Date of publication xxxx 00, 0000, date of current version xxxx 00, 0000.

Digital Object Identifier 10.1109/ACCESS.2021.Doi Number

HEVC Video Encryption with High Capacity Message Embedding by Altering Picture Reference Indices and Motion Vectors

Tamer Shanableh[1], Senior member, IEEE

[1]American University of Sharjah, Sharjah, UAE.

Corresponding author: Tamer Shanableh (e-mail: tshanableh@aus.edu).

ABSTRACT A high capacity message embedding in encrypted HEVC video is proposed in this paper. The challenges addressed in this paper include keeping the encrypted video compliant with standardized decoders, correctly decrypting the video and finally, correctly extracting the message bits. The message embedding is achieved by altering the values of reference picture indices and motion vectors which results in scrambled video. Sixteen picture references are used in this work and therefore, combined with alteration of motion vectors, a maximum of six message bits can be embedded per coding unit. Motion vectors are altered by swapping their *x* and *y* components and/or changing their signs. This is achieved with full compliance with the HEVC video syntax. To extract message bits, an authorized decoder builds a classification model per video sequence and uses it for predicting the true values of the reference indices and motion vectors. As such, message bits are extracted and the video is correctly reconstructed to its unscrambled state. Coding units that result in misclassification are identified at the encoder and excluded from message embedding. This results in slightly lower embedding rates but ensures accurate video reconstruction. Using nine video sequences of various resolutions that are compressed using four different quantization parameters, the experimental results revealed that the true average message embedding rate is 2.7 bits per coding unit or 173 kbit/s. This is achieved with accurate video reconstruction at the expense of increasing the bitrate of the encoder by 3%. Comparison with existing work shows that the proposed solution is superior in terms of embedding capacity whilst reducing the excessive bitrate of the encoder.

INDEX TERMS Data embedding, Machine learning, Video coding, Video encryption

I. INTRODUCTION

To authenticate digital video and ensure its confidentiality and integrity, video encryption [1] and data embedding techniques are used [2]. Video encryption can be used to increase the difficulty of piracy in digital videos and to protect privacy [3], [4]. In all cases, authorized users can restore the encrypted video to its original state [5]. Additionally, with cloud storage becoming feasible and popular, customers might wish to encrypt their videos prior to outsourcing. This is needed as data security is a major obstacle for cloud adoption. For tampering detection, a cloud server manager can embed labeling and authentication data into encrypted video [6].

One approach to protect digital video is through encrypting the entire video standard cipher algorithms such as the Advanced Encryption Standard (AES) [7], however, the encrypted video will be no longer be compliant with standardized decoders which results in prohibiting post processing techniques like video transcoding and watermarking. Thus, video encryption is typically archived by altering selective syntax elements such as the sign of DCT coefficients [8], altering intra prediction modes [9] or altering motion vector difference signs [10].

Video encryption can be combined with data embedding in AVC and HEVC videos as reported in [11], and [12]. The work in [5] proposed a comprehensive solution for encryption and data embedding by altering intra prediction modes, motion vector differences and quantized DCT coefficients of AVC videos. In [13] CABAC bin string substitution is used to embed data in partially encrypted AVC videos. The encryption is performed by altering various syntax elements such that the receiver extract embedded data in the encrypted domain using only the data-hiding key. An improved version of aforementioned solution was reported in [14] where encryption and data embedding did not affect the bitrate and maintained the full bitstream compliance.

Video encryption combined with data embedding is also reported for HEVC videos. A pioneering work was reported in [15] where visual protection of video is achieved by encrypting HEVC-CABAC bin strings whilst maintaining compatibility with standardized decoders and without causing an increase in video bitrate.

In [16], motion vector differences, intra modes and quantized DCT coefficients of HEVC videos are altered to achieve these two tasks. An enhanced version of this work was reported in [17] where the data embedding rate is increased without resulting in excessive video bitrate.

In this work, we perform partial encryption to HEVC videos by altering picture reference indices and motion vectors. To the Such alteration results in scrambled video by means of encryption. Again, to ensure compatibility with standardized HEVC decoders, standard cipher algorithms are not employed in this work.

The major advantage of the proposed work is that we embed up to six message bits in each coding unit of the video. This is made possible through altering the picture reference indices and the motion vectors at the syntax level. As a result, the video becomes decodable by standardized decoders yet scrambled. We show that by using relevant feature variables, a machine-learning model can be built to unscramble the video and extract message bits.

To the best of our knowledge, video encryption and data embedding by altering picture references and motion vectors, as opposed to motion vector differences, is novel. Likewise, the operations at the decoder's side to extract the embedded data and reconstruct the video by predicting the original values of the picture references and motion vectors using machine learning is also novel.

The rest of this paper is organized as follows. The overview of the proposed system is presented in Section 2. The proposed message embedding solution is introduced in Section 3. This is followed by the proposed message extraction solution in Section 4. Sections 5 and 6 present the proposed feature extraction and classification solutions. The experimental results are presented in Section 7 and the conclusion is presented in Section 8.

II. SYSTEM OVERVIEW

In the proposed system, message bits are embedded into a compressed HEVC video by means of altering the reference index of Coding Units (CUs) and their motion vectors. The embedding takes place at the bit stream level and therefore the locally decoded images of the encoder remain intact. However, the generated video bit stream has altered syntax elements and therefore decoding it results in a scrambled video. As such, the output of the proposed encoding process is an encrypted bit stream that embeds message bits. Full information about HEVC syntax and semantics are found in [18]. The proposed encoding arrangement is further illustrated in Figure 1. The details of message embedding and the alteration of the reference indices and motion vectors are presented in Section 3.

A standardized video decoder will be able to decode the bit stream into a scrambled video. This task is archived without crashing the decoder as the generated bit stream is standard compliant in terms of syntax. On the other hand, in the proposed decoding solution, a classification model is employed to predict the correct values of the reference indices and motion vectors. As such, the embedded message bits are extracted and the video can be reconstructed correctly to its unscrambled state. This arrangement is further illustrated in Figure 2.

The proposed message extraction solution is presented in Section 4 and the proposed classification solution is presented in Sections 5 and 6.

