Traffic-Based Side-Channel Attack in Video Streaming

Jiaxi Gu, Jiliang Wang, *Member, IEEE*, Zhiwen Yu, *Senior Member, IEEE*, and Kele Shen

***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, and so on. Despite of the popularity, network traffic during video streaming shows an 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. Compared with traditional video streaming protocols such as Real Time Messaging Protocol (RTMP) and Real Time Streaming Protocol (RTSP), ABS not only has better compatibility with current infrastructures but also provides mature bitrate adaptation strategy. There are various different implementations including HTTP Live Streaming (HLS) from Apple, HTTP Dynamic Streaming (HDS) from Adobe and Microsoft Smooth Streaming (MSS). 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.

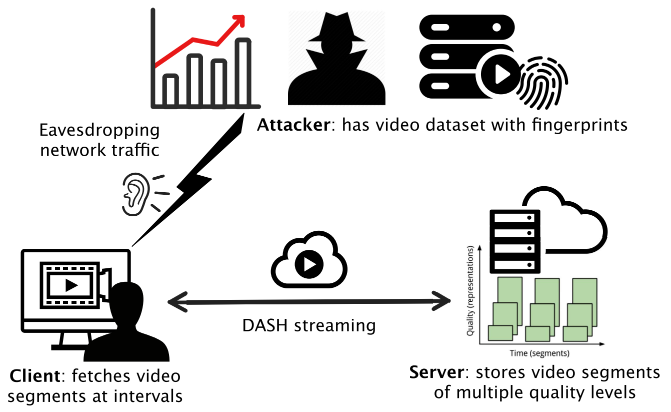

Eavesdropping network traffic
Attacker: has video dataset with fingerprints
DASH streaming
Quality (representations)
Time (segments)
Client: fetches video segments at intervals
Server: stores video segments of multiple quality levels

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.

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

Manuscript received January 23, 2018; revised October 31, 2018; accepted March 10, 2019; approved by IEEE/ACM TRANSACTIONS ON NETWORKING Editor Y. Chen. Date of publication April 11, 2019; date of current version June 14, 2019. This work was supported in part by the National Key R&D Program of China under Grant 2018YFB1004800, in part by the National Science Fund for Distinguished Young Scholars under Grant 61725205, in part by the National Basic Research Program of China under Grant 2015CB352400, in part by the National Natural Science Foundation of China Key Project under Grant 61632013, and in part by the National Natural Science Foundation of China under Grant 61332005, Grant 61772428, Grant 61722210, and Grant 61532012. *(Corresponding author: Jiaxi Gu.)*

J. Gu and Z. Yu are with the School of Computer Science, Northwestern Polytechnical University, Xi'an 710129, China (e-mail: imjiaxi@gmail.com).

J. Wang is with the School of Software, Tsinghua University, Beijing 100084, China.

K. Shen is with the Alibaba Group, Hangzhou 311121, China.

Digital Object Identifier 10.1109/TNET.2019.2906568

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. On the one hand, such traffic-based information leakage is quite serious due to the popularity of video streaming. On the other hand, it is also quite critical with the surging use of encryption and other evasion techniques. Using network traffic information, Internet service provider (ISP) and enterprise managers can spy on users indirectly. Moreover, the profit of video streaming platforms may be damaged if some sensitive information such as video popularity chart is revealed by unauthorized organizations. For users watching online videos, their sensitive video watching history may be used as forensics evidence to conduct blackmail. In addition, video clips of advertisement can also be identified and analyzed using such side-channel attack. Then, users' various types of privacy such as contextual localization and demographic information may be invaded. Also, in this way, understanding human behavior [2], [3] can be possible.

Traffic-based side-channel attack has been brought into focus for years. Related literature involves various scenarios including website browsing [4], instant messaging [5], etc [6], [7]. There are also different approaches targeting video streaming [8]–[10]. Those approaches either need to retrieve remote metadata [8], [9] or re-stream the same video by attackers [10]. 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 or server. Our video fingerprinting process uses video files available on most video providers such as Youtube. Furthermore, 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. In Section III, we investigate some feasible approaches about eavesdropping of network traffic especially from a perspective of adversaries. Section IV introduces the process of video fingerprinting and pattern extraction from traffic traces in DASH. Section V shows our method of video identification based on similarities of sequences. In Section VI, we use a dataset containing videos of various genres and evaluate our method on this dataset. In Section VII, we discuss the defense and countermeasure against such attack proposed in this paper. Section VIII introduces the related work and Section IX concludes this paper.

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 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.

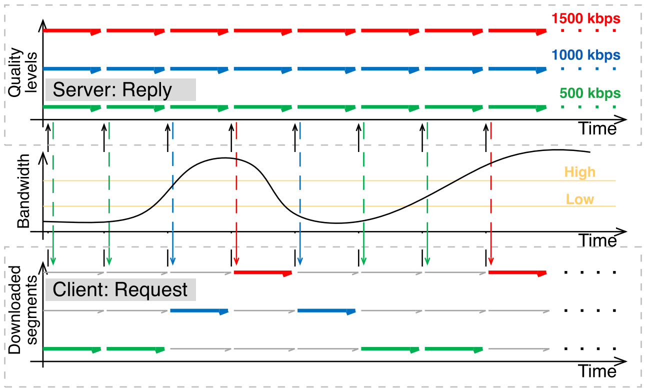


Fig. 2. In the process of DASH streaming, server is responsible for providing video segments of multiple quality levels. At intervals, client sends requests and receives video segments of specific quality levels on demand.

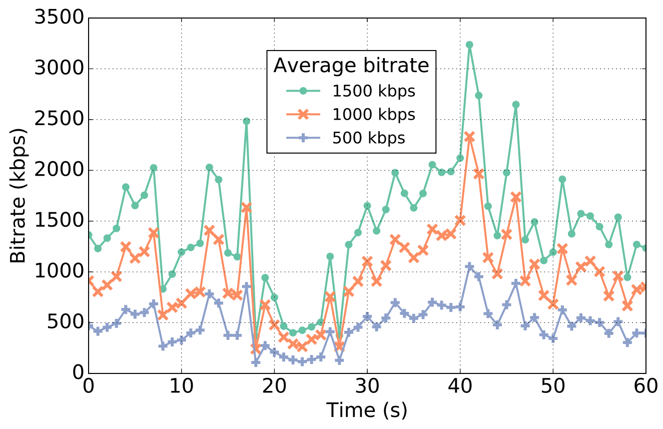


Fig. 3. Video bitrate varies though VBR specifies the quality level. For a video in different quality levels, bitrate trends show similar patterns.

B. VBR Encoding

On the 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 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.

