applied sciences

MDPI

Article

Efficient Transcoding and Encryption for Live 360 CCTV System

Tuan Thanh Le, JongBeom Jeong and Eun-Seok Ryu *

Department of Computer Engineering, Gachon University, Seongnam 13120, Korea; tuanlt@gc.gachon.ac.kr (T.T.L.); uof4949@gc.gachon.ac.kr (J.J.)

* Correspondence: esryu@gachon.ac.kr; Tel.: +82-10-4893-2199

Received: 2 January 2019; Accepted: 18 February 2019; Published: 21 February 2019

Abstract: In recent years, the rapid development of surveillance information in closed-circuit television (CCTV) has become an indispensable element in security systems. Several CCTV systems designed for video compression and encryption need to improve for the best performance and different security levels. Specially, the advent of 360 video makes the CCTV promising for surveillance without any blind areas. Compared to current systems, 360 CCTV requires the large bandwidth with low latency to run smoothly. Therefore, to improve the system performance, it needs to be more robust to run smoothly. Video transmission and transcoding is an essential process in converting codecs, changing bitrates or resizing the resolution for 360 videos. High-performance transcoding is one of the key factors of real time CCTV stream. Additionally, the security of video streams from cameras to endpoints is also an important priority in CCTV research. In this paper, a real-time transcoding system designed with the ARIA block cipher encryption algorithm is presented. Experimental results show that the proposed method achieved approximately 200% speedup compared to libx265 FFmpeg in transcoding task, and it could handle multiple transcoding sessions simultaneously at high performance for both live 360 CCTV system and existing 2D/3D CCTV system.

Keywords: CCTV; 360 videos; live streaming; transcoding

1. Introduction

Nowadays, closed-circuit television (CCTV) is widely deployed in the video surveillance systems and video analysis is a key factor to provide intelligent services. Previously, analog CCTV systems were used to search for intruders in the pre-recorded video by human. Currently, digital CCTV can process automatically using video analytic technologies in a computer system. To meet the necessity of CCTV systems video analytic, improving quality of service (QoS) of video CCTV is one indispensable element. Especially, the existing 2D CCTV system cannot provide high-resolution such as Ultra-HD 4K, or has difficulty adopting new video codec such as High-Efficiency Video Coding (HEVC) [1]. The video converting method is necessary to adapt the various requirements of CCTV systems.

Compared to the H.264/MPEG (Moving Picture Experts Group)-4 Advanced Video Coding (AVC), the HEVC video encoding has achieved approximately twice the compression [2]. It was developed and standardized by a joint team of MPEG known as the Joint Collaborative Team on Video Coding (JCT-VC) [3]. H.264/AVC standard (ITU-T Rec. H.264 | ISO/IEC 14496-10) is reviewed in [4]. HEVC is also standardized as ITU-T H.265 | ISO/IEC 23008-2 (MPEG-H Part 2). HEVC has quickly been deployed in many video services due to its high-performance in compression. Because of a large amount of existing video content encoded with the H.264/AVC format, transcoding H.264/AVC bitstream to HEVC bitstream is very demanding. In addition, most current 2D/3D CCTV systems only support H.264/AVC, thus a transcoder would help upgrade these CCTV systems at a lowest cost. The computational complexity of HEVC coding is extremely high compared to the H.264 standard.

This makes it difficult to implement real-time high-quality HEVC encoder software in multimedia encoding and transcoding systems.

The emergence of 360 videos has brought new trends in video research. It promotes the change of the video experience of users. Moreover, it also opens up more opportunities for video processing area such as efficient encoding, high bandwidth transmission, high-resolution, real-time video interaction, etc. Due to above factors, 360 video technology is promising in changing the surveillance of CCTV systems in the near future. By using the 360 CCTV, the surveillance system will have many changes in monitoring methods as well as surveillance video analysis. However, 360 video streaming requires high bandwidth with low latency, so it is hardly compatible with existing systems. Therefore, a transcoder with the ability to change the bitrate, resolution, and codec is needed to meet the various demands of the terminals.

Encryption is an important factor to ensure user information is secure in security systems. There are many different methods to implement security features in CCTV systems. For example, the encryption can be performed at the Network Abstract Layer (NAL) unit level and affect all NAL units in whole bitstream file. The performance of the encoder is not significant as it will increase the complexity of the encoder and the transcoder. Therefore, in this paper, we propose a transcoding method for multi-core platforms. Moreover, the proposed system encrypts the Video Parameter Set (VPS), Picture Parameter Set (PPS) and Sequence Parameter Set (SPS) NAL units of the HEVC bitstream during transcoding process. The NAL encryption step occurs after NAL encoding step in a serial process. Based on Big O notation theory, the encryption process for each transcoding thread does not increase the complexity of the system. The experimental results show that the proposed system provided significant speed, corresponding to a bit rate for H.264 to HEVC real-time transcoding of 36.4 frames/s for 1080p with six simultaneous threads and 33.6 frames/s for 4K with two simultaneous threads. The conceptual architecture of the proposed system including various CCTV cameras is illustrated in Figure 1.

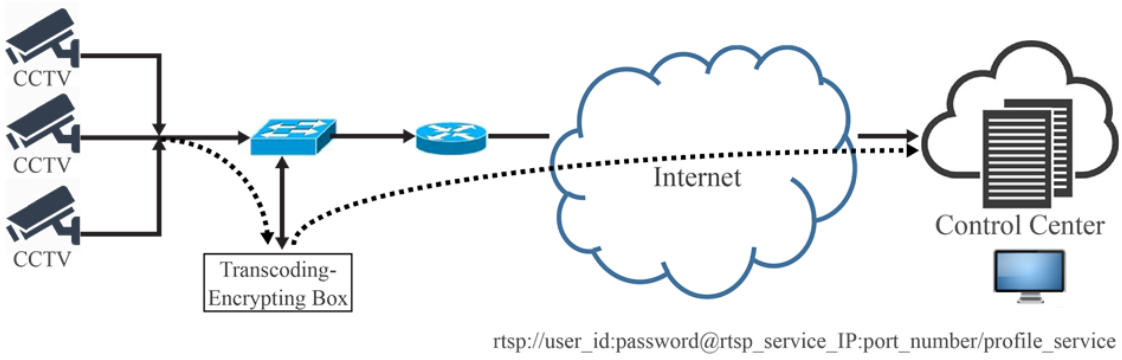


Figure 1. Conceptual architecture of live CCTV System with real-time transcoding.

The rest of the paper is organized as follows: Section 2 describes related works that were considered for proposed system. Section 3 addresses the challenges in real-time CCTV system with or without 360 cameras, and presents the proposed method. Section 4 shows the implemented demonstration and the performance evaluation. Finally, Section 5 presents conclusions about the proposed scheme and future work.

2. Related Work

2.1. Live CCTV System

The availability of CCTV camera poses challenges for massive video processing for transmissions and analysis. CCTV camera data are received from a variety of sources, including traffic intersections, retail shops, convenience stores, and traffic intersections. This condition has promoted the development of computer vision, AI and machine learning. The great point comes from the direct video system that

is extracting the value from the video to impact on the scientific, social and business fields. To meet the requirements of video analytics, CCTV systems need to be developed in real-time transmission, low cost of operation and the ability of accurate data analysis from CCTV live videos. To achieve the increasing demands of CCTV video data, an efficient solution is essential to handle the large bandwidth, real-time, quality of service and quality of experience requirements. Using one transcoder is one of most effective choices to propose reasonable solutions.

2.2. The 360 Video Standard in MPEG

In the 116th MPEG meeting, the MPEG-I group was established and organized to provide the support for immersive media. They began the support project for immersive media by releasing standards related to the format of immersive, and omnidirectional video in 2017 [5]. Figure 2 shows the standardization roadmap of MPEG. MPEG-I group divided the standardization into three phases [6] with some terminologies such as High Dynamic Range (HDR), Media Linking Application Format (MLAF), Common Media Application Format (CMAF) and Descriptors for Video Analysis (CDVA). One of them aims to provide 360 video contents including projection, video coding, and stitching.

The 360 video is mapped into a sphere and then mapped to the 2D plane for transmitting that is known as projection. The omnidirectional media format (OMAF) [7] addressed the standards for omnidirectional 360 video that includes projection types such as cubemap projection (CMP), equirectangular projection (ERP), adjusted cubemap projection (ACP), octahedron projection (OHP), rotated sphere projection (RSP), and segmented sphere projection (SSP).