III. PROPOSED MESSAGE EMBEDDING

To embed message bits in a CU, the following motion information is altered; Vx, Vy and CU reference index (i.e ref_idx). Two bits can be embedded in Vx and Vy and 4 bits can be embedded in ref_idx as follows. For message bits 00, Vx and Vy remain as is. For bits 01, Vx and Vy are swapped and the first vector component is multiplied by -1, which results in (-Vy,Vx). For bits 10, both vector components are negated, which results in (-Vx,-Vy). And lastly, for bits 11, Vx and Vy are swapped and the second vector component is multiplied by -1, which results in (Vy,-Vx). This is arrangement is presented in Equation (1):

$$Altered\ MV = \begin{cases} (Vx,Vy), & message\ bits=00 \\ (-Vy,Vx), & message\ bits=01 \\ (-Vx,-Vy), & message\ bits=10 \\ (Vy,-Vx), & message\ bits=11 \end{cases} \quad (1)$$

These altered MV values are chosen to maximize the Euclidean distance between them and at the same time be reproducible at the decoder for message extraction.

Additional 4 bits are embedded by altering the CU reference index according to the message bits as shown in Equation (2):

$$Altered\ Ref\_idx = \begin{cases} 1, & message\ bits=0000 \\ 2, & message\ bits=0001 \\ & \dots \\ 15, & message\ bits=1110 \\ 16, & message\ bits=1111 \end{cases} \quad (2)$$

The full arrangement of message embedding is listed in Table 1. Notice that in the third column a dash is used in message bits to emphasize that the first 2 bits are a result of MV alteration and the rest of the bits are a result of ref_idx alternation.

TABLE 1.
Altering CU reference indices and MVs for message embedding.

MV alteration	Reference index alteration	Message bits
(Vx, Vy)	ref_idx = 1	00-0000
	ref_idx = 2	00-0001
	…	...
	ref_idx = 15	00-1110
	ref_idx = 16	00-1111
(-Vy, Vx)	ref_idx = 1	01-0000
	ref_idx = 2	01-0001
	…	...
	ref_idx = 15	01-1110
	ref_idx = 16	01-1111
(-Vx, -Vy)	ref_idx = 1	10-0000
	ref_idx = 2	10-0001
	…	...
	ref_idx = 15	10-1110
	ref_idx = 16	10-1111
(Vy, -Vx)	ref_idx = 1	11-0000
	ref_idx = 2	11-0001
	…	...
	ref_idx = 15	11-1110
	ref_idx = 16	11-1111

The alteration of the MVs and reference indices are carried out as a post process at the encoder. This means that the alerted values are stored in the output bit stream but not used for motion estimation and composition. Consequently, an authorized decoder has three tasks to carry out. Firstly, it will predict the values of the unaltered MVs and unaltered reference indices; secondly, it will extract the embedded bits, and finally it will reconstruct the video. These steps are elaborated upon in the sections to follow.

Only inter-coded CUs with MVs can embed 6 message bits in this proposed solution. Intra-coded CUs and skipped CUs are not used for message embedding. However, inter-coded CUs with nil MVs can still be used to embed 4 bits in the reference index. The maximum number of message bits per CU type is listed in Table 2.

TABLE 2.
Maximum number of message bits per CU.

CU Type	Bits in MV	Bits in reference index	Total bits
Inter-coded CU with non-nil MV	2	4	6
Inter-coded CU with nil MV	0	4	4
Skipped CU	0	0	0
Intra-coded CU	0	0	0

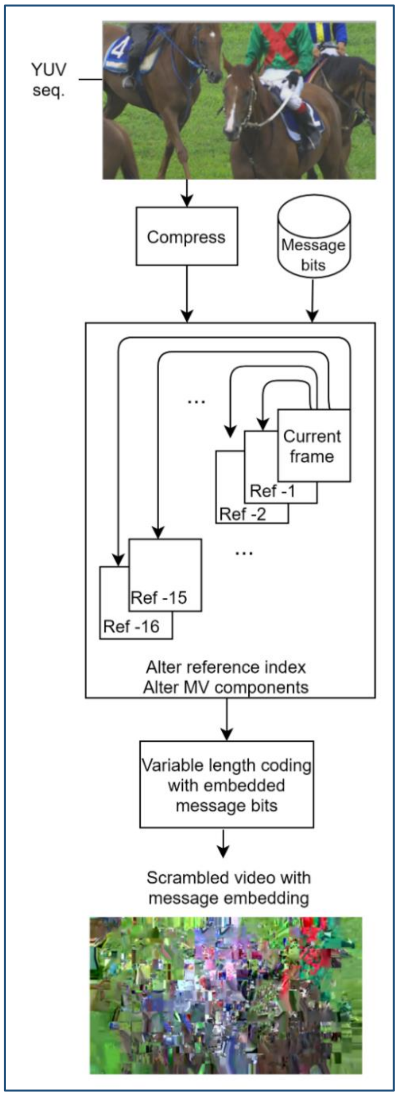

Figure 1. System overview of proposed message embedding and encryption solution.

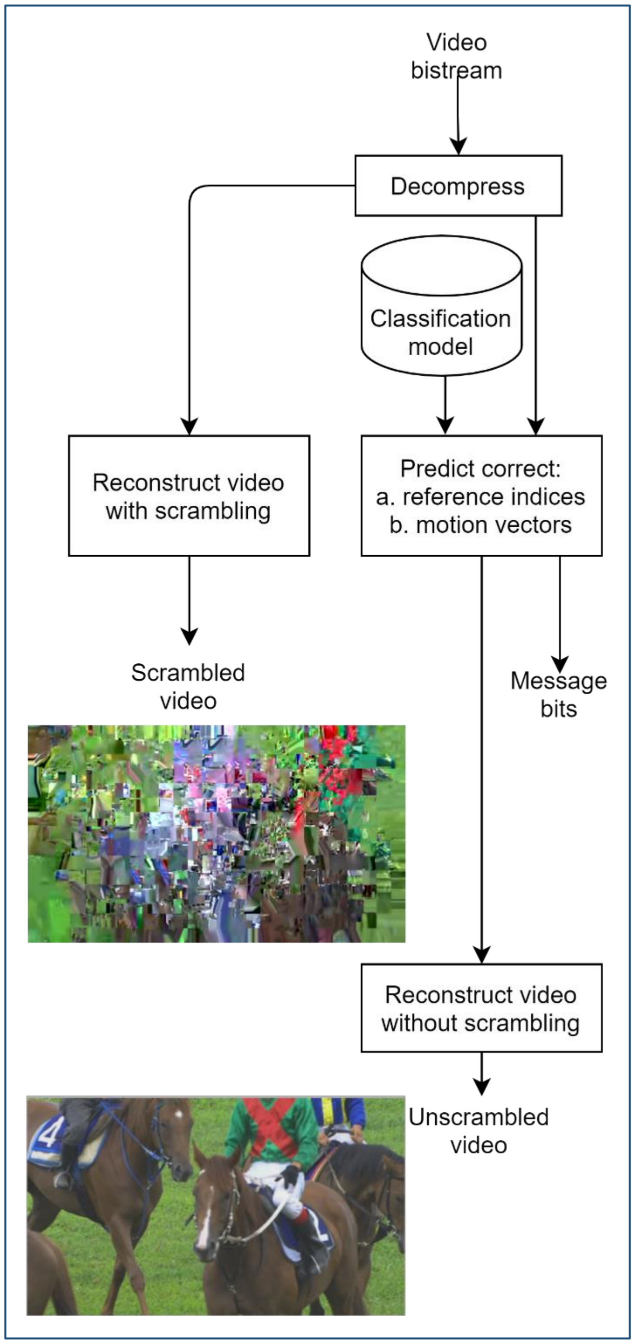

Figure 2. System overview of proposed message extracting and decryption solution.

