Egyptian Informatics Journal (2011) **12**, 115–123

Cairo University

Egyptian Informatics Journal

www.elsevier.com/locate/eij
www.sciencedirect.com

ORIGINAL ARTICLE

Efficient watermark detection by using the longest common substring technique

Taha M. Mohamed *, **Hesham N. Elmahdy**, **Hoda M. Onsi**

Information Technology Department, Faculty of Computers and Information, Cairo University, 5 Dr. Ahmed Zewail St., 12613 Orman, Giza, Egypt

Received 20 February 2011; revised 12 May 2011; accepted 23 May 2011
Available online 21 June 2011

KEYWORDS

H.264 watermarking;
VLC watermarking;
Watermark robustness;
Watermark desynchronization;
Longest common substring

Abstract Large scale of watermarking methods is available in the literature. These methods differ in visibility, capacity, and robustness. In watermarking, the robustness against attacks is the most challenging issue. The desynchronization attacks are the most serious problems facing the watermarking process. The traditional correlation methods fail in watermark detection. Until today there is no widely used algorithm for solving the desynchronization attacks. In this paper, we will introduce a new algorithm for solving the watermark desynchronization attacks. The watermark embedding and detection models are introduced. So, these models are related to the attacker model by presenting four attacking scenarios. We show the effect of each attack scenario on bit rate, signal distortion, and robustness. We conclude that, the attacker could not distort a big part of the watermark. So, we suggest using a probabilistic embedding model combined with the longest common substring technique. This combination is efficient in solving the desynchronization attacks. Results show that, the proposed algorithm is powerful against the attacking scenarios. Moreover, the watermark is still be detected even if only 5% of the watermark is recovered.

1. Introduction

Digital watermarking systems are presented for copyright protection of digital media. The watermark is extra information embedded into digital data. The main requirements of digital watermarking are; invisibility, robustness, blindness, and capacity. Moreover, video streaming applications require real time watermarking. The surviving of watermarks against lossy compression, such as H.264, is an additional feature [1,2]. These requirements are contradicted. The balancing of these requirements is determined by user's application. The watermark can encode copyright information or a cryptographic signature. This information identifies a particular copy of the source digital media [3,4].

* Corresponding author. Tel.: +20 114208725.
E-mail addresses: tahamahdy3000@yahoo.com (T.M. Mohamed), ehesham@fci-cu.edu.eg (H.N. Elmahdy), h.onsi@fci-cu.edu.eg (H.M. Onsi).

Peer review under responsibility of Faculty of Computers and Information, Cairo University.
doi:10.1016/j.eij.2011.05.001

ELSEVIER Production and hosting by Elsevier

H.264 represents an evolution of the existing video coding standards. It is developed to meet the growing needs for higher compression of moving pictures. The standard is jointly developed by the ITU-T VCEG and ISO/IEC MPEG. It achieves higher bit rate and lower distortion ratio compared to MPEG-2. In this paper, the standard is referred to as H.264. H.264 uses the context-based adaptive variable length coding (CAVLC) for entropy coding (VLC coding). CAVLC is only used for encoding quantized transform coefficients. In H.264, CAVLC requires less calculation than other VLC domains. So, CAVLC domain watermarking is more appropriate for real time purposes [5].

Generally, VLC watermark embedding is carried out by modifying, or interchanging, the VLC codes. This modification is performed according to the embedded watermark bit. In watermark detection, the watermark sequence is extracted based on the received VLC codes. The extracted watermark is then correlated with the original reference watermark sequence. The correlation value is used to make a decision about watermark existence.

However, the watermarking robustness is limited in VLC domain. The actual transform coefficients values are ignored during watermarking process. Any bit change in a VLC code causes decoding ambiguities. Moreover, the swapping of two or more VLC codes is possible. This action destroys the watermark [1,6]. The most serious attack is the missing of watermark synchronization which called the desynchronization attacks. In these attacks, the attacker tries to change the length of the extracted watermark sequence. This could be done by removing, or adding, some parts from, or to, the watermarked sequence. Generally, the desynchronization attacks are resulted from intentional attacks made by attackers. Moreover, these attacks could be resulted from signal geometric attacks such as rotation, translation, scaling, and random bending attacks. Whatever the cause of desynchronization, the correlation formulas cannot be applied for watermark detection. That is, the sequences lengths of the extracted watermark and the reference one are not the same.

The watermark robustness problem is a battle between the embedding side and the attacking side. The watermark creators embed the watermark. However, the attackers try to remove or, at least, corrupt the watermark. An important issue is that; the attackers wish to corrupt the watermark without causing high corruption in the original media itself.

There are three possible solution classes to desynchronization attacks [7]. The first solution is the using of invariant transform. In this solution, an invariant transform like Fourier Mellin transform is used. However, this solution suffers from implementation issues. Moreover, it is vulnerable to cropping and random bending attacks (RBAs). The second solution is the template insertion. However, this solution can be tampered by malicious attacks. The attackers could search for the template and remove it. The last solution is the feature based. In this solution, the watermark is embedded into geometrically invariant image features such as edges and corners. This can be done using pattern recognition techniques such as support vector regression [7], support vector machine [8], and neural networks [9]. However, the required exhaustive search is a large problem. Moreover, the watermarking capacity is limited in this solution. The training process is highly computational operation. So, these algorithms are not suitable for real time purposes.

In addition to the previous disadvantages of these solution classes, all of them rely on correlation concepts. So, they all fail in watermark detection when the pair of sequences differ in length. So, a new solution is needed to deal with the desynchronization attacks with the property that; the tested sequences are not of the same length. So, we suggest using the longest common substring (LCS) algorithm to solve the problem. LCS aims to find the longest string that is a substring of two strings. Substrings are consecutive parts of a string. The LCS of the strings "ABAB", "BABA" is "BAB". In this case, the LCS length equals 3. The alignment techniques are optimally implemented using dynamic programming algorithms [10].