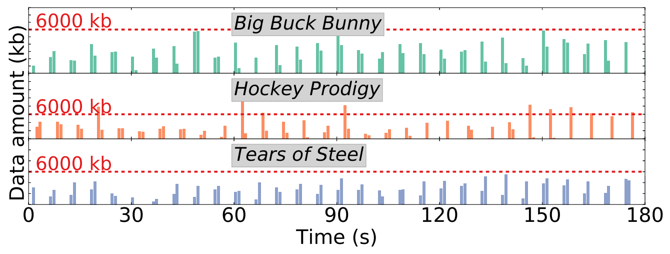


Fig. 4. DASH's segment-based transmission produces traffic peaks while streaming. The resulting traffic traces show distinct patterns while streaming different videos.

Furthermore, 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 the 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.

III. Network Traffic Eavesdropping

Acquiring real-time network traffic while video streaming is a crucial part of our video identification method. In the case of video streaming, network traffic refers to downstream throughput in particular. Network traffic eavesdropping can be conducted in several different phases of data transmission. What is more serious is that traffic eavesdropping can be hardly avoided because of the packet-based data transmission mechanism in modern network structure.

A. Compromised Endpoint Devices

Attacks or eavesdropping aiming at the clients can be seen as the most direct and harmful. Not only the network traffic but also the content of the data transmission may be leaked if the endpoint devices are compromised. Concretely, this kind of attack can be done by malicious software, unauthorized web browser plugin, third-party network proxy, etc. In these cases, though data content can be encrypted with strong encryption methods, real-time network traffic can be effortlessly obtained by attackers.

B. Malicious Network Infrastructures

Because of the multilevel structure of modern Internet, network infrastructures are inevitable in data transmission. Network traffic eavesdropping can be done in various different network devices including routers, switches, etc. Once these network infrastructures are compromised, the transmitted data passing through them can be captured and network traffic can then be calculated. Furthermore, there are also some fake base stations or man-in-the-middle attacks which can seriously leak data transmission including network traffic. Overall, unsafe network environment containing malicious network infrastructures brings probabilities of network traffic leakage.

C. Wireless Sniffer Attacks

In addition to attacks aiming at physical devices, information leakage also occurs in network connections. Wireless sniffer is a typical attack of this type. A wireless sniffer is able to collect and read transmitted data. This type of attack is very hard to detect if the sniffer itself does not send any data. Even though the content of transmitted data may be encrypted, the network traffic can still be inferred.

D. Side-Channel Eavesdropping

Other than direct attacks on network connections and endpoints, there are also some remote attacks using side-channel information of network. A representative example of such attack is utilizing the network congestion by sending probes [11]. Adversaries can send probes remotely and observe queuing delays in routers which reflect current network congestion. By this means, network traffic information can be deduced.

IV. Video Fingerprints and Traffic Patterns

In order to identify videos according to traffic traces of streaming, a database of video identifiers has to be established. We call this process video fingerprinting. Firstly, video identification essentially aims at distinguishing different videos so every identifier needs to be representative enough. Secondly, a video may be encoded in different bitrate levels. It requires to be stable, that is, insensitive to such variations. Lastly, network traffic while video streaming is affected by network conditions, device varieties, etc. Video identifiers should be reliable enough in these cases. In general, we need a video fingerprinting method which is *representative*, *stable* and *reliable*.

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 assume that there are no other processes consuming sizable bandwidth such as downloading large files. It is reasonable as users tend to quit such processes for ensuring a smooth video playback. Some extreme cases, such as very low bandwidth and high packet loss rate, are not in our consideration since users will normally leave video streaming immediately because of bad experience under such circumstances. In addition, continual jumps on the playing progress or frequent switches of video qualities are not considered. These conditions themselves are abnormal and they can not ensure a persistent normal playback.

In general, we have following assumptions:

1) Except for DASH video streaming, there are no other processes consuming large bandwidth at the same time.
2) Network condition meets the requirements for smooth video streaming.
3) A continuous normal video streaming lasts a long enough period of time.
4) There are no abnormal user interferences during video streaming.

In network data transmission, network throughput refers to the amount of data successfully transmitted from one side to another in a given period of time. It is typically measured in bits per second (bps). In this paper, we assume that the raw data an attacker can get is measured by bps, that is, amount of data transmitted every second.

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 is negligible because its amount is very small in comparison with the latter replied video data. In addition, a Media Presentation Description (MPD) document containing metadata of the video needs to be transmitted at the begging of DASH. Its amount which is only several kilobytes also can be ignored. Thus, network traffic can be regarded as video data amount transmitted from server to client. In the 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

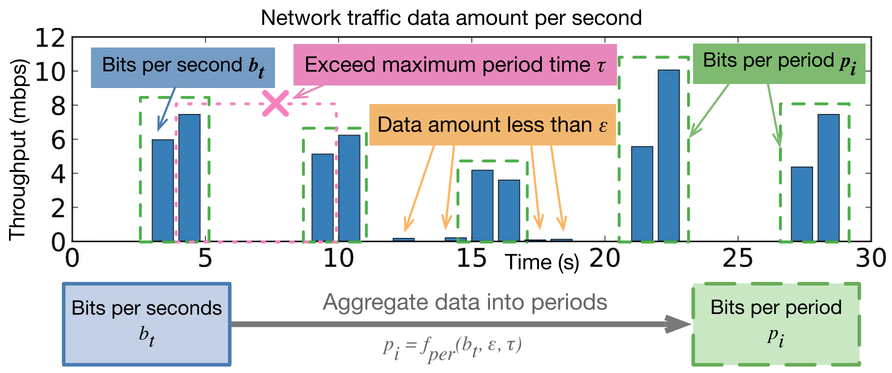


Fig. 5. Network traffic of streaming is measured in every time unit. A threshold is set to filter out too low traffic amount. Video data amount is aggregated into periods in accordance with video segments.

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 normal video streaming process, the interval between two downloads of video segments is normally fixed. 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.

Algorithm 1 $f_{per}(b_t, \epsilon, \tau)$ Aggregates Network Traffic

```
Input  : $b_t (1 \le t \le T)$: Throughput per second.
         $\epsilon$: Throughput threshold.
         $\tau$: 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 $t \leftarrow 1$ to $T$ do
    // There is little data amount.
    if $b(t) < \epsilon$ then continue ;
    // Interval exceeds the maximum.
    if $t - t' > \tau$ then
        $i{+}{+}$ ;    // Skip to the next period.
        $t' = t$ ;           // Update start time.
    end
    $p_i \mathrel{+}= b_t$;       // Add to current period.
end
return $\boldsymbol{p} = (p_1, p_2, \ldots, p_i, \ldots)$
```

V. Video Identification

The overall process of video identification is divided into three steps. The first step is video fingerprinting in the dataset to generate unique fingerprint sequence $\boldsymbol{r}$ for each video. The second step starts with eavesdropping network traffic during streaming. Then, aggregated traffic trace $\boldsymbol{p}$ is computed according to Section IV. In the last step, we try to compute the similarity between video fingerprints and traffic pattern for video identification.

