Cryptography-Based Secure Data Storage and Sharing Using HEVC and Public Clouds

Muhammad Usman[a], Mian Ahmad Jan[b], Xiangjian He[a,*]

[a]*School of Computing and Communications, University of Technology Sydney, Australia*
[b]*Department of Computer Science, Abdul Wali Khan University Mardan, Pakistan*

Abstract

Mobile devices are widely used for uploading/downloading media files such as audio, video and images to/from the remote servers. These devices have limited resources and are required to offload resource-consuming media processing tasks to the clouds for further processing. Migration of these tasks means that the media services provided by the clouds need to be authentic and trusted by the mobile users. The existing schemes for secure exchange of media files between the mobile devices and the clouds have limitations in terms of memory support, processing load, battery power, and data size. These schemes lack the support for large-sized video files and are not suitable for resource-constrained mobile devices. This paper proposes a secure, lightweight, robust, and efficient scheme for data exchange between the mobile users and the media clouds. The proposed scheme considers High Efficiency Video Coding (HEVC) Intra-encoded video streams in unsliced mode as a source for data hiding. Our proposed scheme aims to support real-time processing with power-saving constraint in mind. Advanced Encryption Standard (AES) is used as a base encryption technique by our proposed scheme. The simulation results clearly show that the proposed scheme outperforms AES-256 by decreasing the processing time up to 4.76% and increasing the data size up to 0.72% approximately. The proposed scheme can readily be applied to real-time cloud media streaming.

*Corresponding author
Email address: Xiangjian.He@uts.edu.au (Xiangjian He)

Keywords: Media Clouds, HEVC, Intra, Unsliced Mode, AES, Media Streaming

1. Introduction

Mobile communication plays a vital role in our daily lives and has seen an unprecedented technological growth over the past two decades. Apart from using mobile devices for basic telecommunication services such as messaging and voice, an increasing number of customers is using their phones for sharing multimedia data by taking pictures and capturing videos. As a result, a huge amount of data is generated by mobile users all the time. According to the latest Cisco report, the data generated by mobile users grew up to 69% in 2014 with 2.5 exabytes per month at the end of 2014. During this period, 4G connections produced 10 times more data traffic as compared to non-4G connections. The expected growth of data traffic in 4G connections is shown in Figure 1[1]. The vertical axis represents data in units of exabytes whereas horizontal axis represents the years.

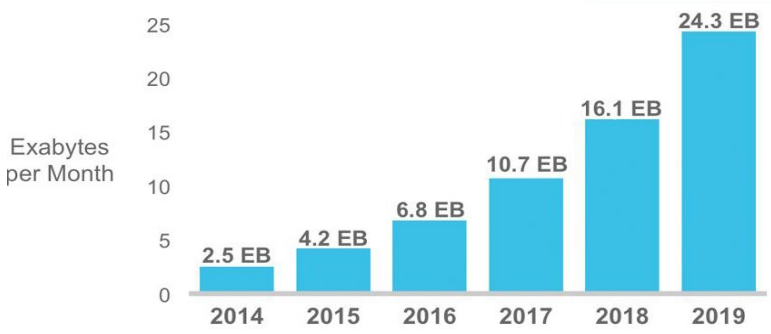


Figure 1: Cisco Forecast for Mobile Data Traffic

Although, the storage capacity of mobile devices has improved significantly, it still cannot meet the ever-increasing demands of mobile users. Cloud plat-

[1]C. V. N. I. Cisco, Global mobile data traffic forecast up date, 2014-2019, white paper.

forms facilitate mobile users by storing their data remotely in media clouds and retrieving later at any point of time. Therefore, they provide an ease for mobile users with an option to enhance their virtual storage [36] as shown in Figure 2. Accessing the data over wireless channels from media clouds has gained popularity among the mobile users, primarily due to the development of various multimedia applications [42]. However, transmission of private data over wireless links is prone to various security breaches. As a result, security provisioning has become a major concern for data residing over the media clouds [35]. As the data can be transferred and stored on a cloud system through wireless links, it is vulnerable to alteration, unauthorized disclosure, and replay attacks [32]. Trust of the users need to be guaranteed while they upload their multimedia and secret data to different clouds. Although, there are existing approaches which provide a facility to share secret information among a number of authorized users, such approaches are applied on simple form of data [6, 13]. In case of videos, especially the large-sized videos, such approaches have limitations primarily due to the data size and the processing capability of end-user's devices. As a result, lightweight but secured schemes need to be designed.

