Start Code-Based Encryption and Decryption Framework for HEVC

MIN KU LEE\textsuperscript{a} AND EUYE SEON JANG\textsuperscript{a}, (Senior Member, IEEE)

Department of Computer Science, Hanyang University, Seoul 04763, South Korea

Corresponding author: Eu Ye Seon Jang (esjang@hanyang.ac.kr)

This work was supported by the Research Fund of the Signal Intelligence Research Center supervised by the Defense Acquisition Program Administration and Agency for Defense Development of Korea.

ABSTRACT In this article, we propose a new selective encryption and decryption framework based on the start code for high efficiency video coding. There is a growing need to encrypt video information to protect video content from privacy invasion and intellectual property infringement caused by information leakage. Although encrypting an entire video is a straightforward approach, the cost to encrypting a large amount of video data is substantial, considering the resulting computational complexity. Consequently, selective encryption algorithms have been actively researched in recent years and have contributed to reducing the computational complexity. However, existing selective encryption algorithms have certain drawbacks. For instance, it is difficult to separate the video encryption algorithm from the video compression algorithm because the encryption framework is based on the syntax elements. Further, a partial reconstruction of the encrypted video bitstream is often unavoidable. To solve these problems, the proposed method encrypts the bitstream based on the start code rather than on the syntax elements. Encrypting the bitstream partially, based on the start code, makes it easy to separate the video encryption algorithm from the video compression algorithm. Furthermore, encrypting the part adjacent to the start code protects the video content, as video reconstruction using a video decoder is impossible, unless the correct start code is returned to the bitstream.

The experimental results show that the proposed method reduced the encryption and decryption times by approximately 97% and 98%, respectively, compared to the encryption and decryption of the entire video bitstream.

INDEX TERMS High efficiency video coding, selective encryption algorithms, start code, video encryption.

I. INTRODUCTION

With the increasing usage of video applications such as video on demand, video conferencing, and video surveillance, it has become very important to protect video content from unauthorized access and usage. Hence, there is a dire need for video content encryption, ranging from commercial videos to home security camera recordings, to protect video data from unauthorized access that could lead to leaked or hijacked videos, and consequently, illegal usage and privacy exposure.

Most video encryption algorithms can be classified into two types: naïve encryption algorithms (NEAs) and selective encryption algorithms (SEAs). A NEA encrypts an entire bitstream of video content using standard encryption algorithms, such as the data encryption standard (DES) \cite{1}, Rivest, Shamir, and Adleman (RSA) method \cite{2}, or advanced encryption standard (AES) \cite{3}. Because it encrypts an entire video bitstream, a NEA provides the best security when confidentiality is the top priority. Further, it is easy to implement when integrated with existing multimedia systems because it does not depend on video compression algorithms. However, it provides its functionality at the cost of a substantially increased computational complexity owing to the encryption of an entire large bitstream.

By contrast, SEAs reduce the computational complexity of encryption by encrypting only the highly sensitive portions of a video, thus overcoming the shortcomings of the NEA. The video bitstreams encrypted by SEAs result in a distorted visual quality when the video is decoded without the appropriate decryption, which degrades the information in the video and renders the video less comprehensible. The SEA might be more suitable for real-time applications owing to its low computational complexity compared to that of the NEA. However, the SEA remains vulnerable to partial decoding...
attempts, making them less desirable for highly sensitive video content demanding full protection. In addition, the SEA is, to a large extent, closely connected to video compression algorithms. It is often difficult to imbed a SEA within a video codec if it is impossible to implement the algorithms during the internal process of the video encoding and decoding (using either software or hardware). Because most standard video codecs are already implemented in hardware, the SEA is highly unlikely to be a viable solution.

In this article, we propose a novel video encryption algorithm that encrypts a portion of the bitstream adjacent to the start code. The proposed method achieves its protection by preventing an unauthorized decoder from decoding the bitstream unless the proper header information is restored. The computational complexity of the proposed method is comparable to that of the SEA because the bitstream is only partially encrypted. In particular, the proposed algorithm encrypts only the header information of the bitstream, which is typically one-byte long. In the worst-case scenario, 256 combinations would be required to restore the header information. The use of many start codes in the bitstream makes it virtually impossible to break the encryption in real time. In addition, the proposed method is easily integrated with existing multimedia systems, similar to the NEA, because it can be applied even after an entire bitstream has been generated.

The remainder of this article is organized as follows. In Section II, we describe the works related to NEAs and SEAs. Details of the proposed method are provided in Section III. In Section IV, we present the results of experiments conducted to evaluate the proposed scheme. Finally, we present the conclusions of the paper in Section V.

