the future.

VII. Conclusion

Traffic-based attack in video streaming is a big threat to user privacy. In this paper, we propose a seamless and efficient attack method by eavesdropping network traffic while streaming. Relying on the invariant of video bitrate trend caused by VBR encoding, we design a robust video fingerprinting method. In various conditions, our identification accuracy can get up to 90% using 3-minute traffic traces. We plan to conduct our method on a larger dataset and explore its performance on online video services such as Youtube and Netflix. Meanwhile, as segment length has critical influence in our algorithm, an automatic detection method of video segment length or even streaming protocol is on our agenda. On the other hand, in face of such information leakage, countermeasures considering both network efficiency and streaming Quality of Experience (QoE) are also worth further studying.

Acknowledgment

This work was supported by the National Science Fund for Distinguished Young Scholars (No. 61725205), the National Basic Research Program of China (No. 2015CB352400), the National Natural Science Foundation of China Key Project (No. 61632013) and the National Natural Science Foundation of China (No. 61332005, 61772428, 61722210, 61532012).

References

[1] C. V. networking Index, "Forecast and methodology, 2016-2021, white paper," *San Jose, CA, USA*, vol. 1, 2016.

[2] T. Wang, X. Cai, R. Nithyanand, R. Johnson, and I. Goldberg, "Effective attacks and provable defenses for website fingerprinting," in *Proceedings of the 23rd USENIX Conference on Security Symposium*, ser. SEC'14. Berkeley, CA, USA: USENIX Association, 2014, pp. 143–157.

[3] W. Wang and D. N. Cheng, "Skype traffic identification based on trends-aware protocol fingerprints," in *Vehicle, Mechatronics and Information Technologies II*, ser. Applied Mechanics and Materials, vol. 543. Trans Tech Publications, Jun. 2014, pp. 2249–2254.

[4] A. K. Das, P. H. Pathak, C. N. Chuah, and P. Mohapatra, "Contextual localization through network traffic analysis," in *IEEE INFOCOM 2014 - IEEE Conference on Computer Communications*, Apr. 2014, pp. 925–933.

[5] H. Li, Z. Xu, H. Zhu, D. Ma, S. Li, and K. Xing, "Demographics inference through wi-fi network traffic analysis," in *IEEE INFOCOM 2016 - The 35th Annual IEEE International Conference on Computer Communications*, Apr. 2016, pp. 1–9.

[6] A. Reed and B. Klimkowski, "Leaky streams: Identifying variable bitrate dash videos streamed over encrypted 802.11n connections," in *2016 13th IEEE Annual Consumer Communications Networking Conference (CCNC)*, Jan. 2016, pp. 1107–1112.

[7] A. Reed and M. Kranch, "Identifying https-protected netflix videos in real-time," in *Proceedings of the Seventh ACM on Conference on Data and Application Security and Privacy*, ser. CODASPY '17. New York, NY, USA: ACM, 2017, pp. 361–368.

[8] R. Schuster, V. Shmatikov, and E. Tromer, "Beauty and the burst: Remote identification of encrypted video streams," in *26th USENIX Security Symposium (USENIX Security 17)*. Vancouver, BC: USENIX Association, 2017, pp. 1357–1374.

[9] P. Tormene, T. Giorgino, S. Quaglini, and M. Stefanelli, "Matching incomplete time series with dynamic time warping: an algorithm and an application to post-stroke rehabilitation," *Artificial intelligence in medicine*, vol. 45, no. 1, pp. 11–34, 2009.

[10] L. J. Latecki, V. Megalooikonomou, Q. Wang, and D. Yu, "An elastic partial shape matching technique," *Pattern Recogn.*, vol. 40, no. 11, pp. 3069–3080, Nov. 2007.

[11] S. Chen, R. Wang, X. Wang, and K. Zhang, "Side-channel leaks in web applications: A reality today, a challenge tomorrow," in *Proceedings of the 2010 IEEE Symposium on Security and Privacy*, ser. SP '10. Washington, DC, USA: IEEE Computer Society, 2010, pp. 191–206.

[12] J. Han, C. Qian, P. Yang, D. Ma, Z. Jiang, W. Xi, and J. Zhao, "Geneprint: Generic and accurate physical-layer identification for uhf rfid tags," *IEEE/ACM Trans. Netw.*, vol. 24, no. 2, pp. 846–858, Apr. 2016.

