

# StreamingTag: A Scalable Piracy Tracking Solution for Mobile Streaming Services

Fan Dang , Member, IEEE, Xinqi Jin , Qi-An Fu , Lingkun Li , Guanyan Peng , Xinlei Chen , Member, IEEE, Kebin Liu , Senior Member, IEEE, and Yunhao Liu , Fellow, IEEE

**Abstract**—Streaming services have billions of mobile subscribers, yet video piracy has cost service providers billions. Digital Rights Management (DRM), however, is still far from satisfactory. Unlike DRM, which attempts to prohibit the creation of pirated copies, fingerprinting may be used to track out the source of piracy. Nevertheless, existing fingerprinting-based streaming systems are not widely used since they fail to serve numerous users. In this paper, we present the design and evaluation of StreamingTag, a scalable piracy tracing system for mobile streaming services. StreamingTag adopts a segment-level fingerprint embedding scheme to remove the need of re-embedding the fingerprint into the video for each new viewer. The key innovations of StreamingTag include a scalable and CDN-friendly delivery framework, an accurate and lightweight temporal synchronization scheme, a polarized and randomized SVD watermarking scheme, and a collusion-resistant fingerprinting scheme. Experiment results show the good QoS of StreamingTag in terms of preparation latency, bandwidth consumption, and video fidelity. Compared with existing methods, the proposed three schemes improve the re-identification accuracy by 4-49x, the watermark extraction accuracy by 2.25x at most and 1.5x on average, and the recall rate of catching colluders by 26%.

Manuscript received 14 May 2023; revised 4 April 2024; accepted 13 August 2024. Date of publication 19 August 2024; date of current version 5 November 2024. This work was supported in part by the National Key R&D Program of China under Grant 2021YFB2900100, in part by the NSFC under Grant 62302259 and Grant 62371269, in part by the Talent Fund of Beijing Jiaotong University under Grant 2022XKRC013, in part by Guangdong Innovative and Entrepreneurial Research Team Program under Grant 2021ZT09L197, and Meituan. Recommended for acceptance by S. Uluagac. (*Fan Dang and Xinqi Jin contributed equally to this work.*) (*Corresponding author: Yunhao Liu.*)

Fan Dang is with the Global Innovation Exchange, Tsinghua University, Beijing 100084, China (e-mail: dangfan@tsinghua.edu.cn).

Xinqi Jin and Guanyan Peng are with the School of Software, Tsinghua University, Beijing 100084, China (e-mail: jinxq21@mails.tsinghua.edu.cn; pgy20@mails.tsinghua.edu.cn).

Qi-An Fu is with the Department of Computer Science and Technology, Tsinghua University, Beijing 100084, China (e-mail: fqa19@mails.tsinghua.edu.cn).

Lingkun Li is with the School of Software, Beijing Jiaotong University, Beijing 100044, China (e-mail: lkli@bjtu.edu.cn).

Xinlei Chen is with the Shenzhen International Graduate School, Tsinghua University, Guangdong 518057, China, also with the Pengcheng Laboratory, Shenzhen 518066, China, and also with the RISC-V International Open Source Laboratory, Shenzhen 518055, China (e-mail: chen.xinlei@sz.tsinghua.edu.cn).

Kebin Liu is with the Fuzhou Fuyao Institute for Advanced Study, Fuzhou 350109, China, and also with Global Innovation Exchange, Tsinghua University, Beijing 100084, China (e-mail: kebinliu2021@tsinghua.edu.cn).

Yunhao Liu is with the Department of Automation & The Global Innovation Exchange, Tsinghua University, Beijing 100084, China (e-mail: yunhao@tsinghua.edu.cn).

Our implementation is at <https://streamingtag.github.io>.

This article has supplementary downloadable material available at <https://doi.org/10.1109/TMC.2024.3445411>, provided by the authors.

Digital Object Identifier 10.1109/TMC.2024.3445411

**Index Terms**—Copyright protection, video streaming, piracy tracing, dynamic time warping, video watermarking, collusion.

## I. INTRODUCTION

**S**TREAMING is regarded as one of the most essential mobile applications. Mobile users are reported to spend dozens of hours on it every month on average [1]. Copyright is crucial in such a large market, as hundreds of thousands of jobs and tens of billions of dollars in GDP are reported to be lost due to video piracy [2]. Actually, anti-piracy strategies have been extensively studied, among which the most widely used one is Digital Rights Management (DRM) [3]. Major streaming service providers such as Netflix and Hulu are all protected by a variety of DRM technologies including Widevine [4], FairPlay [5], and PlayReady [6]. Regrettably, DRM is not a panacea. Specialized hardware is required on mobile terminals for DRM [7], [8], [9]; otherwise, software-only DRM might be compromised [10]. Besides, DRM alone cannot defend against all types of attacks, such as screen recording with a camcorder. Some recent methods [11], [12] prevent recording by exploiting differences between the human visual system (HVS) and cameras, but they either impose a heavy overhead on the client [11] or require installing special devices in the physical environment [12], thereby reducing their applicability. Additionally, they are susceptible to software-based screen recording.

One of the next-best ideas might be to track down and take legal action against the pirate. Assume we can embed a unique fingerprint into the content provided to each user. Once an illegal copy is distributed, the distributor's fingerprint (i.e., the pirate) can be extracted and tracked. However, fingerprint embedding on the client side is not feasible for large-scale deployment, while embedding on the server side is not scalable due to the heavy overhead of generating a uniquely fingerprinted copy for each user. Therefore, this method is not widely used at the moment.

In fact, existing embedding techniques mainly focus on embedding a relatively large amount of data, e.g., the fingerprint, into every selected frame of the video. When it comes to video streaming, however, this type of cumbersome embedding schema is superfluous. Instead, by using a more lightweight and low-density embedding schema, even though only a small amount of data can be embedded in a single frame, sufficient information to identify the pirate can still be extracted using all of the frames. Based on this observation, considering a series of frames selected for watermark embedding (denoted as ‘host frames’ in this paper), we only encode a single bit into each

host frame and prepare two versions of each host frame to represent bit ‘0’ and ‘1’ respectively. Then we can distribute different series of host frames (mixed with the same non-host frames) to different users with their unique fingerprints embedded.

Unfortunately, applying this naïve approach faces many challenges. (1) *A framework compatible with nowadays’ mobile streaming infrastructures is required.* Today’s large-scale streaming relies on caching-based Content Delivery Networks (CDNs) to ensure a consistent level of quality of service (QoS) [13]. However, existing fingerprinting systems are mostly designed to compute offline, i.e., pre-processing the entire video for every user, and so are not suitable for real-time streaming. (2) *A more reliable and confidential host frame identification scheme is required.* Traditional watermarking systems typically identify keyframes as host frames, which could be detected and destroyed by attackers through re-performing keyframe detection. Besides, since bits are sequentially embedded into consecutive host frames, even a single false re-identification in the pirated video cannot be tolerated. (3) *A lighter, more robust, yet imperceptible embedding technique is required.* Existing watermarking algorithms based on singular value decomposition (SVD) are designed to embed much data and have been proven to be robust against single attacks. However, our scenario requires less watermark capacity but robustness against composite attacks. The complexity also needs to be reduced to process numerous videos. (4) *The collusion attack must be considered,* where attackers mix multiple copies and compromise the integrity of the embedded fingerprint.

In this paper, we present StreamingTag, a novel piracy tracking solution intended for streaming services. To address the first challenge, the video is divided into consecutive sections, and two variants (with bit 0 and bit 1 embedded, respectively) are generated from each section, using the digital watermarking technique. For each user requesting the video, a fingerprint is generated at run-time, and StreamingTag instructs users to fetch the corresponding variant of each section in accordance with their fingerprints (e.g., the user fetches the first variant of the fifth video section if the fifth bit of the fingerprint is 0). In this manner, no modification to the mobile terminals is required, and the variants of all video sections can still be cached by CDNs to guarantee the QoS. To address the second challenge, we randomly choose frames in the original video as host frames and design a novel slidingDTW algorithm to synchronize the pirated copy with the original copy, so that we can re-identify the watermarked host frames in the pirated copy. The slidingDTW algorithm is an optimized variant of existing DTW algorithms [14], which can take raw videos (of several Terabytes) as input and presents a linear complexity. With regard to the third challenge, we propose a novel watermarking technique that distinguishes the two variations dramatically by polarizing their singular values in opposite directions and embedding the same bit into multiple randomly-localized blocks. For the last challenge, we use a collusion-resistant fingerprint code whose security has been theoretically demonstrated.

The main contributions are as follows:

- We propose a new video delivery pipeline to trace piracy via fingerprinting, which is scalable, CDN-friendly, and compatible with the existing infrastructures.

- We present a lightweight yet robust slidingDTW algorithm, which improves the recall rate of host frame re-identification by 4-49x more than traditional methods.
- We propose a bipolar and randomized SVD-based watermarking scheme, improving the extraction accuracy by up to 2.25x compared to existing SVD watermarking while incurring no additional costs.
- To defend against collusion, we employ a randomized fingerprinting strategy and carefully select its parameters, achieving a 26% higher recall rate than existing methods.

An early version of this work was published in [15]. The journal version comprises additional research, including the identification and verification of issues with existing keyframe-based host frame identification methods in our scenario. It also entails the design and evaluation of the novel slidingDTW algorithm. *The main symbols are summarized in Section A.1 of the online supplementary material.*

## II. BACKGROUND

### A. Streaming and Live Streaming

HTTP-based streaming protocols, including HLS [16] and MPEG-DASH [17], have been the de facto standard for streaming. They require servers to split video streams into sequential segments. Clients download a manifest file with URLs for each segment and then request each segment.

Large-scale video streaming faces challenges due to the larger size of video data compared to other HTTP-transferred data. CDNs help alleviate this burden. [13] CDNs can be viewed as a hierarchy of caches [18]. In a CDN-enabled system, client requests are routed to a nearby CDN edge server rather than the origin server. The edge server responds directly if the content is cached, or fetches it from higher-level CDN servers or the origin server before caching and responding [19]. CDNs greatly boost content delivery in terms of scalability and latency, so it is crucial to keep compatibility with CDNs while designing StreamingTag.

### B. Watermarking and Fingerprinting

Digital watermarking embeds watermark data into multimedia carriers such as photos [20], music [21], and video [22]. In contrast, digital fingerprinting ensures that the embedded content is unique for each user so that we can trace back the piracy source. When we use the term *watermarking* in this paper, we mean the process of embedding invisible data into a video; while *fingerprinting* refers to the process of encoding a user’s identity. Besides, we refer to the video that has not been watermarked as the *original video*, the legitimate video that is watermarked and available to legitimate users as the *watermarked video*, and the pirated copy which is also watermarked as the *pirated video*.

### C. Keyframe Identification

As embedding a watermark into every frame [23], [24] can be highly time-consuming and vulnerable [25], recent works [26], [27] typically identify some keyframes, embed the watermark into the keyframes, and then re-identify these watermarked frames in the pirated video for watermark extraction. A frame

is deemed as a keyframe if it indicates the start of a new scene (i.e., its histogram difference from the preceding frame exceeds the threshold). In these works, keyframes are always host frames (i.e., frames with the watermark embedded), and vice versa.