The growing popularity of High Definition (HD) videos and the emergence of HD and beyond-HD formats such as 4k×2k or 8k×4k resolutions are creating a benchmark for visual quality [30]. The High Efficiency Video Coding (HEVC) is an emerging standard to deal with such high resolution multimedia contents [4]. This standard provides three basic modes for encoding multimedia contents, i.e., Intra, low-delay and random. The intra mode treats each video frame as an independent image and focuses on quality rather than compression. This mode contains only I-frames and is suitable for the applications including video surveillance and live high-quality video conferences, where no compromise on visual quality can be made. The low-delay mode provides more compression with a compromise on visual quality and provides approximately 33% reduction in bit rate. This mode is a combination of I and P-frames and is suitable for those online applications where compromise on visual quality can be made. An application of the low-delay mode is online gaming, which is usually played on

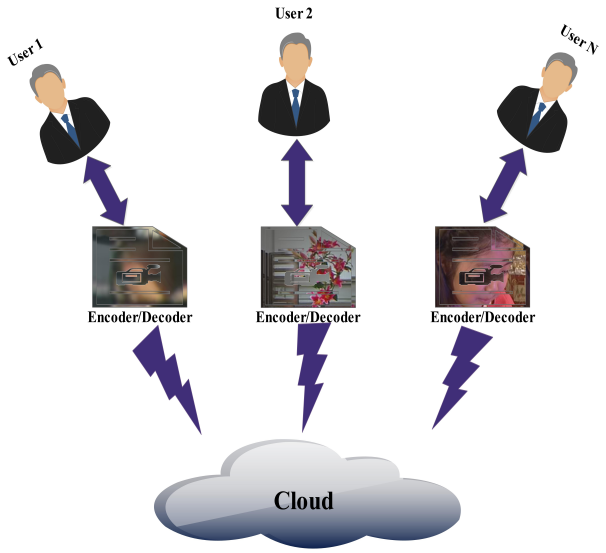


Figure 2: Mobile Data Storage

mobile devices with low resolutions. The random mode is a combination of I, P and B-frames with a facility to access any part of a frame. This mode offers better compression as compared to the low-delay mode and is mainly used for contents storage with almost 31% reduction in bit rate [33].

As a result of the above distinguishing features of HEVC standard, researchers are focusing on developing security techniques which may utilize the encoded contents produced by HEVC standard. Such techniques not only require an analysis of HEVC-produced contents for data protection but also demand authorization by exchanging a key as shown in Figure 3. Security is of utmost importance especially when videos are uploaded to public clouds and later on retrieved by the users. Various encryption algorithms are proposed to protect the video sequences against unauthorized attacks by malicious users who use third-party tools and methods to steal the transmitted video sequences [41, 40]. However, these algorithms focus only on restricting the video access but do not provide any facility to secure secret data hidden in those videos.

Furthermore, the specific requirements of mobile users such as processing capability, hardware resources, and power backup are ignored while downloading and uploading their multimedia data to the public clouds.

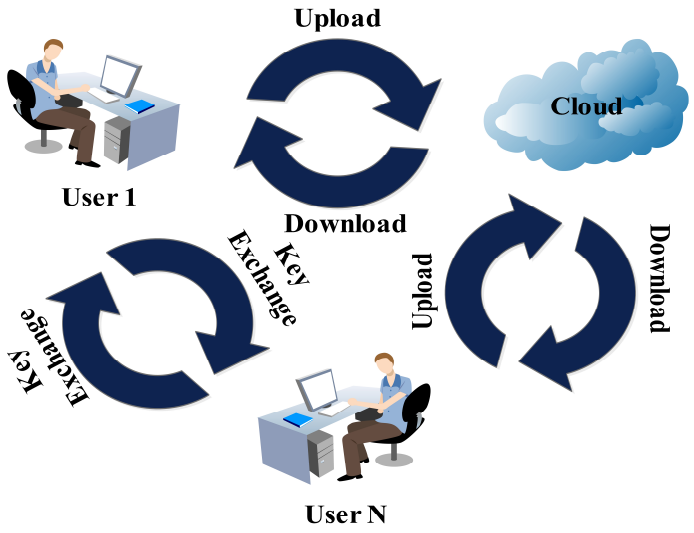


Figure 3: Traditional Security Scheme

Our research has the following major contributions.

1. The HD videos always contain redundant information. If these videos are encoded in Intra domain, the compression is too low for maintaining the visual quality. As a result of low compression, there is always an abundant space available for hiding secret information. We propose a cryptographic algorithm by using Intra-encoded HD video sequences in public clouds. Although it introduces a slight overhead of executing the proposed cryptographic algorithm on the input video sequence, it is still implementable on latest mobile devices.
2. The encrypted videos are shared on a public cloud, which is semi-trusted. If an intruder downloads the video and steals the Private Key (PRK) and Public Key (PUK), it will only be able to decrypt the videos rather than the encrypted secret data. The decryption process can only be performed by an authentic user having a Secret Key (SK). Moreover, the targeted

users are always mobile users, located at different geographic locations across the globe. Our proposed approach supports mobility of the users by authenticating them remotely.

