H.264/AVC video watermarking for active fingerprinting based on Tardos code

Zafar Shahid, Marc Chaumont, William Puech

Signal, Image and Video Processing manuscript No.
(will be inserted by the editor)

H.264/AVC video watermarking for active fingerprinting based on Tardos code

Zafar Shahid · Marc Chaumont · William Puech

the date of receipt and acceptance should be inserted later

Abstract In this paper, we present a novel approach for active fingerprinting of state of the art video codec H.264/AVC. Tardos probabilistic fingerprinting code is embedded in H.264/AVC video signals using spread spectrum watermarking technique in both *luma* and *chroma*. Tardos code is embedded in *intra* as well as *inter* frames. The embedding has been performed in the non-zero quantized transformed coefficients (QTCs), which are above a certain threshold while taking into account the reconstruction loop, to avoid the uncontrollable increase in bitrate of video bitstream. A comprehensive analysis of payload and PSNR trade-off is presented for benchmark video sequences. Different linear and non-linear collusion attacks have been performed in the pixel domain to show the robustness of the proposed approach.

Keywords Tardos fingerprinting code, active video fingerprinting, H.264/AVC, spread spectrum watermarking, traitor tracing applications.

1 Introduction

Multimedia content can be easily copied and modified with the evolution of digital media in the recent past and due to it, concerns regarding its protection and authentication have also surfaced. While watermarking is used for copyright protection and encryption is used for restricted access, active fingerprinting is used for tracing the dishonest users in case of illegal distribution. In fingerprinting, a separate fingerprinting code, identifying a user, is embedded in the personal copy of each user using a robust watermarking technique. If a naive user distributes a copy of his fingerprinted content, then the pirated copy can easily be traced back to the guilty user and hence he will be exposed. Tracing the guilty user becomes more difficult when a collection of users called pirates form a coalition to detect the fingerprints and modify/erase them before illegally distributing the data. These collusion attacks can be linear and non-linear and fingerprinting codes must be

LIRMM Laboratory, UMR 5506 CNRS, University of Montpellier II
161, rue Ada, 34392 MONTPELLIER CEDEX 05, FRANCE
E-mail: firstname.secondname@lirmm.fr

resistant to such type of attacks. A fingerprinting code is designed for each user in a way that enables the distributor to identify **at least one of the pirates** as long as the coalition size does not exceed a certain threshold c, which is one of the design parameters for design of a fingerprinting code.

In the literature, the designs of these two technologies have been made separately, since both of them have evolved in different research areas. Active fingerprinting of digital content have been mostly studied by the cryptographic community and collusion models are thus defined on the sequence space since the pioneering work [5]. Watermarking has mainly been studied by people in the image or signal processing community. Hence, the effect of collusion of watermarked contents on the fingerprinting codes is not the same as the collusion in the fingerprinting sequence space. Recently, Desoubeaux *et al.* have embedded Tardos fingerprinting code to detect traitor's orthogonal zero-bit informed watermark for video content distribution [14]. Similarly, Tardos fingerprinting code is also used with zero-bit broken arrows watermarking by Xie *et al.* in [24]. The limitation of these two methods is that they are presented for raw sequences, and have not taken into account the challenges posed by compressed video content.

In this article, we are presenting the active fingerprinting applied to H.264/AVC video content in compressed domain. H.264/AVC [8] is the state of the art video coding standard of ITU-T and ISO/IEC. It offers better compression as compared to previous video standards. Like previous video standards, an input video frame can be encoded as *intra* or *inter*. In *intra*, spatial prediction is performed while in *inter*, motion compensated prediction is done from previous frames. H.264/AVC video is watermarked off-line in 2^q versions containing different symbols, where q are the bits being embedded in one independent coded unit (called *slice*). We encode a single *intra* frame in more than one *slices* to embed more than 1 bit in it. The online content server just provides the right *slices* according to the user fingerprint sequence. On the decoding side, fingerprint is first extracted, followed by the accusation process accusing some users (or nobody) based on this extracted sequence. First framework for video fingerprinting, which is based on Tardos fingerprinting code using spread spectrum embedding for H.264/AVC, has been already presented in [16]. This previous method proved to be resistant against collusion attacks. In this paper, we have enhanced the previously proposed approach in several aspects. We have used all modes for *intra* frames *i.e.*, intra 4x4, intra 16x16 and intra 8x8 mode. For *inter* frames, all motion estimations block sizes, which includes 16x16 up to 4x4 block sizes, are used. Furthermore, motion estimation up to quarter pixel accuracy has been incorporated in this framework. Spread spectrum embedding has been performed in DC as well as significant AC coefficients in a transform block. We have not inserted watermark in all AC coefficients for two reasons. First, it is not very robust to insert watermark in AC coefficients with low magnitude. Second, it results in relatively larger file size. In case of intra 16x16 modes, the spread spectrum embedding has been performed for Hadamard coefficients too. Hence, the framework presented in this paper is very close to a real-world video encoding system.

This paper has been arranged as follows. In Section 2, an overview of H.264/AVC has been proposed, along with an introduction to fingerprinting codes. Recent work on fingerprinting codes have been discussed in Section 3. In Section 4, we present the proposed algorithm which includes creation of Tardos fingerprinting codes and its embedding in H.264/AVC video using spread spectrum watermark-

ing technique. Section 5 explains the collusion attacks on fingerprinted video in the pixel domain and its performance analysis. It is followed by concluding remarks in Section 6.

2 H.264/AVC fingerprinting, challenges and prospects

Since significant changes have been incorporated in H.264/AVC as compared to the previous video coding standards, an overview of H.264/AVC with an emphasis on transform and quantization is presented in Section 2.1. It is followed in Section 2.2 by an introduction of fingerprinting code design techniques and marking assumption in Section 2.3. In this work, we have used capital letters to represent matrices e.g. A, Y, W and small letters along-with index to represent the elements of matrices e.g. $x(i,j)$ represents j^{th} element in i^{th} row of matrix X.

2.1 Overview of H.264/AVC

H.264/AVC [8] has some additional features as compared to previous video standards. In *baseline* profile of H.264/AVC, it has 4×4 transform in contrast to 8×8 transform of previous standards. DCT transform has been replaced by integer transform (IT) which can be implemented by only additions and shifts in 16 bit arithmetic without any multiplication and hence requires lesser number of computations. H.264/AVC codec uses a uniform scalar quantization. For *inter* frame, H.264/AVC supports variable block size motion estimation, quarter pixel accuracy, multiple reference frames, improved skipped and direct motion inference. For *intra* frame, it offers additional spatial prediction modes. All these additional features of H.264/AVC are aimed to outperform previous video coding standards [23]. The block diagram of H.264/AVC is shown in Fig. 1.

The 4×4 IT has two main advantages. Firstly, it can be implemented with additions and shifts in 16 bit arithmetic only. Secondly, in contrast to floating point arithmetic which gives different results on different platforms, there is no problem of mismatch on the encoder and decoder side for integer arithmetic. A macro-block (MB) is divided into 16 blocks of 4×4 pixels and they are processed one by one.

In *intra* mode, H.264/AVC has three modes, $Intra\_4 \times 4$, $Intra\_16 \times 16$ and I_PCM. In $Intra\_16 \times 16$ mode, Hadamard transform is further used to encode DC coefficients. In $Intra\_16 \times 16$ mode, entire MB is predicted from top and left neighboring pixels and has 4 modes namely *horizontal*, *vertical*, *DC* and *plane* modes. In $Intra\_4 \times 4$ mode, each 4×4 *luma* block is predicted from top and left pixels of reconstructed 4×4 neighbors and has 9 prediction modes. I_PCM mode is used to limit the maximum size of encoded block and is directly entropy encoded by bypassing the transform and quantization stages. The scanning of 4×4 blocks inside MB is not in a raster scan fashion as illustrated with the help of numbers in Fig. 2. In case of $Intra\_16 \times 16$ mode, Hadamard transform coefficients are sent first.

Let a 4×4 block is defined as $X = \{x(i,j)|i,j\epsilon\{0,3\}\}$, where $x(i,j)$ is a pixel as shown in Fig 1. First of all, each of the 16 pixels $x(i,j)$ are predicted from neighboring blocks and we get the residual block:

$$e(i,j) = P(x(i,j), b_1(i,j), b_2(i,j), b_3(i,j), b_4(i,j)), \tag{1}$$

where $P(.)$ is a predictions functions with reconstructed pixels from neighboring blocks and pixels of current block as parameters.