Re-identifying watermarked frames in pirated videos merely by the indices of their corresponding frames in the original video is unreliable. For example, the pirated video may be transcoded to a different frame rate. Also, frame dropping or duplication in video players [28] may cause a difference in the number of frames between the original and pirated videos. Therefore, the same keyframe detection algorithm needs to be performed on the pirated video to locate the watermarked frames. However, keyframe re-identification can hardly be 100% accurate due to the distortion of the pirated video. As a consequence, StreamingTag will encounter a severe problem (see Section IV-A) if we directly use keyframes as host frames. Therefore, we randomly select some frames from the original video as host frames and propose a synchronization scheme (detailedly discussed in Sections IV-B and IV-C) to pinpoint them in the pirated video.

#### D. SVD-Based Video Watermarking

SVD watermarking modulates the watermark information into the singular values of uncompressed video frames. Since the singular values reflect the intrinsic rather than the visual characteristics of frames, embedding the watermark into the singular values can achieve both high imperceptibility and robustness [29]. The *embedding stage* of SVD watermarking consists of the following steps:

- *Step 1:* The input video is decoded into a stream of frames, some of which are identified as host frames. We use the symbol  $r$  to represent embedding regions in the host frames. Typically, the entire host frame is used as the embedding region.
- *Step 2:* Every embedding region  $r$  is converted to the frequency domain through some transform such as discrete cosine transform (DCT) [30], discrete wavelet transform (DWT) [31], or their combination [32], etc. Then the SVD operation is performed on a sub-band at a certain frequency (denoted as  $\mathcal{F}(r)$ ), i.e.,  $\mathcal{F}(r) = USV^T$ , where  $S$  is a diagonal matrix consisting of singular values. Typically, a sub-band at middle or high frequency is used in this step to preserve imperceptibility.
- *Step 3:* The watermark matrix  $S_{wm}$  is modulated into  $S$  through some operation such as matrix addition (i.e.,  $S'_{wm} = S + \alpha S_{wm}$ , where  $\alpha$  is the watermarking strength).
- *Step 4:* The frequency sub-band of the watermarked region,  $\mathcal{F}(r'_{wm})$ , is obtained by  $\mathcal{F}(r'_{wm}) = US'_{wm}V^T$ .
- *Step 5:* The watermarked region  $r'_{wm}$  is obtained by transforming  $\mathcal{F}(r'_{wm})$  back from the frequency domain to the spatial domain through the corresponding inverse operations, such as inverse DCT, inverse DWT, etc.

In the *extraction stage*,  $S'_{wm}$  is extracted by performing the above *Step 1-2* again on the pirated video. To obtain the embedded  $S_{wm}$  (which equals to  $\frac{S'_{wm}-S}{\alpha}$ ),  $S$  should also be extracted from the original (i.e., unwatermarked)  $r$ . Here we use

$S_{adj}$  extracted from the reference region  $r_{adj}$  (which is located at the same position in the adjacent frame from the pirated video and resembles  $r$  due to temporal redundancy) to approximate  $S$ , as it gives better results.

To further improve robustness, one common method is to repeatedly embed the same watermark data into multiple host frames. The host frames can be randomly selected, e.g., using a random sequence to determine their indices [31]. In this paper, we use redundancy-enhanced SVD watermarking as our baseline. In the *Step 2* of our baseline, the 3-level DWT transformation is used, and the  $HL_3$  sub-band (i.e.,  $W_\psi^V(J-3)$  in [33]) is selected as  $\mathcal{F}(r)$ . The input is a video segment, thus the baseline will be called  $m$  times to process a video consisting of  $m$  segments. Only the luminance component of an embedding region is used to improve the fidelity.

### III. SYSTEM DESIGN

#### A. Threat Models

This paper focuses on two types of pirate attacks: the *Naïve Redistribution* and the *Colluding Redistribution Attack*.

*Naïve Redistribution (NR) Attack.* In the NR attack, a single attacker tries to re-distribute a pirated copy. The pirated copy can be generated by downloading it from the streaming platform or screen recording via a software-based recorder. In the former case, the pirated copy may be further modified prior to re-encoding and redistribution, through some common signal processing operations (including scaling, Gaussian noising, and median filter), so that the embedded watermark can be destroyed. However, such a modification is visually insignificant to preserve the visual quality. The latter case resembles a composite of decoding, scaling, and re-encoding. However, it is actually much tougher, as video players may duplicate or drop some frames (as noted in [28]) and we cannot simply locate host frames in the pirated video merely through their indices.

*Colluding Redistribution (CR) Attack.* Several attackers collaborate by mixing different copies in the CR attack. Formally, the set of colluders is denoted as  $C$  (with  $c = |C|$ ), and for each colluder  $j \in \{1 \dots c\}$ , the delivered version of segment  $i$  is denoted as  $segment_{i,j}$ . In this paper, we assume that the colluders will randomly select a segment file from  $\{segment_{i,1}, \dots, segment_{i,c}\}$ , and the probability of selecting any segment file is  $\frac{1}{c}$ . Colluders may prefer this collusion strategy for two reasons: 1) Risk is fairly shared, so the maximum degree of suspicion among all colluders is lowered; 2) Such a strategy allows colluders to easily cooperate even if they do not trust each other. After the selection, the attackers further make minor visual changes to the selected segment file, similar to the NR attack. Apparently, the NR attack is a special and the easiest case of the CR attack.

#### B. Design Challenges

The system should employ fingerprinting to trace pirates. However, three challenges must be overcome to handle the NR attack, and the CR attack brings one more challenge.

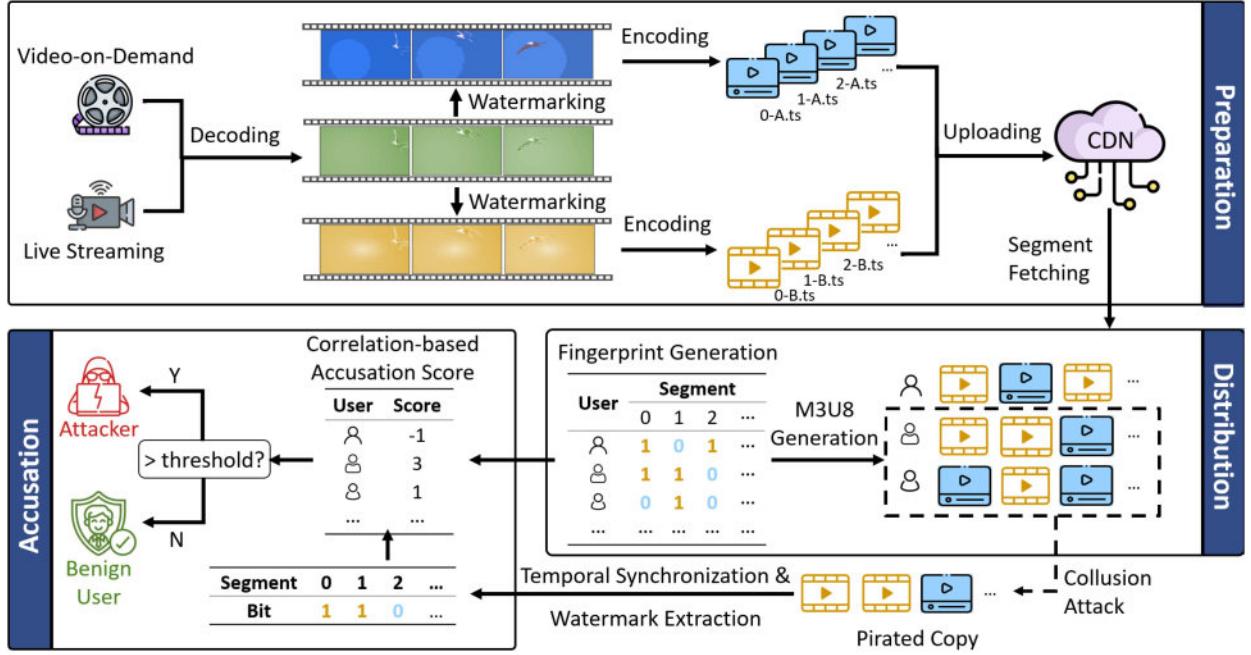


Fig. 1. The workflow of StreamingTag.

- *Challenge 1:* As no modification should be applied to mobile terminals for full compatibility, we should generate all fingerprinted copies on the server side. This can completely negate the QoS improvement brought by CDNs, which capitalize on the fact that multiple users make the same digital content request. Besides, this can impose a significant overhead on the server.
- *Challenge 2:* Video distortion incurred by the NR attack can cause inconsistent detection of keyframes. As StreamingTag embeds bits sequentially, a single inconsistent detection can displace all the following extracted bits (detailed in Section IV-A). Additionally, skilled attackers can also detect these watermarked frames and remove them.
- *Challenge 3:* Existing SVD watermarking techniques are not sufficient. They are only proven to be robust against single attacks, while a chain of attacks can occur in our scenario (detailed in Section V-A). Besides, to support large-scale streaming, the heavy cost of the key operation, SVD, must be reduced.
- *Challenge 4:* The attackers can still be caught even if they launch a CR attack to break down the fingerprint integrity.

### C. System Overview

*StreamingTag in a nutshell.* The workflow of StreamingTag is shown in Fig. 1. In the preparation stage, the input video is split into segments of equal length, and two watermarked variants of each segment are generated, with bit 0 and bit 1 embedded respectively. Following this stage, approximately double-sized video content is generated. Next, in the distribution stage, a fingerprint code of length  $m$  (i.e.,  $X_j$  in Section VI-A) will be generated for any user  $j$  requesting the video. Based on  $X_j$ ,

StreamingTag creates a unique manifest file for user  $j$ , ensuring that the  $i$ th bit,  $X_j^{(i)}$ , is embedded in the  $i$ th segment listed in the manifest file. Consequently, different users are instructed to play distinct copies, and the variants of segments form a bit-stream, i.e., the user's fingerprint, which can be used to track a user. The accusation stage is the last stage. Once the video is illegally distributed, we extract the embedded bits from each segment to hunt out the attackers.

The preparation stage only needs to be performed once for each video, so it introduces almost negligible cost. Besides, as the variants generated in the preparation stage are directly served to different users in the distribution stage, StreamingTag can support large-scale streaming by caching these variants on CDNs. To summarize, StreamingTag effectively addresses Challenge 1 by its lightweight and CDN-friendly delivery pipeline.

*Host Frame Identification (Section IV).* To overcome Challenge 2, we begin by randomly selecting certain frames as host frames and secretly recording their indices in the preparation stage. In the accusation stage, the pirated video is synchronized with the original video on a frame-by-frame basis, enabling us to pinpoint the host frames and extract watermarks. A novel slidingDTW is adopted for synchronization.

*Bipolar and Randomized SVD watermarking (Section V).* Facing Challenge 3, we consider the loose requirement on watermark capacity in StreamingTag, and propose a more robust yet more lightweight watermarking scheme.