3. Our proposed approach is a combination of PRK, PUK, and SK and does not require continuous synchronization among the users as shown in Figure 4. The responsibilities of an uploading user is to encrypt and upload the data and the video on the cloud along with broadcasting an announcement to all authorized users that a new data has been uploaded. Afterwards, it is the choice of the other authorized users whether they want to download and decrypt the secret data, embedded in the uploaded videos. By possessing the same SK, they can download and fully decrypt the hidden data at any time.
4. The proposed approach works efficiently in public clouds by providing the required security level and at the same time eliminating the cost of having private clouds. Our proposed approach efficiently utilizes the computational power of cloud resources by partially completing the decryption process. As a result, the delay occurring in the decryption process can be minimized at the receiver's side. Since, our proposed scheme targets mobile users, it is not necessary for the receivers to have enough computational resources.

The main purpose of the public clouds is to provide enough storage and computational power to the users. Therefore, cloud environments are ideal for processing and storing HD videos. Our proposed scheme increases the compressed video size up to 0.72% on average, as compared to the other state-of-the-art techniques which increase the size up to 9% or higher in some cases [23]. Although our approach affects the size of the transmitted binary video data, the variation of size is relatively small and can easily be ignored as compared to the existing techniques. This variation mainly depends on the quantity of encrypted data. If the quantity is not too large, the increase in size will not be noticeable. This is contrary to the existing approaches, where the increase

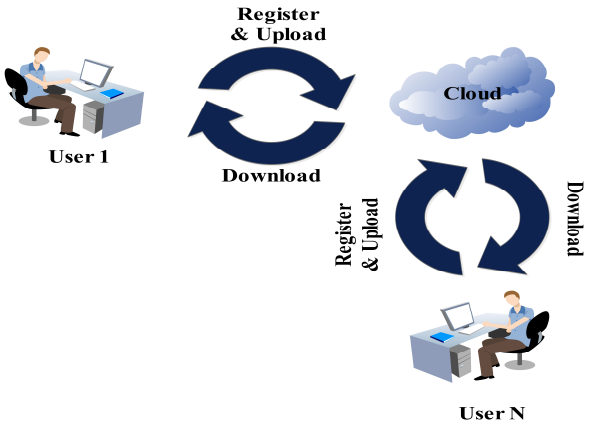


Figure 4: Updated Security Scheme

in size is relatively higher. Although we target HD videos in this paper, our proposed approach is equally applicable to any other resolution of the videos.

The rest of the paper is organized as follows. In Section 2, related work from literature is presented. In Section 3, our proposed approach is explained in detail, followed by simulation setup and results in Section 4. Finally, the paper is concluded in Section 5.

2. Related Work

Mobile devices such as smartphones and tablets are becoming an indispensable part of our lives for entertainment and convenient communication. With the increasing popularity of various mobile devices, there is a phenomenal growth in the development of mobile applications such as email, web browsing, mobile games, terrestrial navigation, mobile health care, and social networking. These applications indicate that mobile devices are quickly becoming the dominant computing platforms for the provisioning of seamless connectivity and entertainment regardless of the user's mobility. Mobile devices, on the other hand, are still restricted in terms of their resources such as computational capabilities, storage, and battery life time. Furthermore, they have limited com-

munication resources such as available bandwidth and connectivity [5]. The resource-constrained nature of these devices limits the support for developing various mobile applications. Mobile Cloud Computing (MCC) has resolved the problem associated with the resource availability of these devices. The MCC allows mobile devices to offload computationally-complex and space-demanding tasks to the clouds [44, 45]. Clouds, on the other hand, have ample resources and provide an ideal platform for resource-consuming mobile applications such as speech recognition and video encoding/decoding [9, 37]. Many other similar applications need to be offloaded to the clouds for processing and utilizing the huge amount of resources available with these clouds. Computational offloading of these applications saves energy and improves the performance of mobile applications [8, 11]. Moreover, the MCC enables the mobile users to store/access major portion of their data on the cloud using wireless networks. This feature of MCC enables mobile devices to utilize their storage and processing power efficiently [7].

