Loss tolerant video streaming authentication in heterogeneous wireless networks

Gabriele Olieria,⇑, Stefano Chessa, Gaetano Giuntac

ISTI-CNR, Area della Ricerca di Pisa, Via G. Moruzzi 1, 56124 Pisa, Italy
Dipartimento di Informatica, Università di Pisa, Largo B. Pontecorvo 3, 56127 Pisa, Italy
Dipartimento di Elettronica Applicata, Università di Roma Tre, Via della Vasca Navale 84, 00146 Rome, Italy

Abstract

Multicast video streaming in heterogeneous networks undergoes to very different physical constraints, in fact, such networks are characterized by different QoS parameters, involving one or more transcod- ing processes between the sender and the receivers. Video streaming authentication algorithms must be robust to transcod- ing processes and must guarantee the copyright of the video owner/producer. We propose a real-time video streaming authentication algorithm that can guarantee the copyright of the video owner and that we prove to be robust to packet loss and transcoding processes.

1. Introduction

The rapid growth of network availability and bandwidth is encouraging video streaming applications such as movie-on-demand, IPTV, video conference and video surveillance. In such scenario the end-users demand for assurance that the received multimedia content comes from the claimed source and has not been manipulated by any unauthorized third party. Packet level authentication algorithms [1–3] are not suitable for multimedia streaming authentication, in fact, all of them rely on hash or MAC (Message Authentication Codes) computations, furthermore, they cannot deal with transcoding processes that can occur during the transmission between the sender and the receivers. Transcoding processes change the bit representation of the video stream but not the content, and therefore, any packet level authentication algorithm based on bit-by-bit authentication needs to be re-executed. Moreover, transcoding processes are an essential step of the broadcast video streaming scenario, that is, when a video stream is transmitted from a sender to various receivers, different networks are involved, with different QoS constraints, furthermore, changing the characteristics of the video streaming could become mandatory. Another important issue in broadcast video streaming scenarios is copyright. Packet level solutions cannot deal with it, since a third party can sniff the transmitted multimedia content, substitute the authentication information, changing the ownership of the video sample, and finally, retransmit the video sample to different receivers. Various solutions have been proposed for the copyright issue, and they are all based on watermarking techniques [4].

Digital watermarking is a multidisciplinary field that combines media and signal processing with cryptography, communication theory, coding theory, signal compression, theory of human perception, and quality of service requirements. In copyright watermark, the embedded signature watermarks the data with the owner's or producer's identification. This is the enabling technology to prove ownership on copyrighted material. There are mainly two different ways to embed data into a multimedia content for guaranteeing the copyright: (i) lossy, by putting a logo in each picture of the video stream, or (ii) no-lossy, by embedding a binary sequence in such a way that can be extracted with no errors. In [5,6], a non-lossy watermarking technique is proposed for transmitting hidden data from the sender to the receivers with no bandwidth overhead, that is, the data is embedded in the video stream by means of a watermarking technique with a negligible impact with respect to the video quality. Under this respect, the watermark can be considered as a reliable communication channel between the transmitter and the receivers [7,8]. The watermark can be used as a side communication channel, for transmitting from the sender to all the receivers the authentication information related to the video stream; moreover, the watermarked data is robust to transcoding processes and cannot be separated from the multimedia content without introducing visible artifacts. Two different approaches can be followed to authenticate multimedia content using watermarking as an out-of-bandwidth communication channel: hard authentication, and fragile authentication. The former
[9,10] achieves multimedia authentication by means of a hash computation performed over one or more pictures. By this way, each picture carries the authentication information (hash) for the subsequent one, and no way for the adversary to change one only bit of the video stream without breaking the authentication propagation. The latter [8] achieves authentication of the multimedia content tolerating little changes to the video quality. In this work we propose a new real time scheme for source and content video stream authentication. The proposed scheme has a negligible computational overhead since there is only one digital signature for a whole video stream, and it has no bandwidth overhead since all the authentication information are embedded in the video stream by means of watermarking. We highlight that our approach can be applied to an existing architecture without changing the currently used hardware, in fact only a software update (the protocol) is needed to the end users to enjoy the authentication procedure, whereas not-compliant users can still play the video content without authenticate it. We deal with transcoding processes and lossy links exploiting video content authentication, and we prove the algorithm robustness against malicious object injection [11]. The paper is organized as follows. Section 2 presents related works on authentication schemes and watermarking. Section 3 introduces a high level overview of the TESLA authentication protocol. Section 4 presents the scenario and the adversary model, while Section 5 gives a high level overview of the proposed solution in a simplified scenario. Sections 6 and 7 show the details for dealing with a generic heterogeneous wireless network, and finally, Section 8 shows the performance evaluation by means of experimental results. Conclusions are drawn in Section 9.

2. Related work

In order to protect multimedia content from unauthorized trading, many techniques have been developed. In [2], authors proposed to use $k$ different keys to authenticate every message with $k$ different MAC. Every receiver knows $m$ keys and can hence verify $m$ MAC. The keys are distributed in such a way that no coalition of $w$ receivers can forge a packet for a specific receiver. However, the communication overhead for this scheme is considerable. Other solutions [12–14] propose to amortize the digital signature by means of hash chains. However, none of these schemes is fully satisfactory in terms of bandwidth and processing time, especially in a setting where the transmission channel is lossy. In [1], authors proposed a new authentication scheme (TESLA) based on a weak synchronization between the sender and the receivers. The computational and communication overheads are low, packet loss is tolerated, but changing the video characteristics, for example by means of a transcoding process, implies to re-execute the algorithm.

