StreamingTag: A Scalable Piracy Tracking Solution for Mobile Streaming Services

Xinqi Jin*
School of Software,
Tsinghua University
jinxq21@mails.tsinghua.edu.cn

Fan Dang*†
Global Innovation Exchange, Tsinghua University
dangfan@tsinghua.edu.cn

Qi-An Fu
Department of Computer Science and Technology,
Tsinghua University
fqa19@mails.tsinghua.edu.cn

Lingkun Li
School of Software,
Beijing Jiaotong University
lkli@bjtu.edu.cn

Guanyan Peng
School of Software,
Tsinghua University
pgy20@mails.tsinghua.edu.cn

Xinlei Chen
Shenzhen International Graduate School, Tsinghua University
Peng Cheng Laboratory
chen.xinlei@sz.tsinghua.edu.cn

Kebin Liu
Global Innovation Exchange, Tsinghua University
kebinliu2021@tsinghua.edu.cn

Yunhao Liu†
Global Innovation Exchange & Department of Automation,
Tsinghua University
yunhao@tsinghua.edu.cn

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, the idea of piracy tracing is not widely used at the moment, since existing fingerprinting-based streaming systems 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, a polarized and randomized SVD watermarking scheme suitable for short segments, and a collusion-resistant fingerprinting scheme optimized for large-scale streaming services. Experiment results show the good QoS of StreamingTag in terms of preparation latency, bandwidth consumption, and video fidelity. Compared with existing SVD watermarking schemes, the proposed watermarking scheme improves the watermark extraction accuracy by 2.25x at most and 1.5x on average. Compared with existing collusion-resistant fingerprinting schemes, the proposed scheme catches more colluders and improves the recall rate by 26%.

*Co-primary authors.
†Co-corresponding authors.

ACM MobiCom '22, October 17–21, 2022, Sydney, NSW, Australia
© 2022 Copyright held by the owner/author(s). Publication rights licensed to ACM.
ACM ISBN 978-1-4503-9181-8/22/10…$15.00
https://doi.org/10.1145/3495243.3560521

CCS Concepts

• Security and privacy → Web application security; **• Applied computing → Evidence collection, storage and analysis**.

Keywords

Copyright Protection, Video Streaming, Piracy Tracing, Collusion Resistance

ACM Reference Format:
Xinqi Jin, Fan Dang, Qi-An Fu, Lingkun Li, Guanyan Peng, Xinlei Chen, Kebin Liu, and Yunhao Liu. 2022. StreamingTag: A Scalable Piracy Tracking Solution for Mobile Streaming Services. In *The 28th Annual International Conference on Mobile Computing and Networking (ACM MobiCom '22), October 17–21, 2022, Sydney, NSW, Australia.* ACM, New York, NY, USA, 13 pages. https://doi.org/10.1145/3495243.3560521

1 Introduction

Streaming services, including live streaming services, are regarded as one of the most essential mobile applications. Mobile users tend to devote a considerable amount of time to these applications. On average, a YouTube user spends 23.7 hours per month, and a TikTok user also spends 19.6 hours per month [16]. Copyright is crucial for content providers in such a large market. Hundreds of thousands of jobs and tens of billions of dollars in GDP are reported to be lost every year in the United States due to video piracy [8].

Numerous anti-piracy strategies have been extensively studied in academia and industry in order to protect the copyright of streaming videos. Among these techniques, the most widely used one is Digital Rights Management (DRM) [34]. The video streaming services provided by major providers such as Netflix, Hulu, Amazon, Apple, are all protected by a variety of DRM technologies including Widevine [11], FairPlay [2], and PlayReady [25]. Regrettably, DRM is not a panacea. To achieve a high level of security, specialized hardware like Trusted Execution Environment (TEE) [42], Trusted Platform Module (TPM) [18], or High-Bandwidth Digital Content Protection (HDCP) [6] is required on mobile terminals; otherwise, software-only DRM might be compromised [19]. Even with top-level DRM, there are still instances where these technologies cannot prevent content leakage on their own, such as screen recording with

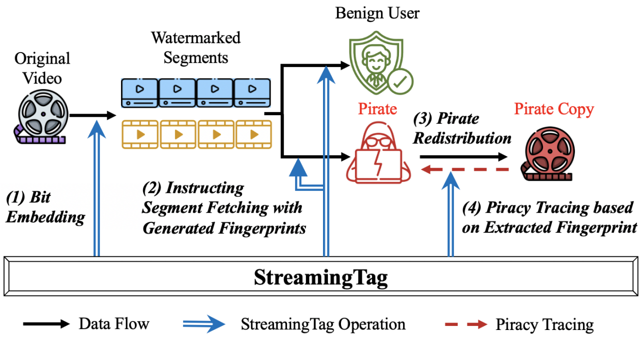


Figure 1: The overview of StreamingTag.

a camcorder. Although recent efforts [44, 45] have been made to prevent recording by exploiting differences between the human visual system and cameras, they either require complex video processing on the client side [44] or attempt to alter the physical environment (*e.g.,* installing an LED flickering at a high frequency [45]), thereby reducing their applicability to mobile streaming. 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 are able to embed unique information (or more precisely, a fingerprint) into the content provided to different users. Once an illegal copy is distributed, the distributor's fingerprint (*i.e.,* the pirate) can be extracted and tracked. Although this concept appears to be quite intuitive and effective, it is not as widely used as DRM at the moment. To make the large-scale deployment of such a framework feasible, no modifications should be made to mobile terminals. We identify the primary impediment to successfully implementing fingerprint-based piracy tracking is that **embedding fingerprints on the server side is not scalable and thus not suitable for large-scale streaming services**. If we use existing fingerprinting-based piracy tracing systems [13, 41] to serve a video to N users, N copies of the video will be generated with N unique fingerprints. Such frameworks are ineffective for a service provider with millions of users due to their high computation and storage overhead. In fact, these 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, we only encode a single bit in each frame and prepare two versions of each frame representing bit '0' and bit '1'. Then we can distribute different series of frames to different users with their identities (*e.g.,* user ids) embedded. Unfortunately, applying this naïve approach has a number of challenges. **(1) A framework compatible with nowadays' mobile streaming infrastructures is required.** Today's large-scale streaming services rely on caching-based Content Delivery Networks (CDNs) to ensure a consistent level of quality of service (QoS) [15, 23]. However, existing watermarking techniques are mostly designed to compute offline, *i.e.,* pre-processing the entire image or video, and so are not suitable for real-time streaming services. **(2) A more lightweight, more robust, yet still imperceptible embedding technique is needed.** Today's singular value decomposition (SVD) watermarking algorithms are the most robust of all original video-based watermarking algorithms, yet they rely on heavy SVD operations. To serve numerous requests, the computation complexity should be reduced to as little as possible. Besides, conventional watermarking algorithms intended for embedding a relatively large amount of data and defending against single attacks are insufficiently resilient against popular streaming pipelines. We must further improve their robustness without significantly changing the visual appearance of video frames. **(3) The *collusion attack* must be considered.** Attackers might collaborate to carry out an effective collusion attack, in which they *mix* multiple copies and compromise the integrity of the embedded fingerprint, making tracking them difficult or impossible. As a result, we must develop a fingerprinting strategy capable of withstanding collusion attacks and identifying potential attackers.

In this paper, we present StreamingTag, a novel piracy tracking solution intended for streaming services. As shown in Fig. 1, StreamingTag can trace back the source of piracy by delivering different copies to different users. 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 user's fingerprint is 0; otherwise the second variant is fetched). 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. With regards to the second 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 numerous randomly-localized blocks. For the third 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 deliver differently fingerprinted copies to different users, which is scalable, CDN-friendly, and compatible with the existing video delivery infrastructure.
- We propose a bipolar and randomized watermarking scheme based on SVD that fits well with the proposed delivery pipeline. By leveraging StreamingTag's low-density requirement for watermark capacity, this approach improves extraction accuracy by up to 2.25x compared to SOTA SVD-based watermarking, without incurring any additional costs.
- To defend against collusion attacks, we employ a randomized fingerprinting strategy and carefully select the fingerprinting generation parameters. The experiments demonstrate that our technique is capable of defending against up to nine colluders with a success rate of greater than 80%.

