Delivery of Live Watermarked Video in CDN: Fast and Scalable Algorithms

Kun He
IMT Atlantique, France
kun.he@imt-atlantique.fr

Patrick Maillé
IMT Atlantique, France
patrick.maille@imt-atlantique.fr

Gwendal Simon
IMT Atlantique, France
gwendal.simon@imt-atlantique.fr

ABSTRACT

To address the problem of illegal re-streaming of video streams, existing solutions are based on watermarking the legal video to track the leak users who re-stream the stream on illegal platform. However, these solutions do not aim at tracking leaks as fast as possible, nor are adaptive to the number of users. We present a CDN-based adaptive delivery architecture for watermarked streaming. We propose an algorithm to generate unique sequences of watermarks for the legal delivery. This algorithm is adaptive in the number of users and optimal for the time needed to detect the leak. It meets the demand of live video providers who do not know in advance the number of clients for a stream. Our algorithm copes with thousands of new clients per seconds and enables leak detection in less than five minutes with only five watermarks for live video streams watched by one billion of clients.

CCS CONCEPTS

• **Security and privacy** → **Digital rights management**; • **Information systems** → *Multimedia streaming*; • **Mathematics of computing** → Network flows;

KEYWORDS

Leak Tracking, Adaptive Live Video, Watermarking, CDN

ACM Reference format:
Kun He, Patrick Maillé, and Gwendal Simon. 2017. Delivery of Live Watermarked Video in CDN: Fast and Scalable Algorithms. In *Proceedings of NOSSDAV'17, Taipei, Taiwan, June 20-23, 2017*, 6 pages.
https://doi.org/http://dx.doi.org/10.1145/3083165.3083174

1 INTRODUCTION

Various law courts have judged illegal the action of re-streaming a live video stream that has been acquired from a copyrighted video service on another video delivery service. The courts target in particular the illegal delivery services that stream live sport events although the rights for these events are restricted to some stakeholders. Unfortunately, these official rules do not prevent illegal streaming services to flourish. Furthermore, since most legal TV providers now offer a live video delivery service on the Internet, the act of re-streaming is easy: each legal customer receives an encoded and packetized version of the live TV, which can thus be forwarded to any illegal delivery service without additional efforts. Let us call *leak* the user who has an account on a legal live video service and who illegally forwards the acquired video stream to another delivery service.

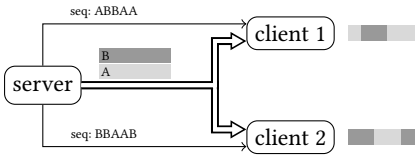

Figure 1: Multicast delivery of watermarked video: The server multicasts two videos and unicasts, for each client, a 5-long sequence. The displayed video alternatively switches between both video versions with respect to the unique sequence.

To address the problem of live video re-streaming, the legal service providers can implement the following scheme: (1) the stream that is legally delivered to the customers is watermarked. The stream that is delivered to every customer is unique and can be identified, (2) the legal service gets the video stream that is delivered on an illegal delivery platform, extracts the watermark, and thus identifies the leak, and (3) the legal video service stops streaming to the leak. However, this solution does not scale: The service provider should prepare and deliver a different watermarked stream for each customer, prevent use scalable group delivery techniques such as *multicast* or Content Delivery Network (CDN).

Since the early 2000s, researchers have addressed the paradoxical problem of *both* multicasting and ensuring the uniqueness of watermarked video. The solution that was initially proposed by Parviainen and Parnes [13] is depicted in Figure 1. The main idea is to *multicast* to every user *two* versions of the same video (one with a watermark A and another with watermark B) and to *unicast* to every user a unique *sequence* of A/B. The video players display unique video streams by alternatively switching between the video with watermark A and watermark B with respect to the unique sequences. This solution, which "only" doubles the bandwidth can also be implemented within a CDN [9]. The generation of unique sequences has been the topic of research, for example to prevent collusion among leaks [10], and is in practice based on randomized algorithms to get 64-long sequences. More recently Jarnikov et al. [6] proposed to implement a similar technique in adaptive streaming systems by switching watermarks at every new video *segment* [5, 6]. In this paper, we go into details of this implementation for CDN and adaptive technologies.