The on-demand nature of cloud computing faces various security threats such as data loss, data leakage, Denial-of-Service (DoS), account or service traffic hijacking, and malicious insiders. A malevolent hacker may modify critical data due to a careless cloud service provider. To tackle such risk, the important data needs to be encrypted and the encryption keys must be protected [20, 1]. If an intruder gains access to a customer credentials stored on a cloud, it may eavesdrop on transactions and activities, maliciously manipulates the data, returns falsified information, and may redirect the customer to illicit sites. DoS attack is another major concern for cloud platforms because most of the organizations are dependent on 24/7 availability of one or more services [20]. Denial of one or more services may be costly to the customers especially, when they are billed based on disk-space consumption and computation cycles. Account or service hijacking is another major threat faced by cloud platforms. Hijacking a service allows a malicious person to sneak into crucial and sensitive areas of a deployed service which may lead to breaching the integrity, availability, and confidentiality of that service. A malicious insider, e.g., a current or former em-

ployee, a business partner, or a contractor, may gain access to the data, network or system for malicious purposes [1]. The situation gets worse when the cloud service provider is solely responsible for data security.

Cloud platforms attract more attacks due to their distributed nature. It is desirable that the data (video contents in this context) is protected and may only be accessed in encrypted form. Directly hiding the data in encrypted HEVC video streams to intact its quality can avoid the leakage of video contents. This feature of data hiding can address the security and privacy concerns associated with the cloud computing [19, 24]. A cloud server has the ability to embed the additional information about a video such as video notation and data authentication, into an encrypted version of HEVC format. Once the data is hidden, the server can verify the integrity without knowing the original contents. As a result, security and privacy of the encrypted data is preserved. In literature, various studies exist on encrypting the data in videos. A novel encryption method for Intra and Inter frames in MPEG videos is presented in [38]. The authors argued that highly sensitive and private videos require encrypting the whole video. Therefore, not only the Intra frames, but the Inter frames were also required to be encrypted. In [39], an encryption scheme for MPEG videos is proposed. The proposed scheme was based on Advanced Encryption Standard-128 bit (AES-128) algorithm. Only Intra frames of a particular video were encrypted because the Inter frames are useless without knowing the corresponding Intra frames associated with them. In [25], a novel scheme based on selective encryption for H.264 data is presented. The proposed scheme ensured a transparent encryption and protection against various attacks. The proposed encryption and decryption was relatively faster at the time of preserving the formation and length of the video streams.

Various clouds such as Google App Engine and Amazon web services have experienced security threats during recent years. These security flaws are exploited by illegal users to steal either secret information or disturb the normal operation of Internet. As a result, robust and lightweight authentication and authorization schemes are required for the devices interacting with the cloud

platforms. Cloud computing is a variant of client-server architecture model, where, thousands of clients use the same infrastructure at a lot larger scales. Identity and access control management is a core requirement for cloud computing [14, 18]. Therefore, stronger authentication is required as compared to conventional client-server interaction model. In [46], the authors proposed a cloud-assisted privacy preserving key management scheme for mobile attacks. Their proposed scheme protected patient's identity and location information. Moreover, the proposed scheme did not take into account the key privacy and updating. Hence, their scheme was not suitable for a real-time cloud computing platform. Bilinear pairing in an elliptic curve has recently gained attention in developing an ID-based cryptosystem [12, 31]. This cryptosystem solved the high cost issue of authentication and public key management derived from traditional public key cryptosystems. The identity of each user was used as a public key. Therefore, a user did not require extra computational cost for verifying the public keys of the other users. Moreover, no extra storage space at the user's device was required to store public keys and corresponding certificates of the other users.

Recently, several studies have applied ID-based cryptosystems in various cloud environments. In [43], a new ID-based authentication scheme is designed for cloud environment. Although the proposed scheme was suitable for a distributed mobile cloud service environment, it lacked the support for user anonymity and untraceability. Most of the authentication schemes which are based on elliptic curve or bilinear pairing [12, 31] are designed for client-server environment. They are not feasible for direct application to distributed service environments, where multiple service providers compete with each other for provisioning of various services. The user needs to manage multiple secret keys learned from each service provider. To resolve this issue, all service providers need to share the same secret key. However, if an adversary acquires the secret key, it may pose as a legitimate service provider to deceive the users. Moreover, an intruder who captures the secret key may acquire the session keys as well. After acquiring one or more session keys, the attacker may eavesdrop on sensitive

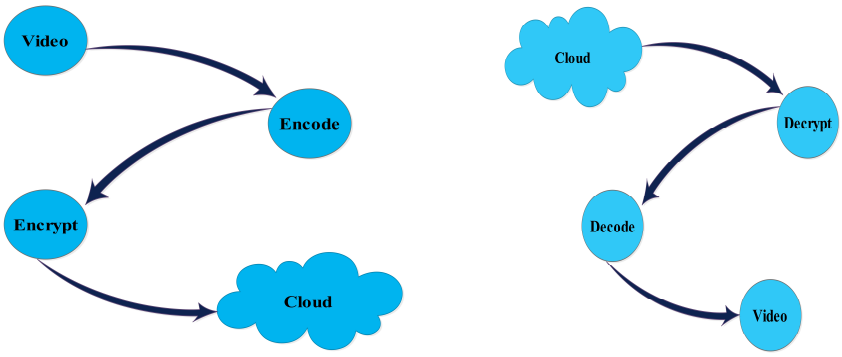
information transmitted between the user and another service provider.