In our previous work [11], we proposed a real time watermarking scheme for H.264. The proposed algorithm models the distribution of the CAVLC blocks *TCNs*. The blocks are modeled according to the geometric distribution. The watermark is embedded into selected CAVLC blocks in the video frame. The choice of these blocks is based on a user defined threshold *T*. The value of *T* is based on the presented model. The proposed watermarking method has many advantages. Such advantages include; high embedding capacity, real time embedding and real time detection. Moreover, the proposed watermark is blind and invisible. The embedding locations are hidden from attackers. However, the watermark is not robust against the desynchronization attacks.

In this paper, we will extend our previous work [11]. Firstly, the watermark desynchronization problems are addressed. So, we detail the embedding and the detection models. Then, we develop the desynchronization attacks models that may be performed at the attacker side. We introduce the modified embedding and detection algorithm for solving the desynchronization attacks. The effect of each attack on video quality and robustness is experimentally shown. Finally, the effects of combining the LCS technique with the probabilistic model are shown.

The rest of this paper is organized as follows; Section 2 contains an overview of the previously related schemes. Section 3 contains the problem formulation and the developed models. The details of the modified algorithm are introduced and explained in Section 4. Section 5 contains the experimental results and the discussion. The paper is concluded in Section 6.

2. Related work

In [12], the authors present the recent basics for image watermarking. A spread spectrum watermark is embedded. The correlation formula is presented for watermark detection. However, the correlation formula cannot be applied if the tested watermarks are not of the same length. In [13], the authors present an audio registration method based on dynamic time warping (DTW) technique. They calculate the distance between the watermarked audio and the attacked one. The algorithm is developed to overcome the geometric attacks such as scaling attack. However, DTW suffers from locality in watermark detection. Moreover, DTW aligns the beginning and ending of the sequences. This is not appropriate in watermark detection. In [14], the authors use the edit distance technique for fast video copy detection. The edit distance is used to measure the distance between two frame descriptors to detect illegal video coping. However, the edit distance technique is not always suitable for watermark detection. If the length of the attacked watermark is highly increased, then the distance is too large.

In [15], the authors proposed an efficient parallel algorithm for solving the longest common subsequence problem. This algorithm is an alternative for the classical dynamic programming techniques. However, the longest common subsequence is not suitable for efficient watermark detection. It is more suitable for DNA comparisons. There are two main differences between DNA sequence and the watermark sequence. The first difference is the sequence length. The DNA sequence is more larger than the watermark sequence. The second difference; the watermark may be attacked. On the contrary, the DNA sequence is never be attacked.

In [16], the computation analysis of the longest common substring (LCS) is outlined. LCS could be computed using the generalized suffix tree. The length and the starting position of the LCS could be found in O($n + m$). In the proposed algorithm, we only concern with the common substring length.

3. Problem formulation

In H.264, CAVLC is used to encode the 4×4 quantized transform block information. The CAVLC block contains some related elements. One of these elements is the Total Coefficients Number (*TCN*). *TCN* is the total number of nonzero coefficients in a 4×4 quantized block. It is an integer value where $0 \leqslant TCN \leqslant 16$.

Let the sequence $B_1 \ldots B_N$ represents all CAVLC blocks in the frame. The function ΩB_i is defined as the *TCN* value of a CAVLC block B_i. In watermark embedding process [11], the user chooses a subset β of the total blocks B; $\beta \subset B$ The cardinality of β is n. The blocks subset β will be watermarked. β is determined by using a user defined threshold T such that; $\Omega(\beta_i) \geqslant T$. So, n is only determined by T. The watermark itself is a discrete, pseudo random, uniformly distributed, sequence. It is generated by using a specified seed (key). The binary valued watermark function f assumes values belong to the set $\{1, -1\}$. The watermark values itself are generated according to a specified probability. The probability of getting a watermark value of 1 is p. In this case, we call this watermark value as *even bit*. So, the probability of getting -1 is $1 - p$. in this case, we call this watermark bit as *odd bit*.

3.1. The watermark embedding model

Let us define two Boolean functions. The first function, $\varepsilon(x)$, which assumes *true* value when x is even. The second function, $\ddot{o}(x)$, which assumes true when x is odd. If the *TCN* of a CAVLC block is even then, we call this block *even block*. So, the *odd block* is a block whose *TCN* is odd. For all i, $1 \leqslant i \leqslant n$, the watermark embedding algorithm is described as:

$$\acute{\Omega}(\beta_i) = \begin{cases} \Omega(\beta_i) - 1; & \text{if } \varepsilon(\Omega(\beta_i)) \text{ and } f(i) = -1 \\ \Omega(\beta_i); & \text{if } \varepsilon(\Omega(\beta_i)) \text{ and } f(i) = 1 \\ \Omega(\beta_i); & \text{if } \ddot{o}(\Omega(\beta_i)) \text{ and } f(i) = -1 \\ \Omega(\beta_i) - 1; & \text{if } \ddot{o}(\Omega(\beta_i)) \text{ and } f(i) = 1 \end{cases} \tag{1}$$

where $\acute{\Omega}(\beta_i)$ is the watermarked total coefficients number (*TCN*) of a block β_i. *f(i)* is the watermarking bit. Note that, in the first and the last cases of Eq. (1), the block's *TCN* is decreased. So, the actual number of the coefficients in the block must be decreased as well. This action is intended to preserve the block's consistency. So, one coefficient should be eliminated from the block. After elimination, the actual total number of coefficients in the block equals the modified *TCN* value. On the contrary, $\Omega(\beta_i)$ is not modified in the second and the fourth cases of Eq. (1).