In practice, the solution of switching watermarks based on unique sequences suffers from a *time-scalability* issue. Indeed, the

NOSSDAV'17, June 20-23, 2017, Taipei, Taiwan

ACM ISBN 978-1-4503-5003-7/17/06…$15.00
https://doi.org/http://dx.doi.org/10.1145/3083165.3083174

video consists of segments, and the duration of one segment in practical adaptive streaming systems ranges from 2 to 10 seconds. With 64-long sequences, the leak identification requires downloading 128 consecutive video segments from the illegal streaming platform, so up to 21 minutes. The time needed to identify the leak is too long for TV providers who deliver hour-long sport events.

We study in this paper the problem of implementing large-scale watermarked video delivery such that the time to identify the leak is minimum. The previous work on watermark sequence generation has addressed the requirements of legal Video on Demand (VoD) providers fighting Peer-to-Peer (P2P) platforms and the sophisticated collusion techniques implemented by teams of leaks. We propose in this paper to revisit the problem of delivering large-scale watermarked video in the case of legal TV providers. We propose a CDN-based delivery system, which is inspired by Jarnikov proposal [5, 6] (see Section 3). We present an algorithm in Section 4 to generate sequences that scale to the number of legal customers, and is thus optimal with regards to the time needed to identify the leak. We study our algorithms from a theoretical standpoint and we provide performance evaluation results in Section 5. Our goal is to show the feasibility of implementing this algorithm in practical systems. In Section 6, we summarize our studies of fast leak tracking in live video stream and we open perspectives regarding the problem of accelerating the leak detection.

2 BACKGROUND AND RELATED WORK

Watermarking. Watermarking is the technique that allows content providers to trace delivered data by embedding hidden information in video, image, and other media types so that the authorised copy of the digital content can be uniquely identified. Katzenbeisser and Petitcolas [8] give an introduction on the fundamental principles of watermarking and watermarked data.

A reliable watermarked delivery solution requires the watermark to meet some requirements [12]. First, it should be difficult for the users to distinguish into watermarked and unwatermarked content, which means the quality of the data should not be affected by the integration of the watermark in the content. Second, the legal provider should be able to extract the watermark even in the case of distortions (typically video transcoding). Third, the watermark information should be difficult to tamper and to forge by attackers.

Adaptive Streaming. We refer to the vocable introduced by the MPEG Dynamic Adaptive Streaming over HTTP (DASH) standard [16]. The live video stream is encoded into multiple video *representations*, which are cut into *segments* of k seconds (k being up to 10). When delivering adaptive live video streams, the CDN should consider three main cost sources: (1) The *preparation cost* for transcoding [18] and watermarking [3] all the representations. (2) The *inner delivery cost* for the transmission of all video representations from the origin server to the set of edge servers that have been selected for the delivery [1]. (3) The *edge delivery cost* for the transmission of the segments from the edge servers to the clients, which cost can be negotiated with the network operators [11]. In addition, the CDN must over-provision resources because the popularity of live streams is not easy to predict. For example, Twitch delivers thousands of live streams concurrently, only a handful of them being watched by more than thousands of clients [14].

The recent proposal from Jarnikov et al. [6] to deliver watermarked videos in adaptive streaming is based on the same principles as the proposal from Kopilovic et al. [9] for one server, which was inspired by the pioneering work from Brown et al. [4] and Parviainen and Parnes [13]. The edge server stores two versions of each segment of each representation. It also stores the sequence of watermarks for each client. When the server gets a request for a given segment from a given user, the server responds by sending the version that corresponds to the right watermark in the sequence associated with this user. This solution doubles the preparation costs and the inner delivery costs but it does not impact the edge delivery costs. It also increases the burden on the edge server (sequence maintaining and alternate segment management).