Interesting solutions to deal with transcoding procedures come from hiding information and watermarking techniques. Hartung and Girod [6] proposed methods for embedding additive digital watermarks into uncompressed and compressed video streams. The first method regards inserting sequences of repeated bits (the redounded mark) in the spatial domain of the video image after their amplification and modulation by means of pseudo noise sequences. They also presented a marking procedure in the MPEG-2 bit stream domain, concluding that in that case watermarking scheme is less robust than its counterpart in the pixel domain. Cox et al. [15,16] proposed a spread spectrum watermarking scheme by embedding data in perceptually insignificant DCT coefficients of the cover image, and they presented a methodology that can be generalized to audio, video, and multimedia data. The basic idea is to spread narrow band watermark signal over many frequency bins of the host image by using pseudo random spreading sequences, so that the watermark energy content for each bin becomes small and can be hardly detected. At the same time, any attempt to remove watermark causes image impairments to an extent that fails to preserve the acceptable quality of the watermarked image.

All the previous techniques guarantee a side communication channel (with no bandwidth overhead) to broadcast information from a sender to a set of receivers, and they introduce a negligible quality degradation in the video pictures. Further, data hiding techniques guarantee that the embedded information survive to image manipulations like transcoding processes and geometrical distortions [17].

By this way, data hiding techniques have been combined with signature amortization to obtain more robust and efficient video stream authentication algorithms. Preliminary works have been proposed in [9,10], where a watermarking technique has been combined with a simple hash chain and an error correction code to obtain multimedia content authentication. The proposed approach is well suitable for the satellite broadcast infrastructure where packet loss is negligible due to project constraints.

A video multicast instant source authentication model based on digital video watermarking and TESLA is proposed in [18]. The authentication information associated to the current picture are directly embedded in the Modified Look-Up Table (MLUT) of the current I-Frame without coding/decoding the MPEG-2 video stream. This approach allows an instant authentication at the receiver side of the current I-Frame but cannot work if the video stream is transcoded between the sender and the receivers, in fact, the transcoding process wipes out the data structures of the compression algorithm. A secure and robust authentication scheme is presented in [8]: watermark is used to embed features extracted from the DCT domain of each video picture. Receivers extract the features and leverage them to authenticate the video content. Video features are computed in such a way that are invariant to transcoding algorithms, therefore the multimedia content can be transmitted and transcoded multiple times, that still carries with itself the authentication information.

Differently from the above approaches, in this paper we present a real-time video streaming authentication algorithm robust to packet loss and transcoding processes with a low computational overhead and a negligible bandwidth overhead. Moreover, despite previous solutions, in this work, we take into account packet loss resilience, real time constraints, and transcoding robustness all together. We combine the strengths of the TESLA algorithm (low computational and bandwidth overhead), with watermarking robustness to transcoding processes, obtaining a new flexible algorithm, that can be used for authenticating video streams transmitted through multiple heterogeneous networks.

We stress that this is a completely new authentication scheme. Only the watermarking embedding and extraction procedures are shared with the previous works [9,10], while the authentication scheme and the extensive measurement campaign to prove its feasibility have never been proposed before.

3. TESLA in a nutshell

The Timed Efficient Stream Loss-tolerant Authentication (TESLA) protocol provides small authentication delay, and robustness against packet loss. TESLA [1] assumes all the nodes are loosely time synchronized, i.e. receivers should be aware of a maximum synchronization error (calculated during the setup phase) between them and the sender. Sender transmits authenticated packets computing a MAC with an undisclosed key for each one of them. A predefined key disclosure schedule will be used for disclosing the key associated to the transmitted MACs. Loosely time synchronization allows receivers
to verify that the MAC key has not yet been revealed when it is received, the MAC and the related key have not been in flight in the same time (in this case the MAC is said to be safe). Each received packet is buffered for the key disclosure delay period, and if the MAC safety condition is met, the receiver leverages the MAC key to verify the authenticity of the packet.

During the setup phase, the sender picks up a random number by means of a pseudorandom function, and uses it as an anchor to generate a hash chain. The last generated key element is used as first, therefore, any node can compute forward keys but not backwards keys. Each key element is used as input to compute the MAC for each transmitted packet. The content of each packet is the following: the data payload, the MAC of the current packet, and the key used for the MAC of a previous packet (how far back depends on the key disclosure delay). Each packet is received and buffered, then, after the MAC key reception, the safe MAC verification is performed: this is possible because the sender and the receivers are loosely time synchronized, and key disclosure delay is compared with the MAC key reception delay. After the safe MAC verification, the key is used to verify the packet authentication through the MAC.

4. Scenario and adversary model

Our scenario consists of a heterogeneous wireless network constituted by different links which undergo different QoS constraints in terms of bandwidth, link latency, and finally, packet loss. A possible scenario is that in which a video stream is broadcasted by a real-time video stream server by means of a satellite network (or another long-range terrestrial wireless network such as UMTS), then the video is forwarded towards its destination by the receiver station by means of short-range wireless networks (for example based on WiFi or Bluetooth). End-to-end video stream authentication is challenging in this scenario, since packet level authentication protocols designed for long-range networks (or even for satellite links) may not be suitable for short-range links; therefore, we propose an authentication algorithm which works at the application level with no specific end-user hardware requirements. The authentication algorithm works on the real-time stream server and embeds the authentication information in the video stream by means of the watermarking technique, with a negligible computational overhead (there is one only digital signature computation), and with no bandwidth overhead (the authentication information are embedded in the video stream and are not perceptible to the human eye). The authentication procedure has to be robust to packet loss, survive to transcoding processes, and finally, to guarantee the video trustiness for clients who want to see the video asynchronously.

