Data Embedding in HEVC Video by Modifying the Partitioning of Coding Units

Tamer Shanableh [1]

[1] Department of Computer Science and Engineering, American University of Sharjah, Sharjah, UAE
*tshanableh@aus.edu

Abstract: This paper proposes a data embedding solution in HEVC videos by modifying the partitioning of Coding Units (CUs). The partitions of a CU are first represented as a sequence of binary flags. The flags pertaining to 16x16 sub CUs are used as a cover for data embedding, where 6 or 4 message bits are embedded per CU. The data embedding algorithm guarantees that a maximum of one partition is modified per message segment, therefore, in a given CU, either 0, 1 or 2 partitions are modified. The Proposed solution is assessed in terms of message payload, number of modified partitions, loss in video quality as indicated by the PSNR results, mean objective scores and excessive bitrate. The proposed solution can embed messages with up to an average payload of 32.6Kbit/s with a corresponding average distortion of less than 0.5dB. Comparisons with existing solutions reveal that the proposed solution maintains similar message payloads with less modifications of CU partitioning and at the same time resulting in less distortions for the cover video.

1. Introduction

Embedding data messages into compressed video has a number of applications including copyright protection [1], access control [2], content authentication [3], and transaction tracking [4]. Less obvious applications can also benefit from data embedding such as real time scene change detection in compressed video [5] and error detection and concealment [6]. In all cases, the process of data embedding should result in minimal video distortion whilst maintaining compatibility with the standardized decoder.

Common approaches to data embedding in compressed videos include modifying quantization scales, DCT coefficients and motion vectors. Data hiding can also be implemented using code-word substitution where the file size is strictly unaltered [7]. An example of data embedding in quantization scales and motion vectors is reported in [8]. Matrix encoding was used to modify the quantization scales and motion vectors in signal layer and scalable video coding. Advanced transcoding techniques were applied to embed messages in a pre-encoded video as well. The work in [9] embeds messages through altering the quantization scales using a machine learning approach. The decoder detects a message bit if the predicted quantization scale is different than the received one. Data embedding using MVs has also been used for video water marking [10]. Additionally, [11] proposed a solution that improves the security of motion vector-based data embedding. Because of its popularity, detection of motion vector-based video data embedding became an important research topic as reported in [12] and [13].

Of relevance to our work is the concept of data embedding using block structure and prediction modes. For instance, the work in [14] proposed the use of intra prediction modes to hide one message bit per 4x4 intra blocks in H264/AVC video. Likewise, the authors of [15] utilized the block types of H.264/AVC blocks to hide message bits. Data embedding is also used with HEVC videos, in which data is embedded by forcing certain partitioning types for the Prediction Units (PUs) [16]. If a message bit is '0' then the PU type is restricted to: NxN, nLx2N, 2NxnU and 2NxN. If the message bit is '1' then the PU type is restricted to: 2Nx2N, nRx2N, Nx2N and 2NxnD. The work reported in [17] also altered the CU coding modes for data embedding. More specifically, a prediction solution was used at the decoder to predict the split pattern of a 32x32 CU and compare that to what is actually received in the bit stream. Based on the comparison, the embedded message bits are revealed.

Motived by the aforementioned work [16] and [17], in this work, we embed data in HEVC video by altering the partition decisions of CUs.

The rest of this paper is organized as follows. Section 2 introduces the data embedding system overview. Section 3 provides the detailed algorithms for data embedding and extraction in CU partitioning flags. Section 4 reports the experimental setup and results and Section 5 concludes the paper.

2. System Overview

In HEVC video coding, video frames are divided into non-overlapping blocks referred to as coding units (CUs). In the luminance part, a typical size of the CU is 64x64 pixels. These CUs are recursively partitioned into square shapes using a quad-tree partitioning approach. Allowed partition sizes are 64x64, 32x32, 16x16 and 8x8. The purpose of the partitioning is to reduce the prediction residual error whilst maximizing the quality of the reconstructed video. The resultant partitions are referred to as Prediction Units (PUs) and they can be partitioned further, however, PUs are not part of the proposed data embedding solution.

