Authentication of H.264 streams by direct watermarking of CAVLC blocks

Bijan G. Mobasseri* and Yatish Naik Raikar

Department of Electrical and Computer Engineering

Villanova University

Villanova, PA 19085

ABSTRACT

In this work we report on the first H.264 authentication watermarker that operates directly in the bitstream with no video decoding or partial decompression. The main contribution of the work is identification of a watermarkable code space in H.264 protocol. The algorithm creates "exceptions" in H.264 code space that only the decoder understands while keeping the bitstream syntax compliant The code space is defined over the Context Adaptive Variable Length Coded(CAVLC) portion of protocol. What makes this algorithm possible is the discovery that most of H.264 code space is in fact unused. The watermarker securely maps eligible CAVLC to unused portions of the code space. Security is achieved through a shared key between embedder and decoder. The watermarked stream retains its file size, remains visually transparent, is secure against forging and is reversible. Since the watermark is placed post compression it remains fragile to re encoding and other tampering attempts.

Keywords: H.264, bitstream watermarking, CAVLC, code space

1. INTRODUCTION

It is a fact that video is rarely available in raw, uncompressed form. Therefore, it is highly desirable to develop watermarking algorithms that can be applied directly in compressed domain. Most existing watermarking algorithms, however, are either applied to raw video or in coefficient domain, requiring at least a partial decompression. If recent industry and consumer trends are any indication, H.264, otherwise known as MPEG-4, Part 10, will displace MPEG-2 as the standard video coding. Already video for hand held devices such as video iPod uses H.264 as their exclusive encoding standard. This interest is also reflected in academic research[1]. Whereas MPEG-2 was an incremental addition to MPEG-1, H.264 is substantially different from previous generation video codecs.

The scope of literature on H.264 watermarking is thin. More importantly, none of the proposed approaches are applied in the bitstream. Recent work reports on an approach where watermarking is accomplished by altering one quantized AC coefficient of macroblock of I frames. There is some work on MPEG-4 watermarking with emphasis on watermarking of video objects[3]. The problem is that video objects are not generally encoded separately in most MPEG-4 streams. H. 264 achieves superior compression even in the absence of video object encoding. Frequently, the watermark is applied to individual frames with the subsequent claim that the watermark is robust to H.264 compression[4]. Another approach is to embed the watermark in the coefficient domain of H.264 standard[5]. This approach has been the most popular approach in watermarking of JPEG and MPEG-2 video. The approach can be classified as embedding in partially decompressed video as access to transform coefficients is needed.

2. WATERMARKING IN CODE SAPCE

Watermarking in the bitstream means watermarking in the code space of the entropy coded portion of the standard. Code space of compression standards is generally not fully utilized. Even if the whole code space is used, it is still possible to construct derivative code spaces with less than 100% occupancy. MPEG-2 is an example[6]. There are a number of code spaces at work here. First is the global code space. This space consists of every valid code word in

* bijan.mobasseri@villanova.edu

the standard. Second is the used code space. This code space is a subset of the global code space and contains all the codes that actually occur in the bitstream. The used code space varies from stream to stream. Third is the unused, but valid code space. These are codes that do not appear in the stream but are nevertheless valid. Assuming the availability of unused but valid code space, watermarking can be achieved by mapping a used code word to the unused but valid portion of the code space. By doing this we have effectively created an "exception". Authorized decoders recognize this exception through a shared secret key and extract the watermark. The challenge for watermarking is that these exceptions, when forced into a stream, do not cause adverse visual impact, be unambiguously decoded, maintain syntax and preserve files size. Since there is more than one exception, an appropriate subset could be selected. Since compression protocols are different, the underlying exceptions to the protocol are different. What constitutes an exception in MPEG-2 is drastically different from H.264. To determine exceptions unique to H.264, we first present an overview of block coding in H.264 and study the associated code space. We then analyze this code space for watermark emplacement.

3. BLOCK CODING IN H.264

Context Adaptive Variable Length Coding is a new feature in H.264. For watermarking we are most interested in how residual transform coefficients are encoded. The primary block size in H.264 is 4x4 and is coded by a field consisting of 7 elements.

CAVLC: {TotalCoeff, T1, T1 sign, levels, TotalZeros, ZerosLeft, run_ before}

TotalCoeff is number of nonzero coefficients in the block, T1 represents the number of trailing ones, T1 signs are the sign of trailing ones, TotalZeros is the number of zero coefficients before the last nonzero coefficient, ZerosLeft and run_before specify the distribution of zeros in the block. {TotalCoeff, T1} is jointly coded as coeff_token. TotalZeros is jointly coded with TotalCoeff in a single VLC. The 7 element CAVLC field is, therefore, coded by 5 VLCs. If there are more than three trailing ones, only the first three are treated as ones and the remaining ±1 values are coded as normal coefficients with amplitude one. Therefore, T1 never exceeds 3. The choice of the Level VLC table keeps changing (adaptive) depending on the magnitude of each successive coded level. There are 7 VLC tables from which we can choose to encode the remaining nonzero coefficients. ZerosLeft and run_before are also jointly coded by a single VLC. As an example, a 4x4 block given by {0,3,0,1, -1,1,0,1} appears in the bitstream as {000010001110010111101101}. The problem now is to embed the watermark directly in this bitstream without decoding it first.