However, there are actually some problems to be solved in these three steps. Firstly, video data is transmitted in segment each of which covers a certain length of video while fingerprints are sampled in second. Thus, we need to achieve segment alignment in the video fingerprints for further identification. Secondly, since bitrate adaptation is a critical feature of DASH, video segments may be of different bitrates in different periods of streaming session. Although such bitrate adaptation is not that frequent, there are still some bitrate adaptation points in timeline which affect the trace pattern of throughput measured per second. Last problem is about calculating similarity between eavesdropped traffic traces and video fingerprints. Network traffic eavesdropping is unlikely to begin from exactly the start of the video. The recorded traffic traces normally correspond to a subset of the streaming video. We need an appropriate method for measuring their similarity.

A. Video Fingerprinting

Briefly, Figure 6 illustrates the process of generating video fingerprints on the video playback timeline. By definition in Section IV, 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

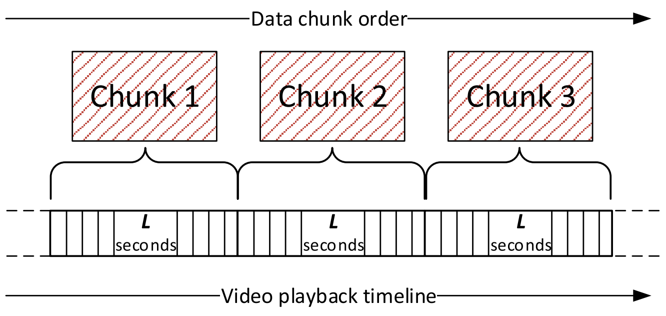


Fig. 6. The size of data chunk is mapped on video playback timeline according to the video segment length.

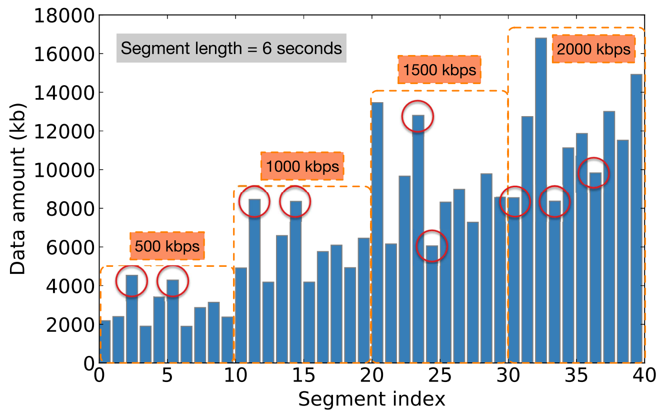


Fig. 7. Bitrate adaptation points are hard to accurately detect using the sequence of segment data amount because of drastic bitrate fluctuations.

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 IV, 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.

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

Theoretically, video bitrate can be arbitrarily adjusted at any segment boundary. d_i^{tr} gets unreliable in such bitrate adaptation points because video segments in adjacent periods are encoded in different bitrate levels. In order to remove the error caused by such bitrate adaptation, the adaptation moments have to be detected very accurately since the traffic trace is described by increment of adjacent segment data amount. To illustrate this issue, we use a segment data amount sequence of adaptive streaming. Segment length is fixed to six seconds and video encoding bitrate varies between 500 kbps, 1000 kbps, 1500 kbps and 2000 kbps. Bitrate level is manually changed at specific interval and the moments are recorded as baseline. The data amount change of such segment sequence is shown in Figure 7. Segments of the same encoding bitrate level are marked in orange dashed rectangles. We can find that segment data amount varies even with the same encoding bitrate. Encoding bitrate only specifies the average bitrate and there are some segments marked with red circle in the figure whose data amount is far larger or smaller than the average value. Furthermore, adjacent bitrate levels for adaptation may have very small gap in average bitrate. As a result, it is very hard to accurately detect bitrate level changes. In practice, however, it is not a good idea for Quality of Experience (QoE) if bitrate level is changed too frequently. Thus, bitrate adaptations are actually occasional in a single DASH streaming session. The error caused by bitrate adaptation has little impact as long as the total eavesdropping time is long enough. This will be evaluated in Section VI.

C. Similarity Measurement

After eavesdropping network traffic while streaming for a period of time, we can extract 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) [12]. 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., $\boldsymbol{d}^{tr}$. Furthermore, 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:

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

Figure 8 illustrates the sequences of video fingerprints and traffic pattern to be matched. They are both normalized to $(0, 1)$. We use DTW [13], a widely used algorithm for calculating similarity [14]. 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 [15], we propose

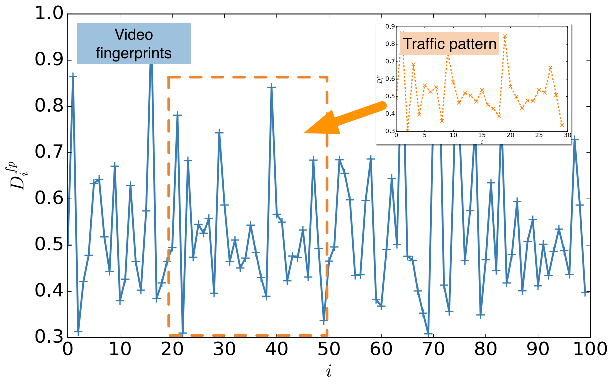


Fig. 8. Similarity between the traffic pattern and video fingerprints is computed using temporal sequence analysis. Traffic pattern may correspond to a part of whole video fingerprints because of short time of eavesdropping.

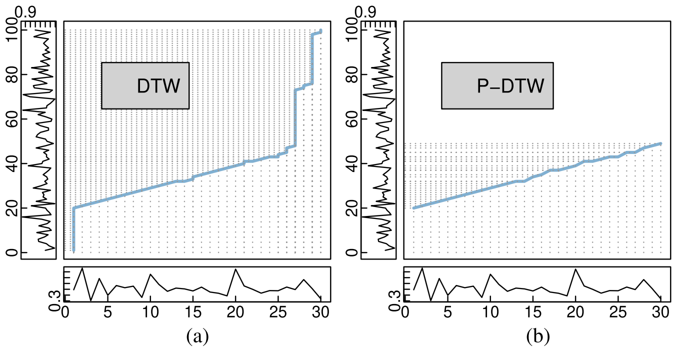


Fig. 9. P-DTW relaxes the constraint of matching every element to support partial matching between sequences. (a) Classic DTW tries to align each element of sequences and leads to unreasonable results in sub-sequence matching. (b) P-DTW allows to relax both the start-point and end-point and makes partial matching possible.

a variant of DTW named P-DTW for partial matching. The difference between classic DTW and P-DTW is illustrated in Figure 9. In classic DTW whose warping curve is shown in Figure 9a, 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 9b, 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 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.