In general, the *TCN* modification decision depends on both the *TCN* value and the embedding bit. To conclude the embedding process, a block is enforced to be *even block* when the watermarking bit is even. Alternatively, the block is enforced to be *odd block* when the watermarking bit is odd.

Let ρ be defined as the probability of even blocks in the frame. So, $1 - \rho$ is the probability of odd blocks in the frame. In watermarking, the number of the modified blocks in the frame, ξ, is computed as:

$$\xi = (\rho(1 - p) + (1 - \rho)p)n \tag{2}$$

where p is the probability that the watermarking bit is even. Moreover, if a block is modified during the watermarking process, then there are resulted distortion and bit saving. Let we define the average resulted block's distortion d and the block's average bit saving $\check{s}$. In watermarking, the total resulted distortion in the frame equals ξd Additionally, the resulted total bit saving in the frame equals $\xi\check{s}$.

3.2. The extraction and detection model

In the detection side, all blocks B are scanned to determine the watermarked set β. β is determined by using the condition that, $\Omega(\beta_i) \geqslant T$. In this case, 1 is extracted from an even block. -1 is extracted from an odd block. The process continues for all watermarked blocks for extracting the watermark sequence. The correlation is tested between the extracted, *possibly attacked*, sequence $\acute{f}$ and the original (reference) watermark sequence f. The correlation is applied by using the normalized dot product:

$$C = \frac{\langle f, \acute{f} \rangle}{n * n}; -1 \leqslant c \leqslant 1 \tag{3}$$

The float valued c equals -1 when there is no similarity at all between the two tested sequences. The correlation value equals 1 for perfect similarity between the tested sequences. Generally, the embedding side wishes that; the correlation value c is maximized at the watermark detector side. The correlation formula is a good similarity measure between any two sequences when two conditions are satisfied. The first condition; the tested sequences should have the same length. This condition is a necessary condition for applying the correlation formula. However, the attackers try to increase, or decrease, the lengths of the extracted sequence. In this case, the correlation formula cannot be applied.

The second condition; the tested sequences should be perfectly synchronized. Here, the perfect synchronization is described as $f(i) = \acute{f}(i); 1 \leqslant i \leqslant n$. Clearly, this condition affects the resulted correlation value. Perfect synchronization gives a correlation value $c = 1$. However, as the synchronization is decreased, the correlation value is decreased as well. The correlation value equals -1 when the synchronization is entirely missed. Again, the synchronization condition may be violated by some watermarking attacks. In this case, the correlation measure is unbeneficial.

3.3. The attacker model

In watermark attack, some blocks *TCNs* are modified. These *TCNs* are increased, or decreased, according to the attacking scenario. In watermarking attacks, if a *TCN* is decreased, then the actual coefficients number in this block should be decreased as well. Similarly, when a *TCN* is increased, the actual coefficients number should be increased as well. These actions are performed to preserve the block consistency. After any attack, the resulted bit rate is increased or decreased. In all cases, there is a resulted video distortion. Let $\check{s}$ represents the average block saving if one coefficient is eliminated from the block. So, if one coefficient is added to the block, then $\check{s}$ is negative. Let R_j be the number of coefficients that are eliminated from the block B_j. So, the total block saving $£_j$ when the block B_j is attacked is computed as:

$$£_j = R_j\check{s} \tag{4}$$

Let N_0 represents the number of the attacked blocks in the frame. The total frame saving, £, is computed as:

$$£ = \sum_{j=1}^{N_0} £_j \tag{5}$$

If the value of £ is positive, then the overall attacking process is a bit saving process. Alternatively, if £ is negative, the overall attacking process is a bit increasing process.

Analogously, the total resulted distortion in the frame could be computed as:

$$D = \sum_{j=1}^{N_0} d_j \tag{6}$$

where d_j is the resulted distortion when the block B_j is attacked. D is the total resulted distortion in the frame. D is always a positive value. It is clear that, the total resulted distortion and bit rate are completely depending on the number of attacked blocks. Moreover, they are depending on the number of attacked coefficients in the block. So, the attacker wants to satisfy two contradicted requirements. The first is; decreasing the resulted distortion. The second is; attacking more blocks to make the watermark undetectable. So, the attacker should optimize these two requirements.

At the attacker side, the attacker does not know the user threshold T. Alternatively, He tries to predict it. If the attacker fails in the threshold prediction process, then some non watermarked blocks may be attacked. This action causes more video distortion without much effect on the watermark itself. The predicted threshold, *PT*, is used to perform one of the following attacking scenarios. These scenarios aim to remove the watermark entirely or at least corrupt it. The watermark corruption increases the difficulty of watermark detection. The attacker tries to minimize the synchronization between the extracted watermark and the reference one. However, the watermarking attacks increase the resulted video distortion. So, the attacking scenarios are discussed and analyzed in terms of watermark robustness, bit rate, and distortion.

Scenario 1: In this scenario, the attacker aims to eliminate the watermark entirely. The attacker predicts a threshold (*PT*). So, the attacker converts the odd blocks to even blocks and vice versa. This attack negates the watermark. Watermark negation is the conversion of the watermark values from 1 to -1 and vice versa. The attacked blocks have the condition that; $\Omega(B_i) > PT$. This attack may be done by using one of the following two options:

$$\acute{\acute{\Omega}}(\beta_i) = \acute{\Omega}(\beta_i) + 1; \quad \acute{\Omega}(\beta_i) > Pt \tag{7}$$

$$\acute{\acute{\Omega}}(\beta_i) = \acute{\Omega}(\beta_i) - 1; \quad \acute{\Omega}(\beta_i) > Pt \tag{8}$$

where $\acute{\acute{\Omega}}(\beta_i)$ is the attacked block's *TCN*. In Eqs. (7) and (8), the average resulted distortion is the same. However, Eq. (7) is not feasible for attackers since it causes larger bit rate. Alternatively, Eq. (8) is the feasible choice for attackers since it decreases the resulted bit rate.