*Fingerprint generation and accusation (Section VI).* To meet Challenge 4, we employ a randomized collusion-resistant fingerprint generation and accusation strategy. Fingerprint data is stored in a database, and fingerprints extracted from pirated copies are compared with stored fingerprints to identify attackers.

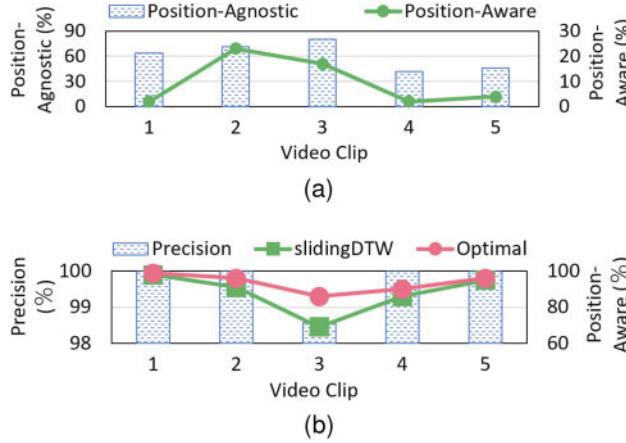


Fig. 2. A comparison of keyframe-based methods and slidingDTW. (a) The performance of keyframe-based methods. (b) The performance of slidingDTW.

#### IV. HOST FRAME IDENTIFICATION

##### A. Robustness of Keyframe-Based Methods

Existing piracy tracing systems [26], [27] typically embed the same watermark into multiple keyframes and integrate the extracted watermarks to obtain the final watermark, by taking the mode [26] or the average at every position. Such methods work well in traditional systems, but they may fail in our scenario for two reasons.

- *Concern 1: Confidentiality of Keyframes.* Keyframes can be detected and removed by the attacker.
- *Concern 2: Possible Displacement.* The inconsistent detection of keyframes can occur as the pirated video can be slightly different from the original video under the NR attack. Such inconsistency can be tolerated in existing systems [26], [27] as they integrate all the extracted watermarks. However, in StreamingTag, bits are sequentially embedded into host frames, and a single inconsistent detection during extraction can displace all the following bits. For example, denote the first keyframe in the original video as  $f$ , and its corresponding frame in the pirated video as  $f'$ . Assuming that the fingerprint is ‘101010’, it could be falsely extracted as ‘01010’ since  $f'$  may be determined as a non-keyframe in the extraction stage.

We conduct preliminary experiments to demonstrate the severity of the two concerns. We randomly select five clips of 100 seconds from the third video in Table II for the experiments. To easily determine whether a detected keyframe in the pirated clip corresponds to some keyframe in the original clip, we pre-process the original clip by drawing a sequence number on every frame of it via the *drawtext* filter in FFmpeg [34]. The determination is made by comparing the sequence numbers. Then, in the pre-processed original clip, we compute the luminance component histogram difference to detect keyframes.

A watermark is embedded into the chrominance component of every keyframe through our watermarking scheme (see Section V-B) to generate a watermarked clip. A pirated clip is generated from each watermarked clip by software-based

screen recording on a Xioami 12S smartphone. We apply the same keyframe detection scheme to the pirated clips. We manually select two distinct thresholds, so that exactly 100 keyframes are detected in both the original and the pirated clip. The sequence numbers of keyframes in the original clip (or pirated clip) are denoted as  $Key^o = \{key_1^o, \dots, key_{100}^o\}$  (or  $Key^p = \{key_1^p, \dots, key_{100}^p\}$ ), where  $key_1^o < \dots < key_{100}^o$  (or  $key_1^p < \dots < key_{100}^p$ ).

We use the position-agnostic recall rate, i.e.,  $\frac{|Key^o \cap Key^p|}{|Key^o|}$ , to quantify the severity of our first concern. Then, we calculate the position-aware recall rate, i.e.,  $\frac{|key_k^o = key_k^p|, 1 \leq k \leq 100}{|Key^o|}$ , to quantify the second concern. As shown in Fig. 2(a), the position-agnostic recall rate can be as high as 80%, while the position-aware recall rate ranges from 2% to 23%. Therefore, such methods are vulnerable and unreliable.

##### B. DTW-Based Temporal Synchronization

Dynamic time warping (DTW) is a well-known algorithm to synchronize two time series if one can be warped to align with another “by stretching or shrinking it along its time axis.” [14] It has been widely used in areas such as speech recognition [35], gesture recognition [36], etc. The inputs to DTW are two time series  $TS^1 = ts_1^1, ts_2^1, \dots, ts_{|TS^1|}^1$  and  $TS^2 = ts_1^2, ts_2^2, \dots, ts_{|TS^2|}^2$ , with  $|TS^1|$  and  $|TS^2|$  being their lengths. The goal is to construct a path  $w = w_1, w_2, \dots, w_K$ , with  $\max(|TS^1|, |TS^2|) \leq K < |TS^1| + |TS^2|$ . The path  $w$  indicates how  $TS^1$  and  $TS^2$  are aligned to each other, and its element  $w_k = (w_k.x, w_k.y)$  implies that  $ts_{w_k.x}^1$  is matched to  $ts_{w_k.y}^2$  in such an alignment. Two conditions must be met to construct a valid  $w$ :

- *Boundary Condition.*  $w_1 = (1, 1)$ ,  $w_K = (|TS^1|, |TS^2|)$ .
- *Continuity Condition.* For any  $1 \leq k < K$ ,  $(w_{k+1}.x - w_k.x, w_{k+1}.y - w_k.y) \in \{(0, 1), (1, 0), (1, 1)\}$ .

The goal is to construct an optimal  $w^*$  which minimizes the distance, i.e.,  $Dist(w^*) = \sum_{k=1}^K \text{Difference}(ts_{w_k.x}^1, ts_{w_k.y}^2)$ . An example of using DTW to synchronize videos is shown in Fig. 3(a). A dynamic programming method can be used to find  $w^*$ , while its space and time complexities are as high as  $O(|TS^1||TS^2|)$ . We seek to synchronize the pirated video with the original one through a complexity-optimized variant of DTW.

*Sliding Window-Based Synchronization.* The lightweight fastDTW algorithm [14] has been proposed to approximate DTW, reducing the space and time complexities to  $O(\max(|TS^1|, |TS^2|))$ . Despite the linear complexity, it is still challenging to apply fastDTW in our scenario, where a video can last for hours and be too large ( $\sim$ Terabytes) in the uncompressed form to fit in memory. Surely, frame hashing can be adopted to reduce the representation size of each frame. [37] Yet, frame hashing inevitably loses some information. For adjacent frames of high temporal redundancy, a lossy representation may not be enough to achieve accurate synchronization of frame-level granularity.

To reduce the memory overhead, we propose to perform DTW on sliding windows. The basic idea is to synchronize a

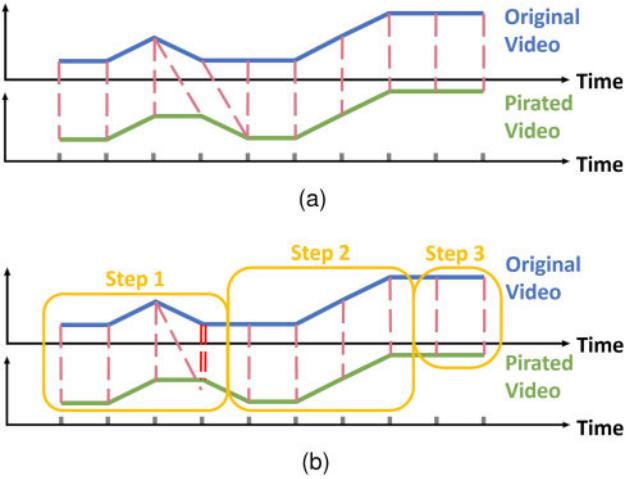


Fig. 3. A comparison of DTW and basicSlidingDTW. (a) Using DTW to synchronize two videos. Here we represent each frame with a one-dimensional value for simplicity. The third frame of the original video is duplicated in the pirated video, while the fifth frame is dropped. (b) An example of basicSlidingDTW. The window length is set to 4 and a total of 3 steps are executed to iterate through the entire sequences. The red double dashed line represents a sub-optimally matched pair due to the boundary condition of DTW.

small window of frames from the pirated video with a window of corresponding frames from the pirated video through the standard DTW. The two windows can share the same length if the two videos are of the same frame rate; otherwise, their lengths can be set proportional to the frame rates. Here we assume the former case to simplify the description. The first window is denoted as  $(f_0^o, \dots, f_{window\_length}^o)$ , with  $f_k^o$  representing the  $k$ th frame in the original video; the second window is denoted as  $(f_0^p, \dots, f_{window\_length}^p)$ . The two windows are repeatedly slid toward future frames until the end of either video. The final result is simply the concatenation of synchronization results between each pair of windows. We refer to this basic algorithm as basicSlidingDTW, where the memory requirement is only related to the window length rather than the entire video length.

*Handling Frame Duplication and Dropping.* As shown in Fig. 3(b), the basicSlidingDTW algorithm can cause sub-optimal synchronization under the NR attack model, where frames from the watermarked video can be dropped or duplicated in the pirated video from time to time. In both cases, moving the two windows by the window length would make the two new windows not well matched to each other, and the drift can accumulate over time.

To alleviate the effect of frame dropping and duplication, we first detect sub-optimally matched frame pairs between the two windows. We determine a pair  $(f_{w_k.x}^o, f_{w_k.y}^p)$  to be sub-optimal if

$$\begin{aligned} & Difference(f_{w_k.x}^o, f_{w_k.y}^p) \\ & \leq \min \left( \min_{\delta k \neq 0, |\delta k| < v} Difference(f_{w_k.x+\delta k}^o, f_{w_k.y}^p), \right. \\ & \quad \left. \min_{\delta k \neq 0, |\delta k| < v} Difference(f_{w_k.x}^o, f_{w_k.y+\delta k}^p) \right), \end{aligned} \quad (1)$$

holds. To make it clear why we check sub-optimal paring by comparing with adjacent frames, consider a case where the true corresponding frame for the frame  $f_{w_k.x}^o$  is  $f_{w_k.y+\delta k}^p$  ( $\delta k \neq 0$ ). Since  $f_{w_k.x}^o$  resembles  $f_{w_k.y+\delta k}^p$  and  $f_{w_k.y+\delta k}^p$  may resemble  $f_{w_k.y}^p$  due to temporal redundancy, it is likely that  $f_{w_k.x}^o$  resembles  $f_{w_k.y}^p$ , making it difficult to determine the occurrence of a sub-optimal pairing by simply comparing  $Difference(f_{w_k.x}^o, f_{w_k.y}^p)$  with an empirically and subjectively pre-defined threshold. Therefore, we turn to look around nearby frames to make a more objective and accurate judgment of sub-optimal pairing, and then remove the sub-optimal pairs (line 12-16 in Algorithm 1, Section A.2 of the supplementary material).