In our scenario, the adversary follows two main goals:

- **Change the video stream content:** Video stream content can be tampered either replacing one or more pictures, or adding new malicious contents to the transmitted video.
- **Change the video stream ownership:** Video stream ownership (copyright) can be tampered removing the existing authentication information from the video stream, and replacing them with new ones. In this way, the video content is still authentic but its ownership has been changed.

The adversary knows all the algorithms used both by the sender and by the receivers, he is able to detect and to read the watermark, but cannot remove or change it. As it will be explained in the next sections, reading and removing the mark implies the knowledge of a shared secret (the pseudo noise sequence used for embedding the mark by the sender), furthermore, the video stream content is always protected, but the ownership can be guaranteed only in that scenario, in which, the pseudo noise sequence is shared as a secret. Finally, the adversary can also add one or more own watermarks to the pictures belonging to the stream video.

5. Authentication overview and an example

In this section we present an overview of our framework that achieves real-time loss tolerant authentication based on TE-SLA [1]. The security overhead is negligible because authentication information are introduced into the video pictures by means of a watermarking technique. The bandwidth overhead sums up only to a digital signature and a single hash value, transmitted before and after the video stream, respectively. Fig. 1 shows the authentication scheme for a simplified scenario with a four blocks video stream. Each block $B_i$ with $i \in \{0, \ldots, 3\}$, is constituted by a sequence of pictures, and embeds the subsequent information by means of a watermarking technique: (i) the content features $F_i$ extracted from the current block $B_i$, (ii) a Message Authentication Code MAC$_{ci}$ computed using as input both $F_i$ and a secret key $k_i+1$, (iii) a secret key $k_i$. Each key $k_i$ is taken by the sender from a precomputed one-way hash chain. In fact, the sender precomputes a one-way hash chain $k_0, \ldots, k_4$, choosing randomly the last element of the hash chain $k_4$, and applying repeatedly a one-way hash function $H$ [19]. Fig. 2 shows the one-way hash function construction and use. The hash chain is revealed in the opposite way respect to its construction. In this way, if a key $k_i$ is lost, the receiver can compute it using the subsequent entry $k_{i+1}$, but a third party cannot compute the subsequent key $k_{i+1}$, knowing the current key $k_i$. Real time constraints are achieved combining a video features extraction algorithm with a delayed authentication scheme, that is, each block carries the authentication information for itself that will be evaluated by the receiver after a fixed discusture delay. Packet loss robustness is guaranteed by the one-way hash chain. In fact, no matter if one or more video blocks are lost, the first received block $B_0$ can be authenticated by the receiver, applying repeatedly the hash function $H$ over the current key $k_i$, until an authenticated entry of the one-way key chain is reached. For example, if block $B_1$ is lost, and $B_0$ is the last authenticated block, the receivers can authenticate $k_0$, and consequently $B_3$, applying repeatedly the
one-way hash function \( H(\cdot) \) to the current key \( k_3 \) until they reach the authenticated key \( k_0 \).

We want to stress that real-time constraints, packet loss and transcoding robustness are presented here for the first time. Previous solutions [9,10] based on both watermarking and bit-by-bit authentication cannot deal with packet loss or transcoding processes due to the “hard authentication” nature of the proposed schemes. Moreover, those schemes need an off-line analysis of the overall multimedia content: the authentication information is embedded starting from the last part of the video stream and going in reverse till the first picture. Nevertheless, we share with those approaches the low computational cost and the reduced bandwidth overhead.

Note, however, that the simplified scenario in Fig. 1 cannot work if two or more blocks are on flight in the same time between the transmitter and the receivers. In fact, a third party could forge a new video block, for example \( C_1 \), by using the authentication information of the block \( B_1 \), and by transmitting the forged block \( C_1 \) to the receiver, before the original block \( B_1 \) is received. To avoid the video block forging attack, the key \( k_2 \) and the related MAC\(_0\) must not be on flight in the same time. Therefore, we define a key disclosure delay \( \Delta \), between the MAC\(_0\) reception and the transmission of the related disclosure key. In this way, the video block carrying the MAC comes before the video block carrying the disclosure key of the related video blocks. The respect of this rule can be committed to the pre- foreclosure for trusting the safety of the video block itself. During the set-up phase, the transmitter (i) evaluates the number of frames per second \( \text{fps} \) of the video stream, (ii) evaluates the number of pictures per GOP \( N_{PG} \) and (iii) sets the number of GOPs per video block \( N_{GB} \) (this is an embedding parameter of the watermarking algorithm), and finally, evaluates the highest link latency \( \lambda_{\text{max}} \) among the various receivers. Subsequently, the transmitter evaluates the number of pictures in a video block as \( N_{PG}\times N_{GB} \), computes the video block rate as \( \frac{\text{fps}}{N_{PG}\times N_{GB}} \), and finally, sets the block disclosure delay as \( \Delta = \left[ \frac{\text{fps}}{N_{PG}\times N_{GB}} \right] \times \lambda_{\text{max}} \). Video blocks received before the \( \Delta \) safe delay are considered unsafe, they are not discarded but signaled as not authenticated to the video application. Table 1 shows the tracks of the authentication algorithm with a block delay \( \Delta = 3 \). The sender discloses the first key \( k_0 \) with the block \( B_1 \), the block \( B_i \) carry the authentic features \( F_i \), the MAC \( (F_i,k_i) \), and finally a disclosed key \( k_{i-\Delta} \) for the buffered block \( B_{i-\Delta} \).