II. RELATED WORK

A. NAÏVE ENCRYPTION ALGORITHM

Naïve encryption entails fully encrypting the video content following compression using standard encryption algorithms such as the DES, RSA, or AES. The NEA processes a video bitstream as a stream of binary data and encodes every word in the bitstream, regardless of the type of video codec. The NEA is one of the most secure video encryption algorithms because it is applied to entire bitstreams using standard encryption algorithms. In general, however, NEA-based encryption is not recommended for use in real-time video transmission applications of large video data because it is computationally expensive.

For the NEA, there are many standard encryption algorithms such as the DES, RSA, and AES. In [4] and [5], following evaluation on video encryption, the AES was reportedly the fastest of these standard encryption algorithms. Furthermore, according to some studies, including [6], the AES reduces the computational complexity to an extent. However, the overall improvement is marginal because the overall complexity of the AES is determined by the size of the input data.

B. SELECTIVE ENCRYPTION ALGORITHM

The primary limitation of the NEA is that it encrypts the entire data; hence, the computation time is directly proportional to the amount of data. For example, applying the NEA to a high-capacity video bitstream, such as a typical two-hour movie that is stored and transmitted in gigabits following compression, presents the problem of computational complexity. Therefore, the SEAs have been studied as an alternative. A video bitstream consists of minimal data through which the original video is reconstructed by exploiting the redundancy of the original video data. Therefore, most (if not all) parts of a video bitstream are interdependent and the corruption of a small fraction of the bitstream may be sufficient to damage the entire bitstream, which could make it impossible to reconstruct the original video. The SEA exploits this characteristic by encrypting the video bitstream only partially. This approach enables it to protect the video data with much less computational complexity compared to that of the NEA.

Meyer and Gadegast [7] proposed a selective video encryption method called Secure MPEG (SEC MPEG). In addition, Maples and Spanos [8] proposed a selective video encryption method called Aegis. Both SEC MPEG and Aegis encrypt only the I-frame or keyframe information, which is critical for a decoder to decode normally. Furthermore, it was considered effective to encrypt only the I-frames using the standard encryption algorithm, the DES, because it made it difficult to properly reconstruct even P- and B-frames that were reconstructed with reference to the I-frame. However, the computational complexity was not significantly improved compared to that of the NEA because the I-frames in the video bitstream usually constitute between 30%–60% of the video size.

Tang [9] proposed the zig-zag permutation algorithm that reordered the transform coefficients in a zig-zag format after a discrete cosine transform (DCT) in the process of generating an I-frame in the video compression process. Because the zig-zag permutation algorithm rearranges the order of data in units of macroblocks constituting I-frames, the computational complexity is low. Experiments have shown that the computational complexity of the zig-zag permutation algorithm was only 1.56% of that of the NEA. Although the zig-zag permutation algorithm has a fast encryption speed, it is problematic in that it increases the size of the bitstream by approximately 50%.

Shi and Bharagava et al. [10] [11], Shi et al. [12] and proposed SEAs, such as the video encryption algorithm (VEA), modified VEA (MVEA), and real-time VEA (RVEA), which encrypt the sign bits of the DCT coefficient of the I-frame and the sign bits of the motion vector of the P- and B-frames. Because the sign bits of the DCT coefficients and motion vectors occupy only a small portion of the entire bitstream, the computational complexity was evaluated to be only 10%, compared to that of the NEA. However, these methods cannot guarantee full security because useful video information can be recovered by simply changing all the encrypted DC

VOLUME 8, 2020

202911
coefficients to 128 and all the encrypted AC coefficients to positive numbers.

Most SEAs designed for the latest video codec, i.e., high efficiency video coding (HEVC), encrypt the bitstream based on the syntax elements [13]–[21]. The syntax elements selected for encryption are encrypted during the compression process; hence, video encryption based on the syntax elements is closely coupled with the video compression algorithm.

Overall, the portion of the entire bitstream encrypted by SEAs is as small as possible and includes the I-frames, DCT coefficients of the I-frame, sign bits of the DCT coefficients of the I-frame, sign bits of the motion vectors of the P- and B-frames, and other syntax elements. This significantly improves the encryption speed compared with the NEA. However, the existing SEAs have critical drawbacks that should be resolved. The computational complexity is not significantly reduced compared to that of the NEA. The encrypted part causes the size of the video bitstream to increase. Because the encrypted part is easily recoverable, the level of security becomes more vulnerable. Moreover, in most SEAs, it is difficult to separate encryption algorithms from compression algorithms.