Sequence Management. The research on sequence generation has followed the advances of the technologies. For example Jin et al. [7] addressed the problem of sequence management in physical devices such as video discs, while Parviainen and Parnes [13] considered that the clients of multicast-based IPTV systems generate sequences in home gateways. These systems let the video player switch watermarks on a short periodic frame, typically at the level of network packets or video frames. Another scheme is to infinitely randomly alternate watermarks without generating false positives (a legal customer being accused of being a leak). Sequences are thus as long as the video and the time needed to identify a leak is not deterministic.

The research on randomized sequence generation has been especially active since the work from Tardos [17] on fingerprinting. The motivation here is to find the shortest sequence length for a given population, a given number of leaks forming a collusion team, and a given minimum threshold for false positive probability. Multiple papers have then refined the optimal bounds in various scenarios [2, 15]. In parallel, Laarhoven et al. [10] have addressed the problem for a dynamic number of cooperating leaks. The line of research matches the requirements of VoD providers who have to deal with a team of leaks by preventing collusion. This work provides theoretical bounds for the sequence length with respect to the number of clients, however it requires in practice the service provider to know in advance the maximum number of clients for the video.

In this paper, we explore the case of live video providers who do *not* know in advance the number of clients for a stream and want to nevertheless minimize the time needed for the leak identification. This is typically the case of user-generated video platforms such as Twitch, Twitter, and YouTube where some videos are watched by millions of users [14]. We reject the idea of implementing a sequence generation algorithm that accommodates the worst case of these most popular streams because it makes the leak identification unnecessarily long for the vast majority of video streams. Our goal is to make the sequence length *adaptive* to the number of users. Moreover, since the live stream is infinite, we consider infinitely repeated sequences of watermarks, similarly as in the earlier work from Jarnikov and

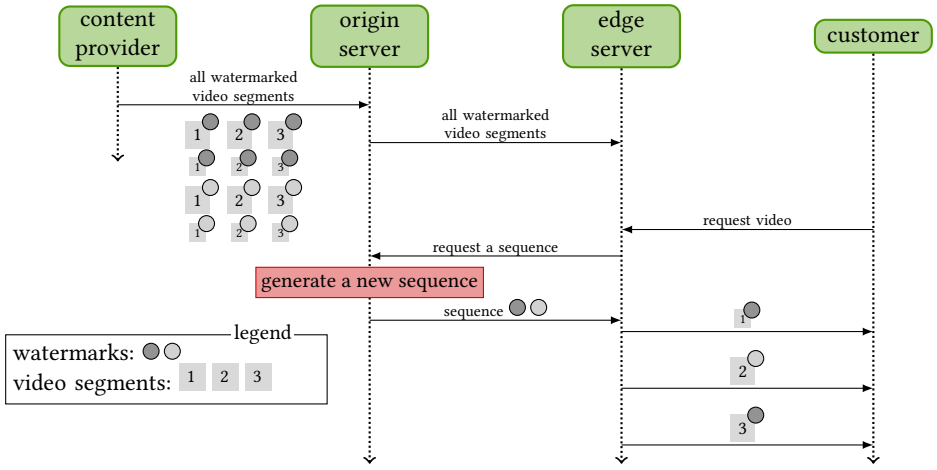

Figure 2: Example of a delivery for a new customer. Here there are two video representations ($n = 2$), two watermarks ($m = 2$), and the length of the sequence is two ($\ell = 2$)

Doumen [5]. Finally, we study how the server copes with a large number of clients concurrently joining the live stream (typically due to social network effects). We thus look for a fast sequence generation algorithm, for example to enable a streaming server to deal with thousands of new clients per seconds.

3 GENERAL DELIVERY ARCHITECTURE

We propose an architecture for live adaptive video streaming, based on CDN, where videos are watermarked and customers associated with unique sequences. This architecture is inspired by the work from Jarnikov et al. [6].