### 6. Video streaming authentication

In this Section we present a novel real time video streaming authentication scheme robust to packet loss. Let \( S = \{B_0,\ldots,B_{N-1}\} \) be a video stream sequence of \( N \) blocks, where each block \( B_i = \{\text{GOP}_0,\ldots,\text{GOP}_{N_i-1}\} \) is constituted by \( N_{GB} \) Group of Pictures (GOP) \[20\].

Let \( F_i = FC(B_i) \) be the features extracted from the block \( B_i \) by means of the extraction algorithm \( FC \), and let \( W(\cdot) \) and \( E(\cdot) \) be the watermarking embedding and extraction algorithms. Let \( \text{MAC}_i = \text{MAC}(F_i,k_i) \) be a Message Authentication Code computed over two parameters: the secret key \( k_i \) and the arbitrary length message \( F_i \), respectively; let \( \text{RSA}(\cdot) \) be a public-key cryptosystem algorithm \[21\]; and finally, let \( H(\cdot) \) be the SHA-1 hash function \[19\].

#### 6.1. Sender side authentication algorithm

The sender precomputes a one-way hash chain \( k_0,\ldots,k_{K-1} \) of length \( K > N \), that is, the chain length must be greater than the number of blocks in the video sequence. It is worth noticing that key-length (\( K \)) does not limit the maximum transmission duration, in fact, in \[22\] authors show how to handle broadcast streams of unbounded duration by switching one-way key chains. The sender picks a random value for \( k_{K-1} \), and computes recursively the hash chain as \( k_i = H(k_{i-1}) \), with \( 0 \leq i < K - 1 \). Algorithm 1 depicts the sender side operations. Firstly, the sender transmits the last element \( k_0 \) of the hash chain by means of a secure channel, i.e. \( \text{RSA}(k_0) \); secondly, it builds the message \( M_1 \) (watermark) containing the authentication information for the current block \( B_0 \), and finally, the key disclosure for the previous block. The watermark is constituted by: (i) \( F_0 \), the features obtained by means of the extraction algorithm, i.e. \( F_0 = FC(B_0) \), (ii) the Message Authentication Code \( \text{MAC}_0 \), with input data \( F_0 \) and \( k_0 \) from the one-way chain and (iii) a key disclosure \( k_{1-\Delta} \). The key disclosure is performed if there are at least \( \Delta \) previously transmitted blocks. The watermark \( M_1 \) is embedded in the block \( B_1 \) obtaining the block \( B'_1 \), i.e. \( B'_1 = \{W(B_1,M_1)\} \). Finally, the watermarked block \( B'_1 \) is broadcasted.

**Algorithm 1.**

Sender side authentication procedure for the block \( B_i \).

<table>
<thead>
<tr>
<th>Time</th>
<th>Block</th>
<th>Feature</th>
<th>MAC</th>
<th>Disclosed key</th>
</tr>
</thead>
<tbody>
<tr>
<td>( t_0 )</td>
<td>( B'_0 )</td>
<td>( F_0 )</td>
<td>( \text{MAC}(F_0,k_0) )</td>
<td>( k_0 )</td>
</tr>
<tr>
<td>( t_1 )</td>
<td>( B'_1 )</td>
<td>( F_1 )</td>
<td>( \text{MAC}(F_1,k_1) )</td>
<td>( k_1 )</td>
</tr>
<tr>
<td>( t_2 )</td>
<td>( B'_2 )</td>
<td>( F_2 )</td>
<td>( \text{MAC}(F_2,k_2) )</td>
<td>( k_2 )</td>
</tr>
<tr>
<td>( t_3 )</td>
<td>( B'_3 )</td>
<td>( F_3 )</td>
<td>( \text{MAC}(F_3,k_3) )</td>
<td>( k_3 )</td>
</tr>
<tr>
<td>( t_4 )</td>
<td>( B'_4 )</td>
<td>( F_4 )</td>
<td>( \text{MAC}(F_4,k_4) )</td>
<td>( k_4 )</td>
</tr>
<tr>
<td>( \ldots )</td>
<td>( \ldots )</td>
<td>( \ldots )</td>
<td>( \ldots )</td>
<td>( \ldots )</td>
</tr>
<tr>
<td>( t_i )</td>
<td>( B'_i )</td>
<td>( F_i )</td>
<td>( \text{MAC}(F_i,k_i) )</td>
<td>( k_{i-\Delta} )</td>
</tr>
</tbody>
</table>
6.2. Receiver side verification algorithm

The receiver cannot authenticate directly the video blocks which are sent by the transmitter. In fact, each video block is authenticated by a key, that is disclosed after a delay of \( \Delta \) video blocks. For this reason, each received video block is authenticated after \( \Delta \) video blocks, and it carries the authentication information for a video block received earlier. The authenticated key authenticated by a key, that is disclosed after a delay of \( \Delta \).

Algorithm 2.