**III. PROPOSED METHOD**

The proposed method can be classified as a SEA-based method. However, it differs from many existing SEAs because it can be operated on top of a video codec. This concept is illustrated in Fig. 1. As shown in Fig. 1(a), conventional SEAs were designed to depend on the video codec using the video encryption algorithms and video decryption algorithms implemented in the video encoder and decoder, respectively. Thus, it is difficult to separate the video security algorithm from the video encoder and video decoder. In this regard, conventional SEAs imbedded in video compression algorithms are less viable because, to integrate the video encryption algorithms, they require the standardized video compression algorithms to be modified. Generally, standardized video codecs are implemented according to the standard specification, which makes it difficult, if not impossible, to modify a standard codec to embed additional encryption algorithms. Therefore, the proposed method is designed to be independent of the video codec and video security systems. This was achieved by applying the video encryption algorithm after video encoding and the video decryption algorithm before decoding, as shown in Fig. 1(b).

Unlike traditional SEAs that encrypt certain syntax elements in the entire video bitstream, the proposed method encrypts an important part of the bitstream that determines the decoding process and is adjacent to the start code in the video bitstream. Because the start code can be searched after or during compression, the proposed method can be incorporated into a video codec, as shown in Fig. 1(a).

As shown in Fig. 2(a), each top-level unit of the bitstream of the general standard video codecs (i.e., H.264/AVC, HEVC, and IVC) contains the start code as a prefix, which is a strong separator between the top-level units. For example, as shown in Fig. 2(b), the video bitstream output by the HEVC encoder can be separated into top-level units by the start code, as shown in Fig. 2(a). The start code of the HEVC bitstream is three-byte (i.e., 0 × 000001) long, as shown in Fig. 2(b). The start code pattern in the HEVC is designed to exploit the high unlikeliness of the arithmetic encoders to generate the same pattern as the start code. Thus, a parser attempting to locate the start code in a bitstream can quickly split the bitstream into top-level units, without having to parse every syntax element of the bitstream. As shown in Fig. 2, in a bitstream compressed using the standard video codec, the header is usually the sequence following the start code. As shown in Fig. 2(b), the first byte of the network abstraction layer (NAL) unit header consists of forbidden_zero, nal_unit_type, and nuh_layer_id. Because the one-bit values of forbidden_zero and nuh_layer_id are zeros, the first byte of the NAL unit header is determined by the six bits of nal_unit_type.

The value of the first byte of the NAL unit header by nal_unit_type in the six-bit range from zero to 63, is listed in Table 1. In addition, as shown in Fig. 2(b), the NAL unit is composed of the header and raw byte sequence payload (RBSP). The RBSP is defined in Table 1 by the nal_unit_type of the header preceding the RBSP. Based on the first byte of the NAL unit header, the RBSP can either be a non-VCL NAL unit payload, such as VPS, SPS, and PPS, or a VCL NAL unit payload, such as TRAIL_N, TRAIL_R,
TABLE 1. First byte value of NAL unit header in HEVC bitstream.

| Class | nal_unit_type | First Byte Value of NAL Unit Header | NAL Unit RBSP |
|-------|--------------|-----------------------------------|---------------|
| Video | 0            | 0x00                              | Trailing Non-reference (TRAIL_N) |
| Coding| 1            | 0x02                              | Trailing Reference (TRAIL_R) |
| Layer | 2            | 0x04                              | Temporal Sub-layer Access Non-reference (TSA_R) |
| (VCL) | 3            | 0x06                              | Temporal Sub-layer Access Reference (TSA_R) |
|       | 4            | 0x08                              | Stepwise Temporal Sub-layer Access Non-reference (STSA_N) |
|       | 5            | 0x0A                              | Stepwise Temporal Sub-layer Access Reference (STSA_R) |
|       | 6            | 0x0C                              | Random Access Decodable Leading Non-reference (RADL_N) |
|       | 7            | 0x0E                              | Random Access Decodable Leading Reference (RADL_R) |
|       | 8            | 0x10                              | Random Access Skipped Leading Non-reference (RASL_N) |
|       | 9            | 0x12                              | Random Access Skipped Leading Reference (RASL_R) |
|       | 10-15        | 0x14-0x1E                         | Reserved |
|       | 16           | 0x20                              | Broken Link Access with Leading Pictures (BLA_W_LP) |
|       | 17           | 0x22                              | Broken Link Access with RADL (BLA_W_RADL) |
|       | 18           | 0x24                              | Broken Link Access without Leading Pictures (BLA_N_LP) |
|       | 19           | 0x26                              | Instantaneous Decoding Refresh with RADL (IDR_W_RADL) |
|       | 20           | 0x28                              | Instantaneous Decoding Refresh without Leading Pictures (IDR_N_LP) |
|       | 21-22        | 0x2A-0x2B                         | Clean Random Access (CRA) |
|       | 22-31        | 0x2C-0x3E                         | Reserved |
| non-VCL| 32           | 0x40                              | Video Parameter Set (VPS) |
|       | 33           | 0x42                              | Sequence Parameter Set (SPS) |
|       | 34           | 0x44                              | Picture Parameter Set (PPS) |
|       | 35           | 0x46                              | Access Unit Delimiter (AUD) |
|       | 36           | 0x48                              | End of Sequence (EOS) |
|       | 37           | 0x4A                              | End of Bitstream (EOB) |
|       | 38           | 0x4C                              | Filler Data (FD_NU) |
|       | 39           | 0x4E                              | Prefix Supplemental Enhancement Information (PREFIX_SEI) |
|       | 40           | 0x50                              | Suffix Supplemental Enhancement Information (SUFFIX_SEI) |
|       | 41-47        | 0x52-0x5E                         | Reserved |
|       | 48-63        | 0x60-0x6E                         | Unspecified |