The rest of the paper is organized as follows. In Sec. 2, we briefly review the preliminary knowledge about video streaming, watermarking, and fingerprinting. In Sec. 3, we present two threat models, highlight design challenges, and then overview the design of StreamingTag. The improved watermarking scheme and fingerprinting scheme are discussed in Sec. 4 and 5, respectively. We evaluate the performance of StreamingTag in Sec. 6. A discussion on limitations and possible extensions is conducted in Sec. 7. We review the related work in Sec. 8 and conclude the paper in Sec. 9. **The main symbols are summarized in Appendix A.3.**

2 Background

2.1 Streaming and Live Streaming

HTTP-based streaming protocols, including the HTTP Live Streaming (HLS) protocol [26] and the MPEG Dynamic Adaptive Streaming over HTTP (MPEG-DASH) protocol [32], have been the de facto standard for video streaming services. When serving a video stream using these protocols, the server side needs to split the original video stream into sequential segments. The client first downloads a manifest file containing the URLs of each segment in the requested video stream and then requests each segment.

Typical video data is of larger size when compared to other kinds of data transferred via HTTP, bringing difficulties to large-scale video streaming. To ease this burden, video data is usually served from CDNs [15]. CDNs can be viewed as a hierarchy of caches [5]. In a CDN-enabled video streaming system, a client's request for a segment listed in the manifest is routed to a closely located CDN edge server instead of the origin server. The edge server directly responds to the client if the requested content hits its cache. Otherwise, it tries to fetch the content from a CDN server higher in the hierarchy and ultimately to the origin server, then sends the response and caches the content for future requests [9]. The caching capabilities of CDNs greatly boost content delivery in terms of scalability and latency. Therefore, we should sufficiently utilize such caching capabilities while designing StreamingTag.

2.2 Watermarking and Fingerprinting

Hidden watermarking (a.k.a. digital watermarking) is a relatively new technology that allows for the discrete integration of watermark data into multimedia carriers such as photos [30], music [4], and video [35]. The embedded data can later be identified to trace copyright infringement and ensure the integrity of data.

Unlike digital watermarking, which often embeds the owners' copyright information into the objects, digital fingerprinting ensures that the encoded content is unique for each user. When a manufacturer finds infringement, it can employ fingerprint data to hunt out the source of the unlawful copy and prosecute the pirate, thereby protecting and preventing copyright infringement.

When we use the word *watermarking* in this paper, we mean the process of embedding invisible data into a video. While the term *fingerprinting* refers to the process of encoding a user's identity.

2.3 SVD-Based Video Watermarking

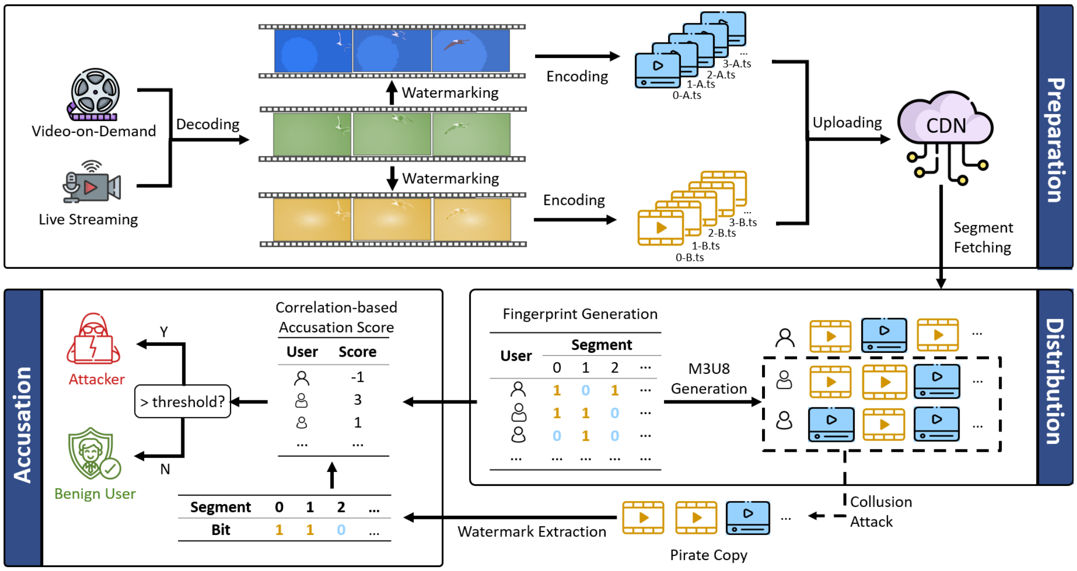
SVD watermarking, a class of original video-based watermarking techniques, modulates the watermark information into the individual values of video frames. Since the singular values reflect the intrinsic rather than the visual characteristics of frames, embedding the watermark into the singular values can achieve both high imperceptibility and robustness [43]. SVD-based watermarking typically consists of two stages, *i.e.,* embedding and extraction. The *embedding stage* consists of the following steps:

- **Step 1**: The input video is decoded to a stream of uncompressed frames. Some frames are selected as embedding regions for watermark embedding. For convenience, we refer to these frames as "host frames" in this paper.
- **Step 2**: Typically, every selected embedding region r is converted to the frequency domain through some transform such as discrete cosine transform (DCT) [24], discrete wavelet transform (DWT) [27], the combination of DCT and DWT [20], *etc.* Then the SVD operation is performed on the sub-band at a certain frequency (denoted as $\mathscr{F}(r)$), *i.e.,* $\mathscr{F}(r) = USV^T$, where S is a diagonal matrix consisting of singular values. Typically, the sub-band at middle or high frequency is used in this step to preserve imperceptibility.
- **Step 3**: A matrix S_{wm} is generated from the watermark image through SVD. Then, S_{wm} is modulated into S through some matrix operation such as matrix addition (*i.e.,* $S'_{wm} = S + \alpha S_{wm}$, where α denotes the watermarking strength).
- **Step 4**: The frequency sub-band of the watermarked region, $\mathscr{F}(r'_{wm})$, is obtained by $\mathscr{F}(r'_{wm}) = U\ S'_{wm}V^T$.
- **Step 5**: The watermarked region r'_{wm} is obtained by transforming $\mathscr{F}(r'_{wm})$ back from the frequency domain to 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 watermarked 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. To avoid the cost of saving r, S is estimated by extracting S_{adj} from the reference region r_{adj}, which is located at the same position within an adjacent frame and resembles r due to temporal redundancy.

To further improve robustness, one common method is to select multiple host frames and repeatedly embed the same watermark into them. The host frames can be randomly selected, *e.g.,* using a random sequence to determine their indices [27]. Denote the probability that an embedded watermark is correctly extracted from a single embedding region as q. For simplicity, we suppose that the extraction correctness from n_e embedding regions (with the same watermark embedded) follows a binomial distribution $Binom(n_e, q)$. Consequently, the overall extraction accuracy is increased to $\sum_{k=\lceil(n_e+1)/2\rceil}^{n_e}\binom{n_e}{k}q^k(1-q)^{n_e-k}$ using a majority vote.

In this paper, we use redundancy-enhanced SVD watermarking as our baseline method. In the **Step 2** of the baseline's embedding stage, the 3-level DWT transformation is used, and the HL_3 sub-band (*i.e.,* $W_{\psi}^{V}(J-3)$ in [10]) is selected as $\mathscr{F}(r)$. The input is video segments, thus the baseline method will be called m times if we want to watermark the entire video consisting of m segments. To improve the fidelity, only the luminance component of an embedding region is used in the baseline and the proposed method.



	Segment			
User	0	1	2	...
	1	0	1	...
	1	1	0	...
	0	1	0	...
...	...	...	...	...

User	Score
	-1
	3
	1
...	...

Segment	0	1	2	...
Bit	1	1	0	...

Figure 2: The workflow of StreamingTag.

3 System Design

3.1 Threat Models

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

Naïve Redistribution (NR) Attack. The attacker simply redistributes the received copy without collaborating with other attackers in this type of attack. To break down any fingerprint embedded in the copy, the attacker can modify the decoded copy prior to re-encoding and redistribution through some common signal processing operations (including scaling, Gaussian noising, and median filter). However, such a modification is visually insignificant in order to preserve the pirated copy's visual quality.