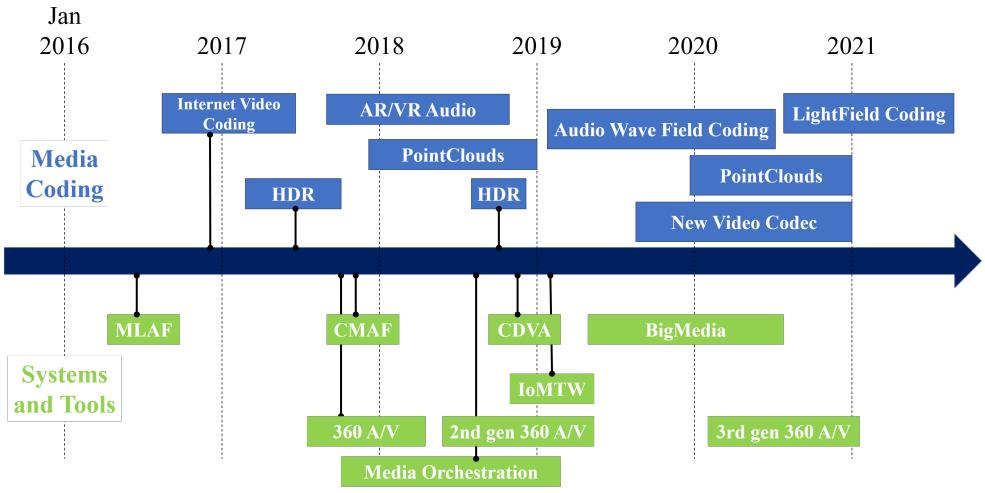
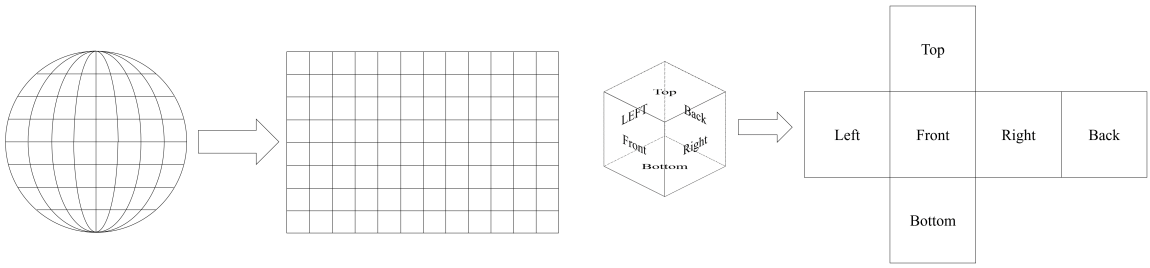


Figure 2. Moving Picture Experts Group (MPEG) standardization roadmap.

According to the definition of projection types, ERP projection is the most popular and widely used format for representing 360 videos. It makes the projection to the video on the sphere into the rectangular 2D plane, as shown in Figure 3a. As top and bottom pixels areas mapped in the sphere are overlapped during projection into 2D rectangular plane, there are some distortions in those areas.



(**a**) (**b**)

Figure 3. (**a**) Equirectangular projection; and (**b**) cubemap projection.

As shown in Figure 3b, the CMP projection is widely used nowadays for 360 video coding with six square faces. By inscribing the sphere with 360 videos in a regular hexahedron, each face is projected into a square. In some cases, it can reduce the size of the video compared to the ERP projection. However, there are some distortions on each edge of the squares. Because CMP projection samples the pixels non-uniformly, the video with CMP projection inefficiently represents the sphere.

As shown in Figure 4, the areas closer to the cube side edges are more densely sampled. Therefore, ACP projection was developed to overcome the explained problems. It can process approximately uniform sphere sampling while preserving the packing scheme.

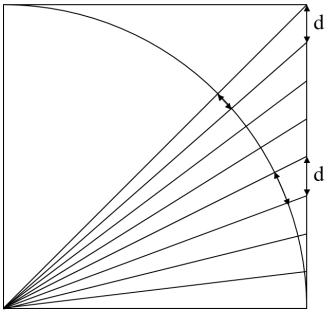


Figure 4. Cubemap sampling.

The function shown in Equation (1)

$$f(x) = sgn(x)\frac{0.34 - \sqrt{0.34^2 - 0.09\,|x|}}{0.18} \tag{1}$$

modifies the lookup vectors, which are used in 2D to 3D conversion. For 3D to 2D conversion, the inverse function of Equation (2) is used when modifying the lookup vectors:

$$g(x) = sgn(x)(-0.36x^2 + 1.36\,|x|) \tag{2}$$

Consequently, ACP projection enlarges the center of the cubemap in CMP, as shown in Figure 5. Figure 5b shows that ACP can make the center of the image become larger when compared to CMP, while CMP focuses on the overall image, as shown in Figure 5a.

(**a**) (**b**)

Figure 5. (**a**) Front face of 3 × 2 cubemap projection (CMP); and (**b**) front face of 3 × 2 adjusted cubemap projection (ACP).

2.3. Video Transcoding

Video transcoder studies are reviewed in [8,9]. Regarding transcoding, there are some concepts commonly used in multimedia tasks:

- Transcoding is the process at a high level of retrieving compressed (or encoded) content, the next is decompressing (or decoding) and then changing somehow to recompress it again. For example, we can change the audio/video format (codec) from MPEG-2 video source to H.264/AVC video and Advanced Audio Coding (AAC) audio. Other tasks might be adding various digital contents to a video stream.
- Trans-rating specifically focuses on changing the bitrate, such as taking the 4K Ultra-HD video input stream at 10 Mbps and converting it to one or more lower bitrates: HD at 4 Mbps, 2 Mbps, 1 Mbps, etc.
- Trans-sizing refers specifically to video frame change (can be understood as up-sampling, down-sampling). Normally, the video source often has high video frame size such as 4K (4096 × 2048), 8K, or even 12K. Thus, the trans-sizing usually is downgrading the resolution, for example downgrading from 4K ultra-high-definition (UHD) resolution (3840 × 2160) to 1080p (1920 × 1080) or 720p (1080 × 720).

Generally, transcoding is the combination or one or all of the above methods. The video conversion requires intensive computational power, so transcoding often requires acceleration capabilities of central processing units (CPUs) or graphics processing unit (GPU). Y. Chen et al. [10] proposed a solution for H.264/AVC to HEVC transcoding using parallel processing based on multicore platforms. Their proposed method speeds up the transcoding time to 5 frames/s for 720p and 1.5 frames/s for 1080p with the specified hardware platform. P.V. Luong et al. [11] proposed a method to reduce the transcoding complexity based on deriving an optimal strategy. They proposed their system for using various transcoding techniques to reduce the complexity in both prediction unit (PU) and coding unit (CU) at optimization levels. The authors achieved an effective approach that can reduce the complexity by around 82% while maintaining the bitrate loss below 3%. D. Honrubia et al. [12] proposed an adaptive fast quadtree level decision (AFQLD) algorithm. The main idea is AFQLD algorithm to exploit the information gathered at the H.264/AVC decoder in order to make faster decisions on CU splitting in HEVC using a Naïve-Bayes probabilistic classifier, which is determined by a supervised process of data mining. The AFQLD algorithm for the three levels achieves a quantitative speedup

of approximate 2.31 times on average with the time reduction of around 56.7%. In this paper, the proposed method uses an efficient library from Intel to implement transcoding task with advanced points. This library supports multiple threads for transcoding in various kinds and allows transcoding both video and audio streams.

2.4. Data Encryption for Surveillance System

In information security systems to secure users data, different encryption algorithms are widely used. Users can access and interact with video streaming systems anywhere on any device such as home, office, video-conferencing, etc. Therefore, user's information is easily accessible by intruders due to tje lack of security protocols. The necessity of safety communications becomes vital as the cost of data loss is increasing. In the field of information security, especially in video surveillance systems, data encryption is used to protect the data.

X. Zhang et al. [13] proposed lightweight encryption method based on automated cell subclassing to protect privacy in video surveillance. The authors extracted region of interests (RoIs) in the initial state of the eight-cell automata, which is controlled by randomly selected rules for the transition states of layered cellular automata (LCA). Therefore, all RoIs are encrypted synchronously and independently. Encrypted RoI is stored on the camera side and that can be interactively authenticated by the user through a required way. Video surveillance without RoI will be accessible in real time by the user. C. Wampler et al. [14] analyzed information leaking in video over IP traffic, including for encrypted payloads. The authors verified leakage by analyzing the metadata of network traffic including video bandwidth, packet size, and arrival time between packets. Event detection using metadata analysis is possible even if common encryption methods such as AES, and Secure Sockets Layer (SSL) are applied to video streaming. The leakage information has been observed on the camera and many codes. Through various experiments of the x264 codec [15], the basis for the ability to detect events through the package time was set. Based on this, event detection can be implemented on commercial streaming video applications. In this paper, the proposed method implements an encryption mechanism for specified Network Abstraction Layer (NAL) units based on ARIA cryptographic.