IV. PROPOSED MESSAGE EXTRACTION SOLUTION

To extract the message bits, the process of Figure 3 is followed. Expressly, the ref_index and Vx and Vy are decoded for a given CU. Four variants of Vx and Vy are created, namely; (Vx,Vy), (-Vy,Vx),(-Vx,Vy) and (Vy,-Vx). Feature vectors are formed using each of these variants with 16 different reference pictures. The resultant 64 feature vectors are classified, only the feature vector belonging to the unaltered MV and unaltered reference index will result in positive classification. Consequently, 6 message bits are extracted according to the MV variant and decoded ref_idx as illustrated in Figure 3. The 2 message bits hid in MVs are extracted by counting the number of 90^0 counterclockwise rotations between the classified MV and the decoded one.

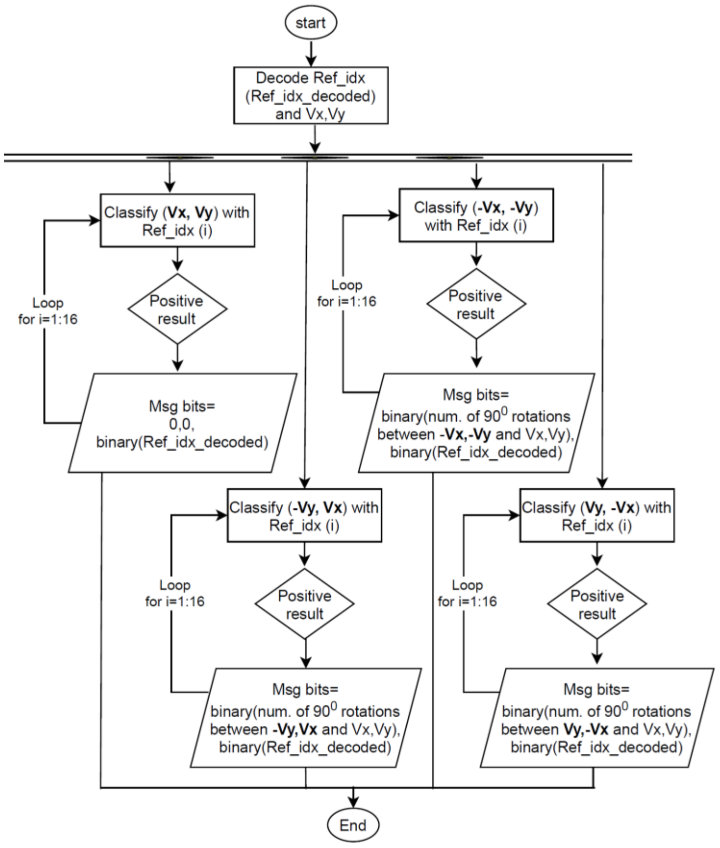
start
Decode Ref_idx (Ref_idx_decoded) and Vx,Vy
Classify (Vx, Vy) with Ref_idx (i)
Positive result
Loop for i=1:16
Msg bits= 0,0, binary(Ref_idx_decoded)
Classify (-Vx, -Vy) with Ref_idx (i)
Msg bits= binary(num. of 90⁰ rotations between -Vx,-Vy and Vx,Vy), binary(Ref_idx_decoded)
Classify (-Vy, Vx) with Ref_idx (i)
Msg bits= binary(num. of 90⁰ rotations between -Vy,Vx and Vx,Vy), binary(Ref_idx_decoded)
Classify (Vy, -Vx) with Ref_idx (i)
Msg bits= binary(num. of 90⁰ rotations between Vy,-Vx and Vx,Vy), binary(Ref_idx_decoded)
End

Figure 3. Data extraction flowchart for one coding unit.

For completion, a simple numerical example of data embedding and extraction is as follows. Assume that a message segment contains the following bits: 110101. Further assume that the motion vector of the underlying coding unit is (-1,2) with a picture reference index of 1. The first 2 message bits are embedded by altering the motion vector value to (2,1) according to Equation 1. The latter 4 message bits are embedded by altering the picture reference index from 1 to 5 (which is the decimal value of 0101). An authorized decoder receives this information and generates 4 alternatives of the received motion vector (i.e (2,1)) which are {(2,1),(-1,2),(-2,-1),(1,-2)} according to Equation 1 above. Each motion vector is then combined with picture references indices 1 to 16 to generate 64 feature vectors as elaborated upon in Section V. The feature vector belonging the combination of the motion vector (-1,2) and picture reference index 1 results in positive classification as elaborated upon in Section VI. The number of 90^0 counterclockwise rotations between the classified MV (-1,2) and the decoded one (2,1) is 3, which is 11 in binary. The decoded picture reference index is 5 which is 101 in binary. The authorized decoder knowns that the number of bits embedded in reference indices are 4 hence it represents 5 as 0101. Combining both sets of bits, the sequence 110101 is extracted at the authorized decoder.