Colluding Redistribution (CR) Attack. Several attackers may collaborate by mixing different copies. 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) Since such a strategy allows colluders to cooperate directly without exchanging received copies with untrusted accomplices, a larger piracy group could be formed by public 'recruitment'. 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.

3.2 Design Challenges

We begin by discussing the NR attack's design challenges. To defend against the NR attack, the system should embed the user fingerprint into the content delivered to any user. However, achieving such a straightforward objective presents two difficulties. (1) CDNs capitalize on the fact that multiple users within a physical region make the same digital content request. By caching the same content for multiple users, the CDN edge servers can deliver the content directly to the user without frequently fetching data from the remotely located origin server. However, fingerprinting-based piracy tracing conflicts with cache-based CDN by nature. As no modification should be applied to mobile terminals for full compatibility, we should generate all fingerprinted copies on the server side and then directly send each unique copy to the corresponding user. As a result, the benefits of using CDNs may be completely negated, and the overall quality of service may suffer significantly. Meanwhile, when n users view the video together, the watermarking algorithm must run n times simultaneously to generate n unique copies, imposing a significant overhead. (2) While SVD-based watermarking techniques are shown to be extremely resistant and universal, they are still insufficient in our scenario. First, to support large-scale streaming services, the cost of SVD operation, which is the key to robust watermarking, must be minimized as much as possible. Second, the watermarking algorithm must provide good imperceptibility and strong robustness (*i.e.,* the watermark should still be recoverable after a chain of attacks consisting of initial encoding at the server side, signal processing conducted by the attacker, and re-encoding for efficient re-distribution) at the same time, which are mutually contradictory by nature.

Even if the above two challenges are overcome, the attacker may collaborate to launch the *Colluding Redistribution Attack*. This takes us to the third issue, which is that the integrity of the embedded fingerprint can be easily broken by a simple CR attack. Rather than producing user fingerprints haphazardly, we must employ a sophisticated fingerprint production and accusation approach.

3.3 System Overview

StreamingTag in a nutshell. The workflow of StreamingTag is shown in Fig. 2. **The preparation stage** is the initial stage.

The input video is decoded and divided into a series of segments of equal duration. Utilizing the proposed watermarking scheme, two variants are generated for each segment, with bit 0 and bit 1 embedded respectively. Following this stage, approximately double-sized video content is generated. **The distribution stage** is the second stage. For any user j requesting the video, a fingerprint code of length m (*i.e.,* X_j in Section 5.1) will be generated.As stated in Section 2.1, content segments are indexed by a manifest file, and the video player works by fetching segments listed in this file. Therefore, during this stage, StreamingTag generates a unique manifest file for the j-th user based on X_j, ensuring that the i-th bit, $X_j^{(i)}$, is embedded into the i-th segment listed in the manifest file. Therefore, different users 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 has been illegally distributed, we must extract the embedded bits segment by segment to obtain the fingerprint. The extracted fingerprint is then used to charge the alleged attackers.

The distribution stage is re-executed for every new viewer of the same video, but the resource-consuming preparation stage only needs to be performed once for the video and 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 real-time streaming services by caching these variants via existing CDNs. To summarize, StreamingTag effectively addresses **the first challenge** by its lightweight and CDN-friendly delivery pipeline.

Bipolar and Randomized SVD watermarking (Sec. 4). We use a novel watermarking scheme in the preparation stage and accusation stage to tackle **the second challenge**. Taking advantage of the loose requirement on watermark capacity, we propose a bipolar and semi-explicit SVD watermarking scheme, which boosts the robustness and efficiency simultaneously. The redundancy-based watermarking enhancement is also slightly modified to adapt to short segments with limited inter-frame variance.

Fingerprint generation and accusation (Sec. 5). To meet **the third challenge** (*i.e.,* the CR attack), we employ a randomized collusion-resistant fingerprint generation and accusation strategy. The fingerprint generation and accusation procedure is conducted independently for each user, preventing attackers from framing innocent users. Each fingerprint is stored in the database. Once a pirated copy is identified, its fingerprint is extracted and compared with stored fingerprints to identify attackers. A theoretical analysis of the model is conducted to provide safety guarantee.

4 Bipolar and Randomized SVD Watermarking

4.1 Robustness of Watermarking

Watermarking technologies can embed invisible data into an image to enable piracy tracing, tampering detection, *etc.* While existing techniques for watermarking original frames 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 a combination of different attacks in both the NR and CR attack models. To be more precise, the attackers are willing to decode the received segment (which is already compressed), then perform malicious operations to remove the embedded watermark, and finally re-encode the modified segment for efficient redistribution. (2) As stated in Section 2.3, the baseline method can utilize redundant embedding to further improve robustness [27, 37]. However, for a rather short video segment, it's highly likely that all the host frames within it are visually 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 within a short segment may always produce the same result, making the assumption of binomial distribution (see Section 2.3) unrealistic and the redundancy-based enhancement useless.

Video ID	Attack				
	H.264	MF (3*3)	H.264+ MF (3*3) + H.264	GN ($\sigma = 5$)	H.264+ GN ($\sigma = 5$) + 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%

Table 1: The watermark extraction accuracy of traditional SVD-based watermarking algorithms under different attacks. 'MF' represents median filter, and 'GN' represents Gaussian noise.

We utilize the baseline SVD-based watermarking introduced in Section 2.3 to quantify the significance of the aforementioned 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.

Two preliminary experiments are conducted with the baseline method. In both experiments a single bit is modulated into every segment, whose duration is fixed to 1 second. The first experiment employs five distinct attacks on the video segments which are watermarked by the baseline method. As shown in Table 1, the third and fifth attacks are composed of a sequence of 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 find out whether redundant embedding really improves robustness. The imposed attack is H.264 encoding, the most applied video coding standard nowadays. The baseline method and an alternative strategy are compared. The alternative strategy first randomly determines the indices of host frames, then randomly selects sub-regions (whose size is 512 × 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

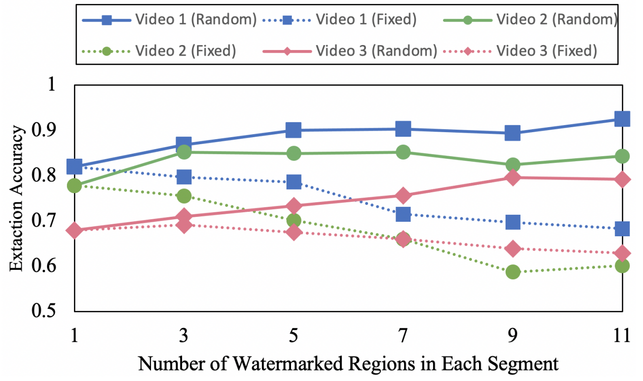


Figure 3: The watermark extraction accuracy of two embedding region selection strategies.

decreased. For the purpose of fair comparison, the baseline method also embeds the watermark into sub-regions (whose size is also 512 × 512) of host frames, but all these sub-regions are at the same spatial position. Fig. 3 illustrates the measurement findings. As the number of embedding regions increases, the 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.

4.2 Watermarking for Streaming Services

As previously stated, existing watermarking methods are ineffective in our circumstance, since the segment is very short and the attack is essentially a combination of multiple discrete attacks. Fortunately, our case provides a lenient restriction on watermark capacity, needing only one bit to be embedded in a segment. Taking advantage of this, we offer two strategies for improving the robustness and computational cost of existing SVD-based watermarking systems. Additionally, we use the results of the second experiment in Section 4.1 to propose a third strategy that is applicable in our circumstances. 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 β via:

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

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. Thus, to embed bit 0 (or bit 1), S_{wm} can be added to (or subtracted from) S, polarizing the two differently watermarked segments in opposite directions and immediately discernible.