In H.264, intra prediction is performed from the reconstructed neighboring pixels, where $b_k(i,j)$ are the pixels from reconstructed neighboring blocks as shown in Fig. 3. Transform and quantization steps are performed together to save the processing power and to avoid multiplications. Once residual block E is computed, comprising of $e(i,j)$. This residual block is then transformed to transform domain using forward and inverse IT 4×4 matrices (A, A_{inv}), which are given as [15]:

$$A = \begin{bmatrix} 1 & 1 & 1 & 1/2 \\ 1 & 1/2 & -1 & -1 \\ 1 & -1/2 & -1 & 1 \\ 1 & -2 & 2 & -1 \end{bmatrix} \qquad A_{inv} = \begin{bmatrix} 1 & 1 & 1 & 1 \\ 2 & 1 & -1 & -2 \\ 1 & -1 & -1 & 1 \\ 1 & -2 & 1 & -1/2 \end{bmatrix}. \tag{2}$$

This residual block E is then transformed using the forward transform matrix A:

$$Y = AEA^T. \tag{3}$$

Scalar multiplication and quantization are defined as:

$$\hat{y}(u,v) = sign\{y(u,v)\}[(\mid y(u,v) \mid \times Aq(u,v) + Fq(u,v) \times 2^{15+Eq(u,v)})/2^{(15+Eq(u,v))}], \tag{4}$$

where $\hat{y}(u,v)$ is QTC, $Aq(u,v)$ is the value from the 4×4 quantization matrix and $Eq(u,v)$ is the shifting value from the shifting matrix. Both $Aq(u,v)$ and $Eq(u,v)$ are indexed by QP (quantization parameter). $Fq(u,v)$ is the rounding factor from the quantization rounding factor matrix. One can note from Eq. 4 that it takes only shift and addition operations to perform integer transform of H.264/AVC. This $\hat{y}(u,v)$ is entropy coded and sent to the decoder side.

On the decoder side, inverse quantization is given by the expression $y'(u,v) = \{[(\hat{y}(u,v) \times (Bq(u,v) \times 2^4)) \times 2^{Eq(u,v)}] + 2^3\}/2^4$, where $Bq(u,v)$ and $Eq(u,v)$ are the values from inverse 4×4 quantization matrix and the shifting factor respectively. $y'(u,v)$ is then inverse transformed to get $E' = (A_{inv} Y' A_{inv}^T + 2^5)/2^6$. The decoded residual signal $e'(i,j)$ is then added to the predicted signal to reconstruct the original signal back.

2.2 Fingerprinting codes

The design objective of a fingerprinting system is to provide a code construction that can trace at least one of the colluders with zero error. Unfortunately, for user $n \geq 3$, and colluders $c \geq 2$, no deterministic solution is possible and one has to opt for probabilistic solutions, allowing some low probability of error.

It is known [5] that any fixed assignment of fingerprints (a deterministic code) cannot satisfy this requirement: namely, there exist several attacking strategies of the coalition that will result in the error probability bounded away from zero irrespective of the decoding employed. For this reason it becomes necessary for the distributor to use probabilistic codes, where the random key is known only to the distributor.

In a probabilistic fingerprinting code, the following parameters are important:

- n: The number of users.
- c: The number of colluders.
- ϵ_1: False positive probability.
- ϵ_2: False negative probability.
- m: All the above four parameters result in a fingerprint code of certain length m.

The code length m influences to great extent the practical usability of a fingerprinting scheme, as the number of segments m that can be used to embed a fingerprint symbol is severely constrained; typical video watermarking algorithms for instance can only embed seven bits of information in a robust manner in one minute of a video clip [20]. Furthermore, distributors are interested in the shortest possible codes that are secure against a large number of colluders, while accommodating a huge number n of users (of the order of $n = 10^6$ or even $n = 10^9$).

Low error probabilities are another central requirement. The most important type of error is accusation of an innocent user called false-positive, denoted by ϵ_1. The probability of such an event must be extremely small; otherwise the distributor's accusations would be questionable, making the whole fingerprinting scheme unworkable. The second type of error is the false-negative, where the scheme fails to accuse any of the colluders, denoted as ϵ_2. In practical situations, fairly large values of ϵ_2 can be tolerated. Often the objective of fingerprinting is to deter unauthorized distribution rather than to prosecute all those responsible for it. Even a 50 % probability of getting caught can be a significant deterrent for colluders.

There are two main setups considered for the fingerprinting problem in the literature namely, the distortion setting and the marking assumption. In this section, we are presenting an overview of marking assumption fingerprinting model.

2.3 Marking assumption

The line of research into the construction of fingerprinting schemes followed in this manuscript applies to systems designed to protect digital content and is based on the following mathematical model:

Let the *content* is a sequence of symbols. Here, we consider a binary alphabet.

1. In the original *content*, there are m locations which we can change the symbol without significantly degrading the content. The codeword, a binary sequence of length m identifying a user will be hidden in the content to such sites.
2. The pirates do not know *a priori* fingerprint positions. The pirate coalition attempts to uncover some of the fingerprint positions by comparing their copies of the data for differences. Once such a difference is located in some position, it is guaranteed to be a fingerprint position. The comparison procedure does not reveal any information regarding the bits that are identical in all the data copies of the coalition members, which can be either information or fingerprint digits.
3. The pirated copy is created by modifying only those positions which are different in the fingerprinted copies of the pirates.
4. The process of accusation knows these places and sample the pirated copy of a sequence, called pirate sequence of m symbols.

It is important to mention that the original unmarked object is never distributed or released. This is especially true with digital content, since simple binary comparison on a marked object versus the original will reveal the locations of all marks even when the marks are not perceivable by human visual system.

3 Recent work on fingerprinting codes

In the literature, several fingerprinting algorithms have been proposed for images. Anti Collusion Code (AAC) was the first anti-collusion forensic code for multimedia content, proposed in [22]. It was based on combinatorial theories with a joint coding and embedding framework. The code is derived from balanced incomplete block design, which is then modulated by Gaussian orthogonal spreading sequences. In [10], ECC-based forensic code was proposed which uses Gaussian sequences to modulate symbols in the codeword along with additive spread spectrum embedding. ECC-based codes are the concatenation of an inner code and an outer code. By saving the inner codes and outer codes, we can easily construct the forensic codes for all the users. In [12], Lin *et al.* have proposed improved ECC-based anti-collusion codes for images which consume fewer resources than Tardos fingerprinting codes.

Tardos [21] constructed fingerprint codes of length at most $m = 100c^2 \ln(\frac{1}{\epsilon_1})$ for c colluders. This construction yields c-secure fingerprinting schemes of rate $1/(100c^2)$. The same paper gave lower bound of $O(c^2 \ln(1/\epsilon))$ on the length of any fingerprint code with the above parameters. The constant 100 in the length $m = 100c^2 \ln(\frac{1}{\epsilon_1})$ was subsequently improved by several papers [18,17,4]. All these papers go through the proof in [21] and optimize various parameters in the proof to improve the code length without fundamentally changing the code construction or the accusation algorithm.

Skoric *et al.* [18] improved the code length if the case the error parameters are in certain intervals *i.e.* the probability of accusing an innocent user is assumed to be $10^{-15} \leq \epsilon_1 \leq 10^{-9}$ and the probability of failing to detect pirates is $0.1 \leq \epsilon_2 \leq 0.5$. For values within these intervals, the code length is calculated experimentally for $t \leq 100$ and it is observed that when t increases the code length approximates $4\pi^2 t^2 \log m$. This represents an improvement by a factor around $2:5$.

Skoric *et al.* [17] made a simplifying assumption about the pirate strategy. Even though the assumption is reasonable there is no mathematical justification for it. With this simplifying assumption they achieve an improvement by a factor of almost 10.

Blayer and Tassa [4] rigorously extract some formulas for the parameters in the proof of [21]. Their experiments confirmed an improvement by a factor of more than 4 and less than 5.

Anthapadmanabhan *et al.* [3] chose a different information theoretic approach. They construct t-secure fingerprinting schemes whose rates for t = 2 and 3 are much higher than previously obtained rates but the rate of their schemes deteriorates exponentially with t. They are the first to prove upper bounds on the rates of t-secure fingerprinting schemes. The upper bounds in their paper is given in terms of a hard to evaluate information theoretic minimax formula. They estimate this formula and prove strong upper bounds on the t-fingerprinting capacity for small values of c (namely 2 and 3) and an $O(1/c)$ asymptotic bound.

Amiri and Tardos [1] combine two approaches represented by [21] and [2] to obtain fingerprinting codes of rates higher than those achieved with either method separately. Codeword construction method is very similar to that of [21]. The optimal distribution is found through a game-theoretic equilibrium and prove the upper bound to be $1.78/t^2 + O(1/t^2)$. On the contrary, accusation algorithm is fundamentally inspired by the work of Anthapadmanabhan *et al.* [2]. For $t = 2$, their codes coincide with the codes given in [2] but for $t \geq 3$, their rates are better than the rates of previously obtained codes. For $t \leq 8$, they prove the $R_t \geq t^{-2}/(2ln2) + O(t^{-2})$ asymptotic bound. This improves the rates of the codes in [21] by a factor over 72 for large values of t.