We mark a frame as “unmatched” if all the DTW pairs containing it are sub-optimal. We slide the window so that in the next step it starts from the first frame from the current window whose following frames in the current window are all “unmatched.” Otherwise, if the last frame in the current window is “matched”, we move the window by its length (line 25-33 in Algorithm 1, Section A.2 of the supplementary material). In this manner, in the case that frame duplication occurs (as shown in Fig. 4(a)), there will exist some unmatched frames located at the end of the original video’s sliding window, and they will be correctly matched in the next step (as shown in Fig. 4(b)). Likewise, in the case that frame dropping occurs (as shown in Fig. 4(b)), the unmatched frames located at the end of the pirated video’s sliding window will be correctly matched in the next step (as shown in Fig. 4(c)). Therefore, we can always keep the two windows well-matched even in case of frame duplication or dropping.

### C. Algorithm Evaluation

The detailed workflow of slidingDTW is presented in Algorithm 1, Section A.2 of the supplementary material. It performs standard DTW to match the two sliding windows (line 11) and remove sub-optimal pairs via (1) (line 12-16). Each standard DTW process introduces a computational complexity of  $O(window\_length^2 \cdot frame\_size)$ , and each sub-optimal matching detection process incurs a computational complexity of  $O(window\_length \cdot v \cdot frame\_size)$ , with  $v$  being a hyper-parameter in (1). The two sliding windows are then slid (line 25-33) so that the algorithm can proceed to match the following frames. As the sliding step is at least one, the entire algorithm has a computational complexity of  $O(\min(m1, m2) \cdot window\_length \cdot (window\_length + v) \cdot frame\_size)$ , with  $m1$  (or  $m2$ ) being the number of all frames in the original video (or the pirated video). The space complexity is  $O(window\_length \cdot frame\_size)$ , as only the frames in the sliding windows need to be kept in memory.

We conduct preliminary experiments to show the excellent performance of slidingDTW, using the same 5 video clips as in Section IV-A. A total of 100 frames are selected in each clip and used as host frames. Denote the number of host frames in an original clip, for each of which slidingDTW finds a matched frame in the pirated clip, as  $n1$ , and the number of host frames

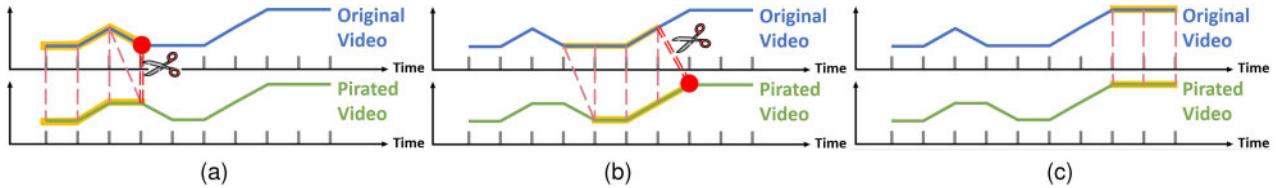


Fig. 4. An example of slidingDTW. The sub-optimal matched pairs are correctly detected by using (1) and removed (represented by the scissors icon in the figure). The ‘mismatched’ frames are represented by red dots. The final result is the concatenation of results at all steps and the same as that of applying DTW on the entire series in this example (see Fig. 3(a)). (a) The first step. (b) The second step. (c) The last step.

in an original clip, for each of which slidingDTW correctly identifies the truly corresponding frame as the matched frame in the pirated clip, as  $n_2$  ( $n_2 \leq n_1$ ). Then we can calculate the precision as  $\frac{n_2}{n_1}$  and the position-aware recall rate as  $\frac{n_2}{100}$ . Furthermore, we can denote the number of host frames not dropped in the pirated video as  $n_3$  ( $n_3 \geq n_2$ ) and define the optimal position-aware recall rate as  $\frac{n_3}{100}$ . As shown in Fig. 2(b), slidingDTW consistently achieves a precision above 98%, which demonstrates the accuracy of our sub-optimal pairing detection scheme shown in (1), as well as a high position-aware recall rate (4-49 times higher than keyframe-based methods). However, when the optimal position-aware recall rate decreases (as in the case of the 3-rd video clip), the position-aware recall rate of slidingDTW drops a little faster. As the optimal recall rate is determined purely by the quality of the pirated video, we conclude from the experimental results that the slidingDTW algorithm is relatively sensitive to the quality of the pirated video. Therefore, applications may need to use a larger number of host frames (i.e., a larger  $n_e$ ) so that a sufficient number of host frames can still be extracted from a low-quality pirated video.

## V. BIPOLAR AND RANDOMIZED SVD WATERMARKING

### A. Robustness of Watermarking

Watermarking technologies can embed invisible data into an image to enable piracy tracing, tampering detection, etc. While existing watermarking techniques are believed to be robust, their reliability in our context remains in doubt for the following reasons: 1) Typically, the robustness of a watermarking algorithm is assessed against a single attack, whereas a watermarked segment is subjected to composite attacks in both the NR and CR attack models. To be more precise, the attackers are willing to decode the received segment (whose embedded watermark is already affected by compression), then perform malicious operations to remove the embedded watermark, and finally re-encode the modified segment for efficient redistribution. 2) As stated in Section II-D, the baseline method can utilize redundant embedding to improve robustness [31], [38]. However, for a rather short video segment, it’s highly likely that all the host frames within it are similar due to temporal redundancy. As the correctness of extracting the watermark depends on the content of the host frame and the embedded watermark, extracting the same watermark repeatedly from the similar host frames may always produce the same result, making the redundancy-based enhancement useless.

TABLE I  
THE WATERMARK EXTRACTION ACCURACY OF TRADITIONAL SVD  
WATERMARKING UNDER DIFFERENT ATTACKS

Video ID	Attack				
	H.264	MF <sup>a</sup> (3*3)	H.264+ MF (3*3) + H.264	GN <sup>b</sup> ( $\sigma = 5$ )	H.264+ GN <sup>b</sup> ( $\sigma = 5$ ) + H.264
1	82.0%	67.8%	57.1%	87.0%	69.0%
2	67.9%	69.2%	58.4%	79.1%	62.6%
3	77.8%	74.5%	56.6%	90.5%	63.5%

<sup>a</sup> MF: Median Filter    <sup>b</sup> GN: Gaussian Noise

We utilize the baseline SVD watermarking introduced in Section II-D to quantify the significance of the above two concerns. As the baseline method was initially designed to embed the singular value matrix of a watermark image rather than a single bit, we employ two different images (whose singular value matrixes are  $S_{wm}^0$  and  $S_{wm}^1$ ) to represent bit 0 and bit 1 in the embedding stage. The extraction stage is slightly modified to map the extracted  $S_{wm}$  to bit 0 or 1 as follows: the extracted bit is 0 if the cosine similarity between the extracted  $S_{wm}$  and  $S_{wm}^0$  is greater than that between  $S_{wm}$  and  $S_{wm}^1$ ; otherwise bit 1 is extracted.

We conduct two preliminary experiments with the baseline method by using it to modulate a single bit into every segment. The first experiment employs five distinct attacks on the watermarked segments. As shown in Table I, the third and fifth attacks are composite attacks, more closely resembling real-world events. Given that composite attacks do really significantly reduce robustness, they must be factored into our architecture. In the second experiment, we change the number of host frames used to repeatedly embed a single bit (which reflects the level of redundancy) to validate the effectiveness of redundant embedding. The imposed attack is H.264 encoding. The baseline method is compared against an alternative strategy, which first randomly determines the indices of host frames, then randomly selects sub-regions (of size  $512 \times 512$ ) located at different positions of these host frames, and finally embeds the same bit into these random regions rather than the entire host frames. In this manner, the contents of these embedding regions differ, and the interdependence among different votes is decreased. For the purpose of fair comparison, the baseline method also embeds the watermark into sub-regions (whose size is also  $512 \times 512$ ) of host frames, but all these sub-regions are at the same spatial position. Fig. 5 illustrates the measurement findings. As the number of embedding regions increases, the

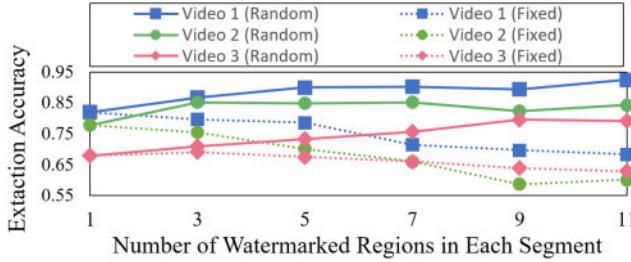


Fig. 5. A comparison of two embedding region selection strategies.

alternative method presents higher robustness, while the baseline method degenerates for two reasons. *First*, as expected, employing similar embedding zones within a brief video piece does not always increase resilience. *Second*, when the number of watermarked blocks increases, spatial and temporal redundancy reduces, pushing H.264 to increase the compression level to maintain a stable bit rate.

### B. Watermarking for Streaming Services

Since the baseline method is ineffective in our scenario, we propose two strategies to further improve its robustness and computational cost, taking advantage of the fact that our case provides a lenient restriction on watermark capacity (i.e., needing only one bit to be embedded in a segment). Additionally, based on the results of the second experiment in Section V-A, we propose a third improvement strategy. The following are the three techniques:

*Polarized Singular Value Modulation.* The baseline method modulates  $S_{wm}$  (which depends on the actual bit to be embedded) to  $S$  by matrix addition (i.e.,  $S'_{wm} = S + \alpha S_{wm}$ ). Instead, we propose to modulate the single bit  $\beta$  via

$$S'_{wm} = S + (-1)^\beta \alpha S_{wm}, \quad (2)$$

where  $S_{wm}$  is a constant matrix. The rationale is that, because watermarked frames are often subjected to a number of attacks in our situation, we should aim to increase the distinction between two different forms of the same frame (with bit 0 and bit 1 embedded, respectively) and make them more easily distinguishable. Following (2), the two forms are polarized in opposite directions and immediately discernible.

*Semi-Explicit SVD-based Watermarking Scheme.* The embedding overhead should be optimized to support real-time watermarking of numerous live video streams. However, the SVD operation on an  $C1 \times C2$  matrix has a time complexity of  $O(C1(C2)^2)$  [39]. Since we only care about  $\beta$  in the extraction stage, we can directly set  $S_{wm}$  equal to  $S$  and re-write (2) as

$$S'_{wm} = (1 + (-1)^\beta \alpha) S. \quad (3)$$

Consequently,  $\mathcal{F}(r'_{wm}) = U S'_{wm} V^T = U(1 + (-1)^\beta \alpha) S V^T = (1 + (-1)^\beta \alpha) \mathcal{F}(r)$ . Based on this, we propose to directly compute  $\mathcal{F}(r'_{wm})$  through scaling  $\mathcal{F}(r)$  by  $(1 + (-1)^\beta \alpha)$  without explicitly performing SVD to obtain  $S$ . This method allows us to polarize the singular values without performing SVD. Since the extraction stage does not require real-time

execution as the embedding stage does, we still perform SVD to obtain  $S'_{wm}$  and  $S_{adj}$ , and determine the extracted bit as 0 if and only if  $|S'_{wm}| > |S_{adj}|$ . As evaluated in Section VII-C, comparing the singular values (i.e.,  $S'_{wm}$  and  $S_{adj}$ ) is of higher robustness than directly comparing  $\mathcal{F}(r'_{wm})$  and  $\mathcal{F}(r_{adj})$ . The reason is that singular values of a matrix capture the intrinsic characteristics of a matrix and are thus more robust than the original matrix when facing attacks. Since this approach performs SVD explicitly only during the extraction stage, we refer to it as a semi-explicit SVD-based watermarking scheme.