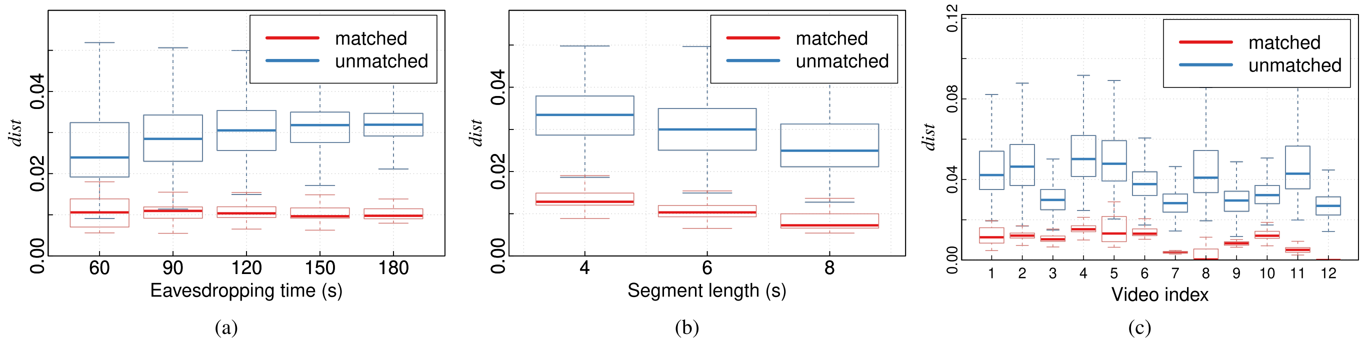


Fig. 10. Similarity distance *dist* keeps stable for measuring similarity between the traffic pattern and video fingerprints. (a) Similarity distance changes with different eavesdropping time. (b) Similarity distance changes with different segment lengths. (c) Similarity distance changes with different video content.