In case of multimedia, the colluders usually apply post-processing after collusion. For instance, the colluders can compress the multimedia to reduce the data size to efficiently redistribute the colluded copy. Therefore, it is important to design a collusion resistant forensic code that is robust to post-processing. In [24], Tardos fingerprinting code has been used with zero-bit broken arrows watermarking scheme for images. They have shown that this combination has ruled out the fusion class of attacks. In this work, our framework composed of Tardos fingerprinting code and spread spectrum embedding for state of the art video codec H.64/AVC as explained in Section 4.

4 Proposed Algorithm

The proposed scheme is composed of two steps. In the first step, Tardos fingerprinting code is generated. While in the second step, embedding of Tardos fingerprinting code in H.264/AVC video is performed using robust watermarking technique. In Section 4.1, we present the Tardos fingerprinting code. The process of spread spectrum watermarking is illustrated in Section 4.2. Embedding of Tardos code in H.264/AVC is explained in Section 4.3. It is followed by Tardos code extraction from pirated video and accusation process in Section 4.4.

4.1 Tardos fingerprinting code generation

For code generation, we have the following three steps:

- For a code of length m, we generate random and independent probabilities $\{p(i)\}_{1 \leq i \leq m}$ with the distribution $f(p) = \frac{1}{\pi\sqrt{p(1-p)}}$ for $p \in [0, 1]$. Practically p is between 0 and t, or $1 - t$ and 1 with $t = 10^{-3}$, hence having high frequency on the edges as shown in Fig. 4.
- The next step is to generate Tardos code. For n users with m code-length, it is a matrix of size $m \times n$ as given in Fig. 5. For the case of binary Tardos code, each line of S is filled with 0 or 1 with $Prob[S(i, j) = 1] = p(i)$. Each column is a fingerprinting code for separate user.
- For accusation process, a sequence Z is extracted from the pirated copy and an accusation score A_j is associated with user j given as:

$$A_j = \sum_{i=1}^{m} U\left(Z(i), S(i, j), p(i)\right), \tag{5}$$

where

$$U(1,1,p) = \sqrt{(1-p)/p} \quad U(1,0,p) = -\sqrt{p/(1-p)},$$
$$U(0,0,p) = \sqrt{p/(1-p)} \quad U(0,1,p) = -\sqrt{(1-p)/p}.$$

A fingerprinting code is analyzed based on its code-length, maximum number of colluders c, false-positive (ϵ_1) and false-negative (ϵ_2) values. For binary asymmetric Tardos code, the length of code is given as $m = 100c^2 \ln(\frac{1}{\epsilon_1})$. The relation between ϵ_1 and ϵ_2 is given as $\epsilon_2 = \epsilon_1^{\frac{c}{4}}$ [21]. It is important that probability to accuse innocent user ϵ_1 should be very small. The code-length is further reduced in its symmetric version by Skoric *et al.* [19] and both ϵ_1 and ϵ_2 were made independent of each other.

4.2 Spread spectrum strategy

Several robust watermarking techniques exist in the literature. Spread spectrum watermarking technique offers robustness [7] and has been selected for our investigation. Spread spectrum embedding is resistant against a number of attacks. It was argued to be highly resistant to collusion attacks, when the watermarks have a component-wise Gaussian distribution and are statistically independent [9]. The basic intuition of this natural strategy is that the randomness inherent in such watermarks makes the probability of accusing an innocent user very unlikely. Spread spectrum embeds the watermark in overlapped regions and this spreading makes it challenging to change even a single bit at will. This confines the effect of a colluder's action to a milder form of collusion from the designer's point of view. Let $S(i,j)$ be the i^{th} bit of Tardos fingerprinting code S_j which is to be embedded into a block of host vector X. To increase the energy of the embedding bit, a scaling parameter α is used. So the watermarked block is given as:

$$Y = X + \alpha C(-1)^{S(i,j)}, \tag{6}$$

with $S(i,j) = 0$ or 1 and C is a bi-polar Gaussian sequence [1, -1]. The attacked watermarked signal is $Z = Y + n$, where n is the noise due to attack. The watermark bit $\tilde{S}(i,j)$ is extracted from Z by the linear correlation of Z and C of length l as:

$$\tilde{S}(i,j) = \begin{cases} 0 \text{ , if } \sum_{j=1}^{l} Z[j]C[j] > 0 \\ 1 \text{ , if } \sum_{j=1}^{l} Z[j]C[j] < 0. \end{cases} \tag{7}$$

4.3 Embedding strategy

For embedding a Tardos code in QTCs of H.264/AVC, embedding can be done in entropy coding stage. It is analogous to embedding watermark in a compressed bitstream. This includes two watermarking approaches. The first approach embeds watermark in VLC (variable length coding) entropy coding domain and bitstream is only to be entropy decoded to use this e.g. as proposed by Lu *et al.* [13]. Another approach embeds watermark in DCT domain and for this approach, bitstream has

to be entropy decoded and inverse quantized e.g. differential energy watermarking [11]. Embedding watermark, after reconstruction loop, creates two problems. First, we do reconstruction with QTC on the encoder side, while on the decoder side with watermarked QTC. This results in a mismatch on the decoder side, which keeps on increasing because of the prediction process and loss in PSNR (peak signal to noise ratio) is very significant even for *intra* frames. Second, Rate Distortion (RD) bit allocation algorithm works in quantization module and any change in bitrate/quality trade-off because of the watermarking of QTCs is not taken into account. To solve both problems, watermark embedding should be performed inside the reconstruction loop as shown in Fig. 6. In this case, we have the same watermarked QTC $\hat{y}_w(u,v)$ on both encoder and decoder side for prediction. In this case, RD bit allocation algorithm is working on $\hat{y}_w(u,v)$ for both *intra* and *inter* frames.

For embedding of m bits of Tardos code S_j, the content is divided into m independent coding blocks (*i.e.* slices) and 1 bit is embedded in each slice. For larger multimedia content, we can have slices of larger size and hence the embedding will be more robust.

To embed 1 bit of Tardos code BIT_{fp} using spread spectrum embedding strategy, a vector X of length l is formed by DC and AC QTCs, with magnitude above a certain threshold. In *intra* 4×4 mode, scanning of 4×4 blocks inside MB is not in a raster scan fashion. Hence we will create a vector X for spread spectrum while taking this scan into account for this mode as illustrated in Fig. 7. Spread spectrum embedding is performed only in significant DC and AC QTCs to avoid uncontrollable increase in bitrate and to make our embedding robust. Each bit BIT_{fp} of Tardos code is embedded into the host vector X using spread spectrum insertion. First of all, Tardos code is modulated with a bi-polar Gaussian sequence C and scaled by a scaling factor α. A threshold for spread spectrum embedding (TH-SS) is set for embedding this modulated sequence. We modify magnitude of QTCs with magnitude between TH-SS +1 and TH-SS $+\alpha$ to TH-SS. It is then followed by embedding the modulated bit by either increasing or decreasing the magnitude of QTC, depending on whether the modulated bit BIT_{fp} is 0 or 1. The whole process is explained in Algorithm 1. For example, let us perform spread

Algorithm 1 The embedding of one modulated bit BIT_{fp} of Tardos fingerprinting code with strength α in QTCs having magnitude above a certain threshold $TH - SS$.

for $i = 1$ to l **do**
  if $|QTC| == (TH - SS + i)$ **then**
    $|QTC| \leftarrow TH - SS$
  end if
end for
if $|QTC| > TH - SS$ **then**
  if $BIT_{fp} == 1$ **then**
    $|QTC| \leftarrow |QTC| + \alpha$
  else
    $|QTC| \leftarrow |QTC| - \alpha$
  end if
end if
end

spectrum embedding with TH-SS = 5 and $\alpha = 2$. As a first step, we will modify

all the QTCs having magnitude 6 or 7 to 5, as shown in first part of the algorithm. In the second step, we will watermark the QTCs with magnitude greater than 5. For example, if QTC is 8, it will be modified to 10 (*i.e.* 8 + α), if the bit to be embedded is 0 or 6 (*i.e.* 8 - α) if the bit to be embedded is 1.

4.4 Extraction and accusation strategy

Extraction and accusation steps are explained in algorithm 2. From the pirated video, Tardos code is extracted and detection process is performed on the extracted code for each user:

- **Extraction step:** For the extraction process, modified spread spectrum extraction is used to extract a single bit of Tardos code BIT_{fp} from the pirated video sequence. First of all, a vector X' of length l is formed by the watermarked QTCs, with magnitude above the threshold TH-SS. In intra 16x16 mode, Hadamard coefficients are also added to this vector X'. While in intra 4×4 mode, scanning of 4 x 4 blocks inside MB is not in a raster scan fashion and the vector X' is created while taking this scan into account as illustrated in Fig. 7. It is followed by linear correlation of X' with bi-polar Gaussian sequence C. If the linear correlation is positive, the extracted bit is '1', otherwise it is '0'. The extracted Tardos sequence Z is formed by concatenation of these extracted bits.
- **Accusation step:** Accusation process is performed on the extracted sequence Z wherein an accusation score A_j is calculated on Z for each user j as given in Eq. 5. A_j for accused users may be modeled with a Gaussian centered at $\mu = \frac{2m}{c\pi}$, while A_j for innocent users may be modeled with a Gaussian centered at 0. Accused users (the traitors) have a score above $\mu - \sqrt{m}$ (*i.e.* $\frac{2m}{c\pi} - \sqrt{m}$), where $\sqrt{m}$ is the standard deviation of the Gaussian. A more precise threshold may be selected as proposed in [6][1].