4. WATERMARKING CAVLC

For our purposes we define the code space of H.264 as the 5-tuple of CAVLC portion of bitstream. Each tuple is a code space onto its own. All valid combinations of 5 code spaces define the global code space. Note that there is a strong dependency among 5 code spaces and they cannot all be independently specified. For example, T1s and number of levels add up to TotalCoeff. Therefore, size of global code space is not simply the product of 5 individual code spaces, and is strongly content-dependent. Running through all valid combinations of 5 code spaces reveals that there are literally millions of different ways that a 4x4 block may be legally encoded. This code space, however, is never fully occupied in any single video. Therein lies the potential for watermarking.

Watermarking in the code space means mapping one 5-tuple to an unused location. But many of the elements of 5-tuples are untouchable. Coeff_token is one of them. There are 4 look-up tables used to encode coeff_token. The choice of the table depends on TotalCoeff in the above and left block. Altering coeff_token in one block has a destructive effect that propagates throughout and affects coding of neighboring blocks. Changing T1 and TotalZeros changes AC coefficients, their order in the zigzag scan and dramatically alter block's appearance. To maintain transparency and manageable code space we parse each block up to 3 levels deep, consisting of {coeff_token, T1 sign, Levels}. All watermarking will take place in this portion. Consider a decoded block given by {TotalCoeffs, T1, T1 sign, Levels}. We keep {TotalCoeffs, T1} unchanged for reasons given above and embed the watermark in the combination of {T1 sign, Levels}. T1 sign consists of T1 mix of +/-1s followed by TotalCoeff-T1 signed integers representing levels. To embed a watermark first identify blocks with T1<3 and at least one level with absolute value of two or less. This set, which is a subset of the used code space, is the watermarkable code space. To watermark an eligible CAVLC, we decrement (increment) a level by one unit to create +/- 1 and increase T1 to match. Watermarking table is built offline and associated mappings define the secret key. This key is a set of pointers in

watermarkable code space. This code space can be built at the decoder using H.264 standard and watermarking rule. Anyone can build this table but without the key it is unusable. If a CAVLC is on this list and is not watermarked then it means a zero is embedded. The following summarizes embedding and decoding of the watermark.

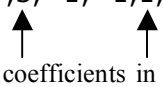
EMBEDDING

1. Parse the H.264 stream.
2. Use the lookup table generated offline to identify watermarkable CAVLC.
3. Replace CAVLC with its watermarked version.
3. Build the key to identify watermarkable code space.

DECODING
1. Acquire key from embedder.
2. Use the key to identify watermarked CAVLC.
3. Decode 1 if watermarkable CAVLC is changed.
4. Decode 0 if unchanged.
5. Strip watermark and restore video.

Here is an example,

Original block: {7,2, -1, -1,2,2, -2,7, -9}
W/m block: {7,3, -1, -1,1,2, -2,7, -8}



There are 7 nonzero coefficients in the block and 2 trailing 1s. The last 5 numbers indicate the levels of nonzero coefficients. In response to a reduction in the first level (from 2 to 1), number of trailing ones (T1) has to go up to 3 to stay within protocol syntax. Note that both embedding and decoding takes effect in the bitstream using a lookup table hence no decoding of the bitstream is needed.

5. COLLISIONS, SECURITY AND FILE SIZE

Watermarking CAVLC may cause a "collision". Collision is defined when 1) a watermarked CAVLC maps to another CAVLC that is already used in the stream or 2) two CAVLCs are watermarked to another CAVLC, even if is unused. In the first case the marked and unmarked CAVLC are indistinguishable. In the second case it is not possible to losslessly recover the watermark because watermarked CAVLC could have come from multiple mappings. For example, {7,3, -1, -1,1,2, -2,7, -8} could have come from either {7 0 -2 -1 1...} or {7 1 -1 -2 1...}. In the first case, when a CAVLC maps to a used CAVLC then it must be skipped and removed from watermarkable code space. In the second case the solution is to watermark only the CAVLC that occurs the most in the stream. This selection will be reflected in the key.

The watermarked block is still within the global code space, hence indistinguishable from unmarked blocks. To identify which blocks are watermarked, the encoder needs to generate a key and share it with the decoder. Since H.264 is an open protocol, global code space can be generated at the decoder without any help. The key that is generated at the encoder points to the watermarkable blocks. These blocks are a subset of global code space and can be similarly indexed at the decoder. For example, watermarkable code space in *foreman* consists of 25,232 CAVLC of which only 192 are actually used. The sheer combinations involved in this selection makes recovering the correct key extremely unlikely.