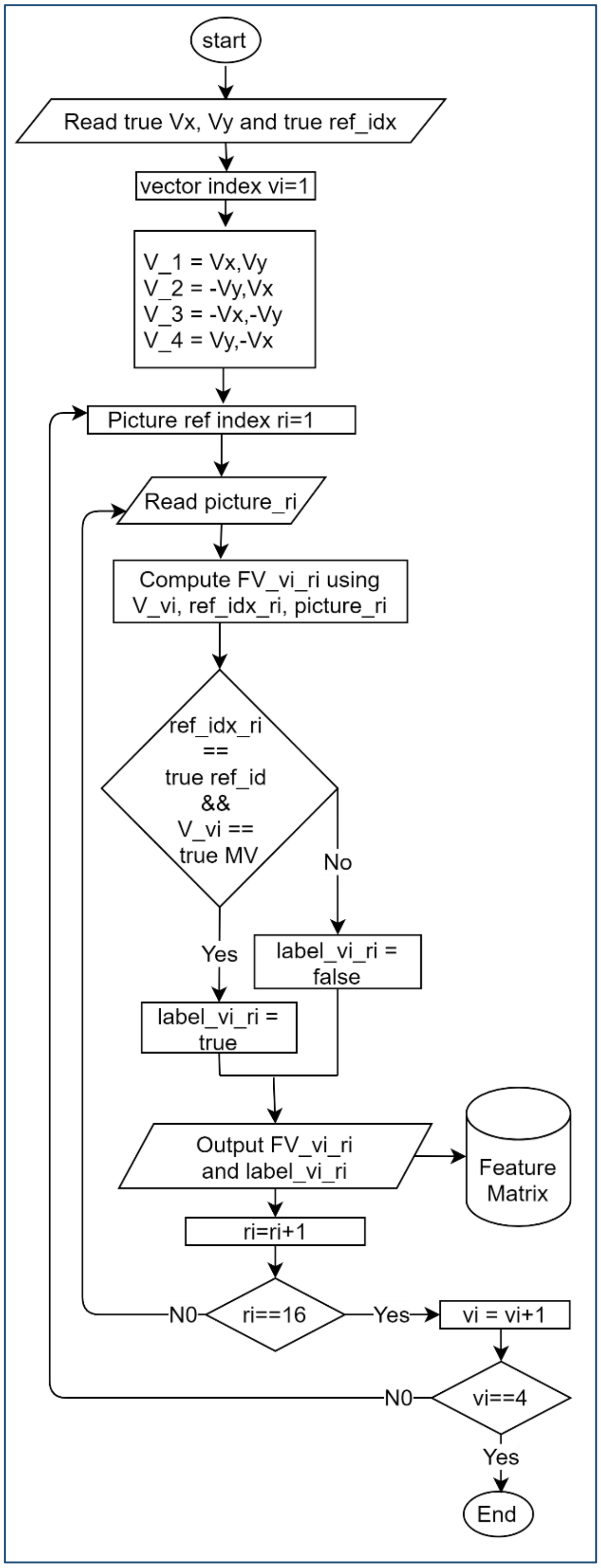

Figure 4. Flowchart of proposed feature extraction approach

Additionally, as a result of this classification process, an authorized decoder identifies the unaltered MVs and unaltered reference indices and therefore be able to correctly decode and reconstruct the video sequence.

In practical situations, the message to be embedded can be prefixed by its length in bits, such that an authorized decoder will know when to stop extracting message bits. Another approach is to append the message with an end-of-message symbol known to the authorized decoder. These extra bits are embedded using the same proposed solution of altering motion vectors and picture reference indices. The proposed embedding solution is HEVC syntax-friendly, hence, a non-authorized decoder can still decode the video but the content will be scrambled as mentioned in Section III above.

V. PROPOSED FEATURE EXTRACTION SOLUTION

In this proposed solution, feature vectors are formed using 16 reference frames and 4 variants of MVs as explained previously. Thus, the total number of feature vectors per CU is 4*16=64. Only one of these 64 feature vectors is labeled as positive and the rest are labeled as negative. The process of generating these feature vectors is illustrated in Figure 4.

The Feature variables used to construct a feature vector are based on bit stream information and reconstructed CUs.

a. Feature variables from received bit stream:

- Reference index differences between the received ref_idx and the ones belonging to the top and left CUs. A smaller difference is preferred.
- MV Phase differences between the received MV and the co-located MV of the previous frame, the MV of the top CU and the MV of the left CU. The difference is discretized into eight categories each of which is 45 degrees. A smaller phase difference is preferred.

Summation of variance of Vx and Vy components of the received MV and its surrounding top and left MVs [19]. A smaller summation of variance is preferred.

b. Feature variables form reconstructed CUs:

- Sum of pixel values outside the range of 0 to 255. A smaller summation is preferred.
- Pixel statistics computed from reconstructed CUs including entropy, average and variance.
- Sum of pixel difference in the x-direction and the y-direction [20].
- Sum of pixel differences across the borders with top and left CUs.

VI. CLASSIFICATION AND CU MARKING

Again, the feature variables introduced in Section 5 are calculated for each combination of reference index and MV variant, this results in a 16x4=64 feature vectors per coding unit. Only one of these feature vectors is labeled as positive and the rest are labeled as negative. Let $FV_{r_i,v_j}^{(n)}$ to denote the feature vector of the nth coding unit with reference index i, and motion vector variant j, where i=1..16 and j=1..4. The feature matrix X for a video sequence is represented as:

$$X = \begin{bmatrix} FV_{r_1,v_1}^{(1)} \\ \dots \\ FV_{r_{16},v_4}^{(1)} \\ \dots \\ \dots \\ FV_{r_1,v_1}^{(N)} \\ \dots \\ FV_{r_{16},v_4}^{(N)} \end{bmatrix} (3)$$

The corresponding label vector can be represented as a vector of Boolean expressions, if the expression evaluates to true then the corresponding feature vector is labeled as positive and vice versa. The label vector is represented in Equation 4:

$$L = \begin{bmatrix} (r_1^{(1)} == true\_ref\_idx)\ and\ (v_1^{(1)} == true\_mv\_idx) \\ \dots \\ (r_{16}^{(1)} == true\_ref\_idx)\ and\ (v_4^{(1)} == true\_mv\_idx) \\ \dots \\ \dots \\ (r_1^{(N)} == true\_ref\_idx)\ and\ (v_1^{(N)} == true\_mv\_idx) \\ \dots \\ (r_{16}^{(N)} == true\_ref\_idx)\ and\ (v_4^{(N)} == true\_mv\_idx) \end{bmatrix}$$
(4)

In this work, we use a sequence-dependent approach to machine learning in which that first 10% of the data is used for training and the rest for testing. This is important as the generated model parameters are more relevant to the underlying video content in comparison to a sequence-independent classification solution where training is performed using other video sequences. In concept, the training can be repeated every K frames to accommodate for scene changes. For example, the first 10 frames of every 100 video frames can be used for training or retraining the model.

This classification arrangement results in very good accuracy as reported in details in the experimental results section. However, if the classification is not 100% accurate, then the decoder will reconstruct some CUs using the wrong reference pictures and/or wrong MVs. Because of the nature of motion compensation, such an error can propagate to future pictures and result in visible distortions.