5 Experimental results

We have used the reference implementation of H.264 JSVM 10.2 in AVC mode, along with CAVLC entropy coding at QP value 18. Nine benchmark video sequences have been used in CIF (352×288) resolution. Each of them represents different combinations of motion (fast/slow, pan/zoom/rotation), color (bright/dull), contrast (high/low) and objects (vehicle, buildings, people). The video sequences 'bus', 'city' and 'foreman' contain camera motion while 'football' and 'soccer' contain camera panning and zooming along with object motion and texture in background. The video sequences 'harbour' and 'ice' contain high luminance images with smooth motion. 'Mobile' sequence contains a still complex background and motion in foreground. Since these sequences contains lesser number of frames, they have been used in repeated fashion.

For generation of binary Tardos code, the parameter values are: $n = 100$, $\epsilon_1 = 10^{-3}$, $c_0 = 20$ and $m = 92104$ [2]. 10 bits of Tardos code were embedded per

[1] It links ϵ_1 and threshold.

[2] Normally, in traitor tracing, $\epsilon_1 = 10^{-6}$. To reduce code-length and hence, the simulation time, we have selected $\epsilon_1 = 10^{-3}$.

Algorithm 2 The extraction and accusation step for the proposed technique wherein spread spectrum extraction is performed on pirated video content in the transform domain. It is followed by accusation step using Tardos fingerprinting code.

for $i = 1$ to l **do**
  if $|QTC| > TH - SS$ **then**
    $X'[i] \leftarrow |QTC|$ {Preparation of array of watermarked QTCs}
  end if
end for
{Extraction of single bit from pirated copy}
$corrValue \leftarrow Corr(X', C)$
if $corrValue > 0$ **then**
  $Z[i] \leftarrow 1$
else
  $Z[i] \leftarrow 0$
end if
{Accusation score A_j for user j}
$A_j \leftarrow 0$
for $i = 1$ to m **do**
  if $Z[i] = 1$ **then**
    if $S[i] = 1$ **then**
      $A_j \leftarrow A_j + Sqrt((1-p)/p)$
    else
      $A_j \leftarrow A_j - Sqrt(P/(1-p))$
    end if
  else
    if $S[i] = 1$ **then**
      $A_j \leftarrow A_j - Sqrt((1-p)/p)$
    else
      $A_j \leftarrow A_j + Sqrt(P/(1-p))$
    end if
  end if
end for
end

frame using robust spread spectrum watermarking. The fingerprinting code has been embedded in both *luma* and *chroma* components with scaling factor $\alpha = 1$. Hence we used 9211 frames (369 seconds of video at 25 fps) of CIF resolution to embed the code. For accusation process, an accusation sum A_j is calculated for each user j. A_j for accused users is considered a Gaussian centered at $\mu = \frac{2m}{c\pi}$, while A_j for innocent users may be modeled with a Gaussian centered at 0.

Section 5.1 explains the linear and non-linear collusion attacks. It is followed by analysis for *intra* sequences in Section 5.2, while *intra* & *inter* sequence is discussed in Section 5.3.

5.1 Collusion attacks

In this work, linear and non-linear collusion attacks have been performed in the spatial domain. For this purpose, the fingerprinted H.264 video content is decompressed and the collusion attacks are performed in spatial domain. The fingerprinted video is compressed again and detection is performed in the compressed domain. Hence the proposed technique is resistant against re-encoding attacks.

Linear attacks include averaging attack. Under these attacks, each pixel in pirated video is average of the corresponding pixels of the fingerprinted videos

associated with the colluders. The typical nonlinear collusion attacks are minimum/maximum/median attacks, minmax attack and modified negative attack. For minimum/maximum/median attacks, each pixel in pirated video is the minimum, maximum, or median, of the corresponding pixels of the fingerprinted videos. For minmax attack, each colluded pixel is the average of the maximum and minimum of the corresponding pixels of the fingerprinted videos. For modified negative attack, each pixel of the attacked video is the difference between the median and the sum of the maximum and minimum of the corresponding pixels of the fingerprinted video signals. Let S_c is a set of all K colluders, the collusion functions can be given as:

$$\begin{aligned} Z_{ave}(j) &= \sum_{k \in S_c} \frac{Y_k(j)}{K} \\ Z_{min}(j) &= min\{Y_k(j)\}_{k \in S_c} \\ Z_{max}(j) &= max\{Y_k(j)\}_{k \in S_c} \\ Z_{med}(j) &= med\{Y_k(j)\}_{k \in S_c} \\ Z_{minMax}(j) &= (Y_{min}(j) + Y_{max}(j))/2 \\ Z_{modNeg}(j) &= Y_{min}(j) + Y_{max}(j) - Y_{med}(j) \end{aligned}$$

where Z is the colluded pixel, and Y is the pixel of fingerprinted video.

5.2 Analysis of *intra* frames

In *intra* frames, we take advantage of the spatial masking to embed fingerprint in QTCs having magnitude above certain threshold. Watermark is naturally embedded in those parts of frames which contain texture and edges because they contains non-zero QTCs.

For experimental simulation, we used all intra prediction modes for $intra4 \times 4$, $intra16 \times 16$ and $intra8 \times 8$. Normally, magnitude of *luma* QTCs is relatively higher as compared to those of *chroma* QTCs. Hence value of TH-SS$_{uv(I)}$ should be slightly less than TH-SS$_{y(I)}$. In *intra* sequence, threshold TH-SS for *luma* and *chroma* components has been selected as:

$$\begin{aligned} \text{TH-SS}_{y(I)} &= \text{TH-SS} \\ \text{TH-SS}_{uv(I)} &= \text{TH-SS} - 1 \end{aligned}$$

Payload capacity and PSNR trade-off is a crucial feature for any watermark embedding system. Decrease in PSNR of all the video sequences due to fingerprint embedding is analyzed in Fig. 8. It is followed by payload analysis for all of three components Y, U & V in Fig. 9. One can note that payload and PSNR variation is different for different video sequences for both of *luma* and *chroma* components. For example, mobile video sequence has lot of texture and details, it has the highest payload among all the video sequences being used for this analysis. Based on this analysis, it is clear that it is not a good idea to have the same threshold for all the sequences. Rather this decision should be addressed along with the rate distortion decision. In these experiments, we have set different thresholds for different sequences.

For analysis of anti-collusion capability against linear and non-linear collusion attacks, we have used a single video sequence which is composed of all the benchmark sequences in a repeated fashion. We have set different threshold TH-SS for different sequences to get a better trade-off between PSNR and payload. For analysis of colluded video, Fig. 10 illustrates the PSNR of the colluded video for different number of colluders, created using linear and non-linear attacks. PSNR of original video is 45.02 dB, 46.85 dB, 47.65 dB for Y, U, V respectively. While for fingerprinted video, PSNR is 43.61 dB, 41.95 dB, 42.94 dB for Y, U, V respectively. Hence, decrease in PSNR because of embedding a fingerprinting code is 1.41 dB, 4.90 dB and 4.71 dB for Y, U, V. It also contains the PSNR of original content and the fingerprinted content before collusion. In case of average/median/minmax attacks, PSNR gets better with increase in number of colluders. While for minimum/maximum attack, the quality of video slightly decreases as the number of colluders increases. Modified negative attack is the strongest among all these attacks and decreases the PSNR up to 15 dB, 33 dB, 23 dB for Y, U , V respectively.

Table 1 shows the number of colluders which have been successfully traced by analyzing a pirated video content. In most of the cases, the colluders have been successfully traced. The attack which makes the video non-traceable *i.e.* modified negative attack also degrades the quality of the video very badly. For example, in case of modified negative attack, we could detect only few of the colluders, but PSNR of the attacked video is 15 dB. While on the other hand, averaging attack does not degrade the visual quality rather PSNR gets better in some cases but we can also detect the colluders with very high confidence value. Fig. 11 shows the visual quality of frame # 0 of benchmark video sequences for different collusion. One can note that modified negative attack makes the accusation process difficult but it severely degrades the video quality among the non-linear attacks.