Based on the threshold prediction process, the watermark may be entirely attacked or partially attacked. The successful prediction of *PT* causes the correlation value equals −1. However, unsuccessful threshold prediction increases, or decreases, the length of the extracted watermark sequence. In this case, the correlation cannot be applied.

Scenario 2: This scenario aims to decrease the length of the extracted watermark sequence. So, the correlation cannot be applied. Here, the attacker tries to preserve the video quality. The attacker modifies the *TCNs* of some watermarked blocks. The *TCNs* are decreased to values less than T. So, the attacker removes a subset of watermarked blocks from the watermarked blocks sequence. In this scenario, if *PT* is successfully predicted, then the extracted watermark length is shorter than the reference one. So, the correlation formula cannot be applied. Generally, to decrease the effect of this attack, the chosen threshold T should be as small as possible. In this scenario, attacking more blocks could be easily detected as the resulted blocks are not geometrically distributed [11].

Scenario 3: As opposite to attacking scenario 2, this scenario aims to increase the length of the extracted watermark sequence. So, the correlation cannot be applied. The attacker converts some un watermarked blocks to appear as watermarked blocks. The attack is carried out by increasing some TCNs values becoming greater than T.

Scenario 4: In this scenario, we assume the attacker knows the threshold T. The attacker tries to convert some watermarked odd blocks to appear as watermarked even blocks and vice versa. In this case, the lengths of the extracted watermark and the reference one are identical. So, the correlation could be applied. However, the correlation value depends on the number of the attacked blocks in this case.

In all previous scenarios, the distortion is increased as the number of attacked blocks is increased. Also, the distortion is increased if any block's *TCN* is highly increased or highly decreased. So, the attacker could not modify more blocks to avoid more distortion. Moreover, the attacker could not distort a big amount of any block.

4. The modified watermarking algorithm

4.1. The embedding and extraction algorithm

We modify our previous scheme [11] to overcome the previously mentioned watermarking desynchronization problems. In the modified algorithm, we combine a probabilistic model with the longest common substring (LCS) technique. The probabilistic embedding model is useful for solving the watermark negation problem. LCS is useful when the extracted watermark length is increased or decreased by some attacks. This modified algorithm still has the previous advantages of [11]. The block diagram of the modified algorithm is illustrated in Fig. 1. Two watermarks are generated w, and its negative $\acute{w}$. The negative of any watermarking bit $w(i)$ is given by $\acute{w}(i) = -w(i)$. The embedding algorithm starts by selecting one of the two watermarks at random. The selection is done using a probabilistic system S with a probability $p(w) = p(\acute{w}) = 0.5$. The two watermarks have the same length. So, the choice step has no effect on the resulted bit rate and the resulted watermark visibility. However, the choice step is very important for avoiding attack effects during watermark detection.

To clarify the watermark negation, consider the following watermark sequence:

The watermark sequence									
1	−1	1	1	1	−1	−1	1	1	1

Its negation is:

The negative watermark sequence									
−1	1	−1	−1	−1	1	1	−1	−1	−1

Next, the embedding blocks are chosen according to a user defined threshold T. All blocks that have $TCN \geqslant T$ will be watermarked. The watermarked blocks r are sent across a noisy channel. The received, *possibly distorted*, $\bar{r}$ may be attacked by some attacking scenario producing $\acute{\acute{r}}$. At the watermark detector $\acute{\acute{r}}$ is received. The watermark sequence l is extracted. Then, l is matched with the original (reference) watermark *wm* using the LCS technique. If the matching value is greater than 5%, the watermark is decided as a detected watermark. If not, the extracted sequence l is negated to form $\bar{l}$. $\bar{l}$ is matched with the original reference sequence again using LCS. If LCS gives a matching percentage of 5%, then the watermark is detected. The negation action is performed to overcome the effects of converting odd blocks to even blocks and vice versa.

So, in the modified algorithm, we do not care about the values of the watermark itself. We care about the contiguous parts of the watermarks. The common contiguous string length is computed between the extracted watermark and the reference one. Also, the common contiguous string length is computed between the extracted watermark and the negation of the reference watermark. If one of the two lengths is more than 5% of the reference watermark length, then the watermark is detected. This may be clarified as; the embedding algorithm embeds one of two watermarks (a watermark and its negative). In the detection side, the detector tries to match the extracted watermark with any one of the two watermarks. If the negated watermark and the extracted watermark are matched using LCS, then we could easily conclude that, the embedded watermark is attacked. This idea is an analogy to the idea of public and private keys in cryptography systems.

So, the detection algorithm is described as:

```
Start Algorithm:
  w = the reference watermark sequence
  $\bar{w}$ = the negative watermark sequence
  e = the extracted watermark sequence
  L1 =  LCS(w,e)
  L2= LCS($\bar{w}$,e)
  If L1> = 5% * length(w) or L2 > = 5% * length(w) then
    The watermark is exist
  else
    The watermark is not exist
  end if
End Algorithm:
```

The proposed algorithm is valid for solving the desynchronization problems of all types of digital media (image, audio, and video). In all media types, the watermark is extracted from the watermarked media. So, the embedded watermark sequence is compared with the detected, *possibly attacked*, sequence using the proposed algorithm. The watermark is still be detected even if the attacking percentage is 95% of the watermarking length. Moreover, if the watermark bits are converted from ones to zeros, or vice versa, then the watermark is still be detected using the probabilistic embedding algorithm.