To avoid such a deficiency, we repeat the same classification arrangement of the decoder at the encoder's side. This is doable as both the encoder and the decoder can use the first 10% of the data for model generation, thus generating the same model weights without having to communicate them. As such, the encoder can use the classification model to identify which coding units are incorrectly classified and exclude them from message embedding. Therefore, only coding units that can be correctly classified at the decoder are used for message embedding. This raises the question of how to indicate to the decoder that a coding unit is excluded from message embedding. One simple approach is to change the quantization parameter of such a coding unit by incrementing or decrementing it by one unit. By alternating between an offset of -1 and +1, the overall bitrate and PSNR of the video remains very similar. The decoder can then exclude such coding units from the message extraction process and perform an accurate reconstruction of all coding units. Clearly, by excluding coding units from message embedding, the embedding rate decreases slightly with the advantage of correct and accurate video reconstruction at the decoder. The details of such a solution and its effect on bitrate, PSNR and embedding rate is presented in details in the experimental results section.

Lastly, in this work we use Random Forests (RFs) as our machine-learning tool. In training and testing, 16 trees are grown. The RF classifier is one of the most used machine learning algorithms as it is known to produce excellent classification results even without hyper-parameter tuning.

VII. EXPERIMENTAL RESULTS

In the experimental results to follow, nine video sequences are used. The sequences has various spatial resolutions as listed in Table 3. The sequences are encoded using HEVC HM13.0 model [21], the total number of frames per sequence is 100, the first frame is intra coded and the rest of the frames are P-frames. Sixteen reference frames are used and the CU size is fixed to 16x16. The videos are coded with four quantization parameters of {22, 27, 32 and 37}. In HEVC video coding, it is recommended to assess proposed solutions using these 4 QPs which result in low, medium-low, medium-high and high bitrates. This reveals the suitability of the proposed solution on 4 different categories of video bitrates.

The message to be embedded is a sequence of ones and zeros randomly generated with a uniform distribution. The sequence of ones and zeros is a binary representation of any encrypted or clear message content to be embedded in coded videos.

The classification accuracies and F-scores of the proposed solution are listed in Table 4. The classification accuracy is calculated using Equation 5:

$$Accuracy = \frac{TP+TN}{TP+TN+FP+FN} * 100\% \quad (5)$$

Where TP and TN stand for true positive and true negative, and FP and FN stand for false positive and false negative. The F-score is computed using Equation 6 where Precision = TP/(TP+FP) and Recall = TP/(TP+FN):

$$FScore = \frac{2*Precision*Recall}{Precision+Recall} \quad (6)$$

TABLE 3
Video sequences used in the experimental results and their resolutions.

ID	Sequence	WxH	f/s
1	RaceHorses	416x240	30
2	BlowingBubbles	416x240	50
3	BasketballPass	416x240	50
4	RaceHorses	832x480	30
5	BQMall	832x480	60
6	BasketballDrill	832x480	50
7	ParkScene	1280x720	24
8	Kimono1	1280x720	24
9	Cactus	1280x720	50

TABLE 4
Classification accuracy and F-scores of the proposed solution

	Classification accuracy			
ID	QP22	QP27	QP32	QP37
1	98.5%	98.6%	98.7%	98.6%
2	91.7%	90.5%	89.8%	90.3%
3	94.8%	95.5%	96.7%	97.4%
4	99.1%	99.0%	98.8%	98.5%
5	95.5%	94.4%	94.4%	94.7%
6	82.3%	85.3%	90.4%	92.4%
7	97.3%	97.6%	97.3%	96.0%
8	99.0%	98.9%	98.5%	97.5%
9	94.0%	95.2%	95.0%	94.2%
Avg	**94.7%**	**95.0%**	**95.5%**	**95.5%**

(a) Classification accuracy

	F-score			
ID	QP22	QP27	QP32	QP37
1	98.5%	98.5%	98.7%	98.6%
2	89.8%	89.6%	89.4%	89.2%
3	96.1%	96.6%	97.0%	96.8%
4	99.1%	98.9%	98.7%	98.5%
5	95.9%	95.5%	95.2%	95.4%
6	83.7%	85.6%	89.7%	91.9%
7	97.0%	97.4%	97.0%	95.7%
8	99.0%	98.7%	98.0%	96.8%
9	94.3%	94.5%	94.5%	93.3%
Avg	**94.8%**	**95.0%**	**95.4%**	**95.1%**

(b) F-scores

The average classification accuracies and F-scores are around 95%. The average classification accuracies and F-scores vary across video sequence, however, there is no clear indication that they constantly increase or decrease as the QP varies. This gives an indication that the proposed solution is suitable for different QP levels.

On the other hand, the QP has a clear effect on the percentage of CUs carrying message bits as shown in Table 5. Higher QPs result in higher percentages of skipped CUs and therefore reduce the percentage of CU carrying message bits. The reported percentages of CUs carrying bits are 66.5%, 53.4%, 40.6% and 29.2% for QPs of 22, 27, 32 and 37 respectively. The percentage of CUs carrying message bits is also affected by the percentage of intra coded-CUs because such CU types do not carry MVs. Likewise inter-coded CUs with nil MVs carry less message bits as previously illustrated in Table 2. As a result, the average message bits per CU is 3.99, 3.2, 2.44 and 1.75 bits for QPs of 22, 27, 32 and 37 respectively.

TABLE 5
Average percentage of CUs carrying message bits and average number of message bits per CU.

	QP22		QP27	
ID	CUs with msg (%)	bits/CU	CUs with msg (%)	bits/CU
1	75.7%	4.54	70.1%	4.21
2	88.2%	5.29	72.2%	4.33
3	47.0%	2.82	39.7%	2.38
4	62.7%	3.76	63.1%	3.78
5	61.5%	3.69	46.1%	2.76
6	60.3%	3.62	43.8%	2.63
7	74.2%	4.45	48.5%	2.91
8	59.5%	3.57	50.3%	3.02
9	69.2%	4.15	46.6%	2.8
Avg	**66.5%**	**3.99**	**53.4%**	**3.20**

(a) QPs 22 and 27

	QP32		QP37	
ID	CUs with msg (%)	bits/CU	CUs with msg (%)	bits/CU
1	59.9%	3.6	47.8%	2.86
2	51.3%	3.08	33.6%	2.01
3	34.2%	2.05	28.6%	1.72
4	52.2%	3.13	36.7%	2.2
5	34.8%	2.09	25.2%	1.51
6	30.6%	1.84	22.1%	1.33
7	30.0%	1.8	16.6%	1
8	39.4%	2.37	27.4%	1.65
9	33.2%	1.99	24.5%	1.47
Avg	**40.6%**	**2.44**	**29.2%**	**1.75**