5.3 Analysis of *inter* frames

For experimental results of *intra* & *inter* sequence, *intra period* has been set 10. Real-world scenario has been simulated by using all the motion estimation estimation block sizes along with quarter pixel accuracy. Non-zero QTCs are found in the parts of frames containing motion and texture. Fingerprinting code is naturally embedded in those areas as these are temporal masking areas in *inter* frames. In *intra* & *inter* sequence, QTCs are lower in magnitude in P frames as compared to I frames. Hence, threshold TH-SS for *luma* and *chroma* components in P frames is kept lower than in I frames. TH-SS in I and P frames is given as:

$$\begin{aligned} \text{TH-SS}_{y(I)} &= \text{TH-SS} \\ \text{TH-SS}_{uv(I)} &= \text{TH-SS} - 1 \\ \text{TH-SS}_{y(P)} &= \text{TH-SS} - 2 \\ \text{TH-SS}_{uv(P)} &= \text{TH-SS}_{y(P)} - 1 \end{aligned}$$

Payload capacity and PSNR trade-off of different video sequences for *intra* & *inter* sequence in analyzed in Fig. 12 and Fig. 13 respectively. One can note that payload and PSNR variation is different for different video sequences for both of *luma* and *chroma* components. For example, *mobile* video sequence has lot of texture and details, and has the highest payload among all the video sequences

in this analysis. Based on this analysis, it is clear that it is not a good idea to have the same threshold for all the video sequences. Rather this decision should be addressed along with the rate distortion decision.

For analysis of anti-collusion capability, all the benchmark sequences have been combined together in a repeated fashion. We have set different threshold TH-SS for different sequences and as a result, threshold for *intra* frames and *inter* frames will be adjusted to get a better trade-off between PSNR and payload. One can observe from Table 2 that our framework is collusion resistant for *intra* & *inter* sequence. Fig. 14 illustrates the PSNR of the colluded video, created using linear and non-linear attacks. It also shows the PSNR of original content and the fingerprinted content before collusion. PSNR of original video is 44.52 dB, 46.68 dB, 47.49 dB for Y, U, V respectively. While for fingerprinted video, PSNR is 42.92 dB, 45.85 dB, 46.60 dB for Y, U, V respectively. Hence, decrease in PSNR because of embedding a fingerprinting code is 1.6 dB, 0.83 dB and 0.89 dB for Y, U, V respectively. In case of average/median/minmax attacks, PSNR gets better with increase in number of colluders. While for minimum/maximum attack, the quality of video slightly decreases as the number of colluders increases.

6 Conclusion

In this work, we have demonstrated the fingerprinting of H.264/AVC using Tardos Fingerprinting codes. Spread spectrum robust watermarking technique has been used for embedding of fingerprinting code in personal copies. Our experimental analysis verifies that embedding of Tardos codes using spread spectrum watermarking technique is a very good design indeed. The colluded video content can be successfully analyzed to trace the colluders until its quality becomes unacceptable. This work has bright prospects and can be extended and improved in several aspects. For example, the accusation process requires $O(nm)$ computations for each Tardos code. We have to run the accusation process for all users. If there are 1 million users, the computation requirement will be very high. Code length increases squarely with increase in the number of colluders, and logarithmically with decrease in value of ϵ_1. For example, for $c = 10$ with $\epsilon_1 = 10^{-4}$, the code length is 23688 bits, while it is 110540 bits for $c = 20$ with $\epsilon_1 = 10^{-6}$. Storage of Tardos code takes lot of memory. Since Tardos code is probabilistic in nature, we have to generate the codes for all the users separately and have to store them. Tardos code needs 64 GB of memory for $n = 10^6$, $c = 5$ and $\epsilon_1 = 10^{-3}$.In future, we intend to extend our work for informed watermark embedding techniques.

References

1. Amiri, E., Tardos, G.: High Rate Fingerprinting Codes and The Fingerprinting Capacity. In: Proc. Annual ACM-SIAM Symposium on Discrete Algorithms, pp. 336–345. Philadelphia, PA, USA (2009)
2. Anthapadmanabhan, N., Barg, A., Dumer, I.: Fingerprinting capacity under the marking assumption. In: Proc. IEEE International Symposium on Information Theory, pp. 706 –710 (2007). DOI 10.1109/ISIT.2007.4557307
3. Anthapadmanabhan, N., Barg, A., Dumer, I.: On the Fingerprinting Capacity Under the Marking Assumption. IEEE Transactions on Information Theory **54**(6), 2678–2689 (2008)

4. Blayer, O., Tassa, T.: Improved versions of tardos' fingerprinting scheme. Designs, Codes and Cryptography **48**(1), 79–103 (2008)
5. Boneh, D., Shaw, J.: Collusion-Secure Fingerprinting for Digital Data. IEEE Transactions on Information Theory **44**(5), 1897–1905 (1998). DOI 10.1109/18.705568
6. Cérou, F., Furon, T., Guyader, A.: Experimental Assessment of the Reliability for Watermarking and Fingerprinting Schemes. EURASIP Journal on Information Security pp. 1–12 (2008)
7. Cox, I., Kilian, J., Leighton, F., Shamoon, T.: Secure Spread Spectrum Watermarking for Multimedia. IEEE Transactions on Image Processing **6**, 1673–1687 (1997)
8. H264: Draft ITU-T Recommendation and Final Draft International Standard of Joint Video Specification (ITU-T Rec. H.264 ISO/IEC 14496-10 AVC). Tech. rep., Joint Video Team (JVT), Doc. JVT-G050 (2003)
9. Hartung, F., Su, J., Girod, B.: Spread Spectrum Watermarking: Malicious Attacks and Counterattacks. In: Proc. SPIE: Security and Watermarking of Multimedia Contents, pp. 147–158 (1999)
10. He, S., Wu, M.: Joint Coding and Embedding Techniques for Multimedia Fingerprinting. IEEE Transactions on Information Forensics and Security **1**(2), 231–247 (2006)
11. Langelaar, G., Lagendijk, R.: Optimal Differential Energy Watermarking of DCT Encoded Images and Video. IEEE Transactions on Image Processing **10**, 148–158 (2001)
12. Lin, W., He, S., Bloom, J.: Performance Study and Improvement on ECC-Based Binary Anti-Collusion Forensic Code for Multimedia. In: Proc. ACM workshop on Multimedia and security, pp. 93–98. New York, NY, USA (2009). DOI http://doi.acm.org/10.1145/1597817.1597834
13. Lu, C., Chen, J., Fan, K.: Real-Time Frame-Dependent Video Watermarking in VLC Domain. Signal Processing: Image Communication **20**(7), 624 – 642 (2005)
14. M. Desoubeaux G. Le Guelvouit, W.P.: Probabilistic Fingerprinting Codes Used to Detect Traitor Zero-Bit Watermark, . In: Proc. IS&T/SPIE Electronic Imaging. San Francisco, CA, USA (2011)
15. Malvar, H., Hallapuro, A., Karczewicz, M., Kerofsky, L.: Low-Complexity Transform and Quantization in H.264/AVC. IEEE Transactions on Circuits and Systems for Video Technology **13**(7), 598–603 (2003)
16. Shahid, Z., Chaumont, M., Puech, W.: Spread Spectrum-Based Watermarking for Tardos Code-Based Fingerprinting for H.264/AVC Video. In: Proc. IEEE International Conference on Image Processing. Hong Kong (2010)
17. Skoric, B., Katzenbeisser, S., Celik, M.: Symmetric Tardos Fingerprinting Codes for Arbitrary Alphabet Sizes. Designs, Codes and Cryptography **46**, 137–166 (2008). URL `http://dx.doi.org/10.1007/s10623-007-9142-x`
18. Skoric, B., Vladimirova, T., Celik, M., Talstra, J.: Tardos Fingerprinting Is Better Than We Thought. CoRR **abs/cs/0607131** (2006)
19. Skoric, B., Vladimirova, T., Celik, M., Talstra, J.: Tardos Fingerprinting is Better Than We Thought. IEEE Transactions on Information Theory **54**(8), 3663 –3676 (2008)
20. Standard, C.: Digital Cinema Initiatives, LLC. Digital Cinema System Specification v1.1. Tech. rep. (2007)
21. Tardos, G.: Optimal Probabilistic Fingerprint Codes. In: Proc. ACM symposium on Theory of computing, pp. 116–125. New York, NY, USA (2003). DOI http://doi.acm.org/10.1145/780542.780561
22. Trappe, W., Wu, M., Wang, Z., Liu, L.: Anti-Collusion Fingerprinting for Multimedia. IEEE Transactions on Signal Processing **51**, 1069–1087 (2003). DOI 10.1109/TSP.2003.809378
23. Wiegand, T., Sullivan, G.J., Bjntegaard, G., Luthra, A.: Overview of the H.264/AVC Video Coding Standard. IEEE Transactions on Circuits and Systems for Video Technology **13**(7), 560–576 (2003)
24. Xie, F., Furon, T., Fontaine, C.: On-Off Keying Modulation and Tardos Fingerprinting. In: Proc. ACM workshop on Multimedia and security, pp. 101–106. New York, NY, USA (2008). DOI http://doi.acm.org/10.1145/1411328.1411347