FIGURE 2. Bitstream structure of video codec: (a) general standard codec and (b) HEVC.

and TSA_N. Each RBSP in the non-VCL class contains basic video information such as the color sampling format, image width and height, and the initial quantization parameter. Each RBSP in the VCL class contains compressed video information by classifying coded pictures such as I-, P-, and B-frames into different coded slices according to the network layer. More information on the RBSP can be found in the HEVC standard [22]. Therefore, the first byte of the NAL unit header is first parsed during the decoding process. The remaining compressed video information is then extracted from the RBSP based on syntax matching with the NAL unit type. Failure to properly assign the first byte of the NAL unit
header makes it very difficult, if not impossible, to decode the video bitstream because how to decode the subsequent RBSP would be unclear. Furthermore, the first byte of the NAL unit header occupies a very small portion of the entire HEVC bitstream. Therefore, the proposed encryption of the first byte of the NAL unit header can effectively improve the encryption speed.

The proposed method encrypts a video bitstream by scrambling the values of the first bytes of the NAL unit header instead of using standard encryption algorithms such as the AES and DES. Most existing SEAs encrypt only a selected part of the bitstream by applying a standard encryption algorithm. Generally, however, standard encryption algorithms introduce computational complexity in terms of the number of word units because they encrypt data using various processes that use these units. The proposed method increases the encryption speed because it does not use a standard encryption algorithm.

The first byte of the NAL unit header can be one of 25 possibilities, excluding “Reserved” and “Unspecified,” which are not yet used in the HEVC standard, as evident from Table 1. The bitstream may contain as many as 25 headers, although overlapping is not permitted. Therefore, in the worst-case scenario, the number of cases required to reconstruct the original bitstream would be 25^25. Assuming a linear search is used, the average number of cases would be (25^25)/2. In addition, after the video decoding process, it can be confirmed whether each case will be recoverable.

Most importantly, in contrast to conventional SEAs, the proposed encryption of the first byte of the NAL unit header by scrambling ensures that decoding is impossible. Therefore, the proposed method can provide security superior to that of the existing SEAs.

A flow chart of the proposed encryption and decryption processes is presented in Fig. 3. As shown in Fig. 3(a), the first step in the encryption process is to generate an encryption lookup table (ELUT) to encrypt and decrypt the first byte of the NAL unit header. For encryption and decryption, the rows of the first and second columns of the ELUT have different values, as shown in Fig. 3. The next step in the encryption process is to read a word corresponding to the start code size from the bitstream and verify whether the current word is the start code (i.e., 0 × 000001). If the current word is the start code, the byte succeeding it in the bitstream is searched in the second column of the ELUT for replacement by the value in the first column corresponding to the second column of the ELUT. The decryption process is repeated according to the loop flow shown in Fig. 3(b) until all encrypted bytes following the start code in the encrypted bitstream are decrypted.

The computational complexity \( (T_{PE}) \) of the proposed encryption algorithm (PEA) 1 is formulated as follows:

\[
T_{PE} = C_A P_A + C_B P_B + C_C P_C + C_D P_D \tag{1}
\]

where \( C_A, C_B, C_C, \) and \( C_D \) indicate the complexity values that are terminated at Processes A, B, C, and D, respectively, and returned to the beginning of the loop in Fig. 3. In addition, \( P_A, P_B, P_C, \) and \( P_D \) are the probabilities corresponding to \( C_A, C_B, C_C, \) and \( C_D, \) respectively. The sum of \( P_A, P_B, P_C, \) and \( P_D \) is 1.