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 $dist = \min(\mathcal{D})$;
```

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:

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

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 10. 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 10a. 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 10b 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. Twelve different video clips are streamed respectively and the results are shown in Figure 10c. 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.

VI. Implementation and Evaluation

A. Experimental Setup

We implement a typical DASH workflow using FFmpeg [16] and GPAC [17] toolkit. FFmpeg is used for video encoding and MP4Box provided by GPAC is used for splitting videos into segments. A remote 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.

The summary of our video dataset for fingerprinting and streaming is shown in Table I. The original dataset for fingerprinting contains 200 videos in total. These videos vary in genres including animation, sports, action, etc. From the original dataset, we randomly select twelve videos for streaming

TABLE I

SUMMARY OF THE VIDEO DATASET

Videos for fingerprinting	
Number of videos	200
Average playback time	726 (s)
Bitrate levels	800, 1200, 1600 (kbps)
Time used for fingerprinting	220 (s)
Storage size of fingerprints	1.872 (MB)
Videos for streaming	
Number of videos	12
Adaptive bitrate levels	500, 1000, 1500, 2000 (kbps)
Segment lengths	4, 6, 8 (s)

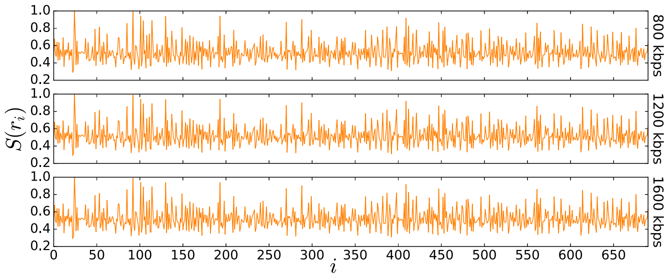


Fig. 11. This example is generated by encoding a certain raw video in three different quality levels. These three copies have very similar bitrate trend.

and further identification. In comparison with the bitrate levels for fingerprinting, the bitrate levels for adaptive streaming are deliberately different as the video information of streaming is unknown in reality. Through this setup, we can evaluate our method in a more realistic environment since our method does not require any prior knowledge about the streaming videos.

B. Video Fingerprinting

Since video quality levels used by video streaming servers are unknown in realistic environment, video encoding bitrate is a noteworthy factor in video fingerprinting. Fortunately, video bitrate differential guarantee the extracted video fingerprints from different encoding bitrate levels show high similarity. To illustrate this, Figure 11 shows the normalized bitrate traces of video copies in different quality levels. To prove this more formally, we calculate the Pearson correlation coefficients between video copies of different encoding bitrate levels. The results are shown in Figure 12. The average Pearson correlation coefficient is around 0.99 so it is reasonable to think that encoding bitrate levels have very little difference in extracting video fingerprints. Finally, Figure 13 shows the extracted fingerprints of the twelve videos for streaming. As we can see, they show observable difference in shape.

C. 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

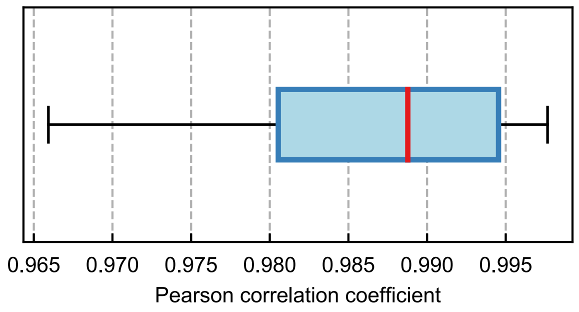


Fig. 12. The Pearson correlation coefficient between video fingerprints extracted from video copies of different encoding bitrate levels. The range of the coefficient is between -1 and 1 where -1 indicates total negative linear correlation, 1 indicates total positive linear correlation and 0 indicates no linear correlation.

that of video segment data in orders of magnitude. Therefore, we use a small $\epsilon = 20$ kilobytes in our experiments. In addition to ϵ, the parameter τ plays a role of distinguishing continuous video segments in network traffic. In the process of stable video streaming, the interval between two continuous traffic peaks is around one segment length, that is L. Thus, the parameter τ can theoretically be any value from 0 to L. Empirically, we set $\tau = L/2$ in our experiments but it is not the only choice.

For different series matching methods including Minimal Variance Matching (MVM) [18] 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 14a. 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 14b 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.

D. Various Parameter Combinations

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

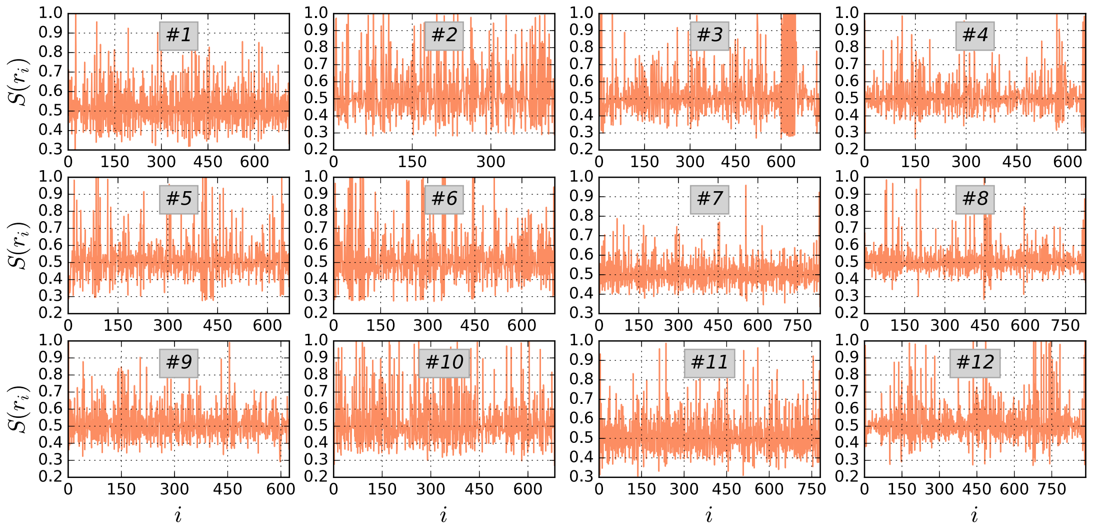
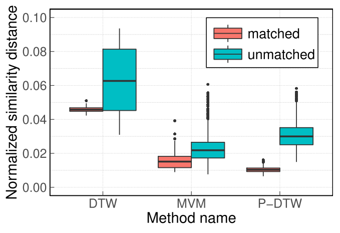
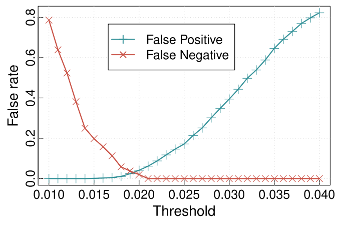


Fig. 13. Twelve videos covering various genres are used as a dataset of our experiments. These are their normalized identifiers represented by $S(r_i)$.



(a)



(b)

Fig. 14. P-DTW is more effective than other methods and a threshold of distance is further computed for video identification. (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.

TABLE II
SIMULATING NETWORK CONDITIONS

Network condition	Description
Wired connection	The client node is connected to the video server directly through wired network.
Wireless connection in office area	A Wi-Fi AP is placed in a corner of a $12 \times 10 \times 3.5$ (m) bullpen office, and the client node is placed in the diagonal corner of the area.
Wireless connection in living area	A Wi-Fi AP is placed in a corner of a $12 \times 10 \times 3.5$ (m) apartment which is divided into 4 rooms by concrete walls with windows, and the client node is placed in the diagonal corner of the area.

our dataset. We evaluate the identification accuracy when eavesdropping time varies. Figure 15 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.

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 16. The actual identification accuracy without manual interference is very close to the upper bound.

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 17 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.

E. Simulations in Various Conditions

In addition to the evaluations above with actual video streaming, we also do extensive simulations of video streaming in different conditions, hoping to achieve a comprehensive evaluation. Generally, the goal of our simulation is to evaluate our method in various network conditions and DASH implementations. The simulation platform we use is a discrete-event network simulator called ns-3 (ver. 3.29) [19] which is very widely used in network-related researches. It can simulate various network typologies, protocols and very fine-grained network configurations. Even the physical environment such as buildings and node mobility potentially affecting network performance can be simulated too. More importantly, multiple DASH implementations can be integrated in ns-3 for simulation so we can evaluate the performance of our method using different implementations of DASH.