Receiver side authentication procedure for the video block \( B_i \).

```
Input: Received video block \( B_i \)
Output: Block Authenticity
Data:
let \( T_0 \) be the time arrival related to the first block \( B_0 \);
let \( t_i \) be the time arrival of the block \( B_i \);
let \( B \) be a buffer for storing the received but not yet authenticated blocks;
let \( j \in N \);
/* Buffering first \( \Delta \) blocks */
if \( i < \Delta \) then
    \( B \leftarrow B_i \);
end
/* Extract key \( k_{i,\Delta} \) from the current block \( B_i \) */
\( k_{i,\Delta} = E(B_i) \);
/* Check for \( k_{i,\Delta} \) safety */
if \( H^{-1}(k_{i,\Delta}) \neq k_0 \) then
    \( k_{i,\Delta} \) is not safe;
    No buffered blocks can be authenticated yet;
end
/* Check for safety of the previously buffered \( \Delta \) blocks */
if \( t_i < T_0 + \left( \left( \frac{\Delta}{2} \right) - 1 \right) \cdot \lambda_{\max} \) then
    MACs \( B_{i,\Delta} \) and all the previously buffered MACs are not safe;
    Block \( B_{i,\Delta} \) and all the previously buffered blocks are not safe;
end
/* Extract features \( F \) from the buffered block */
MAC = E(B_i);
\( F_j = E(B_i) \);
/* Check for \( F \) safety */
if \( MAC(F_j, H^{-(1)}(k_{i,\Delta})) \neq MAC_j \) then
    \( F_j \) is not safe;
    Block \( B_j \) is not safe;
end
```

If \( B_i \) is one of the first \( \Delta \) video blocks \( \{i \leq \Delta\} \), it is buffered. The first \( \Delta \) video blocks are buffered without authenticating them, in fact, the authentication of that blocks \( \{B_0, \ldots, B_{\Delta-1}\} \) starts after the reception of the video block \( B_\Delta \); which, in turn, is carrying the key disclosure \( k_0 \) that allows to authenticate the first block \( B_0 \). The authentication of the current disclosed key \( k_{i,\Delta} \) is performed applying recursively the hash function \( H(\cdot) \) until the authenticated anchor \( k_0 \) is reached. The trusted key \( k_{i,\Delta} \) is now used for authenticating the buffered MACs. Trusting the MAC relies on the verification that MACs and \( k_{i,\Delta} \) have not been in flight in the same time. After the reception of the first block \( B_\Delta \) at time \( T_\Delta \), the time \( \Delta \) is split into \( \Delta \) equally spaced intervals, with \( 0 < \delta < \frac{\Delta}{N}, \) of duration \( \lambda_{\max} \). Each time interval \( \delta \) contains \( \Delta \) video blocks. Keys belonging to time interval \( \delta \) disclose the MACs belonging to time interval \( \delta - 1 \). In this way, the video block \( B_i \) belonging to time interval \( \delta = \left( \frac{\Delta}{2} \right) \), must be received at time \( t_i \) after the time interval \( \delta - 1 \), i.e. \( t_i > T_\Delta + (\delta - 1) \cdot \lambda_{\max} \). If the current disclosed key \( k_{i,\Delta} \) carried by the block \( B_i \) is received before the time slot \( \delta \), it could have been in flight with the MACs, therefore MACs, and all the previously buffered MACs are declared as not safe. The features \( F_i \) are authenticated by means of the trusted MACs. No matters if there is one or more lost block, for example, if the current disclosed key \( k_{i,\Delta} \) refers to a MAC belonging to a lost block \( \{B_{i-1}, \ldots, B_{i-\Delta}\} \), the receiver can authenticate all the previous received blocks, i.e., \( B_j \), with \( j < i - \Delta \). The keys \( k_j \) used for computing the previous MACs can be recovered applying recursively \( \{i - \Delta \leq j \leq i - 1\} \) the hash function \( H(\cdot) \). After the \( F_i \) authentication step, the receiver can trust the block \( B_i \), extracting from it the features \( F \), i.e., \( F = FC(B_i) \); then, it compares the authentic features \( F_i \) with the computed ones \( F \) by means of the \( FC(\cdot) \) function, that is \( FC(F_i, F) \).

6.3. Features extraction and comparison

The features extraction algorithm \( FC(\cdot) \) is used both by the sender and by the receivers. The sender extracts the features \( F_i \) from the video block \( B_i \), that is \( F_i = FC(B_i) \), then it embeds the features \( F_i \) into the video block \( B_i \), by means of the watermarking algorithm \( W(\cdot) \), that is \( B_i = W(B_i, F_i) \). The receiver extracts from \( B_i \) the features \( F_i \) by means of the watermarking algorithm \( E(\cdot) \), that is \( F_i = E(B_i) \), and compares them with those extracted from the received block \( B_i \) by means of the features extraction algorithm \( FC(\cdot) \). Note that the original block \( B_i \) will be different from the received block \( B_i \), in fact, not only the watermarking algorithm and the transcoding process change the video block, but also bits corruption due to the channel transmission. Therefore a features extraction algorithm robust to transcoding and bit error rate is needed. We use the Canny filter algorithm [23] to extract information related to borders and edges from each video block.

The \( FC(\cdot) \) function is used to compare the authenticated features, \( F_i \), extracted from the block \( B_i \), by means of the watermarking algorithm \( W(\cdot) \), with those, extracted from the block \( B_i \), by means of the feature extraction algorithm \( FC(\cdot) \). The \( FC(\cdot) \) function.
function evaluates the difference between the features samples, and returns the Features Difference Indicator, hereafter FDI, an index that expresses the difference between the authentic features $F_i$ of the original block $B_i$ with that one extracted by the receiver from the block $B'_i$.