Semi-Explicit SVD-based Watermarking Scheme. The embedding process should be as simple as possible to allow for real-time watermarking of numerous live video streams. However, the

Algorithm 1 Watermark Embedding

```
  Input:
    $F = (f_1, f_2, ...)$: frames of a segment
    $n_e$: the number of embedding regions
    $size$: the size of each embedding region
    $\alpha$: the watermark strength
    $\beta \in \{0, 1\}$: the bit to be embedded
  $Regions \leftarrow RandomSelect(F, size, n_e)$
  for $region \in Regions$ do
    $r \leftarrow region.luminance$
    $\mathscr{F}(r) \leftarrow DWT(r).HL_3$
    $\mathscr{F}(r'_{wm}) \leftarrow (1 + (-1)^{\beta}\alpha)\mathscr{F}(r)$
    $r'_{wm} \leftarrow inverseDWT(\mathscr{F}(r'_{wm}))$
    $region.luminance \leftarrow r'_{wm}$
  end for
```

Algorithm 2 Watermark Extraction

```
  Input:
    $F = (f_1, f_2, ...), n_e, size$
  Output:
    the bit extracted from $F$
  $Regions \leftarrow RandomSelect(F, size, n_e)$
  $votes \leftarrow [0, 0]$
  for $region \in Regions$ do
    $r'_{wm} \leftarrow region.luminance$
    $r_{adj} \leftarrow f_{region.index-1}.subRegion(region.position).luminance$
    $S'_{wm} \leftarrow SVD(DWT(r'_{wm}).HL_3)$
    $S_{adj} \leftarrow SVD(DWT(r_{adj}).HL_3)$
    $votes.at(|S'_{wm}| > |S_{adj}|?\ 0 : 1)$ += 1
  end for
  return $votes.at(0) > votes.at(1)?\ 0 : 1$
```

SVD operation on an $C1 \times C2$ matrix has a time complexity of $O(C1(C2)^2)$ [38], assuming $C1 \geq C2$. Since we only care about β rather than S_{wm} in the extraction stage, we can directly set S_{wm} equal to S. Then Equation (1) can be re-written as

$$S'_{wm} = (1 + (-1)^{\beta}\alpha)S. \tag{2}$$

Consequently, $\mathscr{F}(r'_{wm}) = US'_{wm}V^T = U(1 + (-1)^{\beta}\alpha)SV^T = (1 + (-1)^{\beta}\alpha)\mathscr{F}(r)$. Based on this, we propose to directly compute $\mathscr{F}(r'_{wm})$ through scaling $\mathscr{F}(r)$ by $(1 + (-1)^{\beta}\alpha)$ without explicitly performing SVD to obtain S. This method allows us to polarize the frame's singular values without resorting to the time-consuming SVD operation during the embedding stage. 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, and determine the extracted bit as 0 if and only if $|S'_{wm}| > |S|$. As evaluated in Section 6.3, comparing the singular values (*i.e.,* S'_{wm} and S) is of higher robustness than directly comparing $\mathscr{F}(r'_{wm})$ and $\mathscr{F}(r)$. 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. Due to the fact that 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 Sec. 4.1, this may reduce the interdependence among different votes, thereby increasing the accuracy of the majority voting-based watermark extraction.

4.3 Algorithm Workflow

We are able to propose a watermarking algorithm based on the aforementioned techniques. The algorithm is formed of two parts: embedding and extraction, which are described in detail in Algorithm 1 and Algorithm 2. The embedding part can embed a single bit into a segment. Both parts make use of the same pseudo-random number generator to determine the location of the embedding regions. The time complexity is $O(n_e \cdot size^2)$ of the embedding stage, and $O(n_e \cdot size^3)$ of the extraction stage, where n_e represents the number of embedding regions in a single segment, and $size$ represents the width of a square embedding region. As for processing the whole streaming video consisting of m segments, the algorithm needs to be called m times, increasing the complexities to $O(mn_e \cdot size^2)$ and $O(mn_e \cdot size^3)$.

5 Fingerprint Generation and Accusation

As proved in Theorem IV.2 of [3], there are no deterministic (*i.e.,* totally secure with no errors) fingerprinting codes against the collusion attack. Following the research trend of collusion-resistant fingerprinting, we employ a randomized fingerprinting strategy to effectively defend against the CR attack. This strategy is based on the Tardos fingerprinting scheme [36]. We begin by introducing the Tardos code and then describe how we build a fingerprinting system for streaming and live streaming services.

5.1 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 η 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.

Every time a user requests the digital content, a fingerprint will be generated independently and added to X as its new row. Before generating any user fingerprint, m parameters (*i.e.,* $p_1^{(i)}$, $1 \le i \le m$) need to be sampled in advance. The probability density function used to sample $p_1^{(i)}$ from $[t, 1-t]$ is

$$f(p) = N_t / \sqrt{p(1-p)}, \tag{3}$$

where the coefficient N_t ensures $\int_t^{1-t} f(p)dp = 1$, 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 $f(p)$ and a small t will be discussed later in Section 5.2. Then, each bit $X_j^{(i)}$ in x_j 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 $1 \le c = |C| \le c_0$, 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 $y = (y^{(1)}, ..., y^{(m)}) = \rho(X_C)$, where ρ is a (deterministic or probabilistic) function determined by the collusion strategy, and $y^{(i)} \in \{0, 1\}$.

The accusation module accuses colluders based on the calculated accusation score. For any user j that once requested the video, the accusation score for the user is computed as $s_j = \sum_{i=1}^m s_j^{(i)} = \sum_{i=1}^m \left(\delta_{y^{(i)}, X_j^{(i)}} g_1(p_{y^{(i)}}^{(i)}) + (1 - \delta_{y^{(i)}, X_j^{(i)}}) g_0(p_{y^{(i)}}^{(i)})\right)$, 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 user's accusation score will thus be increased by $|g_1(p_{y^{(i)}}^{(i)})|$, as $y^{(i)} = X_j^{(i)}$ leads to $\delta_{y^{(i)}, X_j^{(i)}} = 1$. Otherwise, in case that $y^{(i)} \neq X_j^{(i)}$, the user's accusation score 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, $s_j^{(i)}$ has an expectation of 0 and a variance of 1 over all the users (*i.e.,* all the values of j).

The threshold value of the accusation 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 [36, 46] 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).

5.2 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 $\exists \beta \in \{0, 1\}$ *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 $1 \le i \le m$ 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 [21, 36, 46]. To utilize the marking condition, the cutoff parameter t used in the distribution function (see Equation (3)) is typically set close to 0 in early works [36, 46]. As the distribution is biased towards two ends of $[t, 1 - t]$ (in contrast to values around $p = \frac{1}{2}$), 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, they will have no choice on these sections, making their specific collusion strategy have limited influence on their chance of being caught.

Although a value t close to 0 takes advantage of the marking condition, it reveals limited information about the source of piracy at most positions, where nearly all users receive the same variant. As a result, the fingerprint length is rather long, reducing the practicality of the fingerprint code. To avoid such issue, we re-select the value of t by deducing the expectation of the sum of all colluders' accusation scores under the three assumptions of the reasonable CR attack model. We set t to 0.5 with the support of a theoretical analysis (see the following Section 5.3), and greatly reduce the code length needed to defend the same number of colluders (or defend a larger group of colluders with the same code length).

5.3 Parameter Selection

Objectives. When analyzing the Tardos scheme's security level, two often used metrics are P_{FP} (see Section 5.1) 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 assault 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 set as small as possible, so that the number of segments from the streaming video is no less than m.
- **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 5.1** is proved by Škorić *et al.* [46], while the proof of **Theorem 5.2** is conducted in Appendix A.2.

Theorem 5.1 (The sufficient condition for **Objective 1**). *If the parameter values satisfy* $\frac{1.7}{\sqrt{\frac{1-t}{t}}} >= \frac{B}{2c_0A}$, *and*

$$B^2/(4A) \geq 1, \tag{4}$$

***Objective 1** will be achieved.*

Theorem 5.2 (The sufficient condition for **Objective 2**). *If the colluders' strategy conforms to **Assumption 1 - 3**, and* $\tilde{\mu} = \frac{E[\sum_{j \in C} s_j]}{m}$ *satisfies* $\tilde{\mu} \leq 3.4\sqrt{\frac{t}{1-t}}$, *and*

$$A\tilde{\mu} - B \geq 0, \tag{5}$$

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

To meet **Objective 1** and **Objective 2**, we instead try to meet the above sufficient conditions. If we only consider the inequality (4) and (5), 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 A.1** in Appendix, $\tilde{\mu}$ obtains its maximum value 1 if and only if $t = \frac{1}{2}$, so the minimum value of A^* is 4. However, we actually cannot ignore other equations occurring in the above two theorems. Therefore, to meet both the sufficient conditions listed above and the remaining objectives (*i.e.,* **Objective 3 - 4**), StreamingTag instead sets t to $\frac{1}{2}$, A to $4(1 + \sqrt{\phi})$, and B to $4 + 2\sqrt{\phi}$, as long as $\phi = \frac{8ln\frac{1}{\epsilon}}{c_0 ln\frac{n}{\eta}} \leq 1$. With this setting, we can verify that all the inequalities in **Theorem 5.1** and **5.2** hold.