3. Proposed Approach

In this section, we present our proposed data hiding scheme, which is a combination of PRK, PUK and SK. The underlying encryption algorithms used by our proposed scheme are modified AES-256 bit and RSA [10, 16]. The modified AES-256 executes at the user's side while the RSA executes at the cloud's side. The cloud generates a pair of PRK and PUK. The PRK remains at the cloud's side while the PUK is sent to the user for encryption purpose. Our proposed scheme is applied directly to HEVC-encoded video stream. In general, this scheme has three major phases, i.e., HEVC video encoding, video and secret data encryption, half decryption and full decryption with the secret data extraction. The uploading user, also known as data owner, first encodes the video, encrypts the secret data using SK, adjusts the encrypted data into encoded video, and then encrypts the video using PUK to generate an encrypted HEVC Encoded Video Stream (HEVS). Next, the owner uploads the encrypted HEVS over the public cloud, where the cloud sources decrypt the video using the PRK, a technique known as half decryption. The cloud sources have no knowledge about the data hidden in the video stream. The receiver downloads the video, decrypts it using SK, extracts the required data and then either keeps or discards the video stream. The overall process of the proposed scheme is shown in Figures 5a and 5b.

3.1. Video Encoding and Data Encryption

An HD video usually contains N frames, where N is a positive integer. It is to be noted here that, for real-time processing, it is not possible to process and encode an entire video sequence in one go. Moreover, HEVC codec and HD videos require enough hardware resources and computing power. As our targeted users are mobile users, the above limitations need to be considered before starting an encoding process. A simple solution to deal these limitations



(a) Encoding and Encryption (b) Decryption and/or Decoding

Figure 5: Encryption and Decryption Process

is to reduce the total number of frames for encoding. Apart from encoding limitations, it is not compulsory to encrypt entire HEVS. The encryption of an entire HEVS increases the computations and time cost which also affect the format of the HEVS. One simple solution for these later limitations is to nominate a part of HEVS for encryption in order to enhance the security level. To keep the format of the HEVS undisturbed, it is safe to encrypt the secret data in Spatial Information (SI). The Motion Vectors (MVs) information can also be utilized for this purpose.

In this paper, HEVS-based encryption scheme is proposed which is time-efficient, less complex in terms of computations, and does not disturb the format of HEVS. After encoding the HD video using the standard settings in HEVC codec, three main parts in HEVS can easily be utilized to encrypt the secret data, i.e., the SI, the MVs, and Intra Prediction Modes Information (IPMI). As compared to [34], our proposed scheme encrypts the secret data in compressed domain, not in the encoding domain. This feature makes it easy to apply our scheme directly to modify the HEVS. Modified versions of AES have already been applied on H.264/AVC codec-based encoded video streams [26]. The similarity among our proposed scheme and those presented in [26, 34] is the usage of modified AES for data encryption purpose. Our proposed scheme differs from

[26, 34] in the sense that we are using HEVC/H.265 codec while they have used H.264. The former codec is the latest one while the latter is the previous version. We are using the HD videos, which have high resolution and difficult to deal with, while they have used low resolution videos. In this paper, we have used modified AES-256 bit algorithm along with PUK and cloud computing to make it more secure and computationally less complex. The entire computational process of our modified AES is same as the original one, except one main change. In the original implementation of AES, there are four processes involve in the transformation phase, i.e. SubBytes(), ShiftRows(), MixColumns(), and AddRoundKey(). In all these four processes, the MixColumns() is the highly computational one. In our modified AES version, we have replaced the MixColumns() with Permutations() process. The Permutations() process does not affect the entire structure of the AES but reduces its computational time significantly. As our target is mobile users, we cannot expect high computational power, especially the capacity to perform long matrix multiplications, from their mobile devices. The Permutations() process helps us to achieve this goal of low processing. We encrypt the secret data by distributing it into three parts, i.e., first part in the SI, second part in the MVs and the third part in the IPMI, in order to make the compressed HEVS format consistent. After encryption, the HEVS can still be decoded perfectly with any latest version of HEVC codec without showing any visible difference in the decoded videos. The procedure to encrypt secret data in nominated sections of HEVS is very complicated and requires deep knowledge of HEVC encoder. A minor mistake can disturb the HEVS format, and ultimately alert the decoder resulting in either crashing of the decoder or format mismatch errors.