7. Watermarking

In this section we present the procedure for embedding and extracting the authentication information (the mark) from a video stream. As previously discussed in Section 1, our use of watermarking is not standard. Normally, it is acceptable that the extraction of the embedded logo be lossy, that is, the logo can be extracted with errors, provided that it remains recognizable. In our case, we use watermarking to embed authentication information. The extraction of the authentication information must not be affected by errors, otherwise the authentication information for the current block would be lost. For this reason, we propose an embedding technique that provides error-free mark extraction [6,9]. In the following section we describe the technique used to embed the authentication information $M$ into the video stream. In particular, we describe the functions $W(*)$, and $E(*)$ for embedding and extracting the mark, respectively.

7.1. Embedding the mark

Let $M_i = [F_i, MAC(F_i), k_{i-\lambda}]$ be the authentication information (mark) to be embedded into the video block $B_i$, $m_j$ be a symbol belonging to the mark $M_i$, with $m_j \in \{-1,1\}$, $0 \leq j < L_M$, and finally, $L_M$ be the size of the vector $M_i$. Due to specific features of the mark extraction (discussed in Section 7.2), the mark $M_i$ must be represented as a sequence of $\{-1,1\}$. For this reason the vector $M_i$ must be trivially converted into a binary sequence of $\{-1,1\}$ symbols. The mark $M_i$ is replicated according to a replication factor $Rep$, to obtain the spread sequence $H_i = [h_0, \ldots, h_{u_i}]$, where $L_H = Rep \cdot L_M$, that is:

$$m_j = h_j \cdot Rep \leq r \leq (j + 1) \cdot Rep$$  (1)

The purpose of spreading is to add redundancy by embedding one bit of information into Rep pixels of the video signal. The spread sequence $H_i$ is amplified by means of an amplification factor $\alpha > 0$, and is then modulated by a binary pseudo-noise sequence $n_i$, such that $n_i \in \{-1,1\}$, obtaining the amplified and modulated watermark $w_i$, that is:

$$w_i = \alpha \cdot n_i \cdot h_j$$  (2)

Let $v_i$ be the $r$th element of the line-scanned discrete cosine transform (DCT) [24] computed over the luminance components extracted from the sequence of pictures belonging to the block $B_i$. The watermarked video stream $\hat{v}_i$ is thus obtained as:

$$\hat{v}_i = v_i + \left|v_i\right| \cdot w_i$$  (3)

that must be anti-transformed by means of the Inverse-DCT (I-DCT) and re-arranged into a matrix for display.

The chance of successful extraction at the receiver side (Section 7.2) relies on the mark redundancy, therefore, the mark is replicated inside each picture by means of the Rep factor, and outside the picture, using the same mark for all the pictures belonging to the same block. If the number of pictures, in a video block $B$, are not sufficient to guarantee the redundancy, the video block dimension can be increased using more than one GOP for each video block, that is $NG_B > 1$.

7.2. Retrieval of information

The extraction of the symbol $m_j$ from a DCT-transformed and line-scanned video block $\hat{v}$ is performed demodulating the video signal by means of the same pseudo-noise sequence $n$ used by the transmitter.

$$\hat{v}_i \cdot n_i = v_i \cdot n_i + \left|v_i\right| \cdot \left|n_i\right| \cdot n_i = v_i \cdot n_i + \left|v_i\right| \cdot \alpha \cdot n_i$$

Let us consider the sum $s_j$ of all the $h_j$ symbols carrying the same watermark symbol $m_j$:

$$s_j = \sum_{i=1}^{(j+1)\cdot Rep-1} \hat{v}_i \cdot n_i = \sum_{i=1}^{(j+1)\cdot Rep-1} (v_i + \left|v_i\right| \cdot n_i) \cdot n_i$$

If the Rep factor is sufficiently high, the $\sum_1$ value becomes negligible due to the fact that $v_i$ and $n_i$ are two uncorrelated signals. The $s_j$ value becomes:

$$s_j = \sum_{i=1}^{(j+1)\cdot Rep-1} |v_i| \cdot \alpha \cdot n_i = \alpha \cdot Rep \cdot m_j \sum_{i=1}^{(j+1)\cdot Rep-1} |v_i|$$

The $m_j$ value can be computed applying the signum function $\text{SIGN}(s_j)$ to the $s_j$ symbol:

$$\text{SIGN}(s_j) = \text{SIGN}\left(\alpha \cdotRep \cdot m_j \sum_{i=1}^{(j+1)\cdot Rep-1} |v_i|\right) = m_j$$

in fact $\alpha$, $\text{Rep}$, and $\sum_2$ are positive values.

8. Performance evaluation

8.1. Costs and overheads

Bandwidth overheads can be estimated as in Fig. 1. Almost all the authentication information is embedded in the video content by means of the watermarking procedure. Nevertheless, it is still needed the transmission of one only digital signature and one only hash value before and after the video stream, respectively.

Computational costs are twofold: for the crypto functions and for the embedding/extraction watermarking procedures. The former sums up to a digital signature for the whole video stream, and two hashes for each video block. Note that the real-time computations at receiver side sum up only to the two hashes: MAC, and $k_a$, respectively. Watermarking embedding and extraction workloads basically rely on the DCT computations, but we want to highlight that DCT computation is also run by the MPEG-2 decoding algorithm.