*Randomized Embedding Region Selection.* We begin by randomly selecting multiple host frames from each segment and then randomly selecting an embedding region from each selected host frame. The watermark is not embedded in the entire host frames, but only in the selected regions. As discussed in Section V-A, this may reduce the interdependence among different votes, thereby increasing the accuracy of the majority voting-based watermark extraction.

We are able to propose a watermarking algorithm based on the above three techniques, consisting of an embedding part and an extraction part. The slidingDTW algorithm is called before executing the extraction part so that we know which frames to extract the watermark. We detailedly describe its workflow in Algorithm 2 and 3, Section A.2 of the supplementary material. The time complexity of watermarking a video is  $O(m \cdot n_e \cdot \text{size}^2)$ , with  $m$  being the number of segments,  $n_e$  being the number of host frames in each segment, and  $\text{size}$  being the size of each embedding region. Excluding the overhead of pinpointing the host frames via slidingDTW, watermark extraction has a time complexity of  $O(m \cdot n_e \cdot \text{size}^3)$ .

## VI. FINGERPRINT GENERATION AND ACCUSATION

As proved in [40], there are no deterministic (i.e., totally secure with no errors) fingerprinting codes against the collusion attack. Following the Tardos fingerprinting scheme [41], [42], [43], we employ a randomized fingerprinting strategy to defend against the CR attack. We begin by introducing the Tardos code and then describe how we build a fingerprinting system for streaming and live streaming services.

### A. The Tardos Fingerprint Scheme

*Generation.* In the Tardos scheme, the fingerprint length is denoted as  $m = Ac_0^2 \ln \frac{n}{\eta}$ , where  $A$  is a constant coefficient,  $c_0$  is the maximum number of colluders defendable by the system,  $n$  is the number of users, and  $\eta$  is the desired upper bound of  $P_{FP}$  (i.e., the probability that at least one innocent user is falsely accused). All users' fingerprints are denoted as a  $n \times m$  matrix  $X$ , where its  $j$ th row,  $X_j$ , is the fingerprint assigned to the  $j$ th user.

A fingerprint will be sampled independently and added to  $X$  as its new row whenever a user requests the content.  $m$  parameters  $(p_1^{(i)}, 1 \leq i \leq m)$  are sampled before sampling the first row and then will be fixed. The probability density function used to sample  $p_1^{(i)}$  from  $[t, 1 - t]$  is

$$g(p) = N_t / \sqrt{p(1 - p)}, \quad (4)$$

where the coefficient  $N_t$  ensures  $\int_t^{1-t} g(p)dp = 1$ , and the cutoff parameter  $t$  can be tuned to adapt to different performance requirements. Typically  $t$  is set to a small positive number. The reason for using such  $g(p)$  and a small  $t$  will be discussed later in Section VI-B. Then, each element  $X_j^{(i)}$  is sampled independently, with  $P[X_j^{(i)} = 1] = p_1^{(i)}$  and  $P[X_j^{(i)} = 0] = p_0^{(i)} = 1 - p_1^{(i)}$ .

*Accusation.* Denote the set of colluders as  $C$ , where  $c = |C| \leq c_0$ , and the sub-matrix of  $X$  that corresponds to  $C$  as  $X_C$ . The fingerprint embedded in the pirated copy  $y$  can be viewed as  $y = (y^{(1)}, \dots, y^{(m)}) = \rho(X_C)$ , where  $\rho$  is a (deterministic or probabilistic) function determined by the collusion strategy, and  $y^{(i)} \in \{0, 1\}$ .

Colluders are accused based on the calculated accusation score. For any user  $j$  that once requested the video, the accusation score is computed as  $s_j = \sum_{i=1}^m s_j^{(i)} = \sum_{i=1}^m (\delta_{y^{(i)}, X_j^{(i)}} g_1(p_{y^{(i)}}^{(i)}) + (1 - \delta_{y^{(i)}, X_j^{(i)}}) g_0(p_{y^{(i)}}^{(i)}))$ , where  $\delta_{x,y} = \begin{cases} 1, & \text{if } x = y, \\ 0, & \text{if } x \neq y, \end{cases}$ ,  $g_1(p) = \sqrt{\frac{1-p}{p}}$ , and  $g_0(p) = -\sqrt{\frac{p}{1-p}}$ . The intuitions behind such  $s_j$  are as follows: 1) The  $j$ th user should be more suspicious if  $y^{(i)} = X_j^{(i)}$ . Following this formula, the score  $s_j$  will thus be increased by  $|g_1(p_{y^{(i)}}^{(i)})|$ . Otherwise, in case that  $y^{(i)} \neq X_j^{(i)}$ ,  $s_j$  will be decreased by  $|g_0(p_{y^{(i)}}^{(i)})|$ . 2) In the case of  $y^{(i)} = X_j^{(i)}$ , the smaller the value of  $p_{y^{(i)}}^{(i)}$  is, the fewer the number of users having their  $i$ th fingerprint bit equal to  $y^{(i)}$  are, and the more suspicious the  $j$ th user should be (because the  $j$ th user is one of the few users owning that  $y^{(i)}$ ). As  $g_1(p)$  is a monotonic decreasing function, increasing the accusation score by  $g_1(p)$  correctly reflects the relationship between  $p_{y^{(i)}}^{(i)}$  and the degree of suspicion. 3) For any position  $i$  and any innocent user  $j$ ,  $s_j^{(i)}$  has an expectation of 0 and a variance of 1.

The accusation threshold is  $z = Bc_0 \ln \frac{n}{\eta}$ , where  $B$  is a constant coefficient. If and only if  $s_j > z$ , the  $j$ th user will be accused. Using this accusation strategy, it can be theoretically guaranteed [41], [42] that  $P_{FP} < \eta$  holds with carefully selected  $t$ ,  $A$  (the coefficient of the code length  $m$ ), and  $B$  (the coefficient of the accusation threshold  $z$ ).

## B. Discussion of the CR Attack

*Formal description.* The collusion strategy used in the CR attack model can be formalized as follows:

- *Assumption 1: Marking Condition.* If and only if  $\exists \beta \in \{0, 1\} \text{ s.t. } \forall j \in C, X_j^{(i)} = \beta$ , the  $i$ th position will be considered undetectable by colluders, and  $y_i = \beta$  will always hold.
- *Assumption 2: Independent Strategy.* For any  $i \neq j$ , the sampling of  $y_i$  is independent from that of  $y_j$ .
- *Assumption 3: Equiprobable Strategy.* If we define  $b_\beta^{(i)} = |\{j | X_j^{(i)} = \beta, j \in C\}|$ ,  $P(y_i = \beta) = \frac{b_\beta^{(i)}}{c}$  will hold for each position  $1 \leq i \leq m$  and each bit  $\beta \in \{0, 1\}$ . In other words, for each position  $i$ , the colluders randomly select

one member, with the selection rate of each member being  $\frac{1}{c}$ . Then the segment received by the selected member will be used as the  $i$ th part of the pirated copy.

*Improving the Tardos code under the CR model.* The marking condition has been widely used in previous works [41], [42], [44]. To utilize the marking condition, the cutoff parameter  $t$  used in the distribution function (see (4)) is typically set close to 0 [41], [42]. As the distribution is biased towards two ends of  $[t, 1-t]$ , such  $t$  leads most  $p_1^{(i)}$  close to either 0 or 1. Consequently, the colluders will receive the same variant at most positions. According to the marking condition, their specific collusion strategy has no influence on these positions and thus limited influence on their chance of being caught.

Although a  $t$  close to 0 takes advantage of the marking condition, it reveals limited information about the piracy source, since users always receive the same variant. As a result, the fingerprint is too long to be practical. Therefore, we re-select the value of  $t$  by deducing the expectation of the sum of all colluders' accusation scores under the CR attack model. We set  $t$  to 0.5 with the support of theoretical analysis (see Section VI-C), and greatly reduce the code length needed to defend the same number of colluders.

## C. Parameter Selection

*Objectives.* When analyzing the Tardos scheme's security level, two often used metrics are  $P_{FP}$  (see Section VI-A) and  $P_{FN}$  (which represents the probability that no guilty user will be correctly accused of piracy). The precise values of three adjustable parameters, namely  $A$ ,  $B$ , and  $t$ , can affect both  $P_{FP}$  and  $P_{FN}$ . We derive optimal parameter values theoretically for the CR attack model in order to satisfy the following objectives:

- *Objective 1:* when  $1 \leq c \leq c_0$ ,  $P_{FP} \leq \eta$ .
- *Objective 2:* when  $1 \leq c \leq c_0$ ,  $P_{FN} \leq \epsilon$ .
- *Objective 3:*  $A$ , the coefficient of the fingerprint length  $m$ , should be kept small so that we can embed the whole fingerprint by embedding one bit into each segment.
- *Objective 4:* The value of  $t$  should be independent of  $c_0$  (which could possibly be related to  $n$ ),  $n$  and  $m$  (which is related to the unpredictable duration of the video in the case of live streaming services), so that the fingerprint generation process can directly boot up without an arbitrary estimation of these values.

*Sufficient Conditions Theorems.* The sufficient conditions for *Objective 1* and *Objective 2* are summarized in the following two theorems. Theorem 1 is proved by Škorić et al. [42], while the proof of Theorem 2 is conducted in Section A.4 of the supplementary material.

*Theorem 1 (The sufficient condition for Objective 1)* If the parameter values satisfy and

$$B^2/(4A) \geq 1, \quad (5)$$

*Objective 1* will be achieved.

*Theorem 2 (The sufficient condition for Objective 2)* If the colluders' strategy conforms to *Assumption 1 - 3*, and

TABLE II  
THE ATTRIBUTES AND QOS METRICS OF EACH TEST VIDEO

Video ID	1	2	3	4	5	6	7	8	9
Category	Chat	Dance	Western	Musical	Guitar Playing	Variety Show	Science	Documentary	Game Show
Resolution	1280 × 720		1920 × 816			1920 × 1080			
Video Length	01:30:46	01:09:38	01:39:31	01:15:45	01:25:28	01:06:39	01:15:28	01:01:34	01:34:55
Entire Video	EM Duration	08:11	6:18	14:28	15:25	16:16	12:30	13:34	11:51
	Rate (fps)	277	276	170	123	131	133	139	130
Single Region	EM (ms)	6.7	6.7	6.9	7.0	6.9	6.9	6.8	7.0
	EX (ms)	13.9	13.2	16.0	17.8	17.2	17.0	15.4	17.2
	Size Inc (%)	2.7	1.5	7.7	1.8	3.8	7.1	5.4	1.3

**Annotations:** 1. Abbreviations used in the 2nd column: EM - Embedding; EX - Extraction 2. Format of time intervals: (HH:)MM:SS

$$\tilde{\mu} = \frac{E[\sum_{j \in C} s_j]}{m} \text{ satisfies}$$

$$A\tilde{\mu} - B \geq 0, \quad (6)$$

and  $\frac{(A\tilde{\mu} - B)^2}{A} \geq \frac{4\ln\frac{1}{\eta}}{c_0\ln\frac{n}{\eta}}$ , Objective 2 will be achieved.