In this work, we represent the partitions of a CU as a sequence of binary flags as illustrated in Figure 1. The 64x64 CU in this example (i.e. CU_{64x64}) is partitioned into

four CU_{32x32}. The partitions of each CU_{32x32} is represented by 5 flags. The first flag indicates that the top left 16x16 block is further partitioned into 8x8 blocks. The second, third and fourth flags serve the same purpose for the top right, bottom left and bottom right 16x16 blocks respectively. The last or fifth flag indicates that the CU_{32x32} is partitioned into 4 parts.

Referring to Figure, the top left CU_{32x32} has 4 flags pertaining to the CU_{16x16} blocks; 0 0 0 and 1. This means that only the last CU_{16x16} is further partitioned. The last flag, which is the fifth flag belongs to the CU_{32x32} as a whole and indicates that the CU_{32x32} is partitioned into four CU_{16x16}. Therefore, the 5 flags for the first CU_{32x32} are as follows: 0 0 0 1 1.

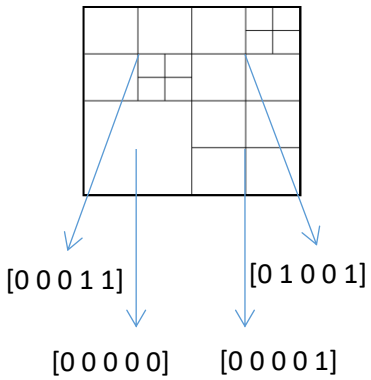

Fig. 1. Representation of CU partitioning with 20 boolean flags.

Notice that the fifth bit pertaining to a CU32x32 can be guessed only if any of its four CU16x16 is further split. For example 1 0 0 0, means that the top left CU16x16 is further split into four CU8x8. In this case, the fifth bit has to be a '1'. However there is a case where the fifth bit cannot be guessed, namely; if the CU32x32 is split into four CU16x16, yet none of the CU16x16 is further split then the partition flags will be 0 0 0 0 and the fifth bit is a '1' as well.

This work proposes to embed data in the first 4 flags of each CU_{32x32}, these flags belong to the partitioning of the CU_{16x16} blocks. Therefore, the total number of flags available for data embedding per CU_{64x64} is 16. In the next section, we introduce the data embedding and extraction algorithms and in the experimental results section we study the effect of modifying the partitioning flags on the quality of the reconstructed video.

3. Data embedding and extraction algorithms

In the context of this work, the purpose of data embedding to hide a sequence of message bits whilst striking a balance between the number of modified CU_{64x64} partitions and the overall message payload.

For data embedding or message hiding in CU partitions, we use the following approach. Assume that two message bits (m_1, m_2) are to be embedded/hidden, in such a case, 3 CU_{64x64} partition flags, i.e. p_1, p_2 and p_3, are needed for data embedding using the following algorithm:

Data Embedding Algorithm:
Input: 2 message bits (m_1, m_2) and 3 CU_{64x64} partition flags, p_1, p_2 and p_3.
Output: CU_{64x64} partition flags with an embedded message.

a. Initialization:
   Initialize m_1_flag, m2_flag and m_1m_2_flag to false
b. Setting the flags:
   If $m_1 \neq p_1$ then m_1_flag ← true
   If $m_2 \neq p_2$ then m_2_flag ← true
   If (m_1_flag && m_2_flag) then m_1m_2_flag ← true
c. Modifying the CU_{64x64} partitions
   If m_1m_2_flag then $p_3 \leftarrow 1\text{-}p_3$
   Else if m_1_flag then $p_1 \leftarrow 1\text{-}p_1$
   Else if m_2_flag then $p_2 \leftarrow 1\text{-}p_2$
   Else no modification is needed

As a consequence, a maximum of one CU_{64x64} partition flag is modified per message segment which is 2 bits in this example. The extension of this algorithm to allow longer message segments of length 3 and 4 bits is straightforward.