3.1.1. Encryption in Spatial Information

To increase the security level, the secret data is divided equally into three parts. The first part is encrypted in the Spatial Information (SI). The SI considers the luminance plane information of pixels in the frames of a video. This information is always computed for the entire video sequence [22]. The HD

videos are encoded in Intra domain, so plenty of spatial information is available to utilize. This phase summarizes the embedding of secret data in the SI. In H.264 and H.265/HEVC standards, the entropy coding is used to encode the quantized coefficient values of the SI [2]. The main difference is that, in HEVC, only CABAC version is used while in H.264, both CABAC and CALVC are used. The details of CABAC and CALVC frameworks can be found in [27, 28]. The CABAC framework has three main parts, i.e., binarization, context modeling, and Binary Arithmetic Coding (BAC). We skip the detailed explanation of these three parts, as it is out of the scope of this paper. Next, we encode the HD videos in Intra domain, where the data compression is very low. As a result, we do not fully utilize the CABAC and our only focus is on the BAC but at a lower level.

The BAC is based on Least Probable Symbol (LPS) and Most Probable Symbol (MPS) rules and is adopted from Shannon-Fano coding [21]. The calculations of LPS and MPS are based on a probability, i.e., by dividing the possibility of occurrences of a symbol within a predefined interval, usually denoted by R. The first half of R, i.e., $[0, 0.5]$ reserves the probability of LPS while the second half, i.e., $[0.6, 1]$ reserves the probability of MPS. The interval $[0.5, 0.6]$ is considered as neutral and the symbols occurring in this interval can either be treated as LPS or MPS. Based on LPS and MPS, we utilize each byte of zero video data to encrypt the secret data. The secret data is first encrypted through modified AES-256 SK. Then, the encrypted secret data is converted into bytes. Next, a 2's complement operation is performed on encrypted secret data's bytes. Finally, the negated bytes are added with zero bytes of the video. In that way, the format and the number of bits in video's data bytes will remain in the same order. Moreover, the original pattern can easily be extracted back at the receiver end by converting those modified video's data bytes into zeros using a simple subtraction operation.

3.1.2. Encryption in Motion Vectors

The Motion Vectors (MVs) play an important role in video coding, especially in Inter and Random access modes. They are the key elements in the motion estimation process during video encoding/compression. In general, they are used to represent a block of pixels in current video frame, based on the position of same block of pixels in the reference video frame[2]. In the case of Intra mode, blocks of pixels are used to reference other blocks but within the same frame. This type of referencing is known as Intra prediction. Basically, the MVs are used to track moving regions and objects. Based on that motion information, compression and information discarding procedure is performed in video coding to keep the texture information of different objects in a video frame.

The HEVC codec can encode MVs with much better precision and less residual errors. The main reason behind this accuracy is the Intra prediction directions. In our scheme, there are 35 Intra prediction directions while in H.264 standard, only 9 directions are present. As stated above, only CABAC version of entropy coding is available in HEVC, so CABAC is used to encode calculated MVs for transmission purpose. After encoding through CABAC, the MVs transform into codewords. Each codeword is a combination of binary 1s and 0s and its length increases with the increasing number of MVs. The structure of the codewords is very simple. The most significant bit in the codewords is always zero. The length of the codewords can easily be calculated using Equation 1.

$$CL = AZ + B_s \tag{1}$$

Here, CL represents the codeword length, AZ represents the number of appended zeros and B_s represents the total number of bits used for binary presentation of a decimal value. The decimal value indicates the MV number. The length of the AZ is always "1 less than the B_s" as shown in Equation 2.

[2]https://hevc.hhi.fraunhofer.de

$$LAZ = LB_s - 1 \tag{2}$$

Here, LAZ represents the total length of the appended zeros and LB_s represents the length of total number of bits, respectively. For example, if a MV is represented by decimal number 13, then its equivalent binary representation is 1101. As this binary representation contains 4 bits, 3 zeros will be appended at the most significant place. As a result, the total number of bits will become 7, which is length of the codeword used to represent that specific MV.

From the above discussion, it is obvious that the most significant part of a codeword is always zero. Therefore, it can easily be utilized to hide the second part of the encrypted data by using the same procedure, as followed in section 3.1.1. This procedure maintains the order and format of the codewords.