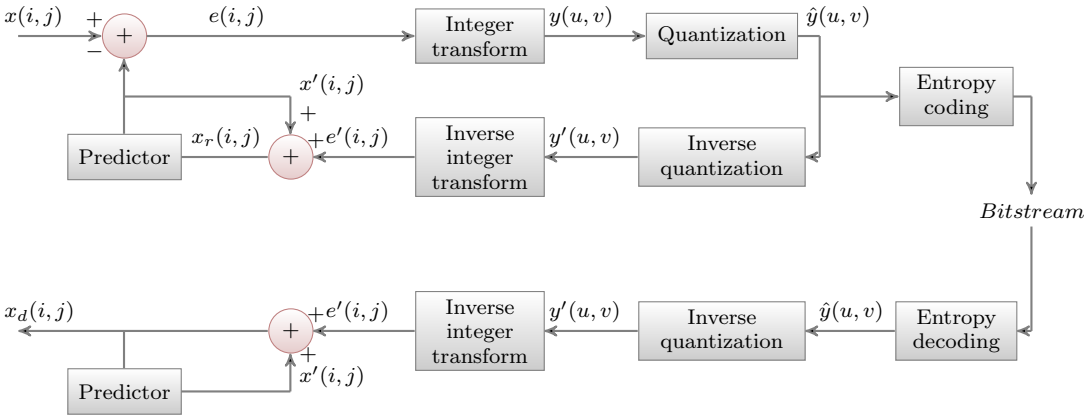

Fig. 1 Detailed block diagram explaining prediction, transform and quantization steps in H.264/AVC.

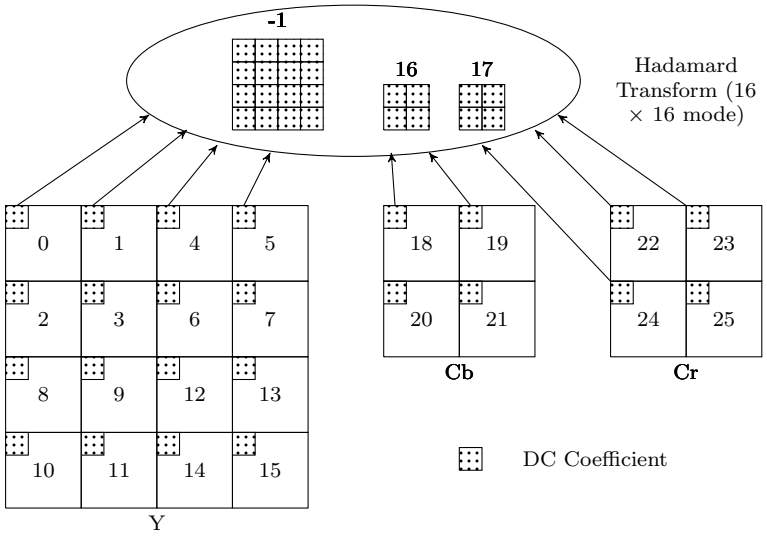

Fig. 2 Order of scan of *luma* and *chroma* $Intra\_4 \times 4$ blocks inside MB.

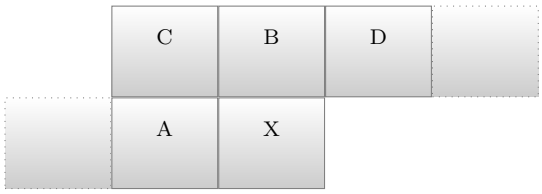

Fig. 3 X is current block while B_1, B_2, B_3 and B_4 are top and left neighboring blocks used for *intra* prediction.

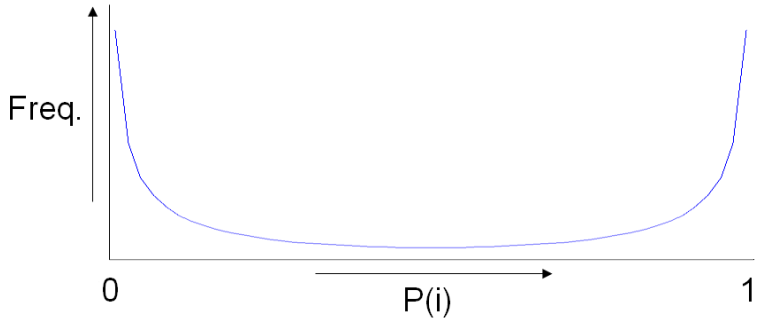

Fig. 4 Probability distribution for the probabilities of Tardos code.

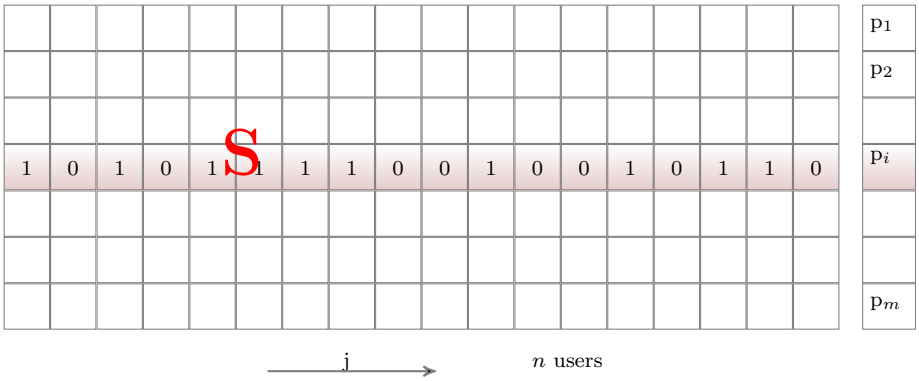

Fig. 5 $m \times n$ Tardos code with code-length of m for n users.

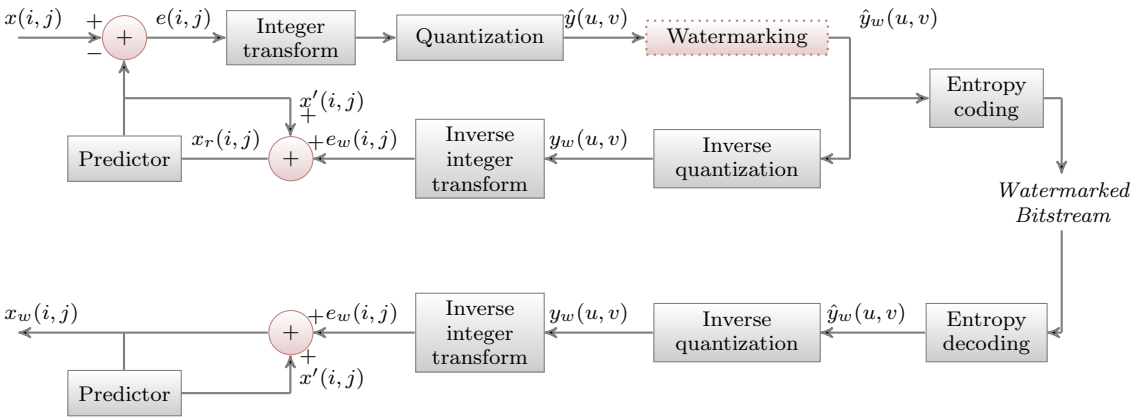

Fig. 6 Watermark embedding in H.264/AVC while taking into account the reconstruction loop.

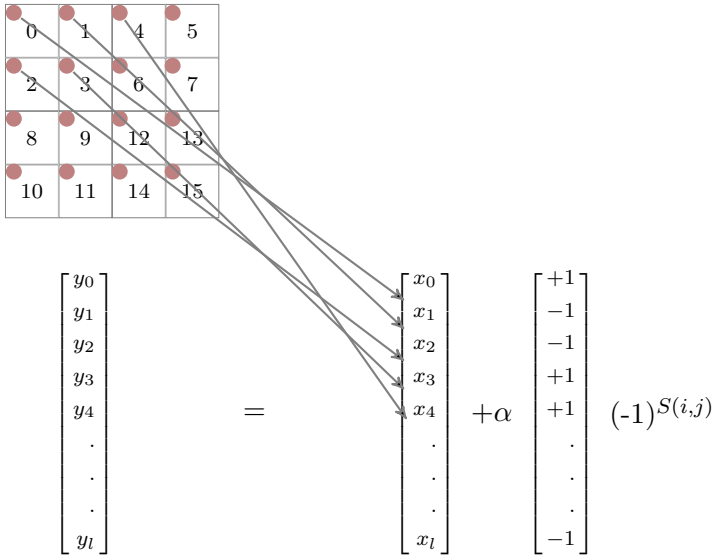

Fig. 7 Spread spectrum embedding of $S(i, j)$ Tardos code bit in copy of user j.