3. Fast Multiple Transcoding and Encryption

3.1. Multiple Transcoding

Instead of using a multiplexer to aggregate various video streams, the proposed method provides a parallel-processing mechanism for transcoding based on Intel Media Software Development Kit (Intel Media SDK), ARIA block cypher encryption and FFmpeg library. Additionally, Real-Time Streaming Protocol (RTSP) is used to handle input/output real-time streams from CCTV cameras to RTSP clients. As shown in Figure 6, each video transcoding session was handled by a CPU-core within one or more threads. One thread can process a number of video frames as a batching or all frames of a video stream. Figure 7 shows an example of parsing input streams, pipelining and multi-threading techniques handle the input streams as parallel processing. Each input stream could be attached to one session. Then, all sessions will be joined together to work simultaneously.

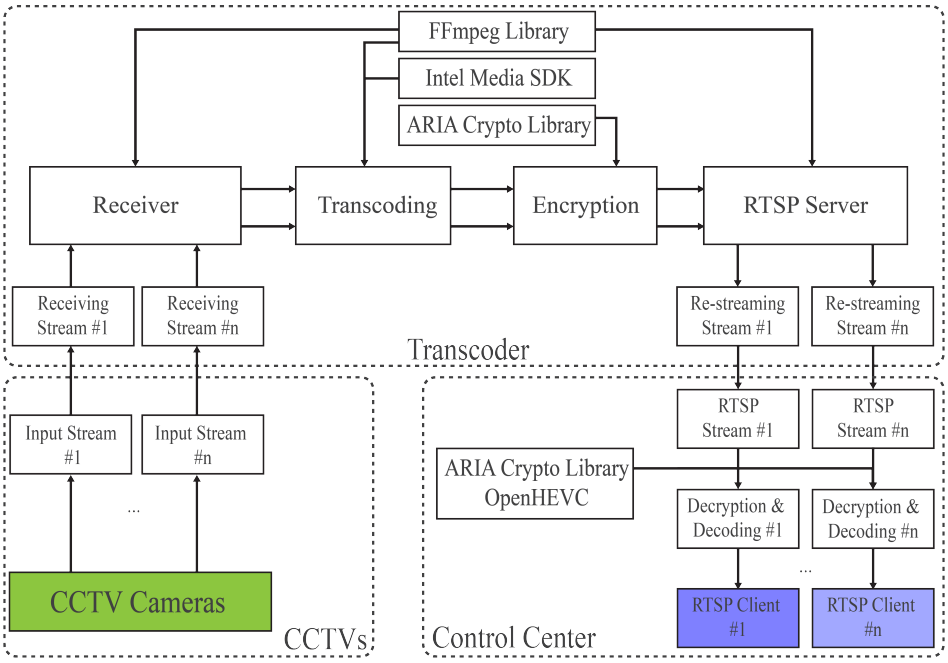


Figure 6. Multiple real-time streaming over transcoder box.

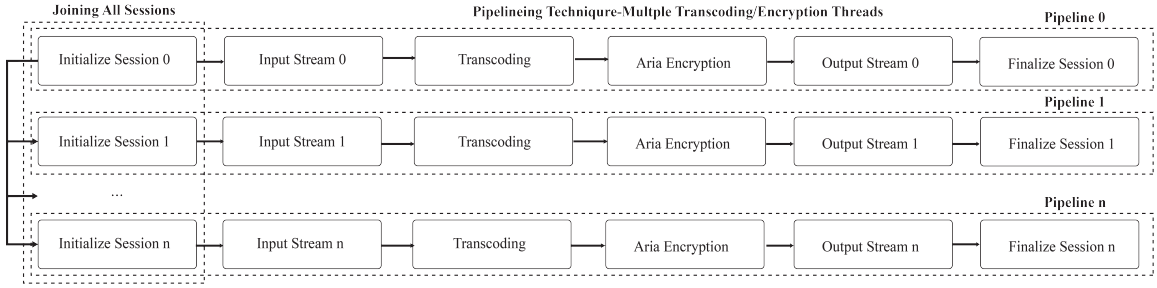


Figure 7. Transcoding and encryption for video streaming on multi-core platforms.

To handle multiple transcoding sessions, FFmpeg library software [16] can also provide a mixing mechanism to multiplex video streams. However, the outputting, re-encoding and decoding multiple times in the same FFmpeg process will slow down some encoders. A few video encoders such libx264 [15] perform the encoding as threading, so they allow for parallel processing. However, audio encoding may be serial and become the bottleneck issue. This leads to FFmpeg treating them as real serial-processing. Thus, FFmpeg may not use all available cores. Another available solution can be achieved by using multiple FFmpeg instances running in parallel or piping from one FFmpeg to another encoding session. All these reasons lead to FFmpeg not working effectively for parallel encoding.

Even though Intel Quick Sync Video technology [17] integrated into FFmpeg can handle multiple sessions, it can only support two sessions of 1080p video streams at the same time with high-latency for the second streaming. Therefore, the proposed method suggests directly using video and audio processing acceleration application programming interfaces (APIs) to improve the parallel processing for the encoders. As shown in Figure 7, to optimize CPU cores and graphics power, the proposed method uses Intel Media Software Development Kit (SDK) APIs to accelerate the video transcoding application as in parallel processing. The main architecture of Intel Media SDK is illustrated in Figure 8.

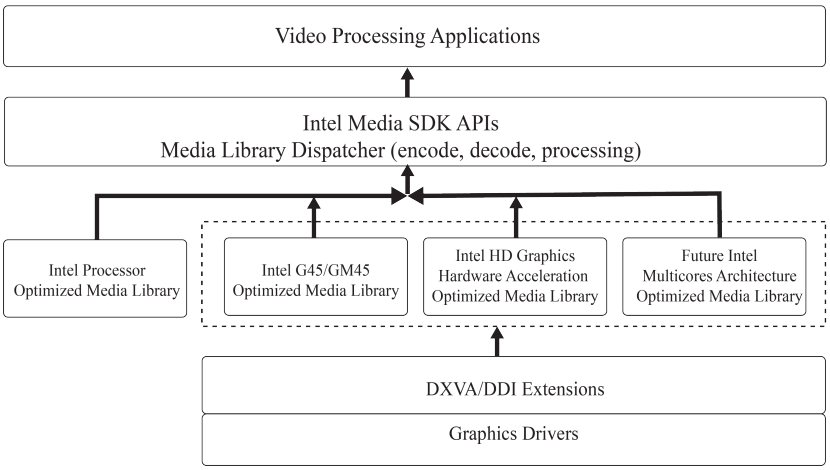


Figure 8. Intel Media SDK architecture.

To encrypt the video and audio simultaneously, we propose a model to transcode multiple audio and video sessions. The audio sessions are classified into two categories. First, audio data from video sessions are demuxed, and then the transcoder convert audio codec into AAC codec, as shown in Figure 9. After that, the muxer combines video and audio elementary stream into one output bitstream. Second, another audio transcoder is designed to adopt independent audio sessions based on importance in emergency situations. These audio microphones are set up near the CCTV camera to provide voice call. The audio transcoder allows converting multiple audio sessions simultaneously.

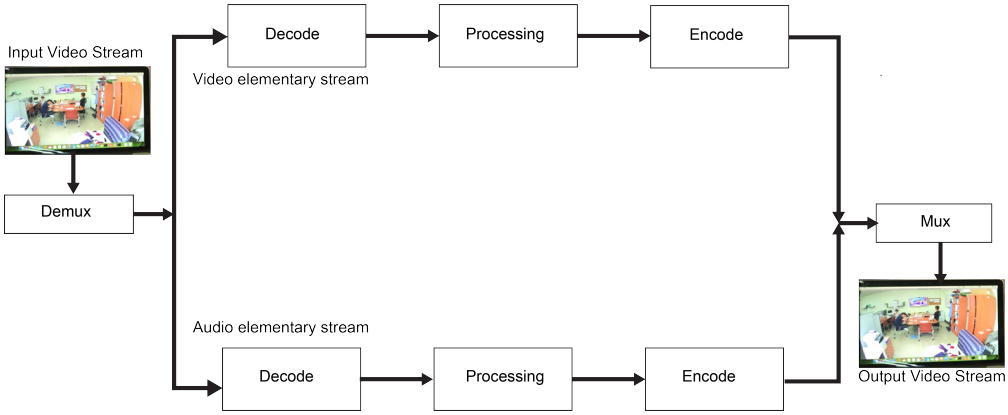


Figure 9. Video transcoding—elementary streams.