In general, if the message segment length is *n* bits then 2^n-1 CU_{64x64} partitions are needed for data embedding. With 16 CU_{64x64} partitions, if the message segment length is 2, then 10 message bits can be hidden as each 2 message bits requires 3 partitions. Likewise, if the message segment length is 3, then 6 message bits can be hidden and lastly, if the message segment length is 4, then only 4 message bits can be hidden. Clearly, the larger the message length, the lower the message payload and as a consequence, the distortion caused to the modified video is reduced.

For data or message extraction, the partitions of non-skipped CUs are retrieved from the bit stream of the video. Continuing with the example of using a message segment length of 2 bits, the extractor arranges the partitions into groups of 3 flags. Then the following algorithm is used for message extraction:

Data Extraction Algorithm:
Input: CU_{64x64} partition flags , p_1, p_2 and p_3, with embedded message.
Output: 2 message bits (m_1, m_2) and 3 CU_{64x64} partition flags.

a. Arrange CU_{64x64} partitions into groups of 2^n-1
b. For each group of partition flags:
   $m_1 \leftarrow (p_1+p_3)$ modulus 2
   $m_2 \leftarrow (p_2+p_3)$ modulus 2

Where the "x modulus y" computes the remainder of dividing x by y. Again, the extension of this algorithm to the case of *n*=3 and *n*=4 is straightforward. These data embedding and extraction algorithms are original and produce the same result as matrix encoding [18].

4. Experimental Results

In the experimental results to follow, message bits are embedded in CU_{16x16} partitioning flags of HEVC video. The closest reported work that can be used for comparison was published in [16] and [17]. We therefore use the same experimental setup as reported in [16] and [17]. The following test video sequences are used: Tennis

(1920×1080@24Hz), FourPeople (1280×720@60Hz), BasketballDrill (832×480@50Hz) and BasketballPass (416×240@50Hz). The video sequences are coded with constant bitrates with the following values: 500Kbit/s, 1Mbit/s, 5Mbit/s and 10Mbit/s. Since the BasketballPass has a low spatial resolution it is coded at 100Kbit/s, 1Mbit/s and 5Mbit/s.

All messages to be embedded are generated using a uniform discrete binary number generator.

We use the HEVC reference software HM13.0 [19]. The video coding structure is IPPP… using 4 reference frames. The maximum CU size is set to the typical maximum size of 64x64 pixels. The asymmetric motion partitions tool and the adaptive loop filter tool are both enabled.

In the results to follow, we report the effect of data embedding in HEVC videos using PSNR, message payload in Kbit per second of video, and amount of modified 16x16 partitioning flags, again measured as Kbit per second averaged over the entire video. Table 1 includes the results of data embedding using two approaches; embedding 4 message bits into 15 CU_{64x64} partitioning flags and embedding 3 message bits into 7 partitioning flags. In the latter approach, since there are 16 partitioning flags in a CU_{64x64}, then data embedding can be applied twice, resulting in embedding 6 message bits per CU_{64x64}.

Table 1 Proposed data embedding solutions with message segment lengths of n=3 and n=4.

		6 msg bits in 14 partition flags			4 msg bits in 15 partition flags		
	Bitrate Mbit/s	PSNR Loss dB	Msg Kbit/s	Modified partitions Kbit/s	PSNR Loss dB	Msg Kbit/s	Modified partitions Kbit/s
Tennis	0.5	1.96	41.14	12.02	1.5	34.6	6.84
	1	1.09	49.65	14.48	0.82	37.3	7.75
	5	0.24	62.68	18.27	0.19	48.1	10.22
	10	0.13	65.05	19.00	0.11	48.9	10.46
	Avg	0.86	54.63	15.94	0.52	42.23	7.13
FourPeople	0.5	0.59	28.67	8.36	0.47	19.1	4.72
	1	0.30	33.43	9.72	0.26	22.3	5.48
	5	0.08	52.63	15.36	0.08	35.1	8.36
	10	0.07	64.10	18.69	0.07	43	10.12
	Avg	0.26	44.71	13.03	0.22	29.88	7.17
Basketball Drill	0.5	0.65	20.23	5.90	0.52	13.5	3.2
	1	0.43	22.71	6.63	0.35	15.1	3.5
	5	0.22	27.41	7.98	0.18	18.3	4.3
	10	0.13	27.86	8.15	0.1	18.6	4.4
	Avg	0.36	24.55	7.16	0.22	16.45	2.86
Basketball Pass	0.1	0.86	5.84	1.71	0.74	3.9	0.96
	0.5	0.33	6.68	1.94	0.29	4.5	1.07
	1	0.25	6.91	2.03	0.22	4.6	1.12
	Avg	0.48	6.48	1.89	0.42	4.33	1.05
Overall	Avg	0.49	32.59	9.51	0.34	23.22	4.55