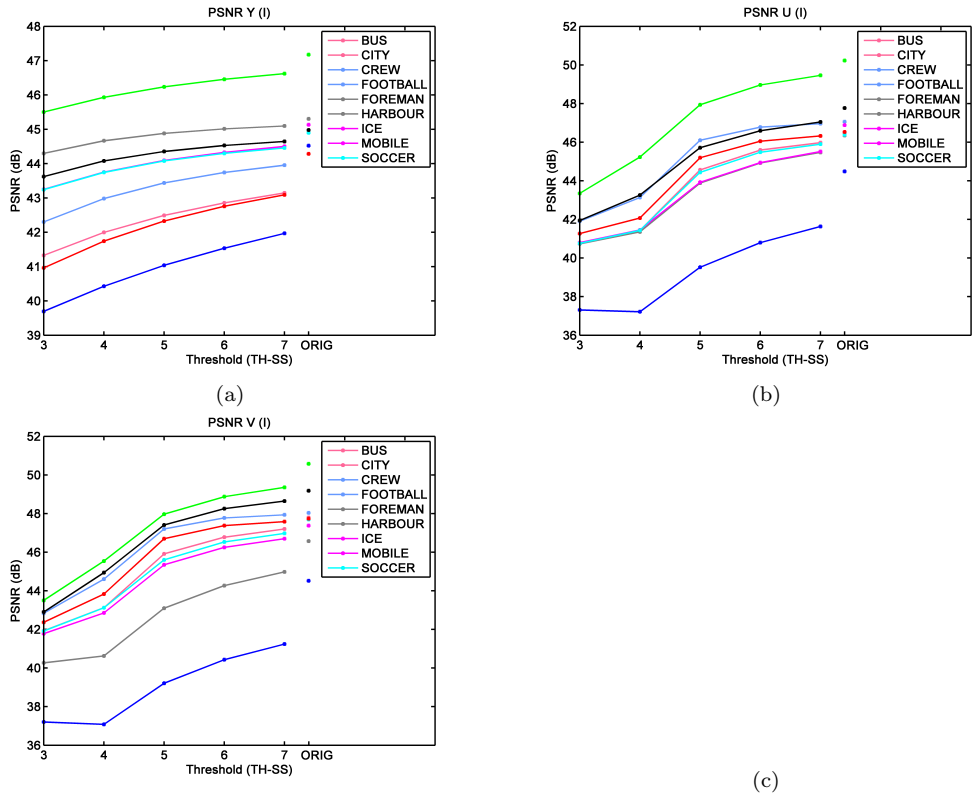

Fig. 8 Analysis of benchmark video sequences for fingerprint embedding at different thresholds: (a) PSNR-Y, (b) PSNR-U, (c) PSNR-V.

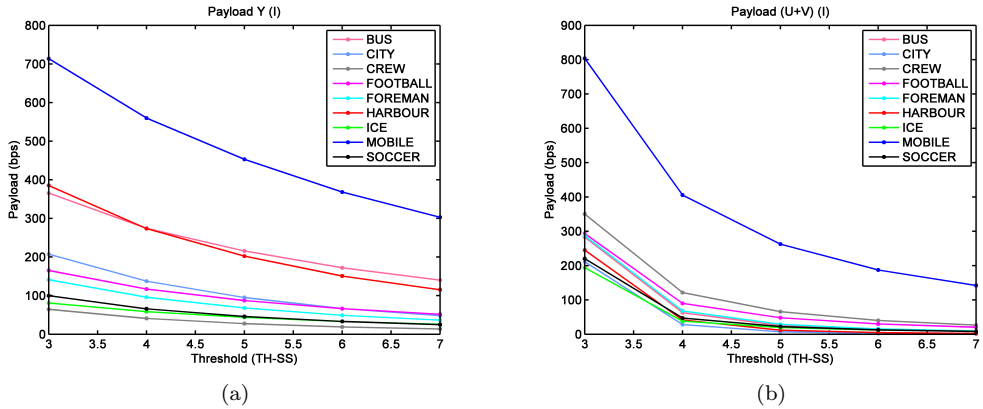

Fig. 9 Analysis of fingerprinted video sequences for different thresholds: (a) Payload-Y, (b) Payload-UV.

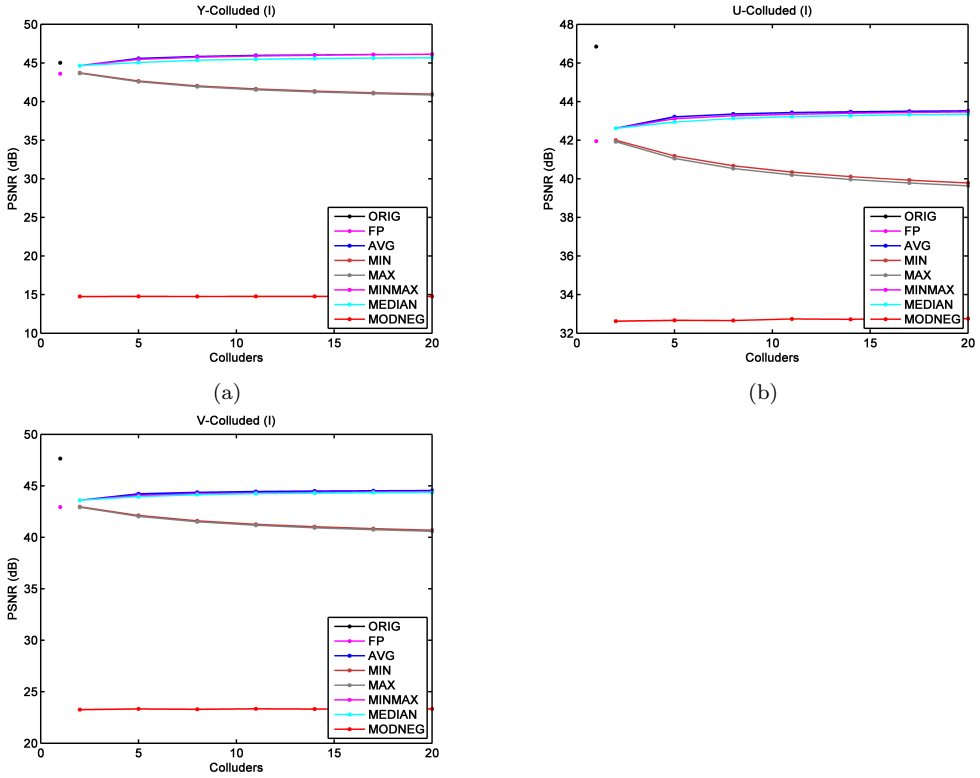

Fig. 10 PSNR of colluded video content, which has been generated using collusion attacks: (a) PSNR-Y, (b) PSNR-U, (c) PSNR-V.

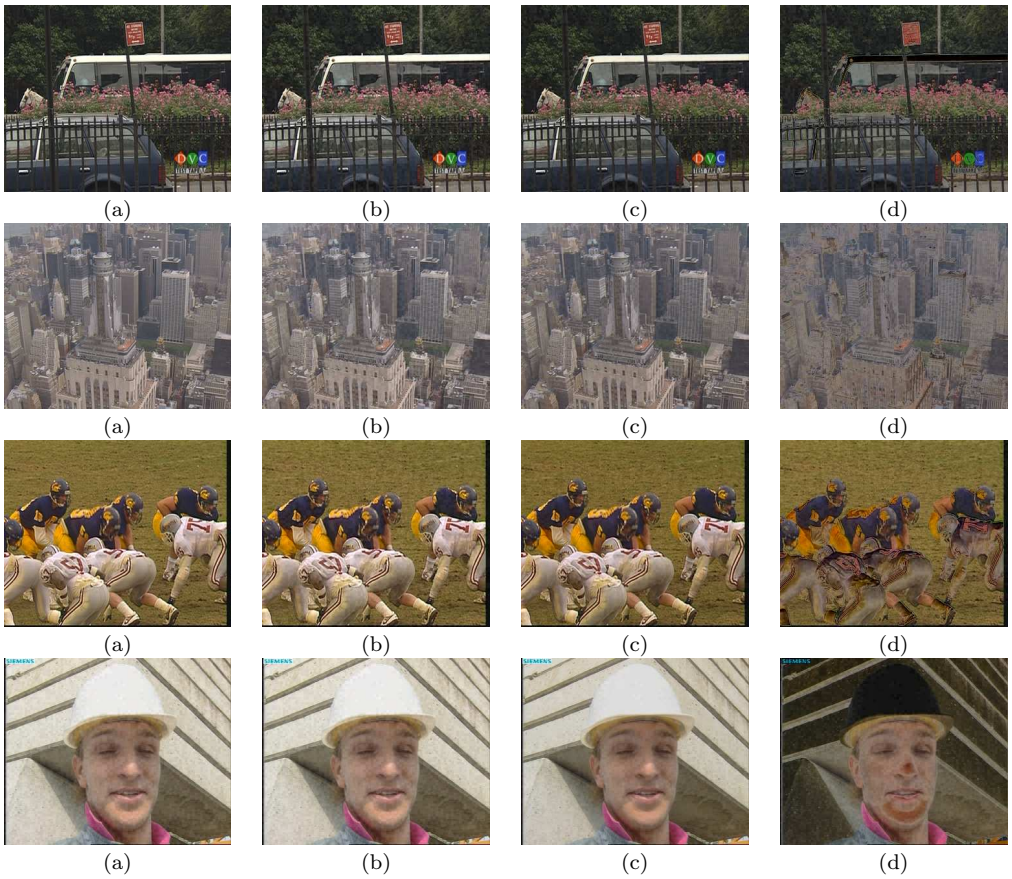

Fig. 11 Analysis of frame # 0 of colluded fingerprinted video sequences of *bus, city, football* and *foreman*: (a) Original frame, (b) Fingerprinted frame, (c) Average Collusion (17 colluders), (d) Modified negative collusion (8 colluders).