When the operation applied in each process is expressed as \( T_A, T_B, \) and \( T_C \), (1) can be approximated as follows:

\[
T_{PE} \approx T_{comp} P_A \tag{3}
\]

The proposed decryption performs the same operation as the proposed encryption shown in Fig. 4; thus, the computational complexity \( (T_{PD}) \) of the proposed decryption is formulated as follows:

\[
T_{PD} \approx T_{comp} P_A \tag{4}
\]

The computational complexity \( (T_{AES,PE}) \) of the AES encryption and that \( (T_{AES,PD}) \) of the AES decryption, which are commonly used in the NEA and SEA, as introduced in Section II, are formulated in [23] as follows:

\[
T_{AES,PE} = (46N_b R - 30N_b T_o) + [31N_b R + 12(R - 1)] - 20N_b T_o + [64N_b R + 96(R - 1) - 61N_b T_s] \tag{5}
\]

\[
T_{AES,PD} = T_{AES,PE} + (96N_b T_o + 72N_b T_o - 32N_b T_s) \times (R - 1) \tag{6}
\]

where \( N_b \) is the block size and \( R \) is the number of rounds. \( T_o, T_s, \) and \( T_{AES} \) indicate the operations of the byte-wise-AND, byte-wise-OR, and byte-wise shift, respectively. Generally, if \( N_b \) is 4 and \( R \) is 10 when the key length is the shortest at 128 bits, (5) expresses the computational complexity for encrypting 16-byte data, which is formulated as follows:

\[
T_{AES,PE} = 1720T_o + 1268T_o + 3180T_s \tag{7}
\]

Equation (6) is the computational complexity of decrypting 16-byte data, which is formulated as follows:

\[
T_{AES,PD} = 5176T_o + 3860T_o + 2028T_s \tag{8}
\]
M. K. Lee, E. S. Jang: Start Code-Based Encryption and Decryption Framework for HEVC

FIGURE 3. Flow chart of proposed method: (a) encryption and (b) decryption.

TABLE 2. Comparing the number of operations for NEA and PEA 1.

| Bitstream Size | PEA 1 Encryption | Decryption | NEA Encryption | Decryption |
|----------------|-------------------|------------|----------------|------------|
| N              | \((N \times P_1)T_{comp}\) | \((N \times P_2)T_{comp}\) | \((N/16) \times (1720T_e + 1268T_c + 3180T_f)\) | \((N/16) \times (5176T_e + 3860T_c + 2028T_f)\) |
| 16             | \((16 \times P_1)T_{comp}\) | \((16 \times P_2)T_{comp}\) | \(1 \times (1720T_e + 1268T_c + 3180T_f)\) | \(1 \times (5176T_e + 3860T_c + 2028T_f)\) |
| 16 \times 10^6 | \((16 \times 10^6 \times P_1)T_{comp}\) | \((16 \times 10^6 \times P_2)T_{comp}\) | \(10^6 \times (1720T_e + 1268T_c + 3180T_f)\) | \(10^6 \times (5176T_e + 3860T_c + 2028T_f)\) |

The comparison of the number of operations for PEA 1, calculated using (3) and (4), and those for NEA, calculated using (7) and (8), are shown in Table 2. Here, \(N\) is the size of a byte unit of the bitstream to be encrypted and decrypted. Based on the experimental results, the average value of \(P_1\) was 0.99. Because the NEA encrypts and decrypts the entire bitstream in units of 16 bytes using the AES, the number of operations for the NEA encryption can be obtained by multiplying (7) by \(N/16\). Thus, because PEA 1 requires fewer operations for encryption and decryption, it is expected to be significantly faster than the NEA, as evident from the results in Table 2.

In addition, we propose PEA 2, which improves on the security of PEA 1. The encryption space is small because PEA 1 scrambles one byte after the start code. Therefore, to improve on the security afforded by PEA 1, PEA 2 increases the encryption space by encrypting the 16 bytes after the start code using the AES.

IV. EXPERIMENTAL RESULTS
The performances of the proposed methods were evaluated in two experiments: verification of the encryption and decryption results and a comparison of the processing speed of the NEA and PEAs. The results of the first experiment are shown
in Figs. 4(a) and (c) and those of the second experiment are shown in Figs. 4(b) and (c). The original bitstreams used in both experiments were 146 conformance bitstreams [24] containing various NAL unit types compressed using various HEVC encoder options. In the HEVC decoder, the HEVC reference software HM-16.20 [25] and the VLC media player were used [26]. This media player is a free, open-source cross-platform multimedia player that can decode and play most multimedia files, including HEVC bitstreams.

V. VERIFICATION OF ENCRYPTION AND DECRYPTION RESULTS

The encryption was verified to determine whether decoding is possible when the bitstream encrypted by the proposed methods is decoded with the HEVC decoder, as shown in Fig. 4(a). The decoding of all encrypted bitstreams using HM-16.20 was terminated when a decoding error occurred. In this case, the size of the reconstructed video output was zero. For the case of the VLC media player, a pop-up indicating that the video could not be played appeared, and no frames were played when all encrypted bitstreams were decoded. The results showed that the bitstream encrypted using the existing SEAs (introduced in Section II) could be decoded, and the reconstructed video was encrypted with a distortion in the video quality. However, it was impossible to decode the bitstream encrypted using the proposed methods and no video information could be found. Therefore, the proposed methods were more secure than the existing SEAs.