3.1 Architecture for Stream Delivery

The steps for the delivery of a watermarked video stream to a user is depicted in Figure 2 and described as follows:

Content Provider encodes the raw video into multiple video representations $R_1, R_2, \ldots, R_n$, each being characterized by a bit-rate, a resolution, and more generally quality parameters. It then embeds m different watermarks $1, 2, \ldots, m$ respectively in each representation R_i ($i = 1, 2, \ldots, n$) to generate *watermarked representations* R_{ij} ($j = 1, 2, \ldots, m$), each of them being then split into segments. Each segment contains complete Group of Pictures (GOP) structures, the duration typically ranging from 2 to 10 seconds. All the watermarked video segments are streamed to the origin server of the CDN.

Origin server generates one unique sequence s of watermarks of length ℓ ($\ell \in \mathbb{N}$), which is then associated to one customer. The sequence s consists of ℓ identifiers of watermarks, which are chosen among the m different watermarks, i.e. $s \in \{0, 1, \ldots, m-1\}^\ell$.

Edge Server delivers the video stream, segment per segment, and takes into account the sequence for each user. When a new customer requests the video, the closest edge server first asks the origin server for one sequence s. The origin server sends s back to the edge server and records the corresponding customer of this sequence s. Then, the edge server sends the different video segments by alternating the watermarked segments (with respect to the representation selected by the customer) in accordance to s.

3.2 Discovery of the Leak

The process for discovering the leak is as follows:

(1) The legal service provider detects a copy of its own video stream on an illegal video streaming service. It downloads this illegal stream during a *detection period* from the illegal service. Let us denote by k the number of segments downloaded by the service provider and by $\ell_{\max}$ the length of the longest sequence. Since the detection begins at any point in the stream and we do not know the length ℓ of the sequence that is embedded in the stream in advance, the detection period should be long enough to make sure that (i) the sequences with maximum length ($\ell = \ell_{\max}$) can be identified, and (ii) sequences with $\ell < \ell_{\max}$ appear twice, so that they are not confused with longer ones. It results that the length of the detection period k should be greater or equal to $2 \times (\ell_{\max} - 1)$.
(2) The legal service provider extracts the watermark symbols one segment by one segment from the detection period, and it then extracts the sequence s from the detection period. For the case that $\ell < \ell_{max}$, it extracts the repeated sequence in the detection period. For the case of sequences that are ℓ_{max}-long, it extracts the first ℓ_{max} watermarks. In Figure 3, we show such an extraction in two cases, one (upper case) where the length of extracted sequence is four (a 4-long sequence), while the length of the extracted sequence in the other case (lower case) is three (a 3-long sequence).
(3) The origin server seeks the customer that is associated with the extracted sequence, and stops streaming to this customer in the legal platform.

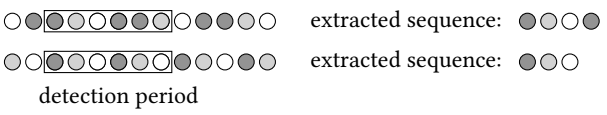

Figure 3: The detection of two watermark sequences in two different video streams. The service provider extracts $k = 13$ segments, the maximum sequence length $\ell_{\max} = 4$, so the detection period should contain at least 6 segments. No repetition in the upper case, the extracted sequence is $\ell_{\max}$-long, while a 3-long sequence repeats in the lower case.

3.3 Trade-offs and Optimal Problems

The legal service provider should address the following trade-off: The number of customers that can be associated with unique watermark sequences depends on both the number of different watermarks m, and the length of the sequences $\ell_{\max}$. The more watermarks (m large), the more identifiable customers for a given length $\ell_{\max}$, but the higher the preparation cost and the inner delivery cost. In addition, the longer is the sequence ($\ell_{\max}$ large), the more customers can be identified for a given number of watermarks m, but the longer the time needed to discover the leak is. Indeed, the detection period increases linearly with the sequence length ℓ.

The solution that we propose is (1) to provision the infrastructure once, so the number of watermarked versions of the video stream m is fixed, typically at a small value ranging from two to five, (2) to increase $\ell_{\max}$ in an iterative manner with the growth of the population. That is, the service provider starts with $\ell_{\max} = 1$, which enables the identification of m different customers, and, when the $(m+1)$th customer requests the video, the server increases $\ell_{\max}$ to generate new watermark sequences.