As expected, the results in Table 1 indicate that increasing the number of message bits from 4 to 6 per CU_{64x64} results in higher average message bitrate. Namely; the averages reported in the Table are 23.2Kbit/s and 32.6Kbit/s for 4 and 6 message bits per CU_{64x64} respectively. As a consequence, the average bitrate of modified partitions increased from 4.6Kbit/s to 9.5Kbit/s and the average PSNR loss increased from 0.34dB to 0.49dB. There are other observations that can be made from the presented results, for example the order of test videos are presented in descending spatial resolution order, therefore, the average message payload ranges from 54.6 Kbit/s to 6.5 Kbit/s. Likewise, it is shown that data embedding at low bitrates for each video sequence, results in higher distortion as indicated by the PSNR loss. Lastly, the message payload increases as the video compression bitrate increases for each sequence. This is so, because the percentage of skipped CUs decreases accordingly and therefore more CUs are available for data embedding.

In Table 2, we compare this work against the work reported in [16] and [17]. On average, 4 partitions per CU_{64x64} are modified in the reviewed work for the embedding of 4 message bits. Therefore, in Table 2, we compare the results of our 4 message bits per CU_{64x64} solution with the existing work as both have the same message embedding rate. The results in the table indicate that the average bitrate of the modified CU_{64x64} partitions using the proposed solution is around one fourth of that of the reviewed work [17], namely, the bitrates are 4.6 Kbit/s and 22.9 Kbit/s respectively. This is so, as in the proposed solution a maximum of one partition is modified for the purpose of embedding 4 message bits as explained in Section 3.

The fact that the modified CU_{64x64} partitions are less, the PSNR loss caused by the proposed solution is 0.34 dB whereas that of the reviewed solution is 0.47 dB. These two conclusions are consistent for each and every test video sequence.

Lastly, the work reported in [16] achieves a high message embedding rate of 80.1Kbit/s, however, as expected, this comes at a high cost in terms of quality degradation. As a result, the average drop in PSNR is around 2dB. The highest drop in PSNR was reported for the Tennis sequence which is 3.8 dB and the lowest drop reported was for the Basketball Drill sequence which is 1 dB. Therefore, increasing the message payload in HEVC video by altering coding modes is not desirable. The proposed solution on the other hand, presents a balanced trade-off between message payload and quality degradation.

In Table 3 we present the percentage of excessive bitrate as a result of data embedding. The excessive bitrate is computed in comparison to regular encoding without data embedding. Namely, the bitrate resulting from data embedding is subtracted from that of regular encoding, and the difference is divided by the bitrate of the latter.

To compute the excessive bitrate, Variable Bit Rate (VBR) coding is required. This is achieved by fixing the quantization scale/parameter (QP). In HEVC coding, it is custom to use the following QP values in VBR testing; 22, 27, 32 and 37, therefore we use these QP values in the results presented in Table 3. We also compare the excessive bitrate of the proposed solution against that in [17].

Table 2 Comparison with existing work with CBR coding.