4.2. Robustness against other attacks

It is important to note that, the embedding and extraction processes are carried out in the compressed domain. That is why the only addressed attack is the desynchronization attacks. To clarify this issue; the proposed algorithm embeds the watermark in the CAVLC domain. The watermark is embedded by decreasing TCN. In fact; TCN is a computed value in 4×4 blocks. It is not a real pixel value in the raw video format. When the watermark is attacked; attackers have two restrictions during the attacking process. Firstly, the attacked (modified) value of the

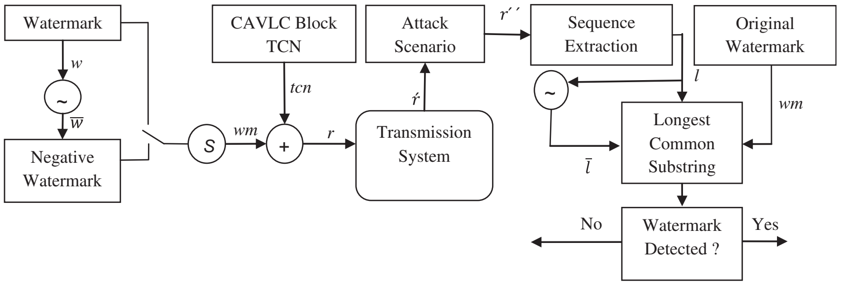


Figure 1 The block diagram of the modified embedding and extraction algorithm.

TCN should be integer valued. As the TCN is the total number of coefficients in the block. Secondly, the modified value of TCN should be bounded by zero and sixteen. If the attackers ignore these restrictions, the compressed video will not be correctly decoded. The first and the second restrictions prohibit attackers from using other attacks such as additive noise, filtering, geometric, and collusion attacks. That is, these attacks are always applied to the raw video pixels. It cannot be applied in the compressed domain. Alternatively, the attackers could apply these attacks to the raw video before starting the coding process. In this case, the watermark is embedded after applying these attacks. In this case, the watermark is fully detected. So, the proposed system is robust against these attacks by default.

However, the attacker could only modifies the TCN value within the above two restrictions as we previously shown in the attacking scenarios. In this case, the synchronization attacks occurred. However, the proposed system is designed to solve these types of attacks. Moreover, the proposed algorithm solves the problem of different lengths in watermark detection. The algorithms in [12,17] and [18] cannot solve the problem. They are using the correlation concepts. These concepts cannot be applied when the watermarks have different lengths.

5. Experimental results and discussion

The proposed algorithm is tested using the official H.264 reference software *JM* ver. 10 [19]. The default encoding parameters are used. All tests are performed using three standard, commonly used, video sequences: *Foreman*, *Mobile*, and *Container*. These sequences vary in texture and frequencies. The watermark randomness, uniqueness, and correlation are measured using Matlab 7.0. The LCS is also implemented using Matlab ver. 7.0. *YUV Tools* software is used for video playing and SNR computation. In this software, any two identical sequences give PSNR value = 100.

5.1. The experimental results

We begin our experimental results by showing the payload of the proposed system. In Fig. 2, the watermarking system capacity is shown by using two tests; Test 1 and Test 2. Test 1 uses a threshold of five. Test 2 uses a threshold of two. It is clear that, Test 1 has more capacity than Test 2. That is, the watermarked blocks when using Test 1 are more than those blocks when using Test 2. Generally, the capacity of the two tests depends on the chosen user threshold T and the video sequence itself. In the proposed algorithm, the Akiyo sequence has the least embedding capacity which equals 139 bits when using Test 2. Comparing this result to Mobasseri's [6], our watermarking system achieves an embedding percentage which is approximately 24 multiples of Mobasseri's percentage [11].

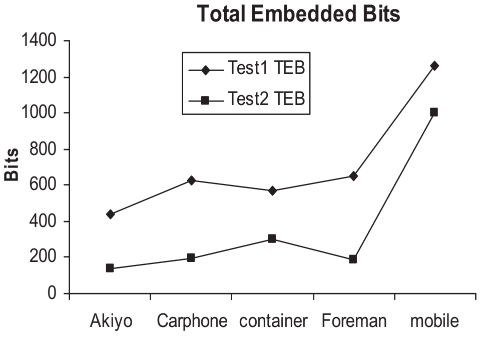


Figure 2 Bit capacity of the proposed system.

To test the algorithm robustness, the randomness quality of the watermark generator is tested. In this test, 200 random watermark sequences are generated. Only two sequences of them have a common seed (seed = 0). The other sequences are generated from different seeds. Fig. 3 shows the correlation value between every two watermark sequences pairs. The correlation value equals one when the two watermarks have the same seed. Otherwise, the correlation values are very small. Concluding that, the correlation value is not large if the tested watermarks are generated from different seeds. So, the watermark randomness quality is good.

Secondly, we evaluate the LCS capability in watermark detection. We perform four tests shown in Table 1. For each test, we perform 100 experiments. In each experiment, two different sequences are generated with the same length. The watermark length is shown in the second column. The LCS between the two sequences is computed and registered. The length of the longest sequence in this register is considered as the LCS. This length is shown in the third column of Table 1. In the last column, this length is computed as a percentage relative to the sequence length.