To meet *Objective 1 and 2*, we instead try to meet the above sufficient conditions. If we only consider the inequality (5) and (6), we will get:  $\frac{B}{\tilde{\mu}} \leq A \leq \frac{B^2}{4}$ . Then the optimal value of A would be  $A^* = \frac{4}{\tilde{\mu}^2}$ . According to *Lemma 1* (in Section A.3 of the supplementary material),  $\tilde{\mu}$  obtains its maximum value 1 if and only if  $t = \frac{1}{2}$ , so the minimum value of  $A^*$  is 4. With further consideration of other inequalities in the above two theorems and the remaining objectives (i.e., *Objective 3 - 4*), StreamingTag sets  $t$  to  $\frac{1}{2}$ ,  $A$  to  $4(1 + \sqrt{\phi})$ , and  $B$  to  $4 + 2\sqrt{\phi}$ , as long as  $\phi = \frac{8\ln\frac{1}{\eta}}{c_0\ln\frac{n}{\eta}} \leq 1$ . With this setting, we can verify that all the inequalities in Theorems 1 and 2 hold.

*Parameter Selection Strategy in StreamingTag.* StreamingTag sets  $t$  to 0.5. Once the preprocessing stage ends, the number of segments is determined, so the fingerprint length  $m$  is fixed. Every time a new pirated copy is captured by the streaming platform, the number of users,  $n$ , may be different, and StreamingTag will update the parameters ( $\eta$ ,  $\epsilon$ ,  $c_0$ ,  $A$ ,  $B$ , and  $z$ ; excluding  $t$  and  $m$ ) so that  $A = 4(1 + \sqrt{\phi})$ ,  $B = 4 + 2\sqrt{\phi}$ , and  $\phi \leq 1$  still hold for the new value of  $n$ . A detailed description is in Section A.5 of the online material.

## VII. EVALUATION

We examine StreamingTag with comprehensive experiments. The preprocessing pipeline is prototyped on top of *FFmpeg* [34], with the proposed watermarking technique implemented as an *FFmpeg* filter. The evaluation consists of three aspects: (a) the impact on QoS via metrics including latency, bandwidth, and fidelity; (b) the robustness against the NR attack; and (c) the resistance against the CR attack.

### A. Experimental Setup

We evaluate StreamingTag using nine videos, whose key attributes are presented in the upper part of Table II. All videos are at 25 frames per second (fps). The processing latency is measured on a desktop computer with an AMD Ryzen 9 5950X processor and 128GiB memory. The resistance against screen recording-based NR attack is evaluated using a Xiaomi 12S smartphone.

We set the watermark strength  $\alpha$  to 0.2. The size of each embedding block is set to 512. We employ the H.264 encoder to encode the watermarked video. The H.264 encoder's keyint and min-keyint parameters are both set to 25, resulting in a 1-second length for each video segment. We set other H.264 parameters as follows: (1) bframes set to 0; (2) rc-lookahead set to 0; (3) preset set to superfast; (4) tune set to zerolatency; and (5) profile:v set to high.

### B. Evaluation of QoS

We set  $n_e$  to 1 and use the following three metrics to evaluate the *QoS*:

- *Processing latency* measures the running performance of the watermarking algorithm, which is the indicator to determine if the algorithm can run in real-time.
- *Video Size* affects QoS as it affects bandwidth consumption.
- *Video fidelity* measures how invisible the watermark is, indicated by two factors, i.e., peak signal-to-noise ratio (PSNR) and structural similarity index (SSIM) [45]. PSNR is determined by the mean square difference, while SSIM better replicates the behavior of the human visual system.

*Processing Latency.* We measure the processing latency to assess the cost introduced by StreamingTag. Specifically, we first measure the duration of generating all watermarked variants from the test video (including decoding, watermark embedding, and re-encoding). Then, we determine how much CPU time [46] our watermarking algorithm requires to embed a single bit into (or extract a single bit from) a single embedding region.

The middle part of Table II summarizes the findings, from which we can deduce that: (a) The rate of processing the entire video is significantly higher than 25 fps, verifying that StreamingTag is suitable for real-time streaming services; (b) Watermarking is quite lightweight as it takes only several milliseconds to process a single region.

*Video Size.* A poorly designed watermarking scheme may reduce redundancy and thus increase the size of the compressed video. To show StreamingTag doesn't suffer from this issue, we compare the average size of watermarked and unwatermarked segments. To generate unwatermarked segments, we transcode the video into consecutive segments, keeping all H.264 encoder parameters the same as generating watermarked segments for a fair comparison. As shown in the last row of Table II, the video size increases are less than 1%, confirming StreamingTag doesn't consume additional bandwidth.

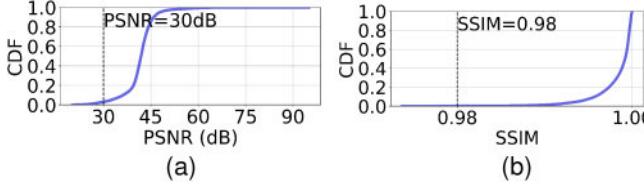


Fig. 6. CDF of PSNR and SSIM. (a) CDF of PSNR. (b) CDF of SSIM.

**Video Fidelity.** A poorly designed watermarking scheme can produce visible artifacts and degrade the QoS. We measure the fidelity of the watermarked video to validate if such a problem occurs in StreamingTag. We generate watermarked and unwatermarked segments in the same way as we measure the video size. PSNR and SSIM are computed between the embedding regions in the watermarked segments and the corresponding regions within the unwatermarked segments. As shown in Fig. 6, StreamingTag does not sacrifice video fidelity, since a PSNR higher than 30 dB (or an SSIM higher than 0.98) is considered good.

### C. Security Under the NR Attack

**Transcoding-Based NR Attack.** We consider the first case of the NR attack, where the watermarked copies are directly downloaded and then re-distributed. Further modification can be applied before re-encoding and re-distribution. We do not use slidingDTW for this experiment as no frame dropping or duplication occurs. We compare our watermarking scheme with three alternatives, using the same embedding regions and the same value of  $n_e$  (=1) for fairness. The first alternative is referred to as bipolar DWT watermarking. Its embedding part is the same as that of our strategy, but it extracts the watermark by calculating  $\frac{\text{sum}(\text{abs}(DWT(r_{wm}).HL_3)))}{\text{sum}(\text{abs}(DWT(r_{adj}).HL_3)))}$ , where  $\text{sum}$  computes the sum of all elements in a matrix,  $\text{abs}$  computes the element-wise absolute values of a matrix,  $DWT(r)$ . $HL_3$  performs 3-level DWT and returns the  $HL_3$  frequency sub-band of  $r$ . The purpose of this comparison is to demonstrate the necessity of explicit SVD for robust watermark extraction. The second alternative is the baseline described in Section II-D. We use two watermark images to represent bit 0 and bit 1 respectively, as discussed in Section V-A. The third alternative embeds the watermark in the DCT domain [38].

We evaluate their performances on all 9 test videos. In addition to the basic case where the attackers perform no modification on the H.264-encoded video, we mimic the attacker's behavior by decoding the watermarked and encoded video, modifying (through Gaussian noising of  $\sigma = 10$ , scaling with a factor of 0.75 or 1.25, or  $15 \times 15$  median filtering), and then re-encoding (with H.264 again) for re-distribution. The modification is conducted by transcoding the watermarked video with built-in FFmpeg filters, i.e., *scale*, *noise*, and *median* filters. To extract the watermark from scaled videos, we first decode them and then re-scale them to the original resolution.

The results are shown in Fig. 7, from which we can conclude that: (a) Both our scheme and bipolar DWT watermarking

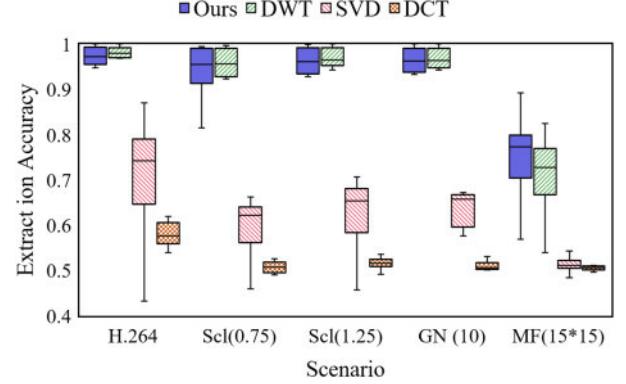


Fig. 7. The extraction accuracy of different watermarking schemes under different attacks.

TABLE III  
SYSTEM PERFORMANCE UNDER THE SCREEN RECORDING-BASED NR ATTACK

Video ID	1 <sup>a</sup>	2 <sup>a</sup>	3 <sup>b</sup>	4 <sup>c</sup>	5 <sup>c</sup>	6 <sup>c</sup>	7 <sup>c</sup>	8 <sup>c</sup>	9 <sup>c</sup>
Useful Segments (%)	82.3	90.4	83.3	90.9	76.7	85.0	83.7	85.3	78.9
Accuracy (%)	79.8	84.7	83.0	66.9	76.2	82.6	84.5	72.9	83.2

achieve higher robustness than the other two alternatives, validating the effectiveness of polarized watermarking. (b) Although bipolar DWT watermarking achieves almost the same level of accuracy as our scheme under most attacks, it drops behind under very strong attacks (i.e., H.264 plus  $15 \times 15$  Median Filter plus H.264), proving that semi-explicit SVD contributes to accurate extraction. (c) Our scheme has an accuracy greater than 90% under most scenarios, realizing an improvement of 2.25x at most and 1.5x on average when compared with traditional SVD watermarking. It can be further improved with a larger  $n_e$ .

**Screen Recording-Based NR Attack.** In this experiment, we use the same test videos and parameters as in Section VII-A, except that  $n_e$  is set to 3 to alleviate the effect of frame dropping. We play the watermarked videos and run the built-in screen recorder to generate pirated copies. The pirated copies are first synchronized with the original videos via slidingDTW. Then, the watermark is extracted from the re-identified host frames. We regard segments containing at least one re-identified host frame as useful segments. As shown in Table III, the ratio of useful segments is higher than 80% for most videos, and the extraction accuracy is around 80% for most videos. The results showcase the strong robustness of our slidingDTW algorithm and semi-explicit SVD watermarking algorithm.

### D. Security Under the CR Attack

**Security under the CR attack.** To demonstrate the superiority of our parameter selection strategy against the CR attack model, we compare it with the strategy in the original Tardos scheme [41] and that in the symmetric Tardos scheme [42]. We consider a video of 7,200 useful segments and assume an extraction accuracy of 90%. We set  $n$  to 100,000 and range  $c$  from 1 to 10. The colluders' attack strategy conforms to the CR model. For each  $c$  and each strategy, the experiment is conducted 100 times.