		Reviewed [17]			Reviewed [16]		Proposed 4 bits in 15 partitions		
	Bitrate Mbit/s	PSNR loss dB	Msg Kbit/s	Modified Partitions Kbit/s	PSNR loss dB	Msg Kbit/s	PSNR loss dB	Msg Kbit/s	Modified partitions Kbit/s
Tennis	0.5	1.45	34.6	27.4	8.21	146.3	1.5	34.6	6.84
	1	0.76	37.3	37.5	5.44	147.2	0.82	37.3	7.75
	5	0.23	48.1	48.1	0.95	188.2	0.19	48.1	10.22
	10	0.13	48.9	50.7	0.51	223.2	0.11	48.9	10.46
	Avg	0.64	42.23	40.93	3.78	176.23	0.52	42.23	7.13
Four People	0.5	0.5	19.1	19.1	4.78	61.9	0.47	19.1	4.72
	1	0.33	22.3	22.3	1.26	66.6	0.26	22.3	5.48
	5	0.1	35.1	35.1	0.22	94.8	0.08	35.1	8.36
	10	0.09	43	42.7	0.18	116.1	0.07	43	10.12
	Avg	0.26	29.88	29.8	1.61	84.85	0.22	29.88	7.17
Basket ball Drill	0.5	0.6	13.5	13.5	1.65	30.1	0.52	13.5	3.2
	1	0.48	15.4	15.1	1.09	38.5	0.35	15.1	3.5
	5	0.36	18.3	18.3	0.71	59.2	0.18	18.3	4.3
	10	0.2	18.6	18.6	0.55	69.9	0.1	18.6	4.4
	Avg	0.41	16.45	16.38	1.00	49.43	0.22	16.45	2.86
Basket ball Pass	0.1	0.7	3.9	3.9	2.25	7.8	0.74	3.9	0.96
	0.5	0.54	4.5	4.5	0.95	13.9	0.29	4.5	1.07
	1	0.47	4.6	4.6	0.76	17.1	0.22	4.6	1.12
	Avg	0.57	4.33	4.33	1.32	12.93	0.42	4.33	1.05
Overall	Avg	0.47	23.22	22.86	1.93	80.86	0.34	23.22	4.55

Table 3 shows that the proposed solution has a lower average excessive bitrate in comparison to the reviewed work. The average is lower for each and every test sequence, the overall averages of the proposed solution and the reviewed work are 9.5% and 11.5% respectively.

It is also shown in the table that the excessive bitrate increases as the QP increases. This is an expected result, as increasing the QP in regular HEVC encoding, a higher percentage of CUs are skipped as more DCT coefficients are quantized to zero. With data embedding on the other hand, the structure of CUs is altered by modifying the split decisions, this results in sub-optimal interframe prediction and therefore a reduction in the percentage of skipped CUs.

Table 3 Comparison with existing work with VBR coding.

		Reviewed [17]	Proposed work
	QP	Excessive bitrate Kbit/s (%)	Excessive bitrate Kbit/s (%)
Tennis	22	7.4	5.5
	27	10.3	8.7
	32	14.5	14.1
	37	21.2	21.8
	Avg	13.4	12.5
FourPeople	0.5	7.0	5.9
	1	11.2	9.4
	5	15.1	14.5
	10	20.6	21.3
	Avg	13.5	12.8
Basketball	0.5	7.2	3.5
Drill	1	9.2	5.4
	5	11.2	8.5
	10	14.3	13.5
	Avg	10.5	7.7
Basketball	0.1	5.3	2.1
Pass	0.5	7.4	3.2
	1	9.6	5.5
		12.4	9.3
	Avg	8.7	5.0
Overall	Avg	11.5	9.5

In Table 4, we present subjective quality assessment results of the proposed solution using the Double-Stimulus Impairment Scale (DSIS) recommended by ITU-R BT.500-11 [20]. Constant bitrate video coding is used with the same compression parameters used to generate the results of Tables 1 and 2 above. In this experiment, 15 subjects are presented with a reference video followed by a video with embedded data without repetition. The average age of the subjects is 23.7 years with a range of 18-45. Since eye stress might bias the results, the presentation order of video sequences is randomized for different subjects. After watching each pair of videos, subjects use a discrete scale of 1 to 5 to score the results. The grading scales from 1 to 5 are: "very annoying", "annoying", "slightly annoying", "perceptible but not annoying" and "imperceptible" respectively.