Concretely, in our simulation, we create a video server providing DASH streaming of 200 videos, each of which has three video quality levels. Meanwhile, we initialize client nodes as

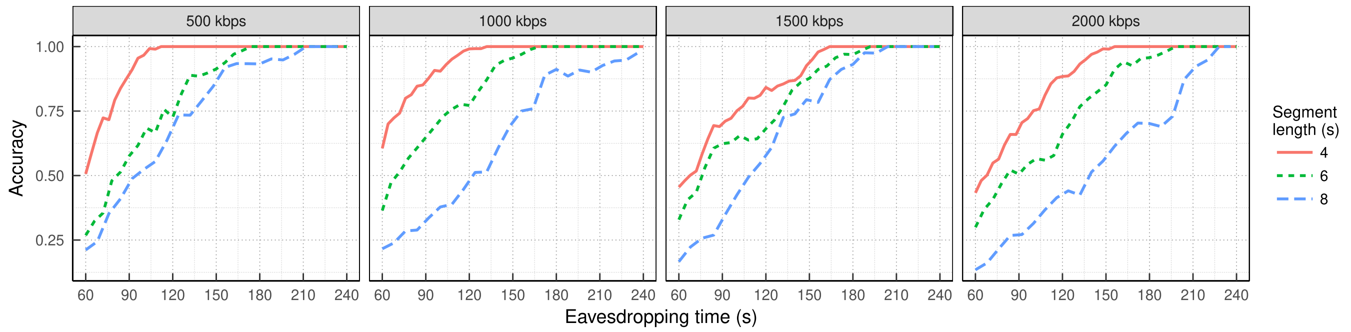


Fig. 15. Identification accuracy with different eavesdropping time in different cases. There are three main variables including eavesdropping time, video segment length and video quality level.

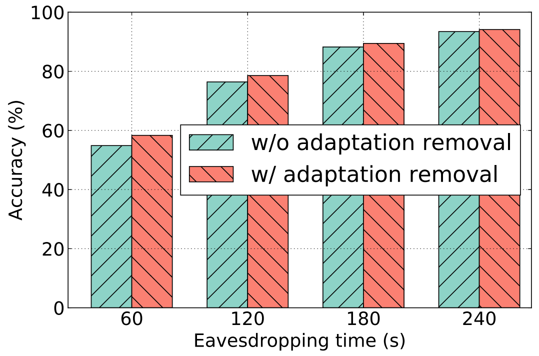


Fig. 16. The influence of extra error caused by bitrate adaptation is limited and it can be decreased with eavesdropping time getting longer.

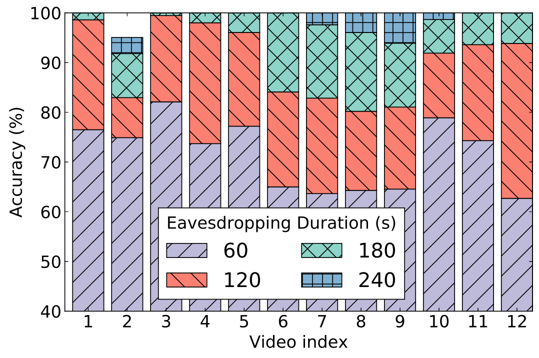


Fig. 17. Identification accuracy of streaming different videos.

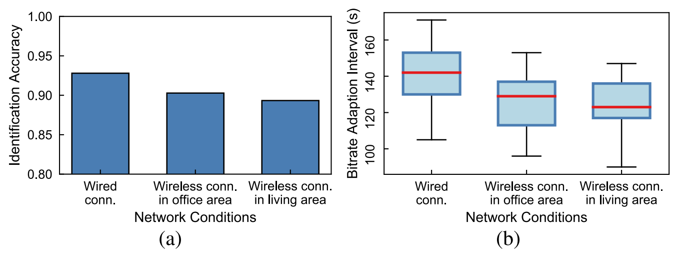


Fig. 18. Simulations with 200 videos are run in different network conditions. (a) Identification accuracy in different network conditions. (b) Bitrate adaptation intervals in different network conditions.

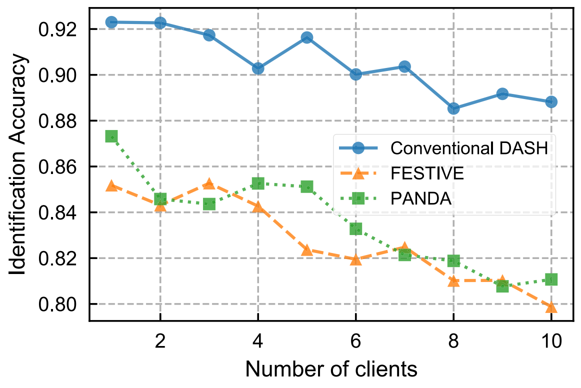


Fig. 19. The video identification performance is evaluated using multiple different DASH implementations and different number of simultaneous video clients.

DASH players requesting video segments from the server. The video segment length is kept 6 seconds and the video quality level varies in 500, 1000, 1500 and 2000 kbps. Eventually, each client finishes streaming the total 200 videos one by one. For network traffic eavesdropping, on the client side, we repeat eavesdropping for a random three minutes 100 times during each client streaming each video. Therefore, in one time of simulation in each client, we generate $200 \times 100 = 20,000$ network traffic samples to identify.

As video bitrate adaptation in DASH is largely affected by network status, simulations of video streaming in different network conditions are necessary. For this purpose, we simulate three typical cases including wired connection, wireless connection in office area and wireless connection in living area. The specific information about our simulation of these three cases is shown in Table II. The data-rate of transmission is set to 100 Mbps and the specific physical-layer standard of Wi-Fi is 802.11n. The final results of video identification in these three network conditions are shown in Figure 18a. We find that video identification accuracy is relatively high in directly wired connection while the results in Wi-Fi connections are also near 90%. Moreover, for evaluating video adaptation frequency in different network conditions, the statistics of bitrate adaptation intervals are shown in Figure 18b.

Since video adaptation strategy plays an important role in DASH and traffic-based video identification as well, multiple different DASH implementations are worth evaluation. We consider three DASH implementations including conventional DASH, FESTIVE [20] and PANDA [21]. Besides, considering video streaming strategies adopted by different DASH implementations, instead of only one client at a time, we use different number of clients simultaneously requesting video segments. The results are shown in Figure 19. In comparison with the conventional DASH implementation, the accuracy in the customized DASH implementations shows different degrees of decline. The reasons are that these customized DASH implementations adopt different video segment transmission strategies for either streaming efficiency or stability. In these cases, the distinct traffic pattern of DASH is interfered. Meanwhile, when the number of connected clients increases, the identification accuracy is also influenced.

VII. Defense and Countermeasures

The generalized traffic-based privacy attack is really hard to detect and defense because such attack stems from packet-based data transmission which is a basic mechanism of modern Internet. The scenarios of video streaming however rely on some specific protocols including VBR and DASH. Thus, countermeasures can be designed against them. The common idea of the countermeasures is adding randomness or disorder to break the recognizable patterns of network traffic. Meanwhile, a practical countermeasure has to be compatible with standard video streaming protocols and requires low overhead. For example, though transmitting video segments in random order or attaching redundant data while streaming theoretically works well as defense. These practices are not good countermeasures as they either rely on a modified streaming protocol or require considerable overhead.

A. Variable Segment Length

As the traffic-based privacy attack in DASH essentially depends on the measured sizes of transmitted video segments, segment length plays an important role in such attack. The segment length in real scenarios is carefully selected considering latency and encoding efficiency. It is even fixed in a specific video streaming platform. In order to design a countermeasure by variable segment length, videos can be segmented with different segment lengths at the acceptable expense of QoE. On the one hand, adversaries can hardly do video identification using the network traffic generated with irregular segment length. On the other hand, video players can still stream the video because the segment length information is included in the video metadata which is requested at the very beginning of the video streaming session.

B. Dynamic Buffer Capacity

Another countermeasure for traffic-based attack is to disrupt the stable process of video segment transmission. The video buffer capacity on the client side is crucial for retrieving new video segments. Video streaming as well as network traffic pattern of DASH keeps stable only if the buffer is as full as possible. The buffer capacity is commonly fixed considering the trade-off between smooth playback and bandwidth utilization. In order to defense against the traffic-based attack, one can use a customized scheme for dynamically adjusting the buffer capacity. For example, one can randomly enable a standby buffer to break the full-buffer state. In this way, long-term stable data transmission can be avoided while video playback can still meet the requirement of QoE.

C. Continuous Bulk Transmission

Since video identification of DASH roots in the orderly and individual transmission of video segments, a countermeasure can be designed by breaking this rule to disrupt the extracted traffic pattern. One realistic practice is to transmit several continuous video segments simultaneously instead of transmitting only one segment at a time. By this means, the eavesdropped network traffic peaks cannot accurately reflect the sizes of each video segments so video identification using this information will fail.