8.2. Security analysis overview

In this section we present simulation results about the algorithm robustness to the video forging attack. Note that two different video forging attacks are possible. The first one has been examined in Section 5, and can be avoided tuning properly the video block disclosure delay, but, another forging attack could be possible escaping the feature comparison algorithm. In fact, the authentication scheme is based on features comparison, therefore, a third party cannot forge a new block using a completely different content but could forge a new video block by including not authenticated contents in the original one. We show how the $\text{FComp}(s)$ function prevents this, and we propose a method for distinguishing between a forged video and a video corrupted by the channel transmission.
All the video samples used in this paper (Akiyo, Foreman, Coastguard, and Stephan [25]) are in the CIF format ([352 × 288]), and have been watermarked in the uncompressed domain with a 512 bit length string, and then coded with the MPEG-2 compression algorithm (GOP length of 5 pictures, 25 frames per second) [20]. We assumed that the watermark length $M_i$ is less than 512 bit, that is, we used the HMAC-MD5 (128 bits) [26] for implementing the MAC($\phi$) function, the MD5 algorithm (128 bits) [27] for the hash function $H(\psi)$, and finally, we reserved 256 bits for a binary vector representing the features $F_i$ extracted from the block $B_i$. Note that, any other configuration can be used, that is, the extraction is still possible with a longer mark $M$ provided that a longer block length is used, see Section 7.

8.3. Forged video samples generation

We set up two different forging attacks: the superimposing and the picture in picture. The first one can be performed merging the original picture with a content not authentic, while the second one can be obtained by building an inset window picture on the original one. We use the merging factor $m_f$ for tuning the superimposing attack, for example, an $m_f = 0.2$ means that the forged video block is obtained merging the not authenticated content with a weight of 0.2 and the authenticated content with a weight of 0.8. Fig. 3 shows a still image obtained using a $m_f = 0.5$. We use the $a_f$ factor for tuning the picture in picture attack. The $a_f$ factor is the ratio of the inset picture area to the authentic picture area. Fig. 4 shows a still image obtained using $a_f = \frac{1}{8}$. Table 2 shows the forged video sequences used for testing the algorithm performances. The superimposing attack has been simulated considering an authentic video, Akiyo, and three other video samples Foreman, Coastguard, and Stephan, used as fake sequences. For each simulation we forged a video sample obtained by merging the Akiyo video sample with one of the faked video samples, for a total of three merged video samples (AF, AC, AS). The previous procedure has been performed using three different merging factors $m_f = 0.2, 0.5,$ and 0.8. The picture in picture attack has been performed similarly to the previous one. The three faked video samples Foreman, Coastguard and Stefan, have been resized and inserted in the middle of the authentic Akiyo video sample, one at a time. We used three different size by means of the $a_f$ factor: $\frac{1}{8}, \frac{1}{16}, \frac{1}{32}$.

8.4. Compare forged with corrupted video samples

In this section we show simulation results about the FComp($\phi$) algorithm performance, comparing the forged video sequences with a set of corrupted video samples. We compare the FDI, computed by means of the FComp($\phi$) over the forged samples, with those, computed over the corrupted video samples. We derive a threshold value that allows to distinguish between a forged and an authentic/corrupted video.

The corrupted video samples have been generated starting from the original Akiyo video sample and simulating the corruptions due to the streaming in a wireless channel. We simulated a Bernoulli channel (independent errors), and generated five corrupted video samples characterized by five different mean bit error rates: $10^{-6}, 5 \cdot 10^{-6}, 10^{-5}, 5 \cdot 10^{-5},$ and $10^{-4}$. We computed the $\{FDI_0, \ldots, FDI_{N-1}\} = FComp(Akiyo, Forged video)$, obtaining an FDI value for each block of the forged video sequence. Fig. 5 shows the quantile 5, 50, and 95 evaluated over the FDI values previously computed. Fig. 5 shows three different simulation results: (i) the superimposing attack, at the top, (ii) the picture in picture attack, in the middle and (iii) the corrupted samples, at the bottom. For example, fixing the superimposing attack, and the forged video sample (AF), increasing the $m_f$ factor, increases the FDI values. Similarly, increasing the bit error rate, increases the FDI values. Exploiting the FDI value, the algorithm can distinguish between a new object insertion and corruptions due to a noisy channel, that is, a bit error rate of $10^{-6}$, with $10^{-3} \leq FDI \leq 2 \cdot 10^{-3}$, can be easily identified respect to a new object insertion, nevertheless, increasing the bit error rate, increases the number of dubious cases. In our simulation scenario, a good threshold for distinguishing between an authentic but corrupted video and a forged video sample is $FDI = 8 \cdot 10^{-3}$.

<table>
<thead>
<tr>
<th>Table 2</th>
<th>Forged sequences configuration.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Authentic video</td>
<td>Fake video</td>
</tr>
<tr>
<td>Superimposing attack</td>
<td></td>
</tr>
<tr>
<td>$m_f$</td>
<td>Akiyo</td>
</tr>
<tr>
<td>Akiyo</td>
<td>Foreman</td>
</tr>
<tr>
<td>Akiyo</td>
<td>Foreman</td>
</tr>
<tr>
<td>Akiyo</td>
<td>Foreman</td>
</tr>
<tr>
<td>Akiyo</td>
<td>Coastguard</td>
</tr>
<tr>
<td>Akiyo</td>
<td>Coastguard</td>
</tr>
<tr>
<td>Akiyo</td>
<td>Coastguard</td>
</tr>
<tr>
<td>Akiyo</td>
<td>Stefan</td>
</tr>
<tr>
<td>Akiyo</td>
<td>Stefan</td>
</tr>
<tr>
<td>Akiyo</td>
<td>Stefan</td>
</tr>
<tr>
<td>Picture in picture attack</td>
<td></td>
</tr>
<tr>
<td>$a_f$</td>
<td>Akiyo</td>
</tr>
<tr>
<td>Akiyo</td>
<td>Foreman</td>
</tr>
<tr>
<td>Akiyo</td>
<td>Foreman</td>
</tr>
<tr>
<td>Akiyo</td>
<td>Coastguard</td>
</tr>
<tr>
<td>Akiyo</td>
<td>Coastguard</td>
</tr>
<tr>
<td>Akiyo</td>
<td>Coastguard</td>
</tr>
<tr>
<td>Akiyo</td>
<td>Stefan</td>
</tr>
<tr>
<td>Akiyo</td>
<td>Stefan</td>
</tr>
<tr>
<td>Akiyo</td>
<td>Stefan</td>
</tr>
</tbody>
</table>

Fig. 3. Example of superimposing attack using a merge factor $m_f$ equal to 0.5.