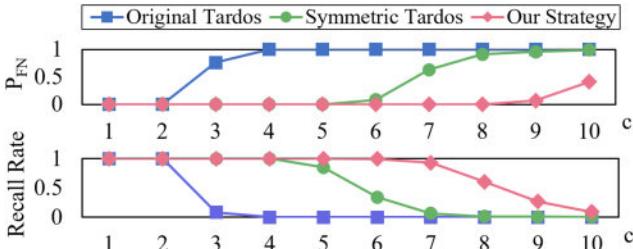


Fig. 8. Collusion resistance.

For our method, we use the parameter selection strategy described in Section VI-C. For the two alternatives, we set  $m$  to 7,200 and  $c_0$  to the maximum number of colluders which can be defended with a false positive rate of  $\eta < 0.1$ . The value of  $t$  is then determined based on  $c_0$ , as required by [41], [42] (e.g.,  $t$  is set to  $\frac{1}{300c_0}$  in [41]).

The results are shown in Fig. 8. Since  $P_{FP}$  equals to 0 in every experiment, we do not present it in this figure. We observe that our method defends more colluders under the CR attack. On average, our strategy improves the recall rate (i.e., the ratio of the number of accused colluders to the number of participating colluders) by 26% when compared with [42], and by 58% when compared with [41].

## VIII. DISCUSSION

*Working with adaptive bitrate streaming.* To adapt StreamingTag to adaptive bitrate streaming, every bit only needs to be embedded once for multiple transcoding bitrates/resolutions for the following two reasons: 1) the watermark embedding process is performed before the encoding process in StreamingTag; 2) the proposed watermarking algorithm is robust to scaling (see Section VII-C).

*Reducing the storage cost.* StreamingTag doubles the video storage costs of the origin server and the CDN as two versions are generated for each segment. Such a heavy overhead can be optimized as the two differently watermarked variants only differ in a few host frames. One possible optimization strategy is to resort to the temporal scalability technique [47] in scalable video coding, where frames are temporally partitioned into distinct layers. Specifically, the base layer, which serves as the foundation, is coded independently of the enhancement layer. We suggest using only frames in the enhancement layer as the host frames and treating all frames in the base layer as the non-host frames. This significantly reduces the storage overhead by limiting duplication to the enhancement layer and maintaining only a single version of the base layer. Yet, the CDN and the client player must be customized so that: 1) at least one version of the enhancement layer is downloaded and decoded; 2) the users cannot figure out which frames are in the enhancement layer; otherwise, they can simply discard the enhancement layer containing all the host frames.

*Fine-grained collusion.* In the CR attack model, we assume that the colluders intermix their copies in a segment-level granularity. However, the colluders may choose to intermix their copies frame by frame so that both two bits (0 and 1) can be

detected in a single segment. StreamingTag can be extended to handle this fine-grained collusion attack. Specifically, StreamingTag can embed a  $q$ -ary bit into each segment, i.e., generating  $q$  different watermarked variants for each segment. By choosing a  $q$  larger than  $n_e$  (the number of host frames), the fine-grained frame-level collusion attack will expose more colluders while not introducing all the  $q$  bits into the pirated segment. As demonstrated in [48], this will lead to a higher risk for colluders, while innocent users are still unlikely to be falsely accused.

*Deep learning-based attack.* Some researchers propose to remove the watermark through neural networks [49], [50]. A possible countermeasure is to use deep adversarial learning, while its overhead must be reduced for large-scale streaming.

*Camcorder recording.* We regard camcorder recording as thornier than software-based recording, since the camcorder may be frequently shaken due to unsteady hands. Consequently, the pirated video may experience translation and rotation. We expect to enforce automatic alignment in the future by using methods such as KAZE feature [38].

*Collusion strategy.* Colluders may employ a collusion strategy different from that in the CR model. One countermeasure is to orthogonally use the proposed fingerprinting code and other codes that resists other collusion strategies. In other words, multiple independent fingerprints are generated for each user and simultaneously embedded into the copy. Another measure is to use  $q$ -ary ( $q > 2$ ) bits to construct fingerprints. The fingerprinting scheme can be designed to be more universal with a larger code space.

*Distribution topologies.* As different streaming platforms may employ different watermarking and fingerprinting technologies, attackers could evade the accusation by fetching different parts of a video from different platforms. Mutual trust among streaming platforms must be established to promote cooperation and respond to this piracy strategy. To address mistrust between service providers and users in piracy tracing for peer-to-peer video distribution, researchers typically resort to a trusted third-party [51], [52] or adopt the blockchain technology [53]. Similar strategies should also be used to maintain trust among platforms.

## IX. RELATED WORK

*DRM.* DRM protects digital content by encrypting it before distribution and requiring users to purchase digital licenses containing the decryption key [3]. However, as discussed in Section I, DRM has several inherent drawbacks.

*Anti-Camcording Technologies.* Anti-camcording technologies are mainly based on the difference between the HVS and the common digital cameras. For example, LiShield [12] deployed LEDs flickering at high frequency and in specialized waveforms, which are imperceptible to the HVS but not to the common digital cameras. Considering that the screen refresh rate can be higher than the video frame rate, KALEIDO [11] re-encoded each video frame into multiple specially designed frames that only looked normal to the HVS. Nevertheless, they require client-side modifications and cannot prevent software-based recorders.

**Collusion-Resistant Forensic Fingerprinting.** Many earlier works are aimed at deterministic fingerprinting strategies, where an error rate of 0 must be guaranteed. Hollmann et al. [54] proposed such a deterministic fingerprinting approach that requires  $c_0 \leq 2$ . Staddon et al. [55] proposed a deterministic fingerprinting strategy, which works under any possible value of  $c_0$ . However, it unrealistically requires the alphabet size  $q$  (i.e., the number of variants for each video segment) is greater than or equal to  $n - 1$ . Due to these inherent limitations [56], recent work has concentrated on non-deterministic fingerprinting, which allows for non-zero  $P_{FP}$  and  $P_{FN}$ . There exist two major categories of non-deterministic fingerprinting schemes, i.e., the Boneh-Shaw scheme [40] and the Tardos scheme [41], [42], [43]. Because the Tardos scheme's code length is nearly equal to the square root of the Boneh-Shaw scheme [41], we base our fingerprinting approach on the Tardos scheme.

## X. CONCLUSION

In this work, we propose StreamingTag, a scalable piracy tracing framework for streaming services. To ensure interoperability with CDNs, StreamingTag embeds fingerprints at the segment level. A novel slidingDTW algorithm is used to pinpoint watermarked frames in the pirated video, and a polarized, semi-explicit, and randomized SVD watermarking algorithm is incorporated for robust watermarking. Additionally, a collusion-resistant code is used to give collusion resistance. The evaluation findings indicate that StreamingTag provides an acceptable level of service and considerably outperforms previous techniques. StreamingTag, we believe, is a significant step towards scalable, reliable, and traceable video streaming.