3.2. *ARIA Encryption*

To adapt secured video transmission over the Internet, an encryption algorithm is applied to encrypt the transcoded video bitstream. From the security point of view, various well known encryption algorithms can be used, such as Advanced Encryption Standard (AES), Digital Signature Algorithm (DSA), etc. Additionally, to provide a real solution for CCTV system specially in Korea, the ARIA encryption algorithm is used, which implemented the first version under the government's standard. The proposed system was considered using cryptographic and plaintext attacks on ARIA to check the secured capability. Moreover, some related studies were verified to estimate the performance of ARIA. Alex Biryukov et al. [18] provided the security and performance of ARIA encryption in specified report. S. Li et al. [19] showed some cryptanalysis analytics on ARIA. A. Pandey et al. [20] showed the comparative survey of different cryptographic algorithms, which included ARIA cryptographic algorithm.

ARIA algorithm is a block cipher designed by a large group of researchers from South Korea. Additionally, the Korean Agency for Technology and Standards voted and selected it as an efficient

standard cryptographic technique in 2004. More information about ARIA encryption can be reviewed in [21–23]. The ARIA algorithm provides a mechanism that uses a substitution-permutation network structure based on the well-known AES encryption. The interface is similar to AES encryption such as the key size of 128, 192, or 256 bits with 128 bits block size. Moreover, the number of rounds depends on the key size with possible values of 12, 14, or 16. The ARIA encryption algorithm uses two 8×8-bit S-boxes, and their inverses in alternate rounds; one of these boxes is the Rijndael S-box [21].

To achieve real-time transcoding/encryption, the proposed system uses ARIA to encrypt each output video stream from the transcoder, as shown in Figures 6 and 7. Instead of encrypting whole output stream, ARIA crypto library [24] was used to encrypt a few special frames of video streams. For example, to perform encryption while transcoding to HEVC, the proposed system only encrypts VPS, PPS and SPS slices. Additionally, the encryption mechanism also allows the transcoder encrypting a part of the video stream as the length of the multiple of 16 bytes. As shown in Figure 10, "encDataLen" is required as multiple of 16 bytes because of the "padding" issue. The padding issue leads to hard work of decryption in client-side or control center. In this paper, we propose the encryption for both video and audio data.

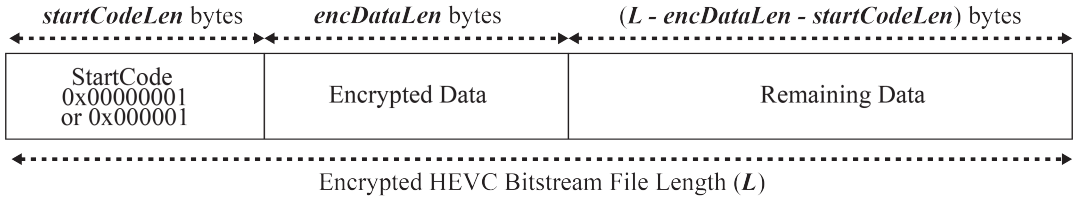


Figure 10. Output encrypted HEVC stream—Type 1.

The master key and round-key can be pre-generated and stored at both client and server sides. The other option is that all keys are attached to full name (RTSP) stream at the server side, as shown in Figure 11. Then, clients can interact with some particular points inside RTSP stream to detach master key and round-key. In this case, the location and length of these keys are stored in a secret process, which is only shared between server and client by another communication channel.

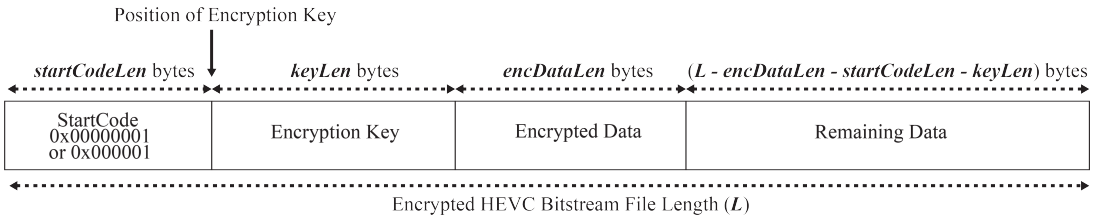


Figure 11. Output encrypted HEVC stream—Type 2.

3.3. RTSP Streaming

Real-time transmission protocol (RTSP) is a network control protocol designed for use in entertainment and communication systems to control streaming media servers. The protocol is used to establish and control communication sessions between endpoints. The transmission of data online is not the task of RTSP. Most RTSP servers use real-time transmission protocols in combination with real-time control protocols to distribute streaming media. However, some vendors implement proprietary transmission protocols. More information about RTSP is extensively explained in [25,26].

To provide live streaming services from CCTV camera to clients, the proposed method is a server–client model using RTSP protocol. This model consists of two part:

1. RTSP server side: Provides multiple real-time RTSP streams at the same time. Each RTSP stream is transcoded and encrypted by transcoder box.
2. RTSP client side: Customer's devices or control center uses computer, smartphone or tablet to decrypt/decode RTSP stream.

As shown in Figure 12, RTSP server functionality was merged into transcoding to avoid overhead of computational complexity of system while adding more functionalities. Client-side consists of two main functionalities: decryption and decoding RTP stream and rendering video. Regarding multiple live streaming sessions, an optimized RTSP based FFmpeg library was implemented to support multiple streaming simultaneously.

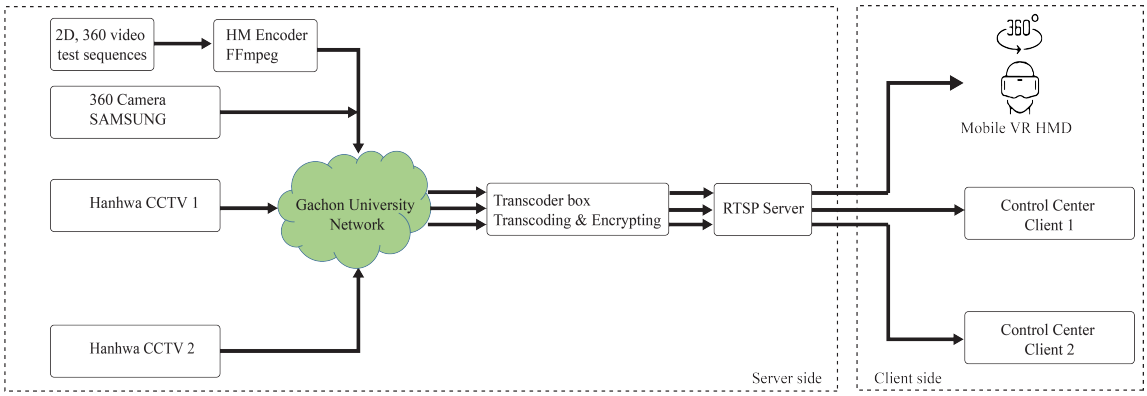


Figure 12. Testbed for live CCTV system.

4. Performance Evaluation

4.1. Testbed Scenario

To demonstrate the operation of the proposed method in a real environment, we implemented a testbed to verify the ability of real-time transcoding and encryption. For experimental measurements, we set up a testbed scenario, as shown in Figure 12. We set up a transcoder box with a Intel(R) Core(TM) i7-7500U 2.70GHz processor (Intel Corporation, Santa Clara, CA, USA), 16 GB of memory with Linux Centos 64-bits version 3.10.0-862.9.1.el7.x86_64 (gcc version 4.8.5 20150623 (Red Hat 4.8.5-28) (GCC)) OS. We also used mobile virtual reality-based bead-mounted-device, desktop, and laptop as clients in the control center. Additionally, we installed two Hanwha CCTV cameras [27] as 2D cameras and a Samsung 360 camera as 360 CCTV camera. Table 1 shows the standard 360 4K video test sequences in detail. These video test sequences were released by JCT-VC, as shown in Figure 13.