Our goal in this paper is to provide the algorithms that generate sequences for any number of watermarks m and any sequence length $\ell_{\max}$ subject that the sequences for m and any ℓ from 1 to $\ell_{\max} - 1$ have been generated.

4 MODELS

We denote by $S(m, \ell)$ the number of unique watermark sequences that can be generated for m watermarks and sequences of length ℓ.

4.1 Number of New Unique Sequences

Our first objective is to quantify the number of unique sequences that can be generated for a given couple (m, ℓ) of watermark numbers and sequence length. In a system where the sequence length $\ell_{\max}$ is fixed, the overall number of generated sequences for m different watermarks and ℓ-long sequences is m^ℓ. However, in our system, we have to discard two types of sequences.

The first type of sequences that should be discarded contain a *periodic pattern*. Let us consider a sequence $s = w_1 \dots w_\ell$. We say that s contains a k-long periodic pattern when the pattern $w_1 \dots w_{k-1}$ is reproduced ℓ/k times in s. Formally:

$$w_{i\times k} \dots w_{i\times k+k-1} = w_{j\times k} \dots w_{j\times k+k-1}, \quad \forall i, j \le \ell/k$$

The sequences containing a periodic pattern should be discarded because they have already been generated for smaller values of ℓ. For instance, let ℓ be equal to four, the sequence may be ●○●○. A periodic pattern with length two can be detected in this sequence, and obviously this pattern ●○ has already been generated as a sequence when $\ell = 2$. More generally, all sequences that have been generated for d among the divisors of ℓ should be discarded. Thus for $\ell = 12$, the sequences that contains 1-long, 2-long, 3-long, 4-long, and 6-long periodic patterns are discarded.

The second type of sequences that should be discarded are circular combinations of another. When the service provider extracts sequences from the detection period, it does not know the "starting watermark" of the extracted sequence. Let s be a sequence ●○●●. Then s cannot be distinguished from s_0: ○●●●, from s_1: ●●○●, and from s_2: ●●●○. More generally, one sequence s cannot be distinguished from $\ell - 1$ sequences and we should discard them.

With respect to these system-related reasons behind the generation and the discarding of sequences, we obtain the following formal result.

Proposition 4.1. *For any positive integers m and ℓ, we have*

$$m^\ell = \sum_{d \text{ divisor of } \ell} d\, S(m, d), \tag{1}$$

therefore we can iteratively compute the values of $S(m, \ell)$ as

$$S(m, \ell) = \frac{1}{\ell}\left(m^\ell - \sum_{d \text{ divisor of } \ell, d<\ell} d\, S(m, d)\right). \tag{2}$$

Proof. Let us consider a sequence $s = w_1 \dots w_\ell$, of length ℓ. There are m^ℓ such different sequences.

We want to exploit the fact that all ℓ circular permutations of s, obtained as $w_{\ell-k} \dots w_\ell w_1 w_2 \dots w_{\ell-k-1}$ for $k = 0, \dots, \ell - 1$, are different if and only if s does not contain a i-long periodic pattern $w_1 \dots w_{\ell/i}$, for a divisor $i < \ell$ of ℓ.

- The "only if" part is obvious: if s contains an ℓ/i-long periodic pattern, a circular permutation of ℓ/i watermarks gives the same sequence s.
- To establish the "if" part, assume that s and the circular permutation of s from $k < \ell$ watermarks, denoted by $\bar{s}$, are identical. Then, any other permutation of k watermarks also gives $\bar{s}$. Besides, s being an ℓ-long sequence, any permutation of ℓ watermarks also gives $\bar{s}$. Consequently, for any integer numbers a and b, a permutation of $ak + b\ell$ watermarks gives $\bar{s}$, and by Bézout theorem, a permutation of $i := \gcd(k, \ell)$ watermarks also gives $\bar{s}$. Hence the first i watermarks of $\bar{s}$ are identical to the next i watermarks, and so on. Since i divides ℓ and $i < \ell$, the property is proven and s contains a periodic i-long pattern.