It is clear from Table 1 that, the maximum percentage in the table equals 0.42. Meaning that; the LCS between any two *different sequences* is less than 5% of the sequence length. *Different sequences* means that, the sequences are generated from different seeds. To clarify this issue; assume we have two watermarks generated from any seed. So, one of the two watermarks is attacked by 95% of its length. In this case, the attacked watermark could be distinguished from other watermarks by using LCS.

In Fig. 4, we illustrate the effect of scenario 1 on video quality. In this scenario, the even blocks are converted to odd blocks and vice versa. To simulate this attack, we use three different threshold values for watermark embedding. The attack is simulated by decreasing all *TCN*s in the frame by one. For block consistency, the last coefficient is eliminated [11]. In Fig. 4.a, 4.c, and 4.e the watermarked sequences are shown using three different thresholds. In Fig. 4.b the attacked versions are shown. From the figure, we noted that, the foreman sequence is disfigured when attacked by attack sceanrio1. Also, the container and mobile sequences are highly degraded. In this case, LCS could not find any matching between the reference watermark and the extracted one. However, in our system, the solution is the watermark negation. This negation step is the removing of scenario 1 effect. In this case, the watermark is fully detected.

In Table 2, we extend the simulation of attack scenario 1. The PSNR values are shown between the watermarked video sequences and their attacked versions. It is clear that, all resulted PSNR values are very small. The attacked sequences are highly degraded as previously shown in Fig. 4. However, the watermark is still be detected by the proposed algorithm.

The results of attack secnario 2 are shown in Table 3. In this test, an embedding threshold $T = 5$ is used. The choice of this threshold value is intended to preserve the original video quality [11]. In this scenario, the attacker tries to attack some watermarked blocks. At the detector, these attacked

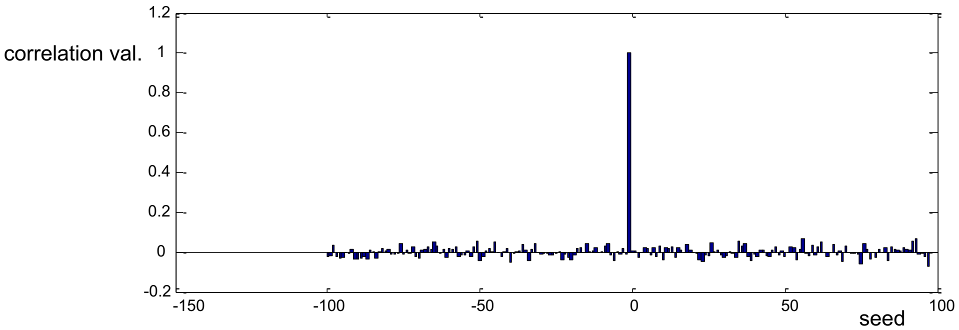


Figure 3 The correlation values of 200 random watermarks.

Table 1 The LCS between watermarks of different random seeds.

No. of experiments	Watermark length	Common substring length	Length percentage
100	500	21	0.042
100	1000	25	0.025
100	1500	27	0.018
100	2000	25	0.012

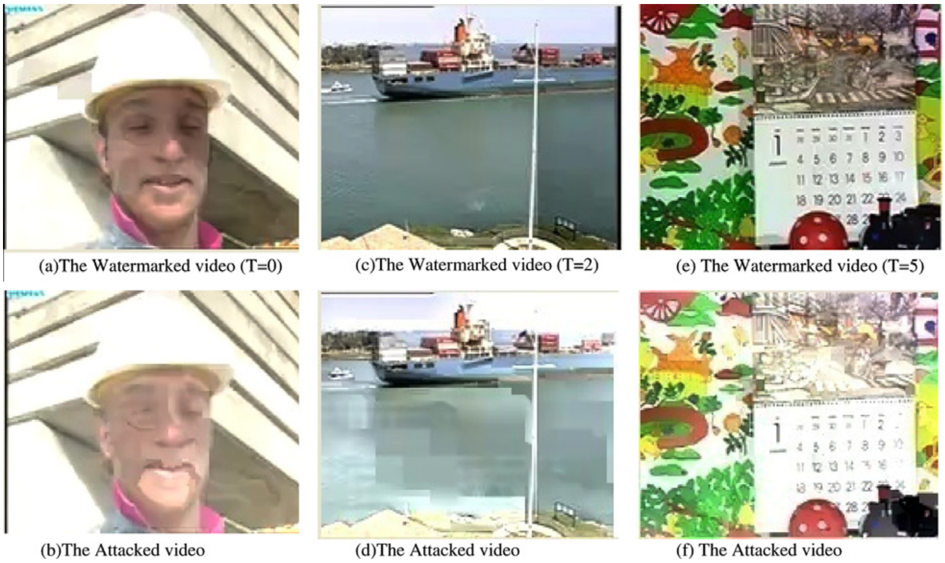
blocks will appear as un watermarked blocks. In fact, the attacker does not know the watermark embedding locations. Alternatively, he tries to predict these locations by predicting the threshold. After predicting the threshold *PT*, the attacker attacks some blocks. The *TCNs* of the attacked blocks are converted to *PT* – 1. The attacker chooses the value, *PT* – 1, to avoid more video degradation.

In Table 3, the first column represents the tested sequences. The second column represents the total number of the watermarked blocks in each sequence before watermark attacking. The third column represents the predicted threshold *PT*. It is clear that, when the predicted threshold *PT* = 3, it may has no effect on the watermark robustness. This case occurs when only the attacked blocks are located between *TCN = 3* and *TCN = 5*. In this case, the effect is only on the resulted video quality.

The fourth column represents the number of the attacked blocks by attack scenario 2. The attack type (*partial or full*) is illustrated in the fifth columns. In this column, *partial* means that: only some watermarked blocks whose *TCN* > *PT* are attacked. However, *full* means that, all blocks whose *TCN* > *PT* are attacked. The attacking percentage is shown in the sixth column. The PSNR value between the attacked video and the original video is shown in the last column.

