Chapter IV
Digital Video Watermarking and the Collusion Attack

Robert Caldelli
University of Florence, Italy

Alessandro Piva
University of Florence, Italy

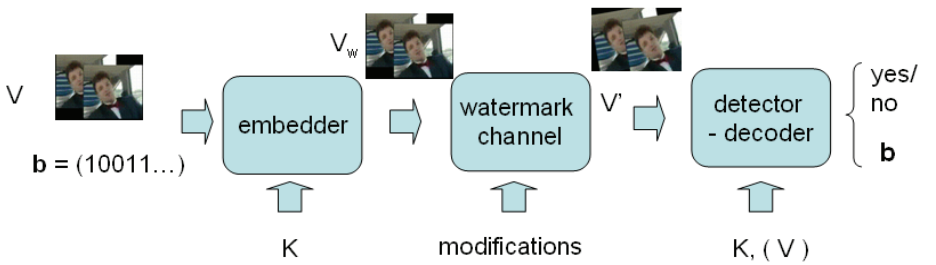
ABSTRACT

This chapter is devoted to the analysis of the collusion attack applied to current digital video watermarking algorithms. In particular, we analyze the effects of collusion attacks, with particular attention to the temporal frame averaging (TFA), applied to two basic watermarking systems like spread spectrum (SS) and spread transform dither modulation (STDM). The chapter describes the main drawbacks and advantages in using these two watermarking schemes and, above all, the fundamental issues to be taken into account to grant a certain level of robustness when a collusion attack is carried out by an attacker.

INTRODUCTION

Digital watermarking technology (Barni & Bartolini, 2004b; Cox, Miller, & Bloom, 2001) allows creators to hide a signal or some information into a digital content (an audio file, a still image, a video sequence, or a combination of the previous), usually named host data, that can be detected or extracted later by means of computing operations to make an assertion about the data. In the beginning the research was mainly devoted to offer a solution to the problem of copyright protection of digital content. In general, watermarking allows creators to provide a communication channel multiplexed into an original content, through which it is possible to transmit some information, depending on the application at hand, from a sender to a receiver.

Figure 1. The proposed model for a digital watermarking system



A digital watermarking system can be modeled as described in Figure 1 (Barni & Bartolini, 2004a). The inputs of the system are certain application dependent information, and the original host content is considered to be a video sequence *V*. The to-be-hidden information is usually represented as a binary string $\mathbf{b} = (b_1, b_2 \ldots., b_k)$, also referred to as the watermark code. The watermark embedder hides the watermark code **b** into the host asset *V* to produce a watermarked content V_w, usually making use of a secret information *K* needed to tune some parameters of the embedding process and allow the recovery of the watermark only to authorized users having access to that secret information.

The second element of the model, the watermark channel, takes into account all the processing operations and manipulations, both intentional and non-intentional, that the watermarked content may undergo during its distribution and fruition, so that consequently the watermarked content can be modified into a new version V_m.

The third element of the model is the tool for the recovery of the hidden information from V_m: the extraction of the hidden data may follow two different approaches: the detector can look for the presence of a specific message given to it as input, thus only answering yes or no, or the system (now called decoder) reads the sequence of bits hidden into the watermarked content without knowing it in advance. These two approaches lead to a distinction between *readable* watermarking algorithms, embedding a message that can be read, and *detectable* watermarking algorithms, inserting a code that can only be detected. An additional distinction may be made between systems that need to know the original content *V* in order to retrieve the hidden information, and those that do not require it. In the latter case we say that the system is *blind*, whereas in the former case it is said to be *non-blind*.

To embed the watermark code into the original content, watermarking techniques apply minor modifications to the host data in a perceptually invisible manner, where the modifications are related to the to-be-hidden data. The hidden information can be retrieved afterwards from the modified content by detecting the presence of these modifications. In general, embedding is achieved by modifying a set of features $\mathbf{f} = (f_1, f_2, \ldots, f_n)$ representing the host content with a watermark signal $\mathbf{M} = (m_1, m_2, \ldots, m_n)$ generated from the vector **b**, according to a proper embedding rule that depends on the particular watermarking scheme, as it will be described in the following.

If we consider the particular case of video watermarking, it has to be pointed out that a video sequence can be considered as a sequence of consecutive and equally time-spaced still images: Video watermarking issue seems thus very similar