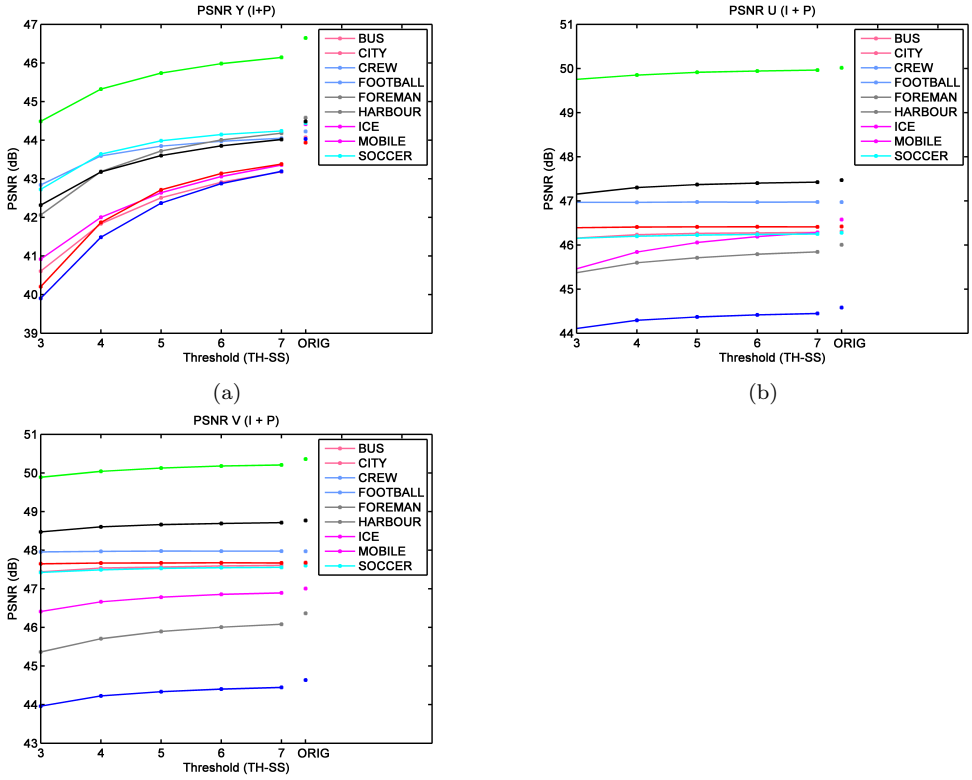

Fig. 12 Analysis of benchmark video sequences for fingerprint embedding at different thresholds: (a) PSNR-Y, (b) PSNR-U, (c) PSNR-V.

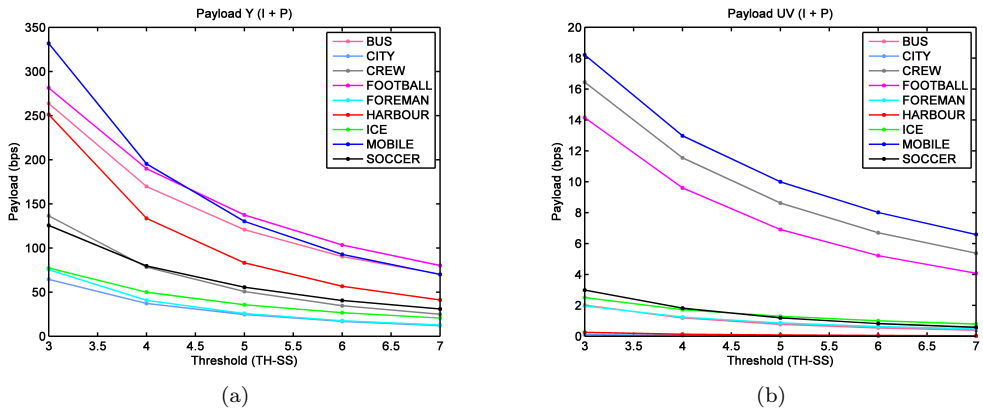

Fig. 13 Analysis of fingerprinted video sequences for different thresholds: (a) Payload-Y, (b) Payload-UV.

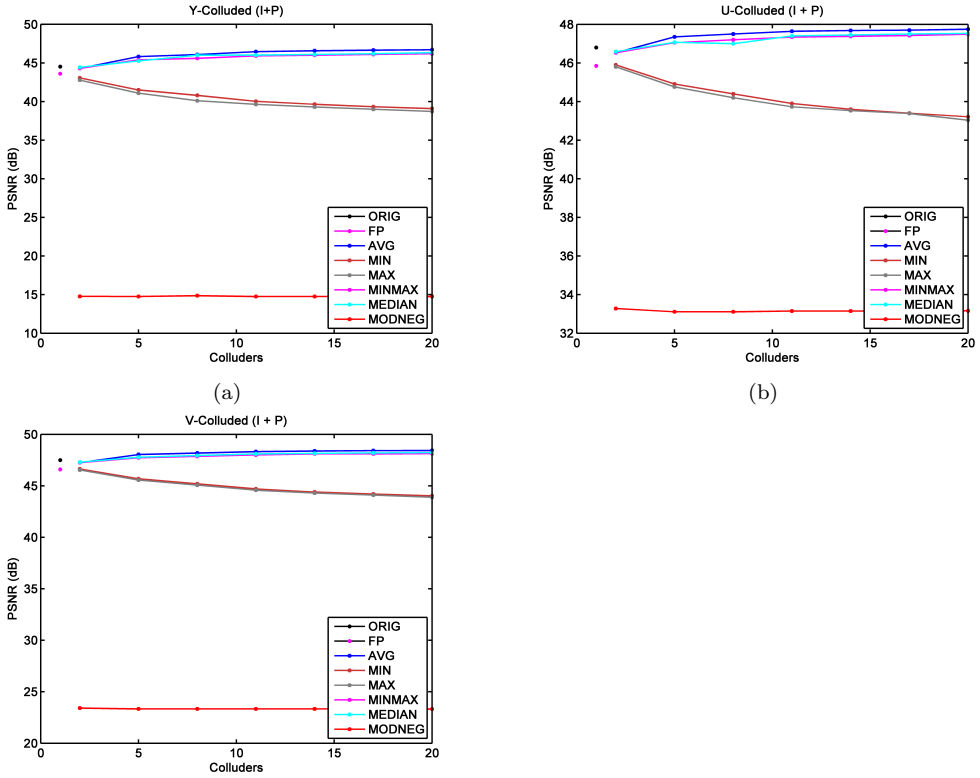

Fig. 14 PSNR of colluded video content, which has been generated using collusion attacks: (a) PSNR-Y, (b) PSNR-U, (c) PSNR-V.

K	Number of colluders detected for attacks					
	avg	min	max	median	minMax	modNeg
2	2	2	2	2	2	2
5	5	5	5	5	5	5
8	8	8	8	8	8	5
11	11	10	10	10	10	6
14	14	13	13	13	13	6
17	16	15	14	14	14	7
20	18	16	16	16	17	8

Table 1 Number of colluders traced from the K-colluded copy, generated using different collusion attacks for *intra* sequence. No innocent user is traced as colluder.

K	Number of colluders detected for attacks					
	avg	min	max	median	minMax	modNeg
2	2	2	2	2	2	2
5	5	5	5	5	5	3
8	8	7	6	7	8	4
11	11	8	8	9	10	5
14	14	10	11	12	13	5
17	16	13	12	13	14	5
20	18	15	15	16	16	6

Table 2 Number of colluders traced from the K-colluded copy, generated using different collusion attacks for *intra*& *inter* sequence. No innocent user is traced as colluder.