Table 4 Statistics of objective scores.

Proposed 4 bits in 15 partitions			
	Bitrate Mbit/s	MOS (out of 5)	Stdev
Tennis	0.5	2.75	1.43
	1	3.58	0.93
	5	4.67	0.74
	10	4.67	0.57
	Avg	3.92	0.92
Four	0.5	3.83	0.88
People	1	4.58	0.61
	5	4.75	0.70
	10	4.83	0.44
	Avg	4.50	0.66
Basketball	0.5	3.42	1.37
Drill	1	4.17	1.02
	5	4.42	0.80
	10	4.67	0.57
	Avg	4.17	0.94
	0.1	3.50	1.65
Basketball	0.5	4.33	0.80
Pass	1	4.42	0.80
	Avg	4.08	1.08
Overall		4.17	0.89

Overall, the average Mean Objective Scores (MOS) for all video sequences at the presented bitrates is 4.17 out of 5 with a standard deviation of 0.89. The results indicate that as the bitrate and quality of the video increase, the mean subjective scores increase as well. At the same time, the standard deviation of the objective scores decrease. This means that data embedding at higher bitrates is less detectable. The results also show that on average, the smallest standard deviation is associated with the highest MOS, which belongs to the FourPeople sequence in this experiment.

5. Conclusion

A data embedding solution was proposed for coded HEVC videos. It was proposed to modify the CU_{64x64} partitioning flags to hide message bits. The message is first divided into segments and each segment is embedded in the partitioning flags such that a maximum of one partition per message segment is modified.

We experimented with message segments of lengths 3 and 4. In the experimental results section, such segment

lengths were shown to generate a good trade-off between message payload and video distortions. With message segment lengths of 3 and 4, six and four message bits can be embedded per CU_{64x64} respectively. As a result, using four video sequences with different resolutions, the average message payload rates were 32.6Kbit/s and 23.2Kbit/s respectively.

The corresponding average losses in PSNR were 0.49dB and 0.34dB respectively. Therefore, increasing the message segment length reduces both the message payload and the overall video distortion. Subjective video testing revealed that the average MOS for all video sequences at various bitrates is 4.17 out of 5 with a standard deviation of 0.89. Comparison with existing work revealed that the proposed solution results in less video distortion in terms of PSNR and excessive bitrate. Yet the proposed solution provided a reasonable balance between message payload and video distortion.

6. References

[1] Tian, L., Zheng, N., Xue, J., Li, C., Wang, X.: 'An integrated visual saliency-based watermarking approach for synchronous image authentication and copyright protection', Signal Processing: Image Communication, October 2011, 26(8-9), pp. 427-437

[2] Chang, F.C., Huang, H.C., Hang, H.M.: 'Layered access control schemes on watermarked scalable media', Journal of VLSI Signal Processing, 2007, 49(2007), pp. 443–455

[3] Su, P.C., Wu, C.-S., Chen, I.-F., Wu, C.-Y., Wu, Y.-C.: 'A practical design of digital video watermarking in H.264/AVC for content authentication', Signal Processing: Image Communication, October, 2011, 26(8-9), pp. 413-426

[4] Emmanuel, S., Vinod, A., Rajan, D., Heng, C.K.: 'An Authentication Watermarking Scheme with Transaction Tracking Enabled', Proc. Digital EcoSystems and Technologies Conference, Inaugural, February, 2007

[5] Kapotas, S., Skodras, A.: 'A new data hiding scheme for scene change detection in H.264 encoded video sequences', Proc. IEEE International Conference on Multimedia and Expo, Hannover, Germany, June 2008, pp.277-280

[6] Yilmaz, A. Aydin, A.: 'Error detection and concealment for video transmission using information hiding', Signal Processing: Image Communication, April 2008, 23(4), pp. 298-312

[7] Xu, D., Wang, R., Shi, Y.Q.: 'Data Hiding in Encrypted H.264/AVC Video Streams by Codeword Substitution', IEEE Transactions on Information Forensics and Security, April 2014, 9(4), pp. 596-606