Table 1. The 360 videos bitstreams by JCT-VC.

Parameters	Value
InputFile	DrivingInCity 3840 × 1920_30fps_8bit_420_erp.yuv
Width	3840
Height	1920
InputFile	KiteFlite 4096 × 2048_30fps_8bit_420_erp.yuv
InputFile	Harbor 4096 × 2048_30fps_8bit_420_erp.yuv
InputFile	GasLamp 4096 × 2048_30fps_8bit_420_erp.yuv
InputFile	Trolley 4096 × 2048_30fps_8bit_420_erp.yuv
SourceWidth	4096
SourceHeight	2048
InputBitDepth	8
InputChromaFormat	420
FrameRate	30
FrameSkip	0
FramesToBeEncoded	300
Level	5.2

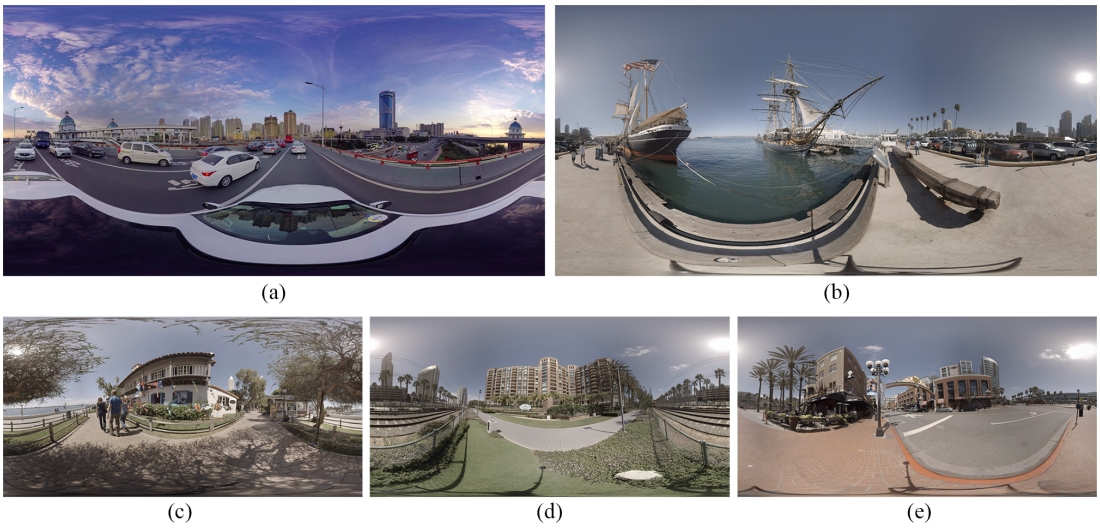


Figure 13. The 360 video test sequences: (**a**) DrivingInCity; (**b**) Harbor; (**c**) KiteFlite; (**d**) Trolley; and (**e**) GasLamp.

Regarding software, HM software with 360 libraries [24] was used as video encoder and openHEVC software [28] as video decoder. Intel Media SDK [29] was used to accelerate transcoding by multiple threads. Crypto++ crypto library [30] was used for bitstream encryption/decryption. The hevcbrowser software [31] was used to perform the status of HEVC bitstream. To evaluate performance, ERP Peak Signal-to-Noise Ratio (PSNR) software [32] and FFmpeg were used as estimation tools. Additionally, HM and 360 libraries were used to process a spherical 360 video with ERP projection [33]. The ERP projection is described in detail in Table A1. The encoding configuration parameters of 360 bitstreams are listed in Table A2. The 2D video test sequences were also encoded using HM version 16.15 within standard configurations for 8 bits.

To evaluate the quality of the output 360 videos, we calculated the distortion in the sphere to properly respond to our view of the 360 videos, instead of calculating the PSNR in rectangle plane. We used PSNR sphere (WS-PSNR) [33] weighted index method for calculating distortion of 360 videos reproduced in the sphere domain to show the difference between the reconstructed video and the original 360 video. The WS-PSNR software [32] provides WS-PSNR calculations for both ERP window and entire ERP system, as shown in Figure 14. To perform performance evaluations for the proposed method, we used a video test sequences as representation data for both 360 CCTV camera and the 2D CCTV camera.

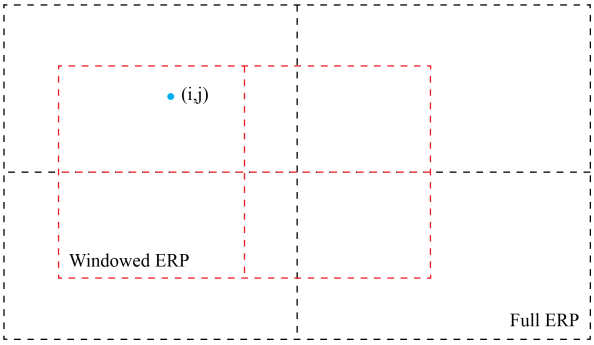


Figure 14. Full equirectangular projection (ERP) and windowed ERP.

4.2. Evaluation

As shown in Figure 15, the input video bitstream had a duration of 10.03 s, and the average transcoding time for the six sessions was approximately 7.05 s. In six transcoding sessions, there were three sessions using ARIA encryption and three sessions without encryption. In sessions with

encryption, transcoding was approximately equivalent to non-encryption sessions, and, in Session 1, the total time was only 7.02 s. For example, for larger videos, a concatenated video from DrivingInCity test sequence was 100.3 s, and transcoding and encryption averaged only 69.04 s, as shown in Table 2. Through the transcoding and encryption of different lengths of various 360 videos, we conclude that the transcoding and encryption system could process two 4K sessions or six 1080p sessions, and only took up around 70–75% of the video duration. The average transcoding rate was 36.4 frames/s for 1080p and 33.6 frames/s for 4K resolution video. Table 3 shows that the proposed method could also transcode 2D video in real-time, and the transcoding time was approximately 60% of 2D video duration.

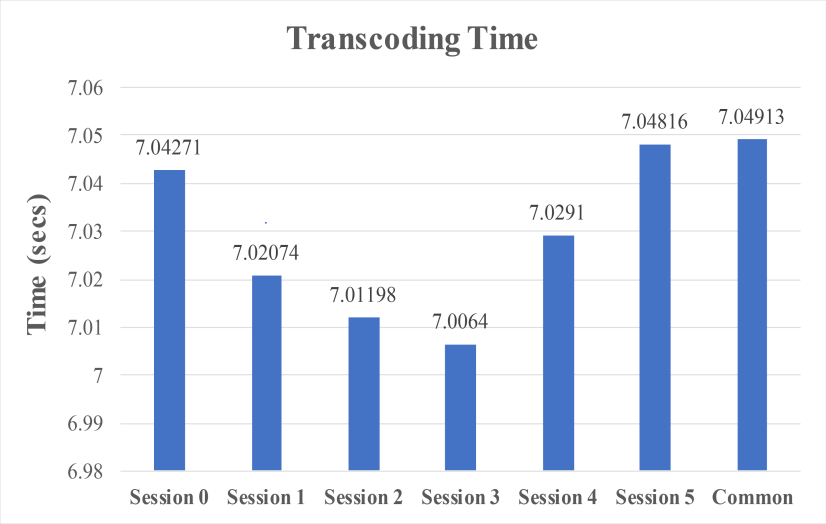


Figure 15. The DrivingInCity test sequence H.264/AVC to HEVC—six 1080p transcoding sessions.

Table 2. The 360 test sequences—common transcoding time.

Test_Sequence	Video Duration (s): Transcoding Time (s)	Video Duration (s): Transcoding Time (s)
DrivingInCity_1920 × 1080	10.03 (s) 7.04 (s)	100.3 (s):69.15 (s)
DrivingInCity_3840 × 1920	10.03 (s):8.92 (s)	100.3 (s):88.36 (s)
GasLamp_1920 × 1080	12.00 (s):8.45 (s)	120.0 (s):84.04 (s)
GasLamp_4096 × 2048	12.00 (s):9.07 (s)	120.0 (s):90.21(s)
Harbor_1920 × 1080	12.00 (s):8.58 (s)	120.0 (s):85.32 (s)
Harbor_4096 × 2048	12.00 (s):9.13 (s)	120.0 (s):90.54 (s)
KiteFlite_1920 × 1080	12.00 (s):8.66 (s)	120.0 (s):85.23 (s)
KiteFlite_4096 × 2048	12.00 (s):9.01 (s)	120.0 (s):90.29 (s)
Trolley_1920 × 1080	12.00 (s):8.64 (s)	120.0 (s):86.12 (s)
Trolley_4096 × 2048	12.00 (s):9.15 (s)	120.0 (s):90.76 (s)

Table 3. JCT-VC 2D test sequences—common transcoding time.