To clarify the table results; consider the first row as an example. The sequence *foreman* is watermarked using a threshold *T = 5*. The total number of watermarked blocks equals 258. At the attacker side, the attacker does not know *T*. So, the attacker predicts a threshold *PT* = 3. In this case, the attacker randomly converts 200 blocks whose *TCNs* values are greater than *PT* (3 in this case*)* to a new value *PT* – 1 (2 in this case). The attacked blocks are missed at the watermark detector. So, this action breaks the watermark synchronization. In this case, the attacking type is partial, which means that, there are more possible watermarked blocks to be attacked.

However, the attacker could not attack all these blocks because attacking more blocks causes a higher video degradation. The attack percentage in this case equals 0.77. It is computed as


(a)The Watermarked video (T=0)

(c)The Watermarked video (T=2)

(e) The Watermarked video (T=5)

(b)The Attacked video

(d)The Attacked video

(f) The Attacked video

Figure 4 The results of applying attack scenario 1.

Table 2 The PSNR values of attack scenario 1.

Forman			Mobile			Container		
T	PSNR	MSE	*T*	PSNR	MSE	*T*	PSNR	MSE
0	19.27	769.27	0	23.84	268.58	0	16.47	1465.82
2	13.55	2871.31	2	13.95	2618.67	2	13.23	3090.87
5	11.17	4966.84	5	12.40	3741.8	5	12.83	3389.07

Table 3 The effect of attack scenario 2.

Video	No. of Watermarked blocks	Predicted threshold (PT)	No. of attacked blocks	Attacking type	Attacking percentage	PSNR	MSE
Forman	258	3	200	Partial	0.77	15.84	1694.65
	258	5	196	Full	0.75	24.39	236.64
Mobile	1151	3	1000	Partial	0.86	15.55	1811.68
	1151	5	1000	Partial	0.86	16.02	1625.85
Container	350	3	300	Partial	0.85	19.43	741.45
	350	5	300	Partial	0.85	21.56	454.03

the total number of the attacked blocks relative to the total number of the watermarked blocks. As indicated in the last column, the PSNR value between the original watermarked video and the attacked one equals 15.84. This PSNR value is very low. It means that, the video sequence is highly attacked.

The maximum PSNR value in the table occurs at the last row which equals 27.11. In this case, the resulted attacked video is also degraded. Although this degradation, the watermark is still be detected. The attacking percentage equals 45% in this case, which it is less than 95%. So, according to the results obtained from Table 1, the watermark is still be detected.

Table 4 shows the attack results when applying attack scenario 3. In this test, we used $T = 5$. Again, the attacker does not know the threshold. The attacker predicts a threshold *PT*, and attacks (*all or some*) non watermarked blocks. These attacked blocks appear as watermarked blocks at the detector. This action breaks the watermark synchronization. The length of the attacked watermark is larger than the length of the reference one. Again, if $PT > T$, there is no effect on watermark synchronization because the watermark length is unchanged. However, the watermark may be partially distorted. The same analogy of Table 3 could be applied when addressing Table 4. It is clear that, when applying this attack, the watermark is still be detected in all cases. That is, all attacking percentages are less than 95%.

Table 5 presents the attack results when applying attack scenario 4. In this test, the used embedding threshold $T = 5$. The attack is simulated by attacking some blocks *TCNs* whose TCN values $> T$. These *TCNS* are converted from even to odd and vice versa. Attack scenario 4 does not break the watermark synchronization. Alternatively, the attack decreases the similarity between the watermark sequence and the reference one. By simulating this attack, the watermark is still be detected using the LCS algorithm. It is also noted that, the effect on video degradation is low. All PSNR values are larger compared to previous mentioned attacks results.

5.2. Discussion

When discussing the previous results we could conclude that, the most dangerous attack scenario is *scenario1* attack. This attack could be efficiently avoided by using the proposed probabilistic embedding model. If the detector receives a negated watermark, then the detector has strong evidence that the watermark is attacked by using attack scenario 1. We believe that, the attacker could not really attack the watermark by using scenario 1. The reason is, the resulted video is highly degraded. This degradation is well illustrated in Fig. 4.

In attacks scenario 2 and scenario 3, the attacker should know the embedding threshold for perfect watermark attack. The attacker does not know the threshold; alternatively he tries to predict it. As we conclude from Tables 3 and 4, the larger the number of the attacked blocks, the larger the resulted degradation. So, the attacker tries to choose a threshold with the target that, only fewer blocks are attacked. In this case, the LCS is an effective method to deal with such attacks. The algorithm only requires a correct percentage equals 5%.

Table 4 The effect of attack scenario 3.

Video	No. of watermarked blocks	Predicted threshold (PT)	No. of attacked blocks	Attacking type	Attacking percentage	PSNR	MSE
Forman	258	3	200	Partial	0.77	21.54	456.12
	258	5	200	Partial	0.77	27.75	109.16
Mobile	1151	3	327	Partial	0.28	19.58	716.28
	1151	5	396	Partial	0.34	25.00	205.63
Container	350	3	300	Partial	0.85	10.30	6068.49
	350	5	300	Partial	0.85	20.89	529.76

Table 5 The effect of attack scenario 4.

Video	Total number of blocks	No. of attacked blocks	PSNR	MSE	Attacking percentage
Forman	258	200	22.07	403.72	0.77
Mobile	1151	1000	25.61	178.68	0.86
Container	350	300	30.36	59.85	0.85

The simulation results show that, this percentage always causes higher video degradation. So, the attacker could not attack the video up to 95%. Moreover, we suggest to choose a threshold $T = 5$ in watermark embedding. This threshold makes a good adjustment for avoiding scenario 2 and scenario 3. Moreover, this threshold preserves the resulted watermark quality.