Parameter Selection Strategy in StreamingTag. StreamingTag sets t to 0.5 for the fingerprint sampling. Once the preprocessing stage ends, the number of segments is determined, and the fingerprint length m is fixed to that number. Since the number of users, n, grows over time, StreamingTag will update the parameters (η, ϵ, c_0, A, B, and z; excluding t and m) as follows for every captured pirated copy:

- **Step 1**: Set n to the number of users that have requested the legitimate video from the streaming platform so far.
- **Step 2**: Init η and ϵ to 0.1.
- **Step 3**: Compute the maximum integer c_0 *s.t.* $\phi = \frac{8ln\frac{1}{\epsilon}}{c_0 ln\frac{n}{\eta}} \leq 1$ and $4(1 + \sqrt{\phi})c_0^2 ln(\frac{n}{\eta}) \leq m$. If no such c_0 exists and $\epsilon < 1$, increase ϵ by 0.01 to decrease ϕ, and then restarts Step 3.
- **Step 4**: Select the value of η so that $4(1 + \sqrt{\phi})c_0^2 ln(\frac{n}{\eta}) = m$ holds. The value of η will always decrease after this step. Intuitively, this indicates that every segment will be used as part of the fingerprint to lower the bound of error rate.
- **Step 5**: Set A to $4(1 + \sqrt{\phi})$, B to $4 + 2\sqrt{\phi}$, and z to $Bc_0 ln(\frac{n}{\eta})$.

Example. Consider a video of 15 minutes. Through properly setting the parameters of the encoder, the length of each segment is fixed to a constant value (*e.g.,* 1 second), and thus 900 independently coded segments will be generated. Each segment has two differently watermarked versions, representing bits 0 and 1, respectively. As suggested above, $t = \frac{1}{2}$ is used to generate fingerprints.

In case a pirated video is redistributed, the streaming platform will use the parameter selection strategy above to determine the parameters. The platform examines its database to confirm that 100,000 users have viewed the video, and thus sets n to 100,000. Then both η and ϵ are initialized to 0.1. Following the above **Step 3**, c_0 is set to 3, the maximum integer *s.t.* $\phi \leq 1$ and $4(1 + \sqrt{\phi})c_0^2 ln(\frac{n}{\eta}) \leq m = 900$ hold. Currently, $4(1 + \sqrt{\phi})c_0^2 ln(\frac{n}{\eta}) \approx 829 < 900$, indicating that some segments will not be used for fingerprint embedding. To avoid such waste, the platform follows the above **Step 4** to set η to approximately 0.0225. Finally, with $\eta \approx 0.225$, $c_0 = 3$, and $\epsilon = 0.1$, the platform further sets A to $4(1 + \sqrt{\phi}) \approx 6.53$, B to $4 + 2\sqrt{\phi} \approx 5.27$, and z to $Bc_0 ln(\frac{n}{\eta}) \approx 241.8$. An accusation score will then be calculated for every one of the 100,000 users, and a user will be accused only if the score exceeds z. A collusion group of no more than three members will be captured with a probability of $1 - \eta \approx 98\%$, if their collusion strategy meets the CR model.

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	17:55
		Rate (fps)	277	276	170	123	131	133	139	130	133
	EX	Duration	01:57	01:19	02:38	02:25	02:31	01:48	01:59	02:15	03:19
		Rate (fps)	1164	1322	945	784	849	926	951	684	716
Single Region	EM (ms)		6.7	6.7	6.9	7.0	6.9	6.9	6.8	7.0	6.9
	EX (ms)		13.9	13.2	16.0	17.8	17.2	17.0	15.4	17.2	18.4
Size Inc (‰)			2.7	1.5	7.7	1.8	3.8	7.1	5.4	1.3	3.7

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

Table 2: The attributes and QoS metrics of each test video.

6 Evaluation

We examine StreamingTag in this section with comprehensive experiments. The preprocessing pipeline is prototyped on top of *FFmpeg* [1], with the proposed watermarking technique implemented as a filter in FFmpeg. The evaluation is primarily concerned with three aspects: (a) we assess StreamingTag's impact on QoS via several metrics, including latency, bandwidth, and fidelity; (b) we assess its robustness against the NR attack; and (c) we assess its end-to-end collusion resistance against the CR attack.

6.1 Experimental Setup

We evaluate StreamingTag's performance using nine distinct films, the key attributes of which are presented in the upper part of Table 2. All test videos are at a frame rate of 25 frames per second (fps). The processing latency of StreamingTag is measured on a desktop computer equipped with an AMD Ryzen 9 5950X processor and 128GiB memory. The proposed watermarking algorithm's resistance to a software-based screen recording attack is evaluated on an Android phone (*i.e.,* the phone is used as a mobile video playing terminal, and a pirated video is recorded by its built-in screen recorder).

We set the watermark strength α in Algorithm 1 to 0.2. We set n_e, the number of watermarked frames within a single segment, to 1, as such setting can ensure strong robustness (as demonstrated later in Section 6.3) with minimal overhead. The size of each embedding block is set to 512. To encode the watermarked video for large-scale distribution, we employ the H.264 encoder. 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. To enable fast encoding, we set the `bframes` parameter to 0, the `rc-lookahead` parameter to 0, the `preset` parameter to `superfast`, and the `tune` parameter to `zerolatency`. To enable the advanced features of H.264 for efficient delivery, we set the `profile:v` parameter to `high`.

6.2 Evaluation of QoS

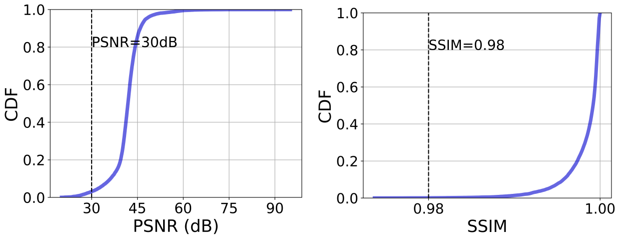
We use three metrics, *i.e.,* processing latency, bandwidth consumption, and video fidelity to evaluate the **QoS** of StreamingTag:

- **Processing latency** measures the running performance of the watermarking algorithm, which is the indicator to determine if the algorithm can run in real-time.
- **Bandwidth consumption** measures the overhead of the watermarking algorithm.
- **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) [39]. PSNR is determined by the mean square difference between the original video and its watermarked version, 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 cost of processing the entire video. The durations of generating all watermarked variants from the test video (decoding, watermark embedding, and re-encoding) and extracting the embedded fingerprint from a complete copy (decoding and watermark extraction) are recorded. The frame rates of these two processes are computed. Then, we determine how much CPU time our watermarking algorithm requires to process a single embedding region. Through adding instrumentation codes [17] into the implemented *FFmpeg* watermarking filter, the time consumed for encoding and decoding is excluded while measuring the CPU time.

The middle part of Table 2 summarizes the findings, from which we can deduce that: (a) The frame rate of processing the entire video is significantly higher than 25 fps, verifying that StreamingTag is suitable for real-time streaming services; (b) The proposed watermarking algorithm is quite lightweight as it takes only several milliseconds to process a single region of shape 512 × 512.

Bandwidth Consumption. Video encoding techniques compress the raw video by exploiting its spatial and temporal redundancy. Since fingerprints are embedded into the original video through watermarking technologies in StreamingTag, the amount of both spatial and temporal redundancy may decrease greatly with a poorly designed watermarking technology. In consequence, the size of a watermarked video could be possibly larger than that of an unwatermarked one, leading to larger bandwidth consumption and a worse QoS. To demonstrate that StreamingTag does not suffer from such a problem, we measure the increase of the video size after watermarking, by comparing the average size of watermarked segments and that of unwatermarked ones.



(a) CDF of PSNR

(b) CDF of SSIM

Figure 4: CDF of PSNR and SSIM.