Test_Sequence	QP22	QP27	QP32	QP37
Basketball_1920 × 1080_500 frames_25fps	11.89 s	11.964 s	11.965 s	11.956 s
Cactus_1920 × 1080_500 frames_25fps	11.92 s	11.676 s	11.529 s	11.075 s
Kimono_1920 × 1080_240 frames_25fps	5.61 s	5.67 s	5.84 s	5.76 s
ParkScene_1920 × 1080_240 frames_25fps	5.596 s	5.643 s	5.678 s	5.715 s

As shown in Tables 4 and 5, a comparison between x265 and proposed transcoding was processed based on measurements of 2D and 360 video test-sequences. We used libx265 [34] with FFmpeg to verify all transcoding sessions for these test sequences. Consequently, we conclude that the proposed method

could speed up the transcoding by up to 195% when compared to x265. The highest achievement was 7.03 s for 360 transcoding time with a speed of 42.8 frames/s, and 2D video transcoding speed could achieve up to 42.88 frames/s.

Table 4. The 2D transcoding comparison.

Test_Sequence	x265	Proposed
Basketball_1920 × 1080_500 frames_25fps_qp22	53.45 s (9.36 fps)	11.89 s (42.05 fps)
Cactus_1920 × 1080_500 frames_25fps_qp22	42.23 s (11.84 fps)	11.92 s (41.94 fps)
Kimono_1920 × 1080_240 frames_25fps_qp22	24.79 s (9.68 fps)	5.61 s (42.7 fps)
ParkScene_1920 × 1080_240 frames_25fps_qp22	21.76 s (11.03 fps)	5.596 s (42.88 fps)

Table 5. The 360 transcoding comparison.

Test_Sequence	x265	Proposed
DrivingInCity_1920 × 1080_300 frames	20.78 s (14.48 fps)	7.03 s (42.8 fps)
GasLamp_1920 × 1080_300frames	22.12 s (13.56 fps)	8.45 s (35.50 fps)
Harbor_1920 × 1080_300frames	22.45 s (13.37 fps)	8.58 s (34.9 fps)
KiteFlite_1920 × 1080_300frames	22.37 s (13.41 fps)	8.65 s (34.68 fps)
Trolley_1920 × 1080_300frames	22.2 9s (13.45 fps)	8.62 s (34.80 fps)

To evaluate the efficiency of ARIA encryption, we used FFprobe (a tool of FFmpeg software) and hevcbrowser software to verify output HEVC bitstreams for both HEVC only and HEVC within encryption. As shown in Figure 16a, FFmpeg tools showed that HEVC bitstream of Session 0 was easy to parse or decode. Figure 16b shows that output HEVC encrypted bitstream of Session 1 could not be parsed or decoded. The reason was the first NAL units were encrypted to a different format. The output 360 DrivingIncCity video was decoded by openHEVC decoder, as shown in Figure 16c. Additionally, by using hevcbrowser software, as shown in Figure 17b, we can confirm that VPS, PPS, and SPS NAL units were encrypted successfully, while Figure 17a shows the output HEVC bitstream of Session 0 without encryption. In Table 6, the WS-PSNR values show unsatisfactory performance compared to other quality metrics when it comes to estimating the quality of images and videos as perceived by humans. We verified the comparison for both 1080p and 4K video. All WS-PSNR values of three channels were higher than 38 (dB), thus the quality of 360 videos at control center was reasonable to feel fully immersed in 360 videos.

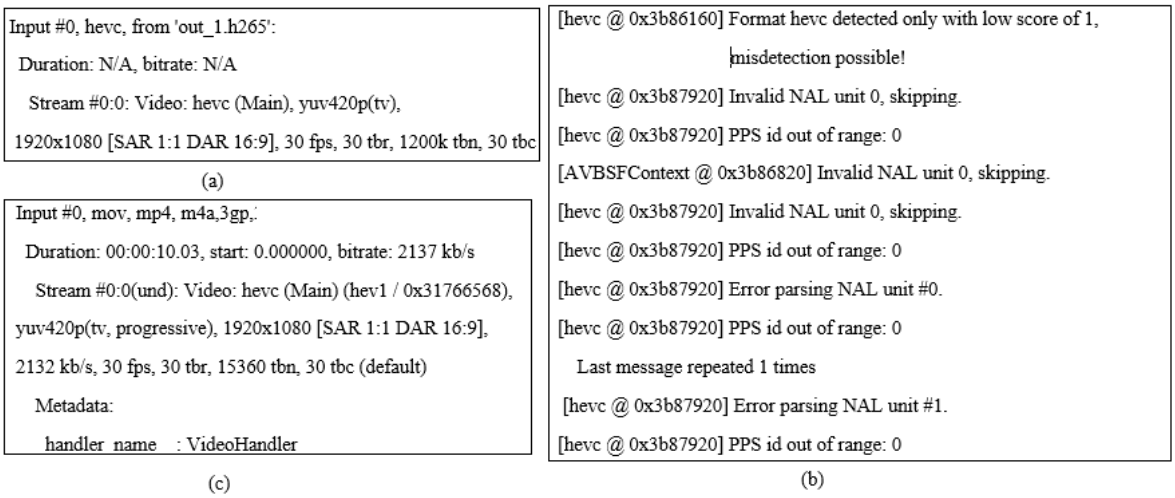


Figure 16. FFmpeg parsing: (**a**) unencrypted HEVC bitstream; (**b**) encrypted HEVC bitstream; (**c**) decoded output video.

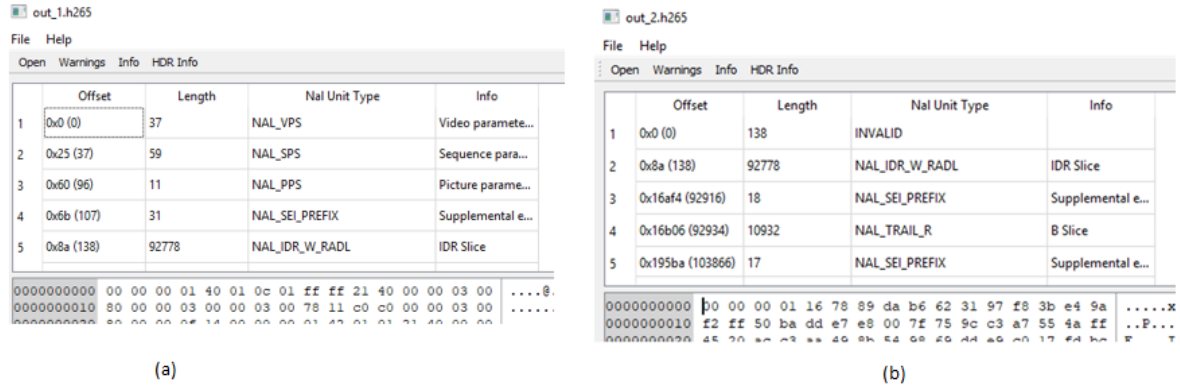


Figure 17. Hevcbrowser parsing: (**a**) unencrypted HEVC bitstream; (**b**) encrypted HEVC bitstream.

Table 6. WS-PSNR (Peak Signal-to-Noise Ratio) comparison.