[13] M. Korczyski and A. Duda, "Classifying service flows in the encrypted skype traffic," in *2012 IEEE International Conference on Communications (ICC)*, Jun. 2012, pp. 1064–1068.

[14] X. Gong, N. Kiyavash, and N. Borisov, "Fingerprinting websites using remote traffic analysis," in *Proceedings of the 17th ACM Conference on Computer and Communications Security*, ser. CCS '10. New York, NY, USA: ACM, 2010, pp. 684–686.

[15] X. Cai, X. C. Zhang, B. Joshi, and R. Johnson, "Touching from a distance: Website fingerprinting attacks and defenses," in *Proceedings of the 2012 ACM Conference on Computer and Communications Security*, ser. CCS '12. New York, NY, USA: ACM, 2012, pp. 605–616.

[16] F. Zhang, W. He, X. Liu, and P. G. Bridges, "Inferring users' online activities through traffic analysis," in *Proceedings of the Fourth ACM Conference on Wireless Network Security*, ser. WiSec '11. New York, NY, USA: ACM, 2011, pp. 59–70.

[17] 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*, vol. 11, no. 1, pp. 114–125, Jan. 2016.

[18] Z. Zhou, Z. Yang, C. Wu, W. Sun, and Y. Liu, "Lifi: Line-of-sight identification with wifi," in *IEEE INFOCOM 2014 - IEEE Conference on Computer Communications*, Apr. 2014, pp. 2688–2696.

[19] X. Chen, X. Wu, X. Y. Li, X. Ji, Y. He, and Y. Liu, "Privacy-aware high-quality map generation with participatory sensing," *IEEE Transactions on Mobile Computing*, vol. 15, no. 3, pp. 719–732, Mar. 2016.

[20] Y. Guo, L. Yang, B. Li, T. Liu, and Y. Liu, "Rollcaller: User-friendly indoor navigation system using human-item spatial relation," in *IEEE INFOCOM 2014 - IEEE Conference on Computer Communications*, Apr. 2014, pp. 2840–2848.

[21] Q. Ma, S. Zhang, T. Zhu, K. Liu, L. Zhang, W. He, and Y. Liu, "Plp: Protecting location privacy against correlation analyze attack in crowdsensing," *IEEE Transactions on Mobile Computing*, vol. 16, no. 9, pp. 2588–2598, Sep. 2017.

[22] M. Conti, N. Dragoni, and V. Lesyk, "A survey of man in the middle attacks," *IEEE Communications Surveys Tutorials*, vol. 18, no. 3, pp. 2027–2051, 2016.

[23] N. Cheng, X. O. Wang, W. Cheng, P. Mohapatra, and A. Seneviratne, "Characterizing privacy leakage of public wifi networks for users on travel," in *2013 Proceedings IEEE INFOCOM*, Apr. 2013, pp. 2769–2777.

[24] T. Ristenpart, E. Tromer, H. Shacham, and S. Savage, "Hey, you, get off of my cloud: Exploring information leakage in third-party compute clouds," in *Proceedings of the 16th ACM Conference on Computer and Communications Security*, ser. CCS '09. New York, NY, USA: ACM, 2009, pp. 199–212.

[25] Z. Ling, J. Luo, Y. Zhang, M. Yang, X. Fu, and W. Yu, "A novel network delay based side-channel attack: Modeling and defense," in *2012 Proceedings IEEE INFOCOM*, Mar. 2012, pp. 2390–2398.

[26] S. Kadloor, X. Gong, N. Kiyavash, T. Tezcan, and N. Borisov, "Low-cost side channel remote traffic analysis attack in packet networks," in *2010 IEEE International Conference on Communications*, May 2010, pp. 1–5.

[27] 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 on USENIX Security Symposium*, ser. SS'07. Berkeley, CA, USA: USENIX Association, 2007, pp. 5:1–5:16.

[28] Y. Liu, A.-R. Sadeghi, D. Ghosal, and B. Mukherjee, "Video streaming forensic - content identification with traffic snooping," in *Proceedings of the 13th International Conference on Information Security*, ser. ISC'10. Berlin, Heidelberg: Springer-Verlag, 2011, pp. 129–135.

[29] Y. Jiang, Z. Li, and J. Wang, "Ptrack: Enhancing the applicability of pedestrian tracking with wearables," in *2017 IEEE 37th International Conference on Distributed Computing Systems (ICDCS)*, Jun. 2017, pp. 2193–2199.