File size after watermarking is also of interest. By upping T1, the VLC representing the new coeff_token actually takes fewer bits. For example, {7,2}->0000 0000 101 watermarks to {7,3}->0000 0010 0. This is a reduction of 2 bits. Increase in trailing ones is accompanied by a reduction in level. There are four VLC tables to choose from. The first level is always picked from Table 0 of the H.264 standard. The following levels are picked from Table 1 if level exceeds 3. In the example shown in Section 4 level 2 is represented by 1. Once level is reduced to 1, it becomes a trailing one and is still represented by a single bit, so no change here. The next level is 2, which is

represented by 001, same as the unmarked block. Although there is no change here, there are scenarios where it takes more bits to represent the level in the marked block, depending on which table is used and the level.

6. RESULTS

We have implemented the proposed algorithm on 400 frames of *foreman*. The first step is to parse the bitstream and extract the CAVLC entropy coded portion for each 4x4 block. This is done using JVT Reference Software Version JM 10.2. Partial table appears below,

Table 1- CAVLC entropy coded portion statistics extracted from *foreman*

Frequency	TC	T1	Sign/Level	Sign/Level
10556	1	1	+/1	+/1
8960	1	1	+/1	
6661	0	0		
3057	1	1	-/1	
2821	1	1	+/1	
2139	2	2	+/1	-/1
1939	2	2	-/1	-/1
1908	2	2	-/1	+/1
1673	2	2	+/1	+/1
881	2	2	+/1	-/1

Frequency is number of times a particular CAVLC appeared in the bitstream. This data supports the expectation that most blocks are very sparsely coded. In fact, none of the above are part of watermarkable code space. Table 2 shows the statistics derived from watermarking software for the entire *foreman* video,

Table 2 - Code space statistics for *foreman*

Feature	# Blocks	Global code space	WM code space	#Used CAVLC	#WM CAVLC	#CAVLC	Cap
Count	88,632	629,424	25,232	2,136	1091	192	2308

First column is the number of 4x4 blocks. Second column is the size of global code space (number of possible CAVLC). This is the enumeration of legal combinations of {Coeff_token, T1, T1 signs, level1, level2, level3}. Although there are many more levels, we are limiting it to three to keep the key size down. Column 3 is the CAVLC in global code space that meets our watermarking condition. Column 4 is the total number of unique CAVLCs that actually occur in foreman. Column 5 is the number of CAVLCs that meets the watermarking criteria. Column 6 is the same as column 5 when collisions are taken out. Last column is payload in bits. It is possible to increase capacity if we relax level shifts to more than 1. At the present before and after video are visually indistinguishable when level shifts are limited to 1. With short videos, key size exceeds embedding capacity. This ratio becomes more favorable because as video length and capacity increases key size remains relatively stable. We also point out that key size and payload are not necessarily in competition. The key can reside on a secure web site and be downloaded independently.

7. CONCLUSIONS

We have proposed a new H.264 watermarking algorithm that is applied and decoded in compressed domain. In contrast to other algorithms that use coefficient watermarking, this paper fully exploits H.264's CAVLC structure to arrive at a reversible, file size preserving algorithm that is applied directly in compressed bitstream. The watermark can be removed, which breaks fails authentication, but cannot be forged or replaced. Since watermarkable code space is a subset of H.264 code space, presence of watermark is well concealed.

REFERENCES

1. Special Issue on 'Emerging H.264/AVC Video Coding Standard', Elsevier J. of Visual Communication & Image Representation (Special issue on Emerging H.264/AVC Video Coding Standard), Vol. 17(2), April 2006.
2. M. Noorkami, R. M. Mersereau, "Compressed-Domain Video Watermarking for H.264," 2005 IEEE International Conference on Image Processing, Volume 2, 11-14 Sept. 2005 Page(s):II - 890-3.
3. M. Barni, F. Bartolini, "Watermarking of MPEG-4 Viodeo Objects," IEEE Trans. On Multimedia, vol. 7, no. 1, February 2005.
4. D. Pröfrock, M. Schlauweg, and E. Müller, "A New Uncompressed-Domain Video Watermarking Approach Robust to H.264/AVC Compression," Proceeding (520) Signal Processing, Pattern Recognition, and Applications, Innsbruck, Austria, 14-Mar-2006.
5. H. Malvar, A. Hallapuro, M. Karczewicz, and L. Kerofsky, "Low-complexity Transform and Quantization in H.264/AVC," IEEE Transactions on Circuits and Systems for Video Technology, vol. 13, pp. 598-603, July 2003.
6. M. P. Marcinak, B. G. Mobasseri, "VLC Pair Tree: a paradigm for MPEG-2 watermarking," SPIE Conference on Security, Steganography and Watermarking of Multimedia Contents VIII, San Jose, CA, January 16-20, 2006.