To generate unwatermarked segments, we first decode the video and then partition it into successive segments, keeping all H.264 encoder parameters constant for a fair comparison. As shown in the last row of Table 2, the video size increases are less than 1% for all test videos. As a result, we can conclude that StreamingTag almost never consumes additional bandwidth.

Video Fidelity. Some visible artifacts may exist in the watermarked video and severely degrade the QoS, if the proposed watermarking algorithm modifies the segment in an inappropriate way. Therefore, we measure the fidelity of the watermarked video to validate if our watermarking algorithm incurs such a problem.

For each test video, we first transcode it to a series of unwatermarked segments. In accordance with StreamingTag's configuration, each segment has a duration of 1 second. Then we use StreamingTag to convert each test video into watermarked segments. PSNR and SSIM are computed between the embedding regions in the watermarked segments and the regions located at the same position of the unwatermarked segments.

The results are shown in Fig. 4. Since a PSNR higher than 30dB (or an SSIM higher than 0.98) is considered good, we can conclude from the figure that StreamingTag does not sacrifice video fidelity.

6.3 Robustness of Watermarking

Comparison to Alternatives. The robust extraction of embedded bits lays the foundation for fingerprinting-based accusation. Therefore, we evaluate the robustness of our watermarking scheme and compare it with several alternatives to demonstrate the effectiveness and necessity of our design.

We compare our watermarking scheme with three alternatives. For a fair comparison, we use the same set of embedding regions while evaluating all the methods. The first alternative is referred to as bipolar DWT watermarking. This alternative uses the same strategy to embed the watermark as our method. However, while extracting the watermark from a watermarked frame, this alternative works by calculating $\frac{sum(abs(DWT(r'_{wm}).HL_3))}{sum(abs(DWT(r_{adj}).HL_3))}$, where *sum* computes the sum of all elements in a matrix, *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 method described in Section 2.3. We use two watermark images to represent bit 0 and bit 1 respectively, as discussed in Section 4.1. The third alternative is DCT-based watermarking [37], which embeds the watermark in the DCT domain.

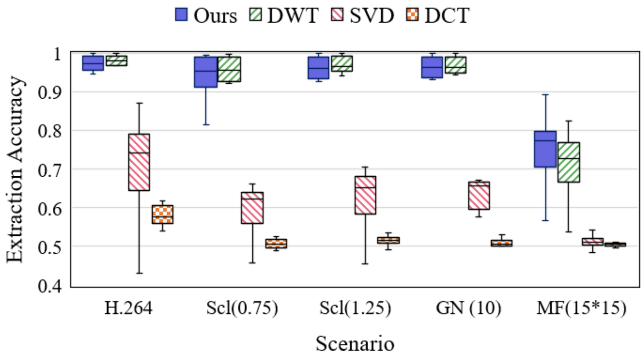


Figure 5: The extraction accuracy of different watermarking schemes under different attacks.

For each test video and each tested method, the measurement is conducted several times with different kinds of visually insignificant modifications performed. Specifically, in addition to the basic case where the attackers perform no modification on the H.264-encoded segment, we mimic the attacker's behavior by decoding the watermarked and encoded segment, modifying (through Gaussian noising of $\sigma = 10$, scaling with a factor of 0.75 or 1.25, or 15×15 median filtering), and then re-encoding (with H.264 again) the modified segments for illegal re-distribution. The attack is conducted by transcoding the watermarked segment with some built-in *FFmpeg* filters, including *scale*, *noise*, and *median* filters. To extract the watermark from scaled segments, we first decode them and then re-scale them to the original resolution.

The results are shown in Fig. 5, where the five settings of attacks are abbreviated for simplicity. We can conclude that: (a) Both our scheme and bipolar DWT watermarking share the same embedding scheme and achieve higher robustness than the other two alternatives. This validates the effectiveness of polarizing two different variants along with opposite directions for higher robustness in the case of low-density watermark embedding. (b) Although bipolar DWT watermarking achieves almost the same level of extraction accuracy as our scheme under most types of attacks, our scheme still outperforms it under very strong attacks, *i.e.,* H.264 plus 15×15 Median Filter plus H.264. This proves that explicit SVD operation in the extraction stage indeed captures the intrinsic characteristics of a matrix and 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-based watermarking. As demonstrated shortly later in Section 6.4, such a level of robustness is enough for fingerprinting-based colluder accusation, which demonstrates the effectiveness of our watermarking scheme. Besides, the robustness can be further boosted by randomly selecting multiple embedding regions from a segment with a $n_e > 1$.

Screen Recording. We first download the entire test videos on an Android phone, and then run the built-in screen recorder while playing them. Then watermarks are extracted from all recorded videos. The accuracy is 94.6% on average for all videos, and reaches

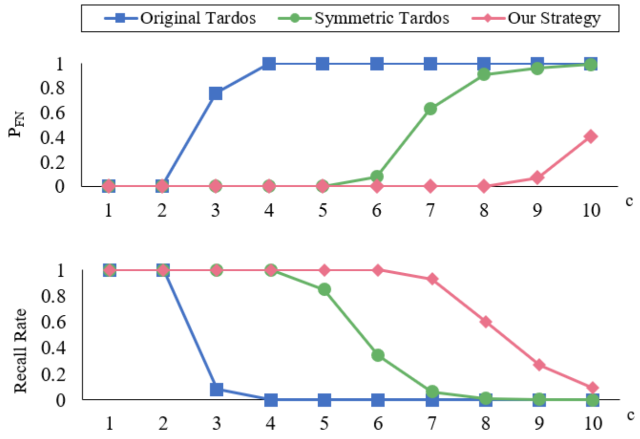


Figure 6: Collusion resistance.

to a maximum value of 99.8% for one pirated video. Such result meets our expectation, since screen recording can be viewed as a composite of decoding, scaling, and re-encoding. As discussed previously in this subsection, our method effectively defends against such a composite attack.

6.4 Security under 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 [36] and that in the symmetric Tardos scheme [46]. We consider a video of 7200 segments in this experiment. Based on the extraction accuracy data in Section 6.3, we assume an extraction accuracy of 90% in this experiment. we set n to 100000 and rang 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.

For our method, we use the parameter selection strategy described in Section 5.3. For the two alternatives, we set m to 7200 and c_0 to the maximum number of colluders which can be defended with a false positive error rate of $\eta < 0.1$. The value of t is then determined based on c_0, as required by [36, 46] (*e.g.,* , t is set to $\frac{1}{300c_0}1$ in [36]). Note that this assumption is only for comparison. In real-world live streaming services, however, the alternatives have to first arbitrarily estimate the number of segments, then determine a value of c_0 so that the code length $m = Ac_0^2 ln\frac{n}{\eta}$ does not exceed the number of segments, and finally select the value of t (which affects fingerprint generation) based on c_0. To avoid an excessively large code length, the alternatives may use a conservative estimate and underutilize the collusion resistance capacity of the code, resulting in inferior performance than our evaluation.

The results are shown in Fig. 6. Since P_{FP} equals to 0 in every experiment, we don 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 [46], and by 58% when compared with [36].

7 Limitations and Future Work

Temporal desynchronization. Though the proposed watermarking scheme is demonstrated to be robust against various attacks, all of them do not alter the temporal index of any frame. Therefore, we can directly identify the host frames from a pirated video by utilizing their indices. However, such property may not always be guaranteed. First, in Internet-based streaming applications, some frames may be lost when there is network congestion. Second, malicious attackers could launch a so-called frame dropping or insertion attack to disrupt the frame indices. To address this issue, many recent works use scene change detection to identify the motion frame (*i.e.,* a frame which is significantly different from the previous frame) as the host frame for watermark embedding [31]. In this manner, even if the index of the motion frame is altered, the motion frame will still be correctly identified in the extraction stage via resorting to scene change detection. We expect to use similar strategies in the future, yet the possibility that a determined attacker uses the same scene change detection algorithm to identify and then damage (or drop) motion frames must be handled well.

Deep learning-based attack. With the advance of deep learning techniques, some researchers seek to compromise watermarking techniques with deep neural networks (DNNs), so that the watermark is removed without degrading the visual quality [12, 22]. One promising countermeasure is to integrate deep adversarial learning into our method, while the heavy overhead of deep models must be optimized to support large-scale streaming services.