## REFERENCES

- [1] Kepios Pte. Ltd., “Digital 2022 global overview report,” 2022. Accessed: Sep. 5, 2022. [Online]. Available: <https://datareportal.com/reports/digital-2022-global-overview-report>
- [2] Forbes, “Pirated video gets viewed over 200 billion times A year [infographic],” 2019. Accessed: Sep. 5, 2022. [Online]. Available: <https://www.forbes.com/sites/niallmccarthy/2019/06/26/pirated-video-gets-viewed-over-200-billion-times-a-year-infographic/>
- [3] S. Subramanya and B. Yi, “Digital rights management,” *IEEE Potentials*, vol. 25, no. 2, pp. 31–34, Mar./Apr. 2006.
- [4] Google, Inc., “Widevine,” 2022. Accessed: Sep. 5, 2022. [Online]. Available: <https://widevine.com/>
- [5] Inc Apple., “FairPlay streaming - Apple developer,” 2022. Accessed: Sep. 5, 2022. [Online]. Available: <https://developer.apple.com/streaming/fps/>
- [6] Inc Microsoft., “Microsoft PlayReady - Home,” 2022. Accessed: Sep. 5, 2022. [Online]. Available: <https://www.microsoft.com/playready/>
- [7] Z. Yan, X. Qian, S. Liu, and R. Deng, “Privacy protection in 5G positioning and location-based services based on SGX,” *ACM Trans. Sensor Netw.*, vol. 18, no. 3, 2022, Art. no. 41.
- [8] S. L. Kinney, *Trusted Platform Module Basics: Using TPM in Embedded Systems*. Oxford, U.K.: Newnes, 2006.
- [9] S. Crosby, I. Goldberg, R. Johnson, D. Song, and D. Wagner, “A cryptanalysis of the high-bandwidth digital content protection system,” in *Proc. Secur. Privacy Digit. Rights Manage.*, 2002, pp. 192–200.
- [10] Krebs on Security, “Google mending another crack in widevine,” 2020. Accessed: Sep. 5, 2022. [Online]. Available: <https://krebsonsecurity.com/2020/10/google-mending-another-crack-in-widevine/>
- [11] L. Zhang et al., “Kaleido: You can watch it but cannot record it,” in *Proc. 21st ACM Annu. Int. Conf. Mobile Comput. Netw.*, 2015, pp. 372–385.
- [12] S. Zhu, C. Zhang, and X. Zhang, “Automating visual privacy protection using a smart LED,” in *Proc. 23rd ACM Annu. Int. Conf. Mobile Comput. Netw.*, 2017, pp. 329–342.
- [13] J. Looney, “Netflix: Streaming entertainment to 200 million members around the world,” *USENIX FAST '21*, Feb. 2021.
- [14] S. Salvador and P. Chan, “Toward accurate dynamic time warping in linear time and space,” *Intell. Data Anal.*, vol. 11, pp. 561–580, 2007.
- [15] X. Jin et al., “StreamingTag: A scalable piracy tracking solution for mobile streaming services,” in *Proc. 28th ACM Annu. Int. Conf. Mobile Comput. Netw.*, 2022, pp. 596–608.
- [16] R. Pantos, “HTTP live streaming 2nd edition,” 2022. Accessed: Sep. 5, 2022. [Online]. Available: <https://datatracker.ietf.org/doc/html/draft-pantos-hls-rfc8216bis>
- [17] I. Sodaig, “Media presentation description and segment formats | MPEG,” 2011. Accessed: Sep. 5, 2022. [Online]. Available: <https://mpeg.chiariglione.org/standards/mpeg-dash/media-presentation-description-and-segment-formats>
- [18] A. Chankhunthod, P. B. Danzig, C. Neerdaels, M. F. Schwartz, and K. J. Worrell, “A hierarchical internet object cache,” in *Proc. 2nd Annu. Conf. USENIX Annu. Tech. Conf.*, 1996, Art. no. 13.
- [19] E. Ghabashneh and S. Rao, “Exploring the interplay between CDN caching and video streaming performance,” in *Proc. 39th IEEE Conf. Comput. Commun.*, 2020, pp. 516–525.
- [20] W. Ren, Y. Xu, L. Zhai, L. Wang, and J. Jia, “Fast carrier selection of JPEG steganography appropriate for application,” *Tsinghua Sci. Technol.*, vol. 25, no. 5, pp. 614–624, 2020.
- [21] L. Boney, A. Tewfik, and K. Hamdy, “Digital watermarks for audio signals,” in *Proc. 3rd IEEE Int. Conf. Multimedia Comput. Syst.*, 1996, pp. 473–480.
- [22] J. Sun, X. Jiang, J. Liu, F. Zhang, and C. Li, “An anti-recompression video watermarking algorithm in bitstream domain,” *Tsinghua Sci. Technol.*, vol. 26, no. 2, pp. 154–162, 2021.
- [23] P. W. Chan, M. Lyu, and R. Chin, “A novel scheme for hybrid digital video watermarking: Approach, evaluation and experimentation,” *IEEE Trans. Circuits Syst. Video Technol.*, vol. 15, no. 12, pp. 1638–1649, Dec. 2005.
- [24] M. Ejima and A. Miyazaki, “A wavelet-based watermarking for digital images and video,” *IEICE Trans. Fundam. Electron. Commun. Comput. Sci.*, vol. 83, no. 3, pp. 532–540, 2000.
- [25] G. Doerr and J.-L. Dugelay, “Secure background watermarking based on video mosaicing,” in *Proc. Secur. Steganogr. Watermarking Multimedia Contents VI*, 2004, pp. 304–314.
- [26] Q. Liu, S. Yang, J. Liu, P. Xiong, and M. Zhou, “A discrete wavelet transform and singular value decomposition-based digital video watermark method,” *Appl. Math. Model.*, vol. 85, pp. 273–293, 2020.
- [27] R. Singh, H. Mittal, and R. Pal, “Optimal keyframe selection-based lossless video-watermarking technique using IGSAs in LWT domain for copyright protection,” *Complex Intell. Syst.*, vol. 8, pp. 1047–1070, 2022.
- [28] Canonical Ltd., “Ubuntu manpage: FFplay - FFplay media player,” 2019. Accessed: Apr. 3, 2023. [Online]. Available: <https://manpages.ubuntu.com/manpages/xenial/man1/ffplay-all.1.html>
- [29] X. Yu, C. Wang, and X. Zhou, “A survey on robust video watermarking algorithms for copyright protection,” *Appl. Sci.*, vol. 8, no. 10, pp. 1–26, 2018.
- [30] K. Meenakshi, K. Swaraja, and P. Kora, “A robust DCT-SVD based video watermarking using zigzag scanning,” in *Proc. Soft Comput. Signal Process.*, 2019, pp. 477–485.
- [31] S. Ponni Alias Sathya and S. Ramakrishnan, “Fibonacci based key frame selection and scrambling for video watermarking in DWT-SVD domain,” *Wirel. Pers. Commun.*, vol. 102, no. 2, pp. 2011–2031, 2018.
- [32] A. Kunhu, N. K. S. Sabnam, M. A., and S. AL-Mansoori, “Index mapping based hybrid DWT-DCT watermarking technique for copyright protection of videos files,” in *Proc. 2nd IEEE Int. Conf. Green Eng. Technol.*, 2016, pp. 1–6.
- [33] R. C. Gonzalez and R. E. Woods, *Digital Image Processing*. London, U.K.: Pearson Education International, 2007, ch. 7.5, pp. 523–532.
- [34] Anon, “FFmpeg,” 2022. Accessed: Sep. 5, 2022. [Online]. Available: <https://ffmpeg.org/>
- [35] L. Muda, M. Begam, and I. Elamvazuthi, “Voice recognition algorithms using mel frequency cepstral coefficient (MFCC) and dynamic time warping (DTW) techniques,” 2010, *arXiv:1003.4083*.
- [36] P. Schneider, R. Memmesheimer, I. Kramer, and D. Paulus, “Gesture recognition in RGB videos using human body keypoints and dynamic time warping,” in *Proc. Robot World Cup XXIII*, 2019, pp. 281–293.
- [37] M. Li and V. Monga, “Twofold video hashing with automatic synchronization,” *IEEE Trans. Inf. Forensics Secur.*, vol. 10, no. 8, pp. 1727–1738, Aug. 2015.
- [38] T. M. Thanh, P. T. Hiеп, T. M. Tam, and K. Tanaka, “Robust semi-blind video watermarking based on frame-patch matching,” *AEU - Int. J. Electron. Commun.*, vol. 68, no. 10, pp. 1007–1015, 2014.

- [39] L. N. Trefethen and D. Bau III, *Numerical Linear Algebra*. Philadelphia, PA, USA: SIAM, 1997.
- [40] D. Boneh and J. Shaw, "Collusion-secure fingerprinting for digital data," *IEEE Trans. Inform. Theory*, vol. 44, no. 5, pp. 1897–1905, Sep. 1998.
- [41] G. Tardos, "Optimal probabilistic fingerprint codes," *J. ACM*, vol. 55, no. 2, 2008, Art. no. 10.
- [42] B. Škorić, S. Katzenbeisser, and M. U. Celik, "Symmetric tardos fingerprinting codes for arbitrary alphabet sizes," *Des. Codes Cryptogr.*, vol. 46, no. 2, pp. 137–166, 2008.
- [43] B. Škorić, T. U. Vladimirova, M. Celik, and J. C. Talstra, "Tardos fingerprinting is better than we thought," *IEEE Trans. Inf. Theory*, vol. 54, no. 8, pp. 3663–3676, Aug. 2008.
- [44] T. Laarhoven, J. Doumen, P. Roelse, B. Škorić, and B. de Weger, "Dynamic tardos traitor tracing schemes," *IEEE Trans. Inf. Theory*, vol. 59, no. 7, pp. 4230–4242, Jul. 2013.
- [45] Z. Wang, A. Bovik, H. Sheikh, and E. Simoncelli, "Image quality assessment: From error visibility to structural similarity," *IEEE Trans. Image Process.*, vol. 13, no. 4, pp. 600–612, Apr. 2004.
- [46] M. Kerrisk, "time(7) – Linux manual page," 2020. Accessed: Sep. 5, 2022. [Online]. Available: <https://man7.org/linux/man-pages/man7/time.7.html>
- [47] H. Schwarz, D. Marpe, and T. Wiegand, "Overview of the scalable video coding extension of the H.264/AVC standard," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 17, no. 9, pp. 1103–1120, Sep. 2007.
- [48] F. Xie, T. Furou, and C. Fontaine, "On-off keying modulation and tardos fingerprinting," in *Proc. 10th ACM Workshop Multimedia Secur.*, New York, NY, USA, 2008, pp. 101–106, doi: [10.1145/1411328.1411347](https://doi.org/10.1145/1411328.1411347).
- [49] M. W. Hatoum, J.-F. Couchot, R. Couturier, and R. Darazi, "Using deep learning for image watermarking attack," *Signal Process.: Image Commun.*, vol. 90, 2021, Art. no. 116019.
- [50] Q. Li et al., "Concealed attack for robust watermarking based on generative model and perceptual loss," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 22, no. 8, pp. 5695–5706, Aug. 2022.
- [51] A. Qureshi, H. Rifà Pous, and D. Megías Jiménez, "Secure and anonymous multimedia content distribution in peer-to-peer networks," in *Proc. 6th IARIA Int. Conf. Adv. Multimedia*, 2014, pp. 91–96.
- [52] A. Qureshi, D. Megías, and H. Rifà-Pous, "Framework for preserving security and privacy in peer-to-peer content distribution systems," *Expert Syst. Appl.*, vol. 42, no. 3, pp. 1391–1408, 2015.
- [53] Y. Xu, C. Zhang, Q. Zeng, G. Wang, J. Ren, and Y. Zhang, "Blockchain-enabled accountability mechanism against information leakage in vertical industry services," *IEEE Trans. Netw. Sci. Eng.*, vol. 8, no. 2, pp. 1202–1213, Second Quarter, 2021.
- [54] H. D. Hollmann, J. H. van Lint, J.-P. Linnartz, and L. M. Tolhuizen, "On codes with the identifiable parent property," *J. Comb. Theory - A*, vol. 82, no. 2, pp. 121–133, 1998.
- [55] J. Staddon, D. Stinson, and R. Wei, "Combinatorial properties of frameproof and traceability codes," *IEEE Trans. Inf. Theory*, vol. 47, no. 3, pp. 1042–1049, Mar. 2001.
- [56] A. Fiat and T. Tassa, "Dynamic traitor tracing," in *Proc. 19th IACR Annu. Int. Cryptol. Conf.*, 1999, pp. 354–371.



**Fan Dang** (Member, IEEE) received the BE and PhD degrees in software engineering from Tsinghua University, Beijing, in 2013 and 2018, respectively. He is a research assistant professor with the Global Innovation Exchange, Tsinghua University, Beijing. He is a member of ACM. His research interests include the industrial intelligence and edge computing.



**Xinqi Jin** received the BE degree in software engineering from Tsinghua University, in 2021. He is currently working toward the PhD degree with the School of Software, Tsinghua University. His research interests include AIoT, video streaming, and edge computing.



**Qi-An Fu** received the BS degree in physics from Tsinghua University, in 2019, and the MS degree in computer science from Tsinghua University, in 2022.



**Lingkun Li** received the PhD degree in computer science from Michigan State University. He is currently a lecturer with Beijing Jiaotong University. His research interests include mobile computing, operating systems, and Internet of Things.



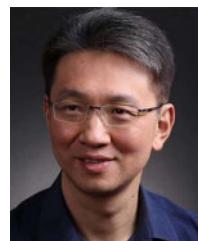
**Guanyan Peng** received the BE degree in software engineering from Zhejiang University, in 2020, and the MS degree in software engineering from Tsinghua University, in 2023. She is currently a research engineer with ByteDance Inc. Her research interests include video streaming and user experience optimization.



**Xinlei Chen** (Member, IEEE) received the BE and MS degrees in electronic engineering from Tsinghua University, China, in 2009 and 2012, respectively, and the PhD degrees in electrical engineering from Carnegie Mellon University. He is an assistant professor with Shenzhen International Graduate School, Tsinghua University. He worked as a postdoctoral research associate with Electrical Engineering Department, Carnegie Mellon University from 2018–2020. His research interests lie in AIoT, pervasive computing, cyber physical system, etc.



**Kebin Liu** (Senior Member, IEEE) received the MS and PhD degrees from Shanghai Jiaotong University, China. He is a research associate professor with Global Innovation Exchange, Tsinghua University, Beijing, China. His research interests include Internet of Things, machine learning, and fault diagnosis.



**Yunhao Liu** (Fellow, IEEE) received the BE degree from the Department of Automation, Tsinghua University, Beijing, in 1995, the MA degree from Beijing Foreign Studies University, Beijing, in 1997, and the MS and PhD degrees in computer science and engineering from Michigan State University, East Lansing, in 2003 and 2004, respectively. He is a professor with the Department of Automation and dean of the Global Innovation Exchange, Tsinghua University, Beijing. He is a fellow of ACM. He is the editor-in-chief of the *Communications of the CCF*. His research interests include the Internet of Things, wireless sensor networks, indoor localization, the industrial Internet, and cloud computing.