3.1.3. Encryption of Intra Prediction Mode Information

In H.264 standard, three types of Intra coding modes are supported, i.e., Intra_4 $\times$ 4, Intra_8 $\times$ 8, and Intra_16 $\times$ 16 [29]. However, in case of HEVC, there are N Intra coding modes, ranging from Intra_2 $\times$ 2 to Intra_64 $\times$ 64 [15]. To perform compression and prediction in H.264 standard, the video frames are divided into blocks of equal sizes, also known as macroblocks (MBs). To provide better accuracy, the concept of MBs is replaced by Coding Tree Unit (CTU) in HEVC. To make the encoding process fast and simple, Intra_32 $\times$ 32 and Intra_64 $\times$ 64 block sizes are selected. The intra prediction is used to recover the lost information of a block of pixels from its surrounding blocks. Both in H.264 and H.265, the video frames are divided into sub-blocks during the encoding process. The intra prediction modes help the encoder to estimate which parts of the current video frame are flat. The Intra Prediction Mode Information (IPMI) is always available in the header of the video block. The header also specifies the patterns, used to perform encoding process. The IPMI is encoded through CABAC to be transformed into codewords. The codewords creation follows the same criteria as described in sections 3.1.1 and 3.1.2. A noticeable

feature in these codewords is that the encoding patterns in consecutive blocks are always same for luma and chroma components [17]. Therefore, two options are available for hiding the third part of the encrypted data. Either, the most significant zero bits are used or one of the coding pattern in consecutive blocks is used. In order to keep the same pattern, we use the most significant zeros to embed the third and last part of the encrypted data by applying the same procedure as followed before in sections 3.1.1 and 3.1.2.

3.2. Half Decryption

As stated earlier, the data owner encrypts and uploads the data on the cloud. The application system on the cloud provides three major services, i.e., Secure Storage Space (SSS), Key Generator (KG), and Half Decryption Function (HDF). The user uploads the encrypted videos on the cloud through KG, which stores the uploaded videos in the SSS. Before storing the encrypted videos in the SSS, they are decrypted by HDF. In order to run such an application system over the public clouds, few assumptions are made:

1. The main purpose behind the concept of public clouds is that they should be used freely with a trust factor. Although it is obvious that cloud owner may see the uploaded contents, it cannot change them. Therefore, it becomes a responsibility of the encryption scheme to secure the data in such environments.
2. A user plays dual roles, i.e., a user may upload and download the data. Therefore, a user may be an owner or a requester.
3. Each user posses the same SK.
4. To protect the data transmission between the user and the cloud over communication channels, it is assumed that it is protected by a security protocol, such as SSH.

These assumptions need to be followed carefully. For example, in our first assumption, if the encryption scheme is not strong enough, it is possible that the video stream may be hijacked and infected during the transmission between the

user and the cloud or a malicious user may upload an infected video to get the control of the application system, running on the cloud. The second assumption facilitates a user, so that it can download and upload videos at the same time. Possession of same SK synchronizes the authentic users. In case of multiple secret keys, the user authentication process will become complicated and the number of secret keys will increase, as more users become authenticated. To protect the transmission channel between two end points, the security protocols are always required. Without such protocols, the transmission channel cannot be trusted.

Since a public cloud cannot be trusted fully, the KG running over the cloud generates PRK and PUK. However, it cannot determine the SK of the users. As the data owner uses PUK, generated by the KG, to encrypt the video, cloud may only be able to decrypt the video by using the same PUK and its PRK. This partial decryption reduces the processing load at the receiver's end. Partial decryption is the responsibility of HDF. For proper functioning of the system, each user registers itself with KG by forwarding the user ID. The KG replies back with a PUK. The user uses this PUK to encrypt its user ID and password and sends it back to cloud's application system to request for a login session. The cloud's application system replies back with the session information. After getting the session's information, the user uses the same PUK to encrypt the video before uploading it.

3.3. Full Decryption with Data Extraction

When a user requests for a specific data, it sends a request to the KG for a login session. First, it looks up the authenticated user's list to search for a match with the user's ID. If the user's ID does not match, it simply drops the connection. If a match is found, then, first a login session is created in the similar way, as is shown in section 3.2. In the meanwhile, the KG checks whether the requested data is available in the SSS or not. If the requested data is not available, it informs the requester about the unavailability of the data, otherwise, it sends back the decrypted video to the requester. After getting the

decrypted video, user decrypts the secret data by using its SK. Upon extracting the secret data, it is up to the requester to either keep the video stream for further use or simply discard it.

There are two scenarios for secret data decryption. In the first scenario, the user is authorized one. In such scenario, the maximum probability is that it will first decrypt the secret data from video stream and then decide either to keep or discard it. In the second scenario, the user is unauthorized one, but has stolen the ID of an authorized user to access the video. In this scenario, the unauthorized user will be able to decode the video stream but will not be able to extract the hidden information without having the SK.

3.4. Proposed Algorithm

The step by step explanation of our proposed scheme is as follows.