A. COMPARING THE SPEED OF NEA AND PEAS

The encryption speed of the NEA and PEAs was compared by measuring the time taken to encrypt all original bitstreams using each algorithm, as illustrated in Figs. 4(b) and (c). The electronic code book (ECB) mode and the 128-bit key of AES were used to encrypt and decrypt the test bitstreams using the NEA. The ECB mode and the 128-bit key were the fastest options for the AES.

The results indicated that the decoded frames and the size of the decrypted bitstreams were completely consistent with those of the original bitstreams. A binary comparison was used to confirm that the videos reconstructed from the decrypted bitstream and the original bitstream were exactly the same. The videos that were reconstructed from all decrypted bitstreams that were decoded using the VLC media player played normally without a distortion in the video quality. The verification results with all the conformance bitstreams clearly demonstrated the efficacy of the proposed methods.
TABLE 3. Comparing the speed of NEA and PEA 1.

| File name            | File size (Bytes) | NEA (s)    | PEA 1 (s)   | PEA 1/NEA (%) |
|----------------------|------------------|------------|-------------|---------------|
| File name            | Encryption       | Decryption | Encryption  | Decryption    | Encryption    | Decryption    |
| AMP_A_Samsung_7      | 265,406          | 4046.4     | 10618.8     | 141.4         | 212.8         | 2.86          | 2.00          |
| CAINIT_A_SHARP_4     | 110,099          | 1963.8     | 4090.2      | 64            | 106           | 3.26          | 2.59          |
| DBLK_A_SONY_3        | 279,115          | 5005.4     | 10358.6     | 155           | 244.8         | 3.10          | 2.36          |
| INITQP_B_Main10_Sony_1| 406,866          | 7254       | 16254       | 218           | 311.4         | 3.01          | 1.92          |
| IPRED_A_docomo_2     | 335,377          | 5929       | 12825.3     | 174.6         | 259.6         | 2.94          | 2.02          |
| MVDLZERO_A_docomo_4  | 1,970,408        | 35860      | 73127.2     | 1136.4        | 1563.2        | 3.17          | 2.14          |
| NUT_A_ericsson_5     | 302,142          | 5389.6     | 11260.4     | 169.2         | 231.2         | 3.14          | 2.05          |
| SLIST_A_Sony_5       | 372,917          | 6980.8     | 13848.2     | 238.4         | 288.6         | 3.42          | 2.08          |
| STRUCT_A_Samsung_7   | 319,053          | 5702       | 13915.6     | 166           | 245.2         | 2.91          | 1.76          |
| TILES_A_Cisco_2      | 484,767          | 9786.8     | 18602.8     | 257.4         | 366.6         | 2.63          | 1.97          |
| TSC1_A_VIDYO_5       | 46,546           | 855.4      | 1732.6      | 113.4         | 124.2         | 13.26         | 7.17          |
| Total (146 conformance bitstreams) | 58,855,215      | 1086383.8  | 2231856.6   | 34060.4       | 48002.8       | 3.14          | 2.15          |

TABLE 4. Comparing the speed of NEA and PEA 2.

| File name            | File size (Bytes) | NEA (s)    | PEA 2 (s)   | PEA 2/NEA (%) |
|----------------------|------------------|------------|-------------|---------------|
| File name            | Encryption       | Decryption | Encryption  | Decryption    | Encryption    | Decryption    |
| AMP_A_Samsung_7      | 265,406          | 4046.4     | 10618.8     | 212.4         | 233.4         | 4.29          | 2.10          |
| CAINIT_A_SHARP_4     | 110,099          | 1963.8     | 4090.2      | 135.4         | 195           | 6.89          | 4.77          |
| DBLK_A_SONY_3        | 279,115          | 5005.4     | 10358.6     | 254           | 284.4         | 5.07          | 2.75          |
| INITQP_B_Main10_Sony_1| 406,866          | 7254       | 16254       | 400           | 415.2         | 5.51          | 2.55          |
| IPRED_A_docomo_2     | 335,377          | 5929       | 12825.3     | 285           | 346.8         | 4.81          | 2.70          |
| MVDLZERO_A_docomo_4  | 1,970,408        | 35860      | 73127.2     | 3120          | 5286          | 8.70          | 7.23          |
| NUT_A_ericsson_5     | 302,142          | 5389.6     | 11260.4     | 270           | 335.6         | 5.01          | 2.98          |
| SLIST_A_Sony_5       | 372,917          | 6980.8     | 13848.2     | 369.4         | 428           | 5.29          | 3.09          |
| STRUCT_A_Samsung_7   | 319,053          | 5702       | 13915.6     | 254           | 280.4         | 4.45          | 2.02          |
| TILES_A_Cisco_2      | 484,767          | 9786.8     | 18602.8     | 430           | 489           | 4.39          | 2.63          |
| TSC1_A_VIDYO_5       | 46,546           | 855.4      | 1732.6      | 113.4         | 124.2         | 13.26         | 7.17          |
| Total (146 conformance bitstreams) | 58,855,215      | 1086383.8  | 2231856.6   | 66286.2       | 83045         | 6.10          | 3.72          |