Test_Sequence	WS-PSNR Y Channel	WS-PSNR U Channel	WS-PSNR V Channel
DrivingInCity_300frames_30fps_1920 × 1080	39.36	45.14	44.52
DrivingInCity_300frames_30fps_3840 × 1920	38.68	45.33	44.63
GasLamp_300frames_30fps_1920 × 1080	41.47	46.74	46.08
GasLamp_300frames_30fps_4096 × 2048	42.19	47.34	46.61
Harbor_300frames_30fps_1920 × 1080	40.92	47.36	47.52
Harbor_300frames_30fps_4096 × 2048	41.24	48.43	48.25
KiteFlite_300frames_30fps_1920 × 1080	38.03	44.22	45.49
KiteFlite_300frames_30fps_4096 × 2048	38.94	46.05	46.83
Trolley_300frames_30fps_1920 × 1080	39.46	45.43	46.81
Trolley_300frames_30fps_4096 × 2048	39.78	47.05	47.60

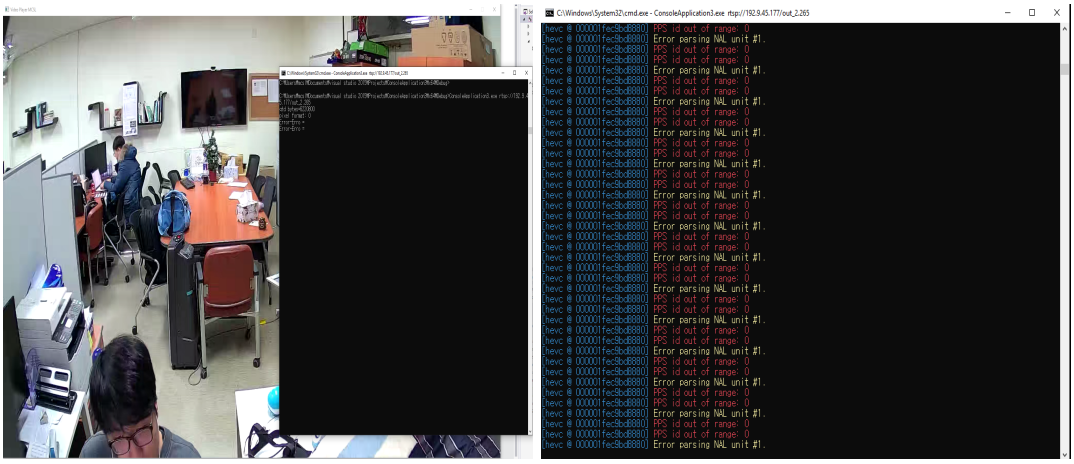
Figure 18 shows the video decryption and video client display. Here, the transcoder box will encrypt the entire video. With correct ARIA keys, as shown in Figure 18a, the video client can show the video after the decryption process. As shown in Figure 18b, video client cannot decrypt the VPS, SPS and PPS NAL units. Thus, it could not find correct video stream info while parsing the RTSP stream. To verify the encryption efficiency, the proposed system was used for experiments with AES and ARIA algorithms. Table 7 shows the comparison between AES and ARIA on the transcoder box with the master key sizes of 128 bits and 256 bits. The results prove that the AES algorithm could provide the encryption speed up to 33.4 times when compared to the ARIA algorithm. Additionally, transcoding and AES encryption process could reduce the total processing time by around 5–10% when compared to the ARIA algorithm, as shown in Table 8. The percentage of AES encryption time over the total processing time decreases when the CCTV video duration time increases. The ARIA encryption gives the result in the same way with higher percentage than AES encryption. Finally, from the security point of view, AES encryption was one of the best choices to secure data.

Table 7. Secure algorithm benchmark on proposed system using Crypto++.

Secure Algorithm	MiB/S	Cycles Per Byte	Microseconds to Setup Key and IV	Cycles to Setup Key and IV
AES (128_bit key)	4648	0.55	0.237	639
AES (256_bit key)	3948	0.65	0.228	617
ARIA (128_bit key)	140	18.36	0.250	675
ARIA (256_bit key)	109	23.71	0.261	705

Table 8. Processing time comparison between ARIA and AES algorithms.

Test_Sequence	ARIA Total Time	ARIA Time	AES Total Time	AES Time
ParkScene_240 frames_25fps	5.596 s (42.88 fps)	3.16%	5.07 s (47.33 fps)	2.82%
Kimono_240 frames_25fps	5.61 s (42.7 fps)	2.53%	5.39 s (44.5 fps)	2.07%
Basketball_500 frames_25fps	11.89 s (42.05 fps)	2.28%	11.55 s (43.29 fps)	1.14%
Cactus_500 frames_25fps	11.92 s (41.94 fps)	2.19%	11.28 s (44.32 fps)	0.95%



(**a**) (**b**)

Figure 18. Video client decrypts encrypted CCTV video bitstream: (**a**) with correct keys; and (**b**) without correct keys.

5. Conclusions

The development of 360 video technology has led to the promised development of CCTV system, which has been maintained with long-standing technologies. In this paper, we propose the transcoding and encryption scheme for live CCTV video streams in real-time. The proposed method can be used for low bandwidth and capacity for 2D/3D CCTV systems that only support H.264 and earlier codecs. Experimental results show that the combination of parallel video processing and encryption enhances the security without increasing the overall complexity of the system, while still ensuring the real-time live streams from CCTV cameras to control center. The proposed system optimized real-time transcoding for six 1080p sessions and two 4K sessions with speeds of 36.4 FPS and 33.6 FPS, respectively.

In the future, we intend to apply cubemap projection extensively into 360 CCTV cameras to improve the quality of video input. Additionally, the application of cubemap projection also offers many advantages in optimizing the encoding of the 360 CCTV cameras. In addition, perfecting the encoding/decoding mechanism is also an important factor in improving the performance of CCTV systems in the coming years. Regarding security issue, we also plan to experiment with applying cryptographic attack and plaintext attack to check the security feature.

Author Contributions: Conceptualization, Data curation and Writing—original draft, T.T.L.; Data curation and Investigation, J.J.; and Project administration, Supervision and Writing—review and editing, E.-S.R.

Funding: This work was supported by Institute for Information & communications Technology Promotion (IITP) grant funded by the Korea government (MSIP) (No. 2017-0-00307, Development of Tiled Streaming Technology for High Quality VR Contents Real Time Service). It also was supported by the MSIT (Ministry of Science and ICT), Korea, under the ITRC (Information Technology Research Center) support program (IITP-2018-2017-0-01630) supervised by the IITP (Institute for Information & Communications Technology Promotion).

Conflicts of Interest: The authors declare no conflict of interest.

Appendix A

Tables A1 and A2 are for 360 video encoding using ERP projection.

Table A1. Equirectangular projection for 360 video.

Parameters	Value	Note
SphereVideo	1	1: 360 video; 0: traditional video;
InputGeometryType	0	0: equirectangular; 1: cubemap; 2: equalarea;
SourceFPStructure	1 1 0 0	frame packing order
CodingGeometryType	0	
CodingFPStructure	1 1 0 0	frame packing order
SVideoRotation	0 0 0	rotation along X, Y, Z;
CodingFaceWidth	0	0: automatic calculation; 4096 for 8K; 3328 for 4K;
CodingFaceHeight	0	0: automatic calculation; 2048 for 8K; 1664 for 4K;
InterpolationMethodY	5	interpolation method for luma, 0: bicubic; 1:NN, 2: bilinear, 3: bicubic, 4: lanczos2, 5: lanczos3
InterpolationMethodC	4	interpolation method for chroma, 0: bicubic; 1:NN, 2: bilinear, 3: bicubic, 4: lanczos2, 5: lanczos3
InternalChromaFormat	420	internal chroma format for the conversion process;
SPSNR_NN	1	enable end-to-end S-PSNR-NN calculation;
SPSNR_I	0	enable end-to-end S-PSNR-I calculation;
CPP_PSNR	0	enable end-to-end CPP-PSNR calculation;
E2EWSPSNR	1	enable end-to-end WS-PSNR calculation;
CODEC_SPSNR_NN	1	enable codec S-PSNR-NN calculation;
WSPSNR	1	enable codec WS-PSNR calculation;
CF_SPSNR_NN	1	enable cross-format S-PSNR-NN calculation;
CF_SPSNR_I	0	enable cross-format S-PSNR-I calculation;
CF_CPP_PSNR	1	enable cross-format CPP-PSNR calculation;
SphFile		../cfg-360Lib/360Lib/sphere_655362.txt
ViewPortPSNREnable	0	1: Yes; 0: No
ViewPortList		2 75.0 75.0 0.0 0.0 75.0 75.0 −90.0 0.0
ViewPortWidth	428	1816 for 8K; 856 for 4K; 428 for Full HD
ViewPortHeight	428	1816 for 8K; 856 for 4K; 428 for Full HD
DynamicViewPort PSNREnable	1	1: Yes; 0: No
DynamicViewPortList		2 75.0 75.0 0 −45.0 −15.0 299 45.0 15.0 75.0 75.0 0 −135.0 −15.0 299 −45.0 15.0
DynamicViewPortWidth	428	1816 for 8K; 856 for 4K; 428 for Full HD;
DynamicViewPortHeight	428	1816 for 8K; 856 for 4K; 428 for Full HD;

Table A2. Coding configuration of 4K 360 video.