Fig. 4. Example of picture in picture attack using an $a_f$ factor equal to 1/8.

| Fig. 3. Example of superimposing attack using a merge factor $m_f$ equal to 0.5. |
| Fig. 4. Example of picture in picture attack using an $a_f$ factor equal to 1/8. |
we can consider them as authentic, at the most corrupted.

than 43.5 dB are characterized by very low inclusions, therefore
and an authentic block. Video blocks with a PSNR value greater

Fig. 6 shows the PSNR estimation for all the video samples pre-
viously considered. We evaluated the PSNR for each block of pic-
ture, the forged video sample approaches the original video
sample. By the same way, decreasing the bit error rate, increases
the FDI values, that is, bit-rate reduction introduces more and more
artifacts that are collected by the $F_{\text{Comp}(\cdot)}$ algorithm. For exam-
ple, reducing the bit-rate to 300 Kbps (for all the video samples),
puts the FDI values in the range $[2 \times 10^{-5}, ... 5 \times 10^{-7}]$. Bit-rate
reduction up to 300 Kbps ($\text{FDI} \leq 5 \times 10^{-2}$) can be easily confused
with a bit error rate of $5 \times 10^{-6}$ ($\text{FDI} \leq 10^{-2}$, Fig. 5), on the contrary
a bit-rate of 150 Kbps introduces artifacts that cannot be tolerated
by our algorithm, and can be confused for a new object insertion.

8.5. Robustness to transcoding processes

In this section we present simulation results about transcoding
robustness of the proposed approach, we focus on transcoding per-
fomed to the purpose of bit-rate reduction. Watermarking tech-
niques are proved to be robust to transcoding processes [8,29],
however, to validate the effectiveness of our approach, we have
to test the $F_{\text{Comp}(\cdot)}$ robustness to the bit-rate reduction. We ap-
ply a bit-rate reduction to the previously considered video sam-
ple, and we look at the FDI values returned by the $F_{\text{Comp}(\cdot)}$. In particu-
lar, starting from the raw video samples (uncompressed), we reduced
the bit-rate to 600 Kbps/300 Kbps, and finally 150 Kbps/
s. Fig. 7 shows the quantile 5, 50, and 95 associated to FDI values
returned by the $F_{\text{Comp}(\cdot)}$. Decreasing the bit-rate, increases the
FDI values, that is, bit-rate reduction introduces more and more
artifacts that are collected by the $F_{\text{Comp}(\cdot)}$ algorithm. For example,
reducing the bit-rate to 300 Kbps (for all the video samples),
puts the FDI values in the range $[2 \times 10^{-2}, ... 5 \times 10^{-5}]$. Bit-rate
reduction up to 300 Kbps ($\text{FDI} \leq 5 \times 10^{-2}$) can be easily confused
with a bit error rate of $5 \times 10^{-6}$ ($\text{FDI} \leq 10^{-2}$, Fig. 5), on the contrary
a bit-rate of 150 Kbps introduces artifacts that cannot be tolerated
by our algorithm, and can be confused for a new object insertion.

Fig. 6 shows the PSNR [28] estimation for all the video samples pre-
viously considered. We evaluated the PSNR for each block of pic-
tures, and we show, for each video sequence, the quantile 5, 50,
and 95 of the PSNR values. For example, considering the picture
in picture attack, the forged video sample approaches the original video
sample. By the same way, decreasing the bit error rate, increases
the PSNR. Comparing Fig. 5 with Fig. 6, it can be observed that the
FDI is a good estimator for the image changes. In particular,
we consider a PSNR threshold at 43.5 dB, having reference to
the $\text{FDI} = 8 \times 10^{-3}$, that can be used for distinguishing between a forged
and an authentic block. Video blocks with a PSNR value greater
than 43.5 dB are characterized by very low inclusions, therefore
we can consider them as authentic, at the most corrupted.

Fig. 5. Superimposing attack, Picture in Picture attack, and Corruptions vs. FDI.

Fig. 6. Superimposing attack, Picture in Picture attack, and Corruptions vs. PSNR.

9. Conclusions and future work

In this paper we have presented a novel approach to video
streaming authentication. The proposed scheme is robust to packet
loss and allows the sender to authenticate a real time video stream.
The authentication is based on a features computation algorithm,
the receiver compares the features computed over the current
block with the authenticated ones, embedded in the current block,
by the sender. Both the bandwidth overhead and the computa-
tional overhead are negligible, in fact, only one digital signature
for a whole video stream is used, and the authentication informa-
tion are embedded in the video stream by means of a watermark-
ing technique. Simulation results are presented, showing the
robustness of the authentication scheme to new object insertion,
packet loss, and finally transcoding processes.

We are currently investigating the use of the one time signa-
tures in place of the Message Authentication Codes.

References

of multicast streams over lossy channels, in: IEEE Symposium on Security and