TABLE 5. Comparison of existing SEAs and PEAs.

| SEAs                  | Overhead relative to NEA (%) | Compliance with standardized video codecs | Decrease in compression efficiency |
|-----------------------|------------------------------|-------------------------------------------|-----------------------------------|
| SECMP [7]             | 30–60                        | Yes                                       | No                                |
| Zig-zag permutation [9]| 1.56                         | No                                        | Yes (approx. 50%)                 |
| RVEA [12]             | 10                           | No                                        | No                                |
| PEA 1                 | 3.14                         | Yes                                       | No                                |
| PEA 2                 | 6.10                         | Yes                                       | No                                |

decryption by PEA 2 are approximately 6% and 4% of those required by the NEA, respectively.

Table 5 presents a comparative analysis of PEAs and the existing SEAs [7], [9], [12] described in Section II. Although the zig-zag permutation algorithm [9] is the fastest, it is impractical because it increases the size of the bitstream by approximately 50% and is not compliant with the standardized video codecs. The existing SEAs for the HEVC [13]–[21] described in Section II are not compliant with the standardized video codecs, and thus they are not included in the comparison in Table 5. Non-compliance with the standardized video codec implies that the encryption algorithms modify the standardized video codec, which makes a standard decoder become not interoperable. Therefore, it is impractical to embed encryption algorithms by modifying standardized video decoding procedures. In this respect, PEA 1 was the fastest encryption method that was compliant with standard video codecs and did not decrease the compression efficiency. Although the encryption speed of PEA 2 was slightly less than that of PEA 1, it was more secure.

VI. CONCLUSION

In this article, we proposed a novel selective encryption and decryption framework based on the start code for HEVC. The proposed methods had several advantages compared to the existing SEAs. First, they were independent of the video codec because they encrypted the portion adjacent to the start code, which was generally included in the video bitstream. Furthermore, the video codec-independent encryption framework enabled compliance with the standard video codec, which was not supported by the existing SEAs that depend on a video codec. Second, the proposed methods disallowed access to any information from a video that could not be decoded. By contrast, existing SEAs partially enable information acquisition from the decoded video with a quality distortion. In addition, because they disallowed decoding,
the proposed methods offered higher security than the existing SEAs. Finally, the proposed methods are highly competitive in terms of the processing speed when compared to selected encryption algorithms. They are also compliant with a standard video codec, with no decrease in the compression efficiency. In this regard, the encryption and decryption framework of the proposed methods based on the start code was practical and valuable.

REFERENCES

[1] US Department of Commerce, FIPS, PUB. 46-3, (1999). Data Encryption Standard. Federal Information Processing Standards, National Bureau of Standards. [Online]. Available: http://csrc.nist.gov/publications/fips/fips46-3/fips46-3.pdf

[2] R. L. Rivest, A. Shamir, and L. Adleman, “A method for obtaining digital signatures and public-key cryptosystems,” Commun. ACM, vol. 21, no. 2, pp. 120–126, Feb. 1978.

[3] US Department of Commerce, FIPS, PUB. 197. (2001). Advanced Encryption Standard (AES), National Institute of Standards and Technology. [Online]. Available: http://csrc.nist.gov/publications/fips/fips-197/fips-197.pdf

[4] M. A. Saleh, N. M. Tahir, E. Hisham, and H. Hashim, “An analysis and comparison for popular video encryption algorithms,” in Proc. IEEE Symp. Comput. Appl. Ind. Electron. (ISCAIE), Langkawi, Malaysia, Apr. 2015, pp. 90–94.

[5] M. M. Ahmad and M. I. Abdullah, “Comparison of encryption algorithms for multimedia,” Rajiashri Univ. J. Sci. Eng., vol. 44, pp. 131–139, Nov. 2016.

[6] P. Deshmukh and V. Kolhe, “Modified AES based algorithm for MPEG video encryption,” in Proc. Int. Conf. Inf. Commun. Embedded Syst. (ICI-CES), Chennai, India, Feb. 2014, pp. 1–5.

[7] J. Meyer and F. Gade gast, “Security mechanisms for multimedia data with the example MPEG-1 video,” Project Description SECMPEG, Tech. Univ. Berlin, Berlin, Germany, 1995.