Now let us use that property: for any sequence s, take the shortest periodic pattern constituting s (it is then of length ℓ/i for i a divisor of ℓ, possibly $i = 1$). Then that pattern, being the shortest, does not contains any shorter patterns, and has therefore been created when generating ℓ/i-long sequences: by assumption there are $\ell/i \times S(m, \ell/i)$ different permutations from such sequences, where the multiplication by ℓ/i comes from our property. Each permuted pattern, repeated i times, gives a different s.

Summarising, counting all ℓ-long sequences gives $\sum_{i \text{ divisor of } \ell} \ell/i \;\; S(m, \ell/i)$, which should equal m^ℓ. This

establishes (1), by noting that i divides ℓ if and only if ℓ/i divides ℓ. The second part of the proposition is straigthforward. □

In particular, Proposition 4.1 gives $S(m,1) = m$, $S(m,2) = \frac{m^2-m}{2}$, and if ℓ is prime, $S(m,\ell) = \frac{m^\ell - m}{\ell}$.

4.2 Algorithm

The algorithm to generate one new sequence must be fast enough to cope with the arrival of new customers. For a popular service provider, the peak of new customers joining a stream can be several hundreds per second [14]. Moreover, the popularity of these streams means that the sequence length $\ell_{\max}$ is large. We should thus design fast algorithms.

We observe that the watermark sequence can be analyzed as being an *integer number*, which is expressed in a *base* (the number of watermarks). That is, if ● means 1 and ○ is 0, then any sequence of ● and ○ can be seen as an integer expressed in a base 2.

To generate one new sequence, the service provider has to know the "latest" sequence length $\ell = \ell_{\max}$, the base m and i, which is the integer that corresponds to the latest sequence that has been generated. The algorithm then iterates for integers larger than i until it finds a sequence that can be generated. Let $j > i$ be the tested integer. The algorithm has two phases.

First, the algorithm should check whether the sequence contain patterns or not. The algorithm is described in Algorithm 1. Let s be the ℓ-long sequence associated with the tested integer j. Let d be a divisor of ℓ. To test whether s contains d-long periodic patterns, we need to compare the d first symbols of s with the next d symbols and so on ℓ/d times. For example, let us assume $\ell = 12$ and divisor 4, we compare $w_1w_2w_3w_4$ with $w_5w_6w_7w_8$ and $w_9w_{10}w_{11}w_{12}$. If the s does not contain any d-long patterns, we switch to the next divisor, until we complete all divisors of ℓ. On the contrary, if s contains a d-long pattern, then we discard the sequence s and we iterate by testing $j+1$ and its associated sequence s'.

Algorithm 1 check periodic patterns

Input: Length of sequence ℓ, last integer i, the number of watermarks m
Output: The next integer that does not contain any periodic pattern

```
begin
    j = i + 1
    while true do
        s ← compute the sequence from integer j
        for d in divisors of ℓ do
            pattern ← s_1 s_2 ... s_{ℓ/d}
            for k ∈ {2, ..., d} do
                pat ← w_{1+k×ℓ/d} ... w_{(k+1)×ℓ/d}
                if pat ≠ pattern then
                    return j
        j = j + 1
        if j > m^ℓ then
            ℓ = ℓ + 1
```

In the second phase, the algorithm checks whether the sequence s related to the integer j is not a circular permutation of a smaller integer. If so, it would mean that this sequence s has already been generated and associated to a customer. The pseudocode is given in Algorithm 2. The idea is to create a new sequence s' by a circular permutation of s from k positions, for every $k \in \{1, \ldots, \ell-1\}$, to compute the integer j' associated to s', and to check whether j' is smaller than j.

Algorithm 2 check circular permutation

Input: Length of sequence ℓ, a pattern-less integer j
Output: Either the index of the sequence or a failure

```
begin
    k ← 0
    w_0 w_1 ... w_ℓ ← compute the sequence from integer j
    while k ≠ ℓ do
        k ← k + 1
        s' ← w_{ℓ-k} ... w_ℓ w_1 w_2 ... w_{ℓ-k-1}
        j' ← integer computed from s'
        if j' < j then
            return failure
    return j
```

5 EXPERIMENTAL RESULTS

The goal of the evaluation part is to check whether the algorithm is fast enough to deal with a high number of incoming users and how fast is our approach to identify the leak in various scenarios. We implemented our algorithm on Python and the tests were performed on an Intel Core 2 Duo CPU running at 1.86 GHz with 4 GB memory.