(b) QPs 32 and 37

When applying the CU marking solution at the encoder as proposed in Section 6, the CUs with MVs that cannot be correctly classified at the decoder are excluded from data embedding. This arrangement results in two consequences. First, all reference indices and MVs of CUs carrying message bits are correctly classified at the decoder and therefore the classification accuracy becomes 100%. Second, since some CUs are excluded from carrying message bits, the average number of message bits per CU is reduced as reported in Table 6. The reduction in message bits varies according to the original classification accuracy. For instance in Table 6, the classification accuracy of Seq. 4 is 98.9% and therefore the average message bits are reduced from 3.22/CU to 3.18/CU. Whereas in Seq. 6 the classification accuracy was 87.6% thus, the reduction in message bits is higher (from 2.36/CU to 1.9/CU).

TABLE 6
Average message bits per CU with and without CU marking at the encoder.

	Without CU marking		With CU marking	
ID	Avg. msg bits/CU	Class %	Avg. msg bits/CU	Class %
1	3.80	98.6%	3.75	100%
2	3.68	90.6%	3.29	100%
3	2.24	96.1%	2.15	100%
4	3.22	98.9%	3.18	100%
5	2.51	94.8%	2.38	100%
6	2.36	87.6%	1.90	100%
7	2.54	97.0%	2.47	100%
8	2.65	98.5%	2.58	100%
9	2.60	94.6%	2.40	100%
Avg	**2.84**	**95.2%**	**2.68**	**100%**

In terms of embedding rate, the average number of message bits per second depends on the number of message bits per CU, spatial and temporal resolutions of the video. The resolutions of the video sequences are reported in Table 3. With the proposed embedding algorithm that incorporates altering CU reference indices and MVs, the message embedding is on average 173Kbit/s as reported in Table 7.

TABLE 7
Average number of message bits per CU and corresponding average number of message bits per second averaged over four QPs

		Embedding bits	
ID	Frames / sec	Bits/CU	Kbit/s
1	30	3.75	42.8
2	50	3.29	62.6
3	50	2.15	41.0
4	30	3.18	145.2
5	60	2.38	217.9
6	50	1.90	144.8
7	24	2.47	222.1
8	24	2.58	232.0
9	50	2.40	450.0
Avg.		**2.68**	**173.16**

The effect of encryption and CU marking at the encoder's side in terms of PSNR and bit rate are examined in Tables 8 and 9. The difference in PSNR is computed as (*PSNR of original video – PSNR of encrypted video)*. Since encryption is carried out as a post process at the encoder then the locally decoded images at the encoder are unaffected. Rather, the effect of encryption will show at the decoder side. Therefore encryption does not have an effect on the PSNR of the video at the encoder's side, and consequently, the PSNR results in Table 8 are computed based on the proposed solution of marking the CUs at the encoder. The effect of this arrangement on the PSNR is reported in Table 8. As can be seen, since the change in QP alternates between -1 and +1, the effect on PSNR is negligible and can result in higher PSNR values which are represented with a negative sign in the table.

TABLE 8
PSNR drop at encoder because of marking CUs to avoid using them for message embedding.

	PSNR drop (at encoder) [dB]			
ID	QP22	QP27	QP32	QP37
1	0.006	-0.008	-0.018	-0.042
2	0.001	-0.016	-0.021	-0.031
3	-0.011	-0.008	-0.027	-0.007
4	0.006	0.000	-0.014	-0.037
5	-0.005	-0.012	-0.032	-0.044
6	-0.003	0.012	-0.031	-0.020
7	-0.013	-0.023	-0.045	-0.062
8	-0.006	-0.033	-0.076	-0.138
9	0.000	-0.006	-0.014	-0.034
Avg.	**-0.003**	**-0.010**	**-0.031**	**-0.046**

On the other hand, encrypting the video for message embedding has an effect on the bitrate of the video at the encoder's side. In fact, the encoder carries out a number of tasks including message embedding, video encryption and marking CUs. The effect all these tasks on the PSNR was presented in Table 8 and its effect on video bitrate is presented in Table 9. The excessive bitrate is computed as shown in Equation 7:

$$excessive\ bitrate = \frac{\#bits(encrypted\ video) - \#bits(originalVideo)}{\#bits(originalVideo)} * 100 \quad (7)$$

As reported in the table, the excessive bitrate is QP and sequence dependent. As the QP increases, the bitrate decreases and therefore the effect of altering the MVs and reference indices becomes more profound. Therefore, the average excessive bitrates per QP as reported in Table 9 are 1.29%, 2.09%, 3.31% and 5.06%. The average excessive bitrate resulting from the proposed encoding solution is also reported in the table in Kbit/s, which is on average 67.2 Kbit/s. In comparison to the average embedding rate of 173 Kbit/s reported in Table 7, the average excessive bit rate indicates the efficiency of the proposed solution.

The rate-distortion curves for all sequences are plotted in Figure 5. The BD-Rate [22] of the proposed solution is also reported to compare the rate-distortion performance of the proposed solution versus normal HEVC coding.

The corresponding BD-Rates for the nine sequences are: 3.57%, 1.88%, 2.97%, 3.46%, 2.98%, 3.0%, 3.3%, 7.2% and 1.8% respectively. The average BD-Rate over all sequences is 3.36%. The BD-Rate results indicate that regular HEVC coding saves 3.36% bits over the proposed solution for the same quality.

TABLE 9
Percentage bitrate increase at encoder because of encryption and marking CUs.

	Excessive bitrate (at encoder) [%]				Avg. excessive bits
ID	QP22	QP27	QP32	QP37	Kbit/s
1	1.3	2.3	4.0	7.5	32.3
2	0.4	1.1	2.1	2.4	15.2
3	1.1	2.2	3.0	5.2	20.6
4	1.3	2.0	4.1	7.1	119.8
5	0.8	2.1	2.7	4.5	88.4
6	1.0	2.1	3.4	5.9	60.8
7	1.2	1.3	2.8	4.3	55.7
8	3.9	4.9	5.5	5.6	113.5
9	0.4	0.9	2.3	3.0	98.0
Avg.	**1.3**	**2.1**	**3.3**	**5.1**	**67.2**

In this work, the fact the data embedding is performed by altering syntax elements (MVs and picture reference indices) as a post process resulted in embedding rates higher than the reported excessive bitrates. If the alteration is not implemented as a post process then a modified MV will result in high prediction error in the motion compensation process as it is a modified version of the best MV selected by the motion estimation algorithm. However, this is not the case as the altered MVs in this work are not used in the motion compensation process. This results in a scrambled video at the decoder side, and the unscrambling can be performed using machine learning by authorized decoders to find the values of the unaltered MVs (the same argument applies to the picture reference indices). In other words, the message bits are implied at the decoder by examining the differences between the received and the classified MVs, hence the high embedding capacity. Additionally, in video coding, MVs are compressed by subtracting them from previously encoded motion vectors, this results in motion vector differences which go through variable length encoding. Neighboring MVs are known to be correlated and therefore altering the values of MVs results in higher MV differences and thus a higher number of coding bits. This is manifested in the reported excessive bitrate.