[8] Shanableh, T.: 'Matrix encoding for data hiding using multilayer video coding and transcoding solutions', Signal Processing: Image Communication, Elsevier, October, 2012, 27(9), pp. 1025-1034

[9] Shanableh, T.: 'Data Hiding in MPEG Video Files Using Multivariate Regression and Flexible Macroblock Ordering', IEEE Transactions on Information Forensics and Security, April, 2012, 7(2), pp.455-464

[10] T. Stütz, T., Autrusseau, F., Uhl, A.: 'Non-Blind Structure-Preserving Substitution Watermarking of H.264/CAVLC Inter-Frames', IEEE Transactions on Multimedia, Aug. 2014, 16(5), pp. 1337-1349

[11] Cao, Y., Zhang, H., Zhao, X., Yu, H.: 'Covert Communication by Compressed Videos Exploiting the Uncertainty of Motion Estimation', IEEE Communications Letters, February 2015, 19(2), pp. 203-206

[12] Wang, K., Zhao, H., Wang, H.: 'Video Steganalysis Against Motion Vector-Based Steganography by Adding or Subtracting One Motion Vector Value', IEEE Transactions on Information Forensics and Security, May 2014. 9(5), 741-751

[13] Tasdemir, K., Kurugollu, F., Sezer, S.: 'Spatio-Temporal Rich Model-Based Video Steganalysis on Cross Sections of Motion Vector Planes', IEEE Transactions on Image Processing, July 2016, 25(7), pp. 3316-3328

[14] Hu, Y., Zhang, C., Su, Y.: 'Information Hiding Based on Intra Prediction Modes H.264/AVC', Proc. IEEE International Conference on Multimedia and Expo, Beijing, China, July 2007, pp.1231-1234

[15] Yang, G., Li, J., He, Y., Kang, Z.: 'An information hiding algorithm based on intra-prediction modes and matrix coding for H.264/AVC video stream', International Journal of Electronics and Communications, April, 2011, (65)4, pp. 331-337

[16] Tew, Y., Wong, K.: 'Information hiding in HEVC standard using adaptive coding block size decision', Proc. IEEE International Conference on Image Processing, Paris, France, October 2014, pp. 5502-5506

[17] Shanableh, T.,: 'Altering Split Decisions of Coding Units for Message Embedding in HEVC', Multimedia and applications, Springer, May 2017, 77(7), pp. 8939–8953

[18] R. Crandall: 'Some Notes on Steganography', 1998, http://dde.binghamton.edu/download/Crandall_matrix.pdf, accessed 20 February 2019

[19] Kim, I.-K., McCann, K., Sugimoto, K., Bross, B., Han, W.-J., Sullivan, G.: 'High Efficiency Video Coding (HEVC) Test Model 13 (HM13) Encoder Description', Document: JCTVC-O1002, Joint Collaborative Team on Video Coding (JCT-VC) of ITU-T SG16 WP3 and ISO/IEC JTC1/SC29/WG11, 15th Meeting: Geneva, CH, 23 Oct. – 1 November 2013

[20] 'Methodology for the Subjective Assessment of the Quality of TV Pictures', Recommendation ITU-R BT.500-11, January 2002

Tamer Shanableh received his Ph.D. in Electronic Systems Engineering in 2002 from the University of Essex, UK. From 1998 to 2001, he was a senior research officer at the University of Essex, during which, he collaborated with BTexact on inventing video transcoders. He joined Motorola UK Research Labs in 2001. During his affiliation with Motorola, he contributed to establishing a new profile within the ISO/IEC MPEG-4 known as the Error Resilient Simple Scalable Profile. He joined the American University of Sharjah in 2002 and is currently a professor of computer science. Dr. Shanableh spent the summers of 2003, 2004, 2006, 2007 and 2008 as a visiting professor at Motorola multimedia Labs. He spent the spring semester of 2012 as a visiting academic at the Multimedia and Computer Vision and Lab at the School of Electronic Engineering and

Computer Science, Queen Mary, University of London, London, U.K . His research interests include digital video processing and pattern recognition.