VIII. Related Work

Information Leakage in Network Traffic: Side-channel attack using network traffic has raised wide 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 [22]. It is ubiquitous in comparison with other features such as hardware features [23]. 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 [5], [24] to detect user behavior. In addition to Skype, websites also can be recognized using network traffic analysis [4], [25], [26]. User activities [27], [28], [42], contextual localization [6] and demographics [7] are likely to be revealed in network traffic. In general package-based communication, there is also critical privacy leakage, e.g., location-based applications [29]–[32], [43].

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 [27]. Routers may be hacked by attackers in the same LAN and used for eavesdropping [33]. Some public access points are probability unsafe [34]. Even, remote attackers can use network congestion to indirectly monitor the traffic pattern of victim's machine [35]. Network delay can be utilized to achieve attacks too [36]. There is also literature [37] introducing side-channel attacks by remotely sending probes and observing round trip time.

Video Fingerprinting and Identifying: T. Scott Saponas et al. propose an early attempt of identifying videos using network traffic [38]. 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 Liu et al. extract short and long range dependencies within video traffic to construct video signatures [39]. Andrew Reed et al. study Youtube video streaming in [8], [9]. They take advantage of DASH and VBR for fingerprinting videos on Netflix by parsing the video metadata. Another work related to ours is from Roei Schuster *et al.* [10]. 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 [40] in the future.

Similarity Measurement Between Sequences: As a vital problem of time series analysis, similarity measurement yields insights in many domains, e.g., speech recognition, medical diagnosis and anomaly detection. It contains two parts including data pre-processing and distance calculating. Data pre-processing mainly involves smoothing, standardization and normalization in order to make sequences reasonably comparable. This step is also responsible for computing representations of data. For some specific applications, data can be projected onto a basis using methods such as Principal Component Analysis (PCA), Singular Value Decomposition (SVD), Fast Fourier Transformation (FFT), etc. In addition, SAX [12] which uses symbolic representation is proven to be an efficient approach for sequence matching. Then, on calculating similarity between data representations, SAX uses its own distance method called MINDIST [41]. DTW is a very popular method for sequence matching. Its variants which differ in input feature space, step pattern, local constraints and so on make it quite flexible. While classic DTW requires that each element of sequence is aligned and is not able to skip any elements, it exposes weakness in partial sequence matching. MVM [18] is proposed for implementing an elastic partial matching which allows for arbitrary number of elements to be skipped. It finds the non-contiguous parts of one sequence which best match the other sequence.

IX. 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 schedule. Moreover, in face of such information leakage, countermeasures considering both network efficiency and streaming QoE are also worth further studying.

References

[1] Cisco Visual Networking Index, "Forecast and methodology, 2016–2021," Cisco, San Jose, CA, USA, Tech. Rep., 2016. [Online]. Available: https://www.cisco.com/c/en/us/solutions/collateral/service-provider/global-cloud-index-gci/white-paper-c11-738085.html

[2] Z. Yu, F. Yi, Q. Lv, and B. Guo, "Identifying on-site users for social events: Mobility, content, and social relationship," *IEEE Trans. Mobile Comput.*, vol. 17, no. 9, pp. 2055–2068, Sep. 2018.

[3] Z. Yu, H. Du, F. Yi, Z. Wang, and B. Guo, "Ten scientific problems in human behavior understanding," *CCF Transactions on Pervasive Computing and Interaction*. Singapore: Springer, 2019. doi: 10.1007/s42486-018-00003-w.

[4] T. Wang, X. Cai, R. Nithyanand, R. Johnson, and I. Goldberg, "Effective attacks and provable defenses for website fingerprinting," in *Proc. 23rd USENIX Security Symp.*, San Diego, CA, USA: USENIX Association, 2014, pp. 143–157.

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

[6] A. K. Das, P. H. Pathak, C.-N. Chuah, and P. Mohapatra, "Contextual localization through network traffic analysis," in *Proc. IEEE Conf. Comput. Commun.*, Toronto, ON, Canada, 2014, pp. 925–933.

[7] H. Li *et al.*, "Demographics inference through Wi-Fi network traffic analysis," in *Proc. 35th Annu. IEEE Int. Conf. Comput. Commun.*, San Francisco, CA, USA, Apr. 2016, pp. 1–9.

[8] A. Reed and B. Klimkowski, "Leaky streams: Identifying variable bitrate DASH videos streamed over encrypted 802.11n connections," in *Proc. 13th IEEE Annu. Consum. Commun. Netw. Conf.*, Las Vegas, NV, USA, Jan. 2016, pp. 1107–1112.

[9] A. Reed and M. Kranch, "Identifying HTTPS-protected Netflix videos in real-time," in *Proc. 7th ACM Conf. Data Appl. Secur. Privacy*, Scottsdale, AZ, USA, 2017, pp. 361–368

[10] R. Schuster, V. Shmatikov, and E. Tromer, "Beauty and the burst: Remote identification of encrypted video streams," in *Proc. 26th USENIX Secur. Symp.*, Vancouver, BC, Canada, 2017, pp. 1357–1374.

[11] X. Gong, N. Borisov, N. Kiyavash, and N. Schear, "Website detection using remote traffic analysis," in *Proc. 12th Int. Symp. Privacy Enhancing Technol.*, Vigo, Spain, 2012, pp. 58–78.

[12] J. Lin, E. Keogh, S. Lonardi, and B. Chiu, "A symbolic representation of time series, with implications for streaming algorithms," in *Proc. 8th ACM SIGMOD Workshop Res. Issues Data Mining Knowl. Discovery*, San Diego, CA, USA, 2003, pp. 2–11.

[13] D. J. Berndt and J. Clifford, "Using dynamic time warping to find patterns in time series," in *Proc. AAAI Workshop Knowl. Discovery Databases*, Seattle, WA, USA, 1994, pp. 359–370.

[14] H. Du *et al.*, "Recognition of group mobility level and group structure with mobile devices," *IEEE Trans. Mobile Comput.*, vol. 17, no. 4, pp. 884–897, Apr. 2018.

[15] 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," *Artif. Intell. Med.*, vol. 45, no. 1, pp. 11–34, 2009.

[16] S. Tomar, "Converting video formats with FFmpeg," *Linux J.*, vol. 2006, no. 146, p. 10, 2006.