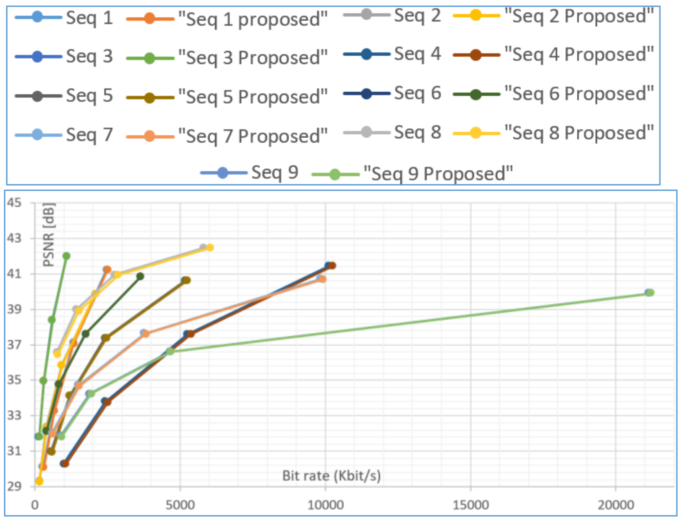

Figure 5. BD-Rate curves for proposed solution versus HEVC coding.

The proposed solution generates promising results in terms of the embedding capacity and the excessive overhead at the encoder's side. However, these advantages are achieved at the expense of increasing the computational time required for model generation. This is because the proposed solution has 64 different alteration options per coding unit as explained previously, therefore, each coding unit has 64 features vectors which results in an increased model generation time as reported in Table 10. The table also lists the classification time at the decoder's side which includes classifying 64 combinations of reference frames and motion vectors. The results are computed on a laptop with Intel Core i7-37400QM, 2.7-GHz CPU with a 16-GB DDR3 RAM.

It is shown in the table that the average model generation time varies according to the spatial resolution of the video. It is also shown that within the same spatial resolution, the time varies as the percentage of coding units used for message embedding change from one sequence to the other. For instance, in Table 2 above, the percentage of CUs with message bits for sequences 1, 2 and 3 are around 63%, 61% and 37% respectively. Therefore sequence 3 has the lowest model generation time. At the decoder side, the 64 combinations of picture references and motion vectors are classified to find the correct values used at the encoder. The time required for this arrangement is also reported in table. Depending on the spatial resolution of the video, the average reported times are 6.1s, 23.8s and 60.2s.

TABLE 10
Model generation time and classification time at decoder

Seq. ID	WxH	Model generation time (sec)	Decoder classification time (sec)
1	416x240	9.2	7.7
2	416x240	7.7	6.5
3	416x240	4.5	4.0
Avg.		**7.1s**	**6.1s**
4	832x480	46.4	37.0
5	832x480	25.3	20.4
6	832x480	16.6	14.1
Avg.		**29.4s**	**23.8s**
7	1280x720	93.7	65.3
8	1280x720	105.1	70.7
9	1280x720	63.8	44.6
Avg.		**87.5s**	**60.2**
Overall Avg.		**41.3s**	**30.0s**

Note that since message extraction need not be performed in real-time, therefore we do not claim that the proposed solution is suitable for real-time applications.

In terms of comparison with existing work, the authors in [16] and [17] reported a message embedding solution in HEVC encrypted videos. In both solutions, bits are embedded in HEVC quantized transform coefficients. In [16], the reported embedding rate is 3.3 Kbit/s and the reported increase in bitrate the encoder side is 7.8%. Provided that periodic I-frames are used, a negligible drop in PSNR was reported. In [17], the reported message bits in HEVC videos is 4.86 Kbit/s without any increase in bitrate. The comparison with the proposed work is summarized in Table 11. It should be noted that the work in [16] and [17] have low complexity as the encryption and embedding is implemented using a procedure that uses the modulus operator and simple arithmetic/logic operations, likewise the extraction is implemented using simple conditional statements.

TABLE 11
Comparison with existing solutions in terms of message bits/s and excessive bitrates.

	Proposed	Ref.[16]	Ref.[17]
Kilo Bits per second	173	3.3	4.86
Excessive bitrate	2.94%	7.8%	0%

In all cases, our proposed work has high embedding capacity, results in accurate video reconstruction at the decoder's side and does not require periodic I-frames. All of which are achieved at an expense of 3% excessive bitrate at the encoder's side.

TABLE 12
Comparison with existing solutions in terms of message bits per coding unit and drop in PSNR.

	Bits/coding unit	PSNR drop[dB]
Proposed	2.68	0
Ref. [5]	0.75	0.51
Ref. [14]	1.64	0.47
Ref. [13]	0.05	0.2
Ref. [23]	0.003	0.08

Additionally, Table 12, summarizes the message embedding rates and drop in PSNR as reported in [5], [13], [14] and [22]. All these solutions embed data in encrypted videos and were summarized in the introduction. The work in [23] encrypts the sign bits of nonzero quantized coefficients and motion vector differences. Consequently, data hiding is performed in the encrypted domain by modifying absolute level of quantized DCT coefficients. As listed in the table, the proposed solution has a clear advantage in terms on embedding capacity (as detailed in Section III and the experimental results presented in Tables 5,6 and 7) and at the same time it results in accurate reconstruction of the video without loss in PSNR at the decoder as detailed in Section VI.