Parameters	Value	Note
InputFile	videosequences 4096×2048_30fps_8bit_420_erp.yuv	Format name
InputBitDepth	8	Input bitdepth
InputChromaFormat	420	Ratio of luminance to chrominance samples
FrameRate	30	Frame Rate per second
FrameSkip	0	Number of frames to be skipped in input
SourceWidth	4096	Input frame width
SourceHeight	2048	Input frame height
FramesToBeEncoded	300	Number of frames to be coded
Level	5.2	
DynamicViewPort PSNREnable	1	
DynamicViewPortList	2 75.0 75.0 0 210 −18 299 300 12 75.0 75.0 0 30 −44 299 120 −14	
DynamicViewPortWidth	856	(1816 for 8K; 856 for 4K; 428 for Full HD)
DynamicViewPortHeight	856	(1816 for 8K; 856 for 4K; 428 for Full HD)

References

1. JCT-VC. High Efficiency Video Coding (HEVC). Available online: https://hevc.hhi.fraunhofer.de/ (accessed on 17 February 2019).
2. Ohm, J.-R.; Sullivan, G.J.; Schwarz, H.; Tan, T.K.; Wiegand, T. Comparison of the coding efficiency of video coding standards—Including High Efficiency Video Coding (HEVC). *IEEE Trans. Circuits Syst. Video Technol.* **2012**, *22*, 1669–1684. [CrossRef]
3. JCT-VC—Joint Collaborative Team on Video Coding. Available online: https://www.itu.int/en/ITU-T/studygroups/2013-2016/16/Pages/video/jctvc.aspx (accessed on 18 February 2019).
4. Wiegand, T.; Sullivan, G.J. Overview of the H.264/AVC video coding standard. *IEEE Trans. Circuits Syst. Video Technol.* **2003**, *13*, 560–576. [CrossRef]
5. WG11 (MPEG). *MPEG Strategic Standardisation Roadmap*; Technical Report ISO/IEC JTC1/WG11; MovingPicture Experts Group (MPEG): Villar Dora, Italy, 2016.

6. Champel, M.L.; Koenen, R.; Lafruit, G.; Budagavi, M. *Working Draft 0.4 of TR: Technical Report on Architectures for Immersive Media*; Technical Report ISO/IEC JTC1/WG11; Moving Picture Experts Group (MPEG): Villar Dora, Italy, 2017.
7. Wang, X.; Chen, L.; Zhao, S.; Lei, S. From OMAF for 3DoF VR to MPEG-I media format for 3DoF+, windowed 6DoF and 6DoF VR. In Proceedings of the 119th MPEG Meeting of ISO/IEC JTC1/SC29/ WG11, MPEG 119/m44197, Torino, Italy, 17–21 July 2017.
8. Kalva, H. Issues in H.264/MPEG-2 video transcoding. In Proceedings of the 1st IEEE Consumer Communications and Networking Conference, Las Vegas, NV, USA, 5–8 January 2004; pp. 657–659.
9. Zhou, Z.; Sun, S.; Lei, S.; Sun, M.-T. Motion information and coding mode reuse for MPEG-2 to H.264 transcoding. In Proceedings of the 2006 IEEE International Symposium on Circuits and Systems, Kobe, Japan, 23–26 May 2005; pp. 1230–1233.
10. Chen, Y.; Wen, Z.; Wen, J.; Tang, M.; Tao, P. Efficient software H.264/AVC to HEVC transcoding on distributed multicore processors. *IEEE Trans. Circuits Syst. Video Technol.* **2015**, *25*, 1423–1434. [CrossRef]
11. Luong, P.V.; De Praeter, J.; Van Wallendael, G.; Van Leuven, S.; De Cock, J.; de WalleE, R.V. Efficient bit rate transcoding for high efficiency video coding. *IEEE Trans. Multimed.* **2016**, *18*, 364–378.
12. Díaz-Honrubia, A.J.; Martínez, J.L.; Cuenca, P.; Gamez, J.A.; Puerta, J.M. Adaptive fast quadtree level decision algorithm for H.264 to HEVC video transcoding. *IEEE Trans. Circuits Syst. Video Technol.* **2016**, *26*, 154–168. [CrossRef]
13. Zhang, X.; Seo, S.H.; Wang, C. A lightweight encryption method for privacy protection in surveillance videos. *IEEE Access* **2018**, *6*, 18074–18087. [CrossRef]
14. Wampler, C.; Uluagac, S.; Beyah, R. Information leakage in encrypted IP video traffic. In Proceedings of the 2015 IEEE Global Communications Conference (GLOBECOM), San Diego, CA, USD, 6–10 December 2015.
15. VLC x264 Library. Available online: https://www.videolan.org/developers/x264.html (accessed on 17 February 2019).
16. FFmpeg Software. Available online: https://www.ffmpeg.org/ (accessed on 17 February 2019).
17. Intel. Quick Sync Video Technology. Available online: https://www.intel.com/content/www/us/en/architecture-and-technology/quick-sync-video/quick-sync-video-general.html (accessed on 17 February 2019).
18. Security and Performance Analysis of Aria. Available online: https://www.esat.kuleuven.be/cosic/publications/article-500.ps (accessed on 17 February 2019).
19. Li, S.; Song, C. Improved impossible differential cryptanalysis of ARIA. In Proceedings of the 2008 International Conference on Information Security and Assurance (isa 2008), Busan, Korea, 24–26 April 2008.
20. Pandey, A.; Rizvi, M.A. Comparative survey of different cryptographic algorithm. *Int. J. Sci. Eng. Res.* **2017**, *8*, 41–44.
21. Lee, J.; Kwon, D.; Kim, C. *IETF RFC: ARIA Encryption Algorithm*; NRSI: Syosset, NY, USA, 2010. Available online: https://tools.ietf.org/html/rfc5794 (accessed on 20 February 2019).
22. Kwon, D.; Kim, J.; Park, S.; Soong, S.H.; Sohn, Y.; Song, J.H.; Yeom, Y.; Yoon, E.-J.; Lee, S.; Le, J.; et al. Aria Encryption KISA. In Proceedings of the Information Security and Cryptology (ICISC 2003), Seoul, Korea, 27–28 November 2003; LNCS 2971, pp. 432–445. Available online: http://www.math.snu.ac.kr/~jinhong/04Aria.pdf (accessed on 20 February 2019).
23. Kim, W.; Lee, J.; Park, J.; Kwon, D. *IETF RFC: ARIA Cipher Suites to Transport Layer Security (TLS)*; NRSI: Syosset, NY, USA, 2011. Available online: https://tools.ietf.org/html/rfc6209 (accessed on 20 February 2019).
24. HM Software and 360 Library. Available online: https://hevc.hhi.fraunhofer.de/ (accessed on 18 February 2019).
25. Schulzrinne, H.; Rao, A.; Lanphier, R. *IETF RFC: Real Time Streaming Protocol*; RealNetworks: Seattle, WA, USA, 1998. Available online: https://tools.ietf.org/html/rfc2326 (accessed on 20 February 2019).
26. Schulzrinne, H.; Rao, A.; Lanphier, R.; Westerlund, M.; Stiemerling, M. *IETF RFC: Real-Time Streaming Protocol Version 2.0*; Westerlund Ericsson, M., Stiemerling, M., Eds.; University of Applied Sciences Darmstadt: Darmstadt, Germany, 2016. Available online: https://tools.ietf.org/html/rfc7826 (accessed on 20 February 2019).

27. Hanwha CCTV Camera—Model LNO-6010R. Available online: https://www.hanwha-security.com/en/products/camera/network/bullet/LNO-6010R/overview/ (accessed on 19 February 2019).
28. Open HEVC Decoder. Available online: Available:http://openhevc.github.io/openHEVC/ (accessed on 17 February 2019).
29. Intel Media SDK. Available online: https://software.intel.com/en-us/media-sdk (accessed on 17 February 2019).
30. Crypto++ Library. Available online: https://www.cryptopp.com/ (accessed on 17 February 2019).
31. Hevcbrowser Software. Available online: https://github.com/virinext/hevcesbrowser/projects (accesed on 19 February 2019).
32. ERP WS-PSNR Software. Available online: http://mpegx.int-evry.fr/software/MPEG/Explorations/3DoFplus/ERP_WS-PSNR (accesed on 19 February 2019).
33. Yu, M.; Lakshman, H.; Girod, B. A framework to evaluate omnidirectional video coding schemes. In Proceedings of the 2015 IEEE International Symposium on Mixed and Augmented Reality (ISMAR), Fukuoka, Japan, 29 September–3 October 2015.
34. VLC x265 Library. Available online: https://www.videolan.org/developers/x265.html (accessed on 19 February 2019).