[17] J. Le Feuvre, C. Concolato, and J. Moissinac, "GPAC: Open source multimedia framework," in *Proc. 15th Int. Conf. Multimedia*, Augsburg, Germany, 2007, pp. 1009–1012.

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

[19] G. F. Riley and T. R. Henderson, "The ns-3 network simulator," in *Modeling and Tools for Network Simulation*. Berlin, Germany: Springer, 2010, pp. 15–34.

[20] J. Jiang, V. Sekar, and H. Zhang, "Improving fairness, efficiency, and stability in HTTP-based adaptive video streaming with festive," *IEEE/ACM Trans. Netw.*, vol. 22, no. 1, pp. 326–340, 2014.

[21] Z. Li *et al.*, "Probe and adapt: Rate adaptation for HTTP video streaming at scale," *IEEE J. Sel. Areas Commun.*, vol. 32, no. 4, pp. 719–733, Apr. 2014.

[22] S. Chen, R. Wang, X. Wang, and K. Zhang, "Side-channel leaks in Web applications: A reality today, a challenge tomorrow," in *Proc. 31st IEEE Symp. Secur. Privacy*, Berkeley/Oakland, CA, USA, May 2010, pp. 191–206.

[23] J. Han *et al.*, "GenePrint: Generic and accurate physical-layer identification for UHF RFID tags," *IEEE/ACM Trans. Netw.*, vol. 24, no. 2, pp. 846–858, Apr. 2016.

[24] M. Korczyński and A. Duda, "Classifying service flows in the encrypted skype traffic," in *Proc. IEEE Int. Conf. Commun.*, Ottawa, ON, Canada, Jun. 2012, pp. 1064–1068.

[25] X. Gong, N. Kiyavash, and N. Borisov, "Fingerprinting Websites using remote traffic analysis," in *Proc. 17th ACM Conf. Comput. Commun. Secur.*, Chicago, IL, USA, 2010, pp. 684–686.

[26] X. Cai, X. C. Zhang, B. Joshi, and R. Johnson, "Touching from a distance: Website fingerprinting attacks and defenses," in *Proc. ACM Conf. Comput. Commun. Secur.*, Raleigh, NC, USA, 2012, pp. 605–616.

[27] F. Zhang, W. He, X. Liu, and P. G. Bridges, "Inferring users' online activities through traffic analysis," in *Proc. 4th ACM Conf. Wireless Netw. Secur.*, Hamburg, Germany, 2011, pp. 59–70.

[28] M. Conti, L. V. Mancini, R. Spolaor, and N. V. Verde, "Analyzing Android encrypted network traffic to identify user actions," *IEEE Trans. Inf. Forensics Security*, vol. 11, no. 1, pp. 114–125, Jan. 2016.

[29] Z. Zhou, Z. Yang, C. Wu, W. Sun, and Y. Liu, "LiFi: Line-of-sight identification with WiFi," in *Proc. IEEE Conf. Comput. Commun.*, Toronto, ON, Canada, Apr./May 2014, pp. 2688–2696.

[30] X. Chen *et al.*, "Privacy-aware high-quality map generation with participatory sensing," *IEEE Trans. Mobile Comput.*, vol. 15, no. 3, pp. 719–732, Mar. 2016.

[31] Y. Guo, L. Yang, B. Li, T. Liu, and Y. Liu, "RollCaller: User-friendly indoor navigation system using human-item spatial relation," in *Proc. IEEE Int. Conf. Comput. Commun.*, Toronto, ON, Canada, Apr./May 2014, pp. 2840–2848.

[32] Q. Ma *et al.*, "PLP: Protecting location privacy against correlation analyze attack in crowdsensing," *IEEE Trans. Mobile Comput.*, vol. 16, no. 9, pp. 2588–2598, Sep. 2017.

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

[34] N. Cheng, X. O. Wang, W. Cheng, P. Mohapatra, and A. Seneviratne, "Characterizing privacy leakage of public WiFi networks for users on travel," in *Proc. IEEE Conf. Comput. Commun.*, Turin, Italy, Apr. 2013, pp. 2769–2777.

[35] 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 *Proc. ACM Conf. Comput. Commun. Secur.*, Chicago, IL, USA, 2009, pp. 199–212.

[36] Z. Ling *et al.*, "A novel network delay based side-channel attack: Modeling and defense," in *Proc. IEEE Conf. Comput. Commun.*, Orlando, FL, USA, Mar. 2012, pp. 2390–2398.

[37] S. Kadloor, X. Gong, N. Kiyavash, T. Tezcan, and N. Borisov, "Low-cost side channel remote traffic analysis attack in packet networks," in *Proc. IEEE Int. Conf. Commun.*, Cape Town, South Africa, May 2010, pp. 1–5.

[38] T. S. Saponas, J. Lester, C. Hartung, S. Agarwal, and T. Kohno, "Devices that tell on you: Privacy trends in consumer ubiquitous computing," in *Proc. 16th USENIX Secur. Symp.*, Boston, MA, USA, 2007, pp. 55–70.

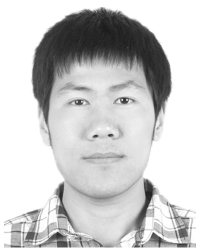
[39] Y. Liu, A.-R. Sadeghi, D. Ghosal, and B. Mukherjee, "Video streaming forensic—Content identification with traffic snooping," in *Proc. 13th Int. Conf. Inf. Secur.*, Boca Raton, FL, USA, 2010, pp. 129–135.

[40] Y. Jiang, Z. Li, and J. Wang, "PTrack: Enhancing the applicability of pedestrian tracking with wearables," in *Proc. 37th Int. Conf. Distrib. Comput. Syst.*, Atlanta, GA, USA, 2017, pp. 2193–2199.

[41] J. Lin, E. Keogh, L. Wei, and S. Lonardi, "Experiencing SAX: A novel symbolic representation of time series," *Data Mining Knowl. Discovery*, vol. 15, no. 2, pp. 107–144, 2007.

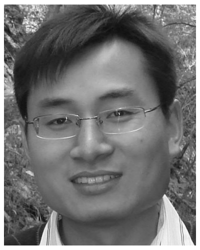
[42] M. Yang, X. Gu, Z. Ling, C. Yin, and J. Luo, "Anactivede-anonymizing attack against tor Web traffic," *Tsinghua Sci. Technol.*, vol. 22, no. 6, pp. 702–713, 2017.

[43] J. Chen, K. He, Q. Yuan, M. Chen, R. Du, and Y. Xiang, "Blindfiltering at third parties: An efficient privacy-preserving framework for location- based services," *IEEE Trans. Mobile Comput.*, vol. 17, no. 11, pp. 2524–2535, 2018.

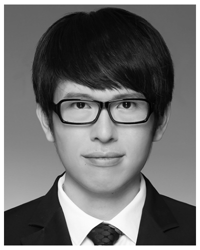


Jiaxi Gu received the B.E. degree in computer science and technology from Northwestern Polytechnical University, China, where he is currently pursuing the Ph.D. degree with the School of Computer Science. His research interests include mobile computing and privacy-aware video streaming.

Jiliang Wang (M'12) received the B.E. degree in computer science and technology from the University of Science and Technology of China and the Ph.D. degree in computer science and engineering from The Hong Kong University of Science and Technology. He is currently an Associate Professor with the School of Software and TNList, Tsinghua University, China. His research interests include wireless and sensor networks, Internet of Things, and mobile computing.



Zhiwen Yu (SM'13) received the Ph.D. degree from Northwestern Polytechnical University, China. He was a Research Fellow with Kyoto University, Japan, from 2007 to 2009, and an Alexander Von Humboldt Fellow with Mannheim University, Germany, from 2009 to 2010. He is currently a Professor and the Vice-Dean with the School of Computer Science, Northwestern Polytechnical University. His research interests include pervasive computing and human–computer interaction.



Kele Shen received the Ph.D. degree in computer science from the School of Computer Science and Technology, Tsinghua University, China, in 2016. He finished his post-doctoral position at the School of Software, Tsinghua University, in 2018. He is currently an Algorithm Expert with Alibaba Group. His research interests fall in the areas of hardware security, mobile computing security, and deep learning.