Camcorder recording. We regard camcorder recording as more thorny than software-based screen recording, since the camcorder may be frequently shaken due to unsteady hands. Consequently, the screen (which plays the streaming video) may experience translation and rotation in the recorded video. We expect to enforce automatic alignment in the future, *e.g.,* using KAZE feature points to calculate and then canceling out the relative rotation [37].

Collusion strategy. Some colluders may employ a collusion strategy different from that in the CR model. For example, they could compare the received segment files at each position to find out which ones are equal, and then randomly select one from all the different variants. In that case, 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 distributed copy. Another measure is to use q-ary ($q > 2$) bits to construct fingerprints, scattering users' fingerprints into a large code space. Consequently, colluders are unlikely to receive all variants at many positions, and the mark used at each position of the pirated copy always has a higher correlation to the marks received by colluders than those of innocent users, whichever collusion strategy is adopted.

Distribution topologies. As different platforms may employ different watermarking and fingerprinting technologies and refuse to exchange relevant information for business reasons, attackers could escape from accusation by fetching different parts of a video from multiple streaming platforms to construct the entire pirated copy. 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 [28, 29] or adopt the blockchain technology [40]. We envision future deployments where similar strategies are adopted to maintain trust among streaming platforms.

8 Related Work

DRM. DRM protects digital content by encrypting it before distribution and requiring users to purchase digital licenses containing the decryption key [34]. However, as discussed in Section 1, DRM has several inherent drawbacks.

Anti-Camcording Technologies. Anti-camcording technologies are mainly based on the difference between the human visual system and the common digital cameras. For example, KALEIDO [44] utilized the disparity between the refresh rate (which can be as large as 240fps in the case of a high-end display screen) and the video rate (24fps or 30fps typically) and re-encoded the original video frames into multiple different frames, which are specially designed to prevent camcording as well as preserve the human viewing experience. LiShield [45] deployed smart LEDs flickering at high frequency and in specialized waveforms, which are imperceptible to the human visual system but not to the common digital cameras. Nevertheless, these technologies require client-side modifications and cannot prevent software-based screen 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.* [14] proposed such a deterministic fingerprinting approach that requires $c_0 \leq 2$. Staddon *et al.* [33] 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 [7], 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 [3] and the Tardos scheme [36, 46, 47]. Because the Tardos scheme's code length is nearly equal to the square root of the Boneh-Shaw scheme [36], we base our fingerprinting approach on the Tardos scheme.

9 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. The solution for robust fingerprint embedding incorporates a polarized, semi-explicit, and randomized SVD watermarking algorithm. 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 in terms of robustness and collusion resistance. StreamingTag, we believe, is a significant step towards scalable, reliable, and traceable video streaming.

Acknowledgments

We thank the anonymous shepherd and reviewers for their insightful comments to improve the quality of our work. This work is supported in part by the National Key R&D Program of China under grant 2021YFB2900100.

A Appendix

A.1 Derivation of the Value of $\tilde{\mu}$

Lemma A.1. *If the collusion strategy meets* ***Assumption 1-3****, the exact value of $\tilde{\mu}$ will only depend on the parameter t, and always satisfy $\tilde{\mu} \leq 1$. The equality holds if and only if $t = \frac{1}{2}$.*

Proof: With the symbol b defined in **Assumption 3** we have:

$$\begin{aligned}\tilde{\mu} =&E_p\Bigg[E_{b_1^{(i)}\sim Binom(c,p_1^{(i)})}\Bigg[\frac{b_1^{(i)}}{c}\cdot\left(b_1^{(i)}\left(\sqrt{\tfrac{1-p_1^{(i)}}{p_1^{(i)}}}+\sqrt{\tfrac{p_1^{(i)}}{1-p_1^{(i)}}}\right)-c\sqrt{\tfrac{p_1^{(i)}}{1-p_1^{(i)}}}\right)\\&+\frac{c-b_1^{(i)}}{c}\cdot\left(c\sqrt{\tfrac{p_1^{(i)}}{1-p_1^{(i)}}}-b_1^{(i)}\left(\sqrt{\tfrac{1-p_1^{(i)}}{p_1^{(i)}}}+\sqrt{\tfrac{p_1^{(i)}}{1-p_1^{(i)}}}\right)\right)\Bigg|p\Bigg]\Bigg]\\=&2E_p[\sqrt{p_1^{(i)}(1-p_1^{(i)})}].\end{aligned} \tag{6}$$

From (6) we easily verify this lemma.

A.2 Proof of Theorem 5.2

Let $s_C = \sum_{j\in C} s_j$ and $\tilde{\sigma}^2 = \frac{E_{yXp}[s_C^2]-E_{yXp}^2[s_C]}{m}$. Following the equation (21) in [46] we have

$$\begin{aligned}&\tilde{\sigma}^2+\tilde{\mu}^2\\=&E_p\Bigg[E_{b_1^{(i)}\sim Binom(c,p_1^{(i)})}\Bigg[(b_0^{(i)}\sqrt{\frac{1-p_0^{(i)}}{p_0^{(i)}}}-b_1^{(i)}\sqrt{\frac{p_0^{(i)}}{1-p_0^{(i)}}})^2\Bigg|p\Bigg]\Bigg]\\=&c.\end{aligned}$$

Combining the above equation and the proof process in Section 4.2 of [46], we easily prove **Theorem 5.2**.

A.3 Legend of Symbols

The main symbols are as follows:

Symbol	Meaning	Symbol	Meaning
α	watermark strength	r	embedding region
S	a matrix containing singular values	n_e	number of embedding regions
m	fingerprint length	C	set of colluders
n	number of users	j	index of a user
c_0	StreamingTag works for $c \leq c_0$	p	parameter of generating X
i	index of a segment (or a fingerprint bit)	X	a matrix of all fingerprints
s	accusation score	z	accusation threshold
y	fingerprint in the pirated copy	A	tuneable coefficient of m
B	tuneable coefficient of z	t	tuneable cutoff parameter
P_{FP}	probability of false positive accusation	P_{FN}	probability of falsely negative accusation
η	we hope $P_{FP} \leq \eta$	ϵ	we hope $P_{FN} \leq \epsilon$

References

[1] Anon. 2022. FFmpeg. https://ffmpeg.org/, last accessed on Sept. 5, 2022.

[2] Apple, Inc. 2022. FairPlay Streaming - Apple Developer. https://developer.apple.com/streaming/fps/, last accessed on Sept. 5, 2022.

[3] D. Boneh and J. Shaw. 1998. Collusion-Secure Fingerprinting for Digital Data. *IEEE Trans. Inform. Theory* 44, 5 (1998), 1897–1905.

[4] L. Boney, A.H. Tewfik, and K.N. Hamdy. 1996. Digital Watermarks for Audio Signals. In *Proceedings of the 3rd IEEE ICMCS*. 473–480.

[5] Anawat Chankhunthod, Peter B. Danzig, Chuck Neerdaels, Michael F. Schwartz, and Kurt J. Worrell. 1996. A Hierarchical Internet Object Cache. In *Proceedings of the 2nd USENIX ATC*.

[6] Scott Crosby, Ian Goldberg, Robert Johnson, Dawn Song, and David Wagner. 2002. A Cryptanalysis of the High-Bandwidth Digital Content Protection System. In *Security and Privacy in Digital Rights Management*. 192–200.

[7] Amos Fiat and Tamir Tassa. 1999. Dynamic Traitor Tracing. In *Proceedings of the 19th IACR CRYPTO*. 354–371.

[8] Forbes. 2019. Pirated Video Gets Viewed Over 200 Billion Times A Year [Infographic]. https://www.forbes.com/sites/niallmccarthy/2019/06/26/pirated-video-gets-viewed-over-200-billion-times-a-year-infographic/, last accessed on Sept. 5, 2022.

[9] Ehab Ghabashneh and Sanjay Rao. 2020. Exploring the Interplay between CDN Caching and Video Streaming Performance. In *Proceedings of the 39th IEEE INFOCOM*. 516–525.

[10] Rafael C. Gonzalez and Richard E. Woods. 2007. *Digital Image processing*. Pearson Education International, Chapter 7.5, 523–532.

[11] Google, Inc. 2022. Widevine. https://widevine.com/, last accessed on Sept. 5, 2022.

[12] Makram W. Hatoum, Jean-François Couchot, Raphaël Couturier, and Rony Darazi. 2021. Using Deep learning for image watermarking attack. *Signal Process.: Image Commun.* 90 (2021), 116019.