VIII. CONCLUSION

This paper proposed a message embedding solution in encrypted HEVC videos. An encrypted video remained compliant with standardized HEVC decoders. The message embedding is achieved by altering the values of reference picture indices and motion vectors of CUs. When altering the values of motion vectors and reference indices, it was shown that a maximum of six message bits can be embedded per CU. Motion vectors were altered by swapping their Vx and Vy components and/or changing their signs. Message bits were extracted using an authorized decoder by generating a classification model and using it for predicting the true values of the reference indices and motion vectors. Consequently, message bits were extracted correctly whilst correctly reconstructing the video to its unscrambled state. Coding units that resulted in misclassification were identified at the encoder and excluded from message embedding. Such an arrangement resulted in lower embedding rates and ensured accurate reconstruction of the video. In the experimental results, nine video sequences are coded using a HEVC encoder with four different QPs. It was shown that the average message embedding rate is 2.7 bits per CU which corresponds to 173 Kbit/s. This is achieved at the expense of increasing the bitrate of the encoder by 3% in addition to the time required to generate the classification model per video sequence. Comparison with existing work revealed that the proposed solution is superior in terms of embedding capacity whilst reducing the excessive bitrate of the encoder with the extra requirement of model generation time.

REFERENCES

[1] T. Stütz and A. Uhl, "A survey of H.264 AVC/SVC encryption," IEEE Transactions on Circuits Systems and Video Technology, 22(3), pp. 325_339, March 2012.

[2] Y. Tew, K. Wong, R.CW. Phan, et al., "Separable authentication in encrypted HEVC video," Multimedia Tools and Applications, 77, 24165–24184, 2018.

[3] M. Farajallah, W. Hamidouche, O. Déforges and S. E. Assad, "ROI encryption for the HEVC coded video contents," 2015 IEEE International Conference on Image Processing (ICIP), 2015, pp. 3096-3100.

[4] M. A. Taha, N. Sidaty, W. Hamidouche, O. Dforges, J. Vanne and M. Viitanen, "End-to-End Real-Time ROI-Based Encryption in HEVC Videos," 2018 26th European Signal Processing Conference (EUSIPCO), 2018, pp. 171-175

[5] P. Su, T. Tsai and Y. Chien, "Partial frame content scrambling in H.264/AVC by information hiding," Multimedia Tools and Applications, 76(5), pp. 7473–7496, March 2017.

[6] D. Xu, S. Su, "Reversible data hiding in encrypted images with separability and high embedding capacity," Signal Processing Image Communication, vol. 95, 2021

[7] "NIST, Advanced Encryption Standard (AES), Federal Information Processing Standard 197", Nov. 2001.

[8] Y. Wang, M. O'Neill, and F. Kurugollu, "A tunable encryption scheme and analysis of fast selective encryption for CAVLC and CABAC in H.264/AVC," IEEE Trans. Circuits Syst. Video Technol., vol. 23, no. 9, pp. 1476_1490, Sep. 2013.

[9] B. Boyadjis, C. Bergeron, B. Pesquet-Popescu, and F. Dufaux, "Extended selective encryption of H. 264/AVC (CABAC)-and HEVC-encoded video streams," IEEE Transactions on Circuits Systems and Video Technology, 27(4), pp. 892_906, April 2017.

[10] Y. Wang, M. O'Neill and F. Kurugollu, "The improved sign bit encryption of motion vectors for H.264/AVC," Proceedings of the 20th European Signal Processing Conference (EUSIPCO), Romania, pp. 1752-1756, 2012.

[11] Y. Yao, W. Zhang, and N. Yu, "Inter-frame distortion drift analysis for reversible data hiding in encrypted H.264/AVC video bitstreams," Signal Process., vol. 128, pp. 531_545, November 2016.

[12] M. Long, F. Peng, and H. Y. Li, "Separable reversible data hiding and encryption for HEVC video," J. Real-Time Image Process., 14(1), pp. 171_182, January 2018.

[13] D. Xu, R. Wang, Y. Zhu, "Tunable data hiding in partially encrypted H.264/AVC videos," Journal of Visual Communication and Image Representation, vol. 45, 2017.

[14] D. Xu, R. Wang, Y. Shi, "An improved scheme for data hiding in encrypted H.264/AVC videos," Journal of Visual Communication and Image Representation, vol. 36, 2016.

[15] Z. Shahid and W. Puech, "Visual protection of HEVC video by selective encryption of CABAC binstrings", IEEE Trans. Multimedia, 16(1), pp. 24-36, January 2014.

[16] D. Xu, "Commutative encryption and data hiding in HEVC video compression," IEEE Access, vol. 7, 2019.

[17] B. Guan, D. Xu and Q. Li, "An Efficient Commutative Encryption and Data hiding Scheme for HEVC Video," IEEE Access, April 2020.

[18] ISO/IEC 23008-2:2013: Information technology—high efficiency coding and media delivery in heterogeneous environments—Part 2: High efficiency video coding (2013)

[19] E. Peixoto, T. Shanableh and E. Izquierdo, "H.264/AVC to HEVC Video Transcoder based on Dynamic Thresholding and Content Modeling, " IEEE Transactions on Circuits and Systems for Video Technology, 24(1), January 2014

[20] T. Amestoy, A. Mercat, W. Hamidouche, D. Menard and C. Bergeron, "Tunable VVC Frame Partitioning Based on Lightweight Machine Learning," in IEEE Transactions on Image Processing, vol. 29, pp. 1313-1328, 2020.

[21] I.-K. Kim, K. D. McCann, K. Sugimoto, B. Bross, W.-J. Han and G. J. Sullivan, "High Efficiency Video Coding (HEVC) Test Model 13 (HM13) Encoder Description," JCT-VC O1002, 15th meeting of Joint Collaborative Team on Video Coding of ITU-T SG 16 WP 3 and ISO/IEC JTC 1, November 2013.

[22] G. Bjentegaard, "Calculation of average PSNR differences between RD-curves (VCEG-M33)", *VCEG Meeting (ITU-T SG16 Q.6)*, Apr. 2-4 2001

[23] D. Xu, "Data hiding in partially encrypted HEVC video," ETRI Journal, 42(3), March, 2020.

Tamer Shanableh a senior member of the IEEE and a professional engineer. Was born in Scotland, UK in 1972. He studied at the University of Essex where he received his Ph.D. in electronic systems engineering in 2002 and his MSc in software engineering in 1998.

He then worked as a senior research officer at the University of Essex for three years, during which, he collaborated with BTexact on inventing video transcoders. He then joined Motorola UK Research Labs and 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. During the summer breaks, Dr. Shanableh worked as a visiting professor at Motorola Labs in five different years. He spent his sabbatical leave as a visiting academic at the Multimedia and Computer Vision and Lab at Queen Mary, University of London, U.K. Dr. Shanableh authored more than 80 publications and has six patents. His research interests include digital video processing and pattern recognition.