[8] G. A. Spanos and T. B. Maples, “Performance study of a selective encryption scheme for the security of networked, real-time video,” in Proc. 4th Int. Conf. Comput. Commun. Netw., Las Vegas, NV, USA, 1995, p. 210.

[9] L. Tang, “Methods for encrypting and decrypting MPEG video data efficiently,” in Proc. 4th ACM Int. Conf. Multimedia, Boston, MA, USA, 1996, pp. 219–229.

[10] C. Shi and B. Bhargava, “A fast MPEG video encryption algorithm,” in Proc. 6th ACM Int. Conf. Multimedia, New York, NY, USA, 1998, pp. 81–88.

[11] C. Shi and B. Bhargava, “An efficient MPEG video encryption algorithm,” in Proc. 17th IEEE Symp. Reliable Distrib. Syst., West Lafayette, IN, USA, Oct. 1998, pp. 381–386.

[12] C. Shi, S. Y. Wang, and B. Bhargava, “MPEG video encryption in real-time using secret key cryptography,” in Proc. Int. Conf. Parallel Distrib. Process. Techn. Appl., Las Vegas, NV, USA, 1999, pp. 2822–2828.

[13] G. Van Wallendael, A. Boho, J. De Cock, A. Munteanu, and R. V. D. Walle, “Encryption for high efficiency video coding with video adaptation capabilities,” IEEE Trans. Consum. Electron., vol. 59, no. 3, pp. 634–642, Aug. 2013.

[14] H. Hofbauer, A. Uhl, and A. Unterweger, “Transparent encryption for HEVC using bit-stream-based selective coefficient sign encryption,” in Proc. IEEE Int. Conf. Acoust., Speech Signal Process. (ICASSP), Florence, Italy, May 2014, pp. 2005–2009.

[15] Z. Shahid and W. Puech, “Visual protection of HEVC video by selective encryption of CABAC binstrings,” IEEE Trans. Multimedia, vol. 16, no. 1, pp. 24–36, Jan. 2014.

[16] Y. Tew, K. Minemura, and K. Wong, “HEVC selective encryption using transform skip signal and sign bin,” in Proc. Asia-Pacific Signal Inf. Process. Assoc. Annu. Summit Conf. (APSIPA), Honolulu, HI, USA, Dec. 2015, pp. 963–970.

[17] F. Peng, H.-Y. Li, and M. Long, “An effective selective encryption scheme for HEVC based on Rossler chaotic system,” in Proc. Int. Symp. Nonlinear Theor. Appl., Hong Kong, 2015, pp. 1–4.

[18] M. Yang, L. Zhuo, J. Zhang, and X. Li, “An efficient format compliant video encryption scheme for HEVC bitstream,” in Proc. IEEE Int. Conf. Prog. Informat. Comput. (PIC), Nanjing, China, Dec. 2015, pp. 374–378.

[19] V. A. Memos and K. E. Psannis, “Encryption algorithm for efficient transmission of HEVC media,” J. Real-Time Image Process., vol. 12, no. 2, pp. 473–482, Aug. 2016.

[20] A. I. Sallam, E.-S.-M. El-Rabaie, and O. S. Faragallah, “CABAC-based selective encryption for HEVC using RC6 in different operation modes,” Multimedia Tools Appl., vol. 77, no. 21, pp. 28395–28416, Nov. 2018.

[21] B. Boyadjis, C. Bergeron, B. Pesquet-Popescu, and F. Dufaux, “Extended selective encryption of H.264/AVC (CABAC)- and HEVC-encoded video streams,” IEEE Trans. Circuits Syst. Video Technol., vol. 27, no. 4, pp. 892–906, Apr. 2017.

[22] Series H: Audiovisual and Multimedia Systems, Infrastructure of Audiovisual Services—Coding of Moving Video, High Efficiency Video Coding, Standard ITU-T, Recommendation H.265, Dec. 2016.

[23] M. R. Doonan, K. M. Sunjiv Sojayaiah, and D. Bundho, “Energy consumption and computational analysis of rijndael-AES,” in Proc. 3rd IEEE/IFIP Int. Conf. Central Asia Internet, Tashkent, Uzbeksep, 2007, pp. 1–6.

[24] Series H: Audiovisual and Multimedia Systems, Infrastructure of Audiovisual Services—Coding of Moving Video, Conformance Specification for ITU-T H.265 High Efficiency Video Coding, Standard ITU-T, Recommendation H.265.1, Oct. 2018.

[25] HEVC Test Model (HM), Accessed: Mar. 8, 2019. [Online]. Available: https://hevc.hhi.fraunhofer.de/svn/svn_HEVCSoftware/tags/

[26] VLC Media Player. Accessed: Jul. 20, 2019. [Online]. Available: https://www.videolan.org/