[13] Yingliang He, Gaobo Yang, and Ningbo Zhu. 2012. A Real-Time Dual Watermarking Algorithm of H.264/AVC Video Stream for Video-on-Demand Service. *AEU - Int. J. Electron. Commun.* 66, 4 (2012), 305–312.

[14] Henk D.L Hollmann, Jack H van Lint, Jean-Paul Linnartz, and Ludo M.G.M Tolhuizen. 1998. On Codes with the Identifiable Parent Property. *J. Comb. Theory - A* 82, 2 (1998), 121–133.

[15] Ken Florance. 2016. How Netflix Works With ISPs Around the Globe to Deliver a Great Viewing Experience. https://about.netflix.com/en/news/how-netflix-works-with-isps-around-the-globe-to-deliver-a-great-viewing-experience, last accessed on Sept. 5, 2022.

[16] Kepios Pte. Ltd. 2022. Digital 2022 Global Overview Report. https://datareportal.com/reports/digital-2022-global-overview-report, last accessed on Sept. 5, 2022.

[17] Michael Kerrisk. 2020. time(7) — Linux manual page. https://man7.org/linux/man-pages/man7/time.7.html, last accessed on Sept. 5, 2022.

[18] Steven L Kinney. 2006. *Trusted Platform Module Basics: Using TPM in Embedded Systems*. Newnes.

[19] Krebs on Security. 2020. Google Mending Another Crack in Widevine. https://krebsonsecurity.com/2020/10/google-mending-another-crack-in-widevine/, last accessed on Sept. 5, 2022.

[20] Alavi Kunhu, Nisi K, Sadeena Sabnam, Majida A, and Saeed AL-Mansoori. 2016. Index Mapping based hybrid DWT-DCT Watermarking Technique for Copyright Protection of Videos Files. In *Proceedings of the 2nd IEEE IC-GET*. 1–6.

[21] Thijs Laarhoven, Jeroen Doumen, Peter Roelse, Boris Škorić, and Benne de Weger. 2013. Dynamic Tardos Traitor Tracing Schemes. *IEEE Trans. Inf. Theor.* 59, 7 (2013), 4230–4242.

[22] Qi Li, Xingyuan Wang, Bin Ma, Xiaoyu Wang, Chunpeng Wang, Suo Gao, and Yunqing Shi. 2022. Concealed Attack for Robust Watermarking Based on Generative Model and Perceptual Loss. *IEEE Trans. Circuits Syst. Video Technol.* 22, 8 (2022), 5695–5706.

[23] Jonathan Looney. 2021. Netflix: Streaming Entertainment to 200 Million Members Around the World.

[24] K. Meenakshi, K. Swaraja, and Padmavathi Kora. 2019. A Robust DCT-SVD Based Video Watermarking Using Zigzag Scanning. In *Soft Computing and Signal Processing*. 477–485.

[25] Microsoft, Inc. 2022. Microsoft PlayReady - Home. https://www.microsoft.com/playready/, last accessed on Sept. 5, 2022.

[26] Roger Pantos. 2022. HTTP Live Streaming 2nd Edition. https://datatracker.ietf.org/doc/html/draft-pantos-hls-rfc8216bis, last accessed on Sept. 5, 2022.

[27] S. Ponni alias Sathya and S. Ramakrishnan. 2018. Fibonacci based Key Frame Selection and Scrambling for Video Watermarking in DWT-SVD Domain. *Wirel. Pers. Commun.* 102, 2 (2018), 2011–2031.

[28] Amna Qureshi, David Megías, and Helena Rifà-Pous. 2015. Framework for Preserving Security and Privacy in Peer-to-peer Content Distribution Systems. *Expert Syst. Appl.* 42, 3 (2015), 1391–1408.

[29] Amna Qureshi, Helena Rifà Pous, and David Megías Jiménez. 2014. Secure and anonymous multimedia content distribution in peer-to-peer networks. In *Proceedings of the 6th IARIA MMEDIA*. 91–96.

[30] Weixiang Ren, Yibo Xu, Liming Zhai, Lina Wang, and Ju Jia. 2020. Fast Carrier Selection of JPEG Steganography Appropriate for Application. *Tsinghua Sci. Technol.* 25, 5 (2020), 614–624.

[31] Kh. Manglem Singh. 2018. A Robust Rotation Resilient Video Watermarking Scheme based on the SIFT. *Multimedia Tools Appl.* 77, 13 (2018), 16419–16444.

[32] Iraj Sodagar. 2011. Media presentation description and segment formats | MPEG. https://mpeg.chiariglione.org/standards/mpeg-dash/media-presentation-description-and-segment-formats, last accessed on Sept. 5, 2022.

[33] J.N. Staddon, D.R. Stinson, and Ruizhong Wei. 2001. Combinatorial Properties of Frameproof and Traceability Codes. *IEEE Trans. Inf. Theory* 47, 3 (2001), 1042–1049.

[34] S.R. Subramanya and B.K. Yi. 2006. Digital Rights Management. *IEEE Potentials* 25, 2 (2006), 31–34.

[35] Jing Sun, Xiaoping Jiang, Jin Liu, Fan Zhang, and Congying Li. 2021. An Anti-Recompression Video Watermarking Algorithm in Bitstream Domain. *Tsinghua Sci. Technol.* 26, 2 (2021), 154–162.

[36] Gábor Tardos. 2008. Optimal Probabilistic Fingerprint Codes. *J. ACM* 55, 2, Article 10 (2008), 24 pages.

[37] Ta Minh Thanh, Pham Thanh Hiep, Ta Minh Tam, and Keisuke Tanaka. 2014. Robust Semi-blind Video Watermarking based on Frame-patch Matching. *AEU - Int. J. Electron. Commun.* 68, 10 (2014), 1007–1015.

[38] Lloyd N Trefethen and David Bau III. 1997. *Numerical linear algebra*. Siam.

[39] Zhou Wang, A.C. Bovik, H.R. Sheikh, and E.P. Simoncelli. 2004. Image Quality Assessment: From Error Visibility to Structural Similarity. *IEEE Trans. Image Process.* 13, 4 (2004), 600–612.

[40] Yang Xu, Cheng Zhang, Quanrun Zeng, Guojun Wang, Ju Ren, and Yaoxue Zhang. 2021. Blockchain-Enabled Accountability Mechanism Against Information Leakage in Vertical Industry Services. *IEEE Trans. Netw. Sci. Eng.* 8, 2 (2021), 1202–1213.

[41] Takaaki Yamada, Michiro Maeta, and Fuminori Mizushima. 2016. Video Watermark Application for Embedding Recipient ID in Real-time-encoding VoD Server. *J. Real-Time Image Process.* 11, 1 (2016), 211–222.

[42] Zheng Yan, Xinren Qian, Shushu Liu, and Robert Deng. 2022. Privacy Protection in 5G Positioning and Location-Based Services Based on SGX. *ACM Trans. Sen. Netw.* 18, 3, Article 41 (2022), 19 pages.

[43] Xiaoyan Yu, Chengyou Wang, and Xiao Zhou. 2018. A Survey on Robust Video Watermarking Algorithms for Copyright Protection. *Appl. Sci.* 8, 10 (2018), 1–26.

[44] Lan Zhang, Cheng Bo, Jiahui Hou, Xiang-Yang Li, Yu Wang, Kebin Liu, and Yunhao Liu. 2015. Kaleido: You Can Watch It But Cannot Record It. In *Proceedings of the 21st ACM MobiCom*. 372–385.

[45] Shilin Zhu, Chi Zhang, and Xinyu Zhang. 2017. Automating Visual Privacy Protection Using a Smart LED. In *Proceedings of the 23rd ACM MobiCom*. 329–342.

[46] Boris Škorić, Stefan Katzenbeisser, and Mehmet U. Celik. 2008. Symmetric Tardos Fingerprinting Codes for Arbitrary Alphabet Sizes. *Des. Codes Cryptogr.* 46, 2 (2008), 137–166.

[47] Boris Škorić, Tatiana U. Vladimirova, Mehmet Celik, and Joop C. Talstra. 2008. Tardos Fingerprinting is Better Than We Thought. *IEEE Trans. Inf. Theory* 54, 8 (2008), 3663–3676.