The results of attack scenario 4 show that, the attacked blocks could not affect watermark synchronization. Alternatively, it may decrease the correlation values. In this case, the watermark is still be detected when using the ordinary correlation. If the watermark is highly attacked, then the correlation value is very low. If the attacker tries to attack more blocks, the resulted video is highly attenuated. If the watermark is entirely attacked, then LCS value is low. Alternatively, the probabilistic model is used to solve the problem. The extracted watermark is negated to restore the original watermark.

Finally, we should note that, the watermarking domain is H.264 compressed video. The domain is a lossy domain. Moreover, the watermarking process is another loss. When the watermarked video is attacked, a new resulted loss is presented. This accumulated loss causes more video degradation.

6. Conclusion

In this paper, we introduce a novel scheme for solving the watermark desynchronization attack problems. This scheme is valid for all type of digital media images, audio, and video. We introduced the embedding and extraction model followed by addressing four attacks scenarios. The traditional correlation fails in solving these types of attacks. We show that, the attacker could not attenuate a big part of the watermark. This action causes video destruction. Moreover, the attacker does not know the embedding threshold and the embedding locations. So, a resulted video distortion is introduced when predicting the threshold. The resulting distortion and bit rate depend on the attacking scenario. The proposed algorithm is designed by mixing a probabilistic embedding model with the longest common substring algorithm. Results show that, the watermark is still be detected in all attack scenarios. The detection is occurred even if the watermark is attenuated up to 95%. Moreover, the probabilistic model is very useful when the watermark values are negated. The proposed scheme could be combined with other work. This combination achieves additional required watermarking features such as visibility, bit rate, real time conditions, and high capacity.

References

[1] Pranata S, Wahadaniah V, Guan Y, Chua H. Improved bit rate control for real-time MPEG watermarking, EURASIP Journal on Applied Signal Processing 2004;2004(14):2132–41.

[2] Lua C, Chena J, Fan K. Real-time frame-dependent video watermarking in VLC domain. Journal of Signal Processing: Image Communication 2005;20(7):624–42.

[3] Yea D, Zoub C, Daib Y, Wang Z. A new adaptive watermarking for real-time videos. Journal of Applied Mathematics and Computation 2007;185(2):907–18.

[4] Foote J, Adcock J, Girgensohn A. Time base modulation: a new approach to watermarking audio. I EEE International Conference on Multimedia and Expo, Maryland, USA 2003;I:221–4.

[5] Wiegand T, Sullivan G, Bjøntegaard G, Luthra A. Overview of the H.264/AVC video coding standard. IEEE Transactions on Circuits and Systems for Video Technology 2003;13(7):560–76.

[6] Mobasseri B, Raikar Y. Authentication of H.264 streams by watermarking CAVLC blocks. In: SPIE Proceeding on security, steganography and watermarking of multimedia contents IX, Vol. 6505, Issue 1W, San Jose, CA, USA, Jan. 2007.

[7] Wang X, Cui C. A novel image watermarking scheme against desynchronization attacks by SVR revision. Journal of Visual Communication and Image Representation, Elsevier 2008;19(5):334–42.

[8] Sweilam N, Tharwat A, Abdel Moniem N. Support vector machine for diagnosis cancer disease: a comparative study. Egyptian Informatics Journal, Faculty of Computers and Information 2010;11(2):81–92.

[9] Abd El-Wahed W, Zaki E, El-Refaey A. Artificial immune system based neural networks for solving multi objective programming problems. Egyptian Informatics Journal, Faculty of Computers and Information 2010;11(2):59–65.

[10] Senin P. Dynamic time warping algorithm review, Tech. Report, Information and Computer Science Department, University of Hawaii, Manoa, Honolulu, USA, Dec. 2008.

[11] Mahdy T, El-Mahdy H, Onsi H. Real time watermarking of H.264 video sequences. International Journal of Digital Image Processing, CiiT Research 2011;3(1):34–42.

[12] Cox I, Kilian J, Leighton T, Shamoon T. A secure spread spectrum watermarking for multimedia. IEEE Transactions on Image Processing 1997;6(12):1673–87.

[13] Xu C, Lim Y, Feng D. Recovering modified watermarked audio based on dynamic time-warping technique, digital image computing techniques and applications, DICTA2002, Melbourne, Australia, Jan. 2002.

[14] Yeh M, Cheng K. Video copy detection by fast sequence matching. In: Proceeding of the ACM international conference on image and video retrieval, CIVR, Santorini, Greece, Jul. 2009.

[15] Yang J, Xu Y, Shang Y. An efficient parallel algorithm for longest common subsequence problem on GPUs. In: Proceedings of the world congress on engineering, WCE, Vol. I, London, UK, Jul. 2010.

[16] Dan G. Algorithms on strings, trees and sequences: computer science and computational biology. USA: Cambridge University Press; 1999, ISBN 0-521-58519-8.

[17] Hernández J, Amado M, González F. DCT-domain watermarking techniques for still images: detector performance analysis and a new structure. IEEE Transactions on Image Processing 2000;9(1).

[18] Usman I, Khan A. BCH coding and intelligent watermark embedding: employing both frequency and strength selection. Applied Soft Computing 2010;10(1):332–43.

[19] The H.264/AVC Reference Software JM 10, web site, last visit Jan. 20, 2011, online available at: < http://www.iphome.hhi.de/suehring/tml/download/old_jm/ > (last visit 12.05.11).