In Figure 4, we show the relation between the time to generate new sequences and the values of ℓ and m. In our test, for $m \in \{2,3,4,5\}$, we calculate the execution time to generate one sequence for each ℓ. Since the time needed to generate a given sequence depends on the previously generated sequence, we generated ten thousands times randomly picked sequences (for every sequence length and every number of watermarks) and we compute the average time. The results show that this algorithm supports thousands to ten thousands new clients per second with execution time always below 500 μs (although using standard hardware and not optimized programming). In Figure 4 we also show the linear growth of the sequence generation time and we observe that the number of watermarks has a lower impact on the sequence time. Finally, we show that some sequence lengths are longer to generate due to the number of divisors.

In Figure 5, we show the relation between the number of segments in the detection period and the number of identifiable users with a logarithmic scale for the y-axis. We can see here typical trade-offs for live streaming providers. For videos that are watched by billions of users, the number of watermarks we suggest is typically five (which means multiplying by five the preparation cost and the inner delivery costs in the CDN) to enable leak period in less than 300 s (5 minutes). On the contrary, for videos that are likely to be far less popular (for example thousands), two watermarks is enough to stick below the 5 minutes mark. This result demonstrates the interest for our approach based on adaptive sequence length.

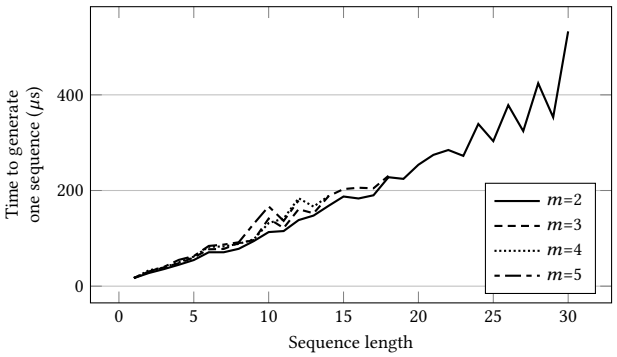

Figure 4: The execution time about generating one sequence with different length.

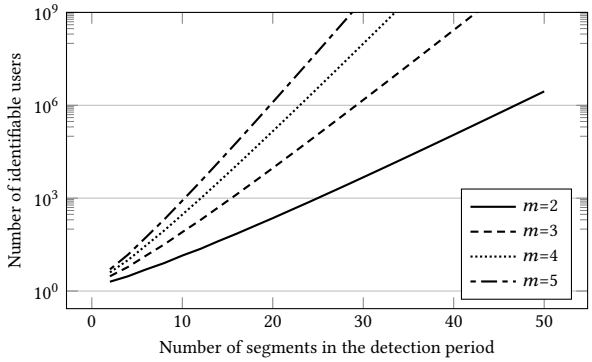

Figure 5: Number of detection users versus the number of segments in the detection period.

6 CONCLUSION AND PERSPECTIVES

In this paper, we revisit the problem of leak detection in video streaming systems by addressing the new problem of fast detection. We deal with live video re-streaming of sport events where the goal of the legal provider is to be as fast as possible to identify the leak. Another key new problem that we address is the fact that the number of legal users is not known in advance, so the watermark sequence generation should be adaptive to the number of clients. We present a delivery architecture and we propose an algorithm that is optimal in the time needed to detect one leak among any growing number of users. We show that our algorithm can be implemented in practice since it can support thousands of incoming new users, and that it is efficient, even to find a leak in less than five minutes among billions of clients. This work is to our knowledge the first one to address the fast detection of leaks in live streaming.

Our work opens various perspectives. First, our approach does not consider the collusion attacks, where two or more attackers combine their sequences together to generate a new one. This attack is much less frequent in live streaming than in VoD re-streaming because it requires the collusing leaks to synchronize and re-encode their video flows. The delay can be significant, which makes the problem less critical in the case of sport event where the liveness matters. Though, further studies should be conducted to find a trade-off between sequence length and collusion detection. Second, the reduction of the inner delivery cost in the CDN is our priority since the approach represents a significant extra-cost for the video providers (multiplying by two the required bandwidth), which already face bandwidth requirements over two terabytes per second [14]. We would like to study whether the popularity of representations and the aggregation of users in edge servers can be exploited to reduce the inner delivery cost, typically by not sending some segments to all edge servers, for example the least popular representations.

ACKNOWLEDGMENT

This work has been funded by Région Bretagne, Rennes Métropole and BPIFrance. The project includes industrial partners: Broadpeak and Nagra.

REFERENCES

[1] M. Adler, R. K. Sitaraman, and H. Venkataramani. Algorithms for optimizing the bandwidth cost of content delivery. *Computer Networks*, 55(18):4007–4020, 2011.
[2] E. Amiri and G. Tardos. High rate fingerprinting codes and the fingerprinting capacity. In *ACM SODA*, 2009.
[3] T. Bianchi and A. Piva. Secure watermarking for multimedia content protection: A review of its benefits and open issues. *IEEE Signal Process. Mag.*, 30(2):87–96, 2013.
[4] I. Brown, C. Perkins, and J. Crowcroft. Watercasting: Distributed watermarking of multicast media. In *Proc. of NGC Workshop*, 1999.
[5] D. Jarnikov and J. M. Doumen. Watermarking for adaptive streaming protocols. In *Proc. of Workshop on Secure Data Management*, 2011.
[6] D. Jarnikov, E. Hietbrink, M. Arana, and J. M. Doumene. A watermarking system for adaptive streaming. In *Proc. of IEEE ICCE*, 2014.
[7] H. Jin, J. Lotspiech, and S. Nusser. Traitor tracing for prerecorded and recordable media. In *Proc. of ACM Workshop on Digital rights management*, 2004.
[8] S. Katzenbeisser and F. Petitcolas. *Information hiding techniques for steganography and digital watermarking*. Artech house, 2000.
[9] I. Kopilovic, V. Drugeon, and M. Wagner. Video-dna: Large-scale server-side watermarking. In *Proc. of IEEE EUSIPCO*, 2007.
[10] T. Laarhoven, J. Doumen, P. Roelse, B. Skoric, and B. de Weger. Dynamic tardos traitor tracing schemes. *IEEE Trans. Info. Theory*, 59(7):4230–4242, 2013.
[11] P. Maillé, G. Simon, and B. Tuffin. Vertical integration of CDN and network operator: Model and analysis. In *Proc of IEEE MASCOTS*, 2016.
[12] J.-S. Pan, H.-C. Huang, and L. C. Jain. *Intelligent Watermarking Techniques:(With CD-ROM)*. World scientific, 2004.
[13] R. Parviainen and P. Parnes. Large scale distributed watermarking of multicast media through encryption. In *Proc. of IFIP CMS*. 2001.
[14] K. Pires and G. Simon. Youtube live and twitch: a tour of user-generated live streaming systems. In *Proc. of ACM MMSys*, 2015.
[15] B. Škorić, S. Katzenbeisser, and M. U. Celik. Symmetric tardos fingerprinting codes for arbitrary alphabet sizes. *Designs, Codes and Cryptography*, 46(2):137–166, 2008.
[16] T. Stockhammer. Dynamic adaptive streaming over HTTP -: standards and design principles. In *Proc of ACM SIGMM MMSys Conf.*, 2011.
[17] G. Tardos. Optimal probabilistic fingerprint codes. *Journal of the ACM (JACM)*, 55(2):10, 2008.
[18] L. Toni, R. Aparicio-Pardo, K. Pires, G. Simon, A. Blanc, and P. Frossard. Optimal selection of adaptive streaming representations. *TOMCCAP*, 11(2s):43:1–43:26, 2015.