Step 1: Registration Phase

1. Each MU_i forwards its user ID to KG function on the nominated public cloud.
2. KG authorizes the MU_i for uploading/downloading data by forwarding KG's generated PUK.
3. The data owner encrypts the user ID and password using the PUK and sends it back to cloud for login session.
4. The KG replies back with the session information.

Step 2: Encoding Phase

1. Videos are encoded using HEVC codec in Intra mode.
2. This phase results in a Binary File (BF), also known as HEVS.

Step 3: Calculation Phase

1. Total number of blocks (TBs) in the generated BF are calculated.
2. DC/zero Blocks (DCBs) and AC/non-zero blocks (ACBs) are separated from each other.

Step 4: Pre-Encryption Phase

1. The AES-256 bit key, known as SK, is generated in the same way, as is generated in the original implementation.
2. The State matrix is set up.
3. Each Mobile User (MU_i) possess the same SK, where $i = 1, 2, \cdots, n$ and MU_i is either data owner or requester.

Step 5: Encryption Phase

1. *Input*: Video frames or Binary Packets
2. The secret data is encrypted through modified AES's SK and converted into byte format.
3. 2's complement of the encrypted secret data is computed.
4. Information about the SI, MV and IPMI is extracted from the frames header.
5. The encrypted data is added with the extracted information.
6. After adjusting the encrypted data, the HEVS is encrypted by the PUK.

Step 6: Uploading Phase

1. After authorization, data owner (MU_i) alerts the KG function that data is ready for uploading.
2. The KG function acknowledges MU_i and starts receiving the data.

Step 7: Half Decryption Phase

1. This step involves cloud computing resources.
2. The encrypted video is decrypted at the cloud to reduce the user's processing load by the HDF.
3. The PRK generated by the KG is used to perform this task.
4. After decrypting the complete data file, the KG stores it in the SSS.

Step 8: Downloading/Decryption Phase

1. The MU_i is authenticated.

2. After authentication, the MU_i downloads the required video, decrypts the secret data, and/or decodes the video by using the SK and/or HEVC codec, respectively.

4. Simulation Setup and Results

In this section, we present the experimental setup and results of our proposed scheme. Our scheme is mainly based on modified version of AES-256 but we also compare our results with other versions of AES and DES. Although AES-128, AES-192 and DES-56 are also very popular, to assure better security, large-sized security keys are always preferable and difficult to crack. Table 1 shows a summarized comparison among AES, 3DES and DES techniques.

Characteristics	**AES**	**3DES**	**DES**
Length (in bits)	128, 192, and 256	112 and 168	56
Type	Symmetric	Symmetric	Symmetric
Resistance	Defensive against linear, differential, and interpolation attacks	Vulnerable against differential attacks	Vulnerable against linear and differential attacks
Keys Production	2^{128}, 2^{192}, and 2^{256}	2^{112} and 2^{168}	256
Approximated Cracking Time	2^{128}: 5×10^{21} years 2^{192}: 7.5×10^{32} years 2^{256}: 10×10^{42} years	2^{112}: 800 days 2^{168}: 1200 days	2^{56}: 400 days

Table 1: Comparison between AES and DES

All our experiments are performed on Dell machine with processor Intel CoreTM i5 − 3470 CPU @ 3.20 GHz and having memory 8 GB. For encoding purpose, we use HEVC/H.265 reference software version HM 16.6 with Intra configuration settings [3]. For simulation dataset, we use standard HD videos[3]. We select video sequences, i.e., PeopleOnStreet, Kimono, Cactus, and Rush_Hour, having 2560×1600 and 1920×1080 resolutions, respectively. The frame rate of the test videos varies from one video to another. In Table 2, the details of test sequences used during the experiments are shown. As shown in the table, the test sequences have different frame rates. Similarly, these sequences contain multiple objects which are moving with different velocities. For testing purpose, we encode first 150 frames from each video sequence. Default Quantization Parameter (QP) value, i.e. 32, is being followed while encoding the videos. The Group of Pictures (GOP) is set to 1 because of Intra coding mode. The proposed technique is applied to Intra-frames, i.e., I-frames, in unsliced mode. In case of I-frames, the compression is always low. As a result, we have enough space available in video frames to hide the data. This techniques can also be applied to Inter-frames (P or B-frames), both in sliced and unsliced modes, but it increases computational time of encryption and decryption and network latency because Inter-frames are always dependent on I-frames. Another major issue is that if an I-frame is lost during transmission due to any reason, all dependent Inter-frames will also get dropped.

For secret dataset, we use standard test images[4]. We select color images in bitmap format, i.e., barbara, baboon, lenna, pepper, airplane, BoatsColor, and goldhill. The resolution of the test images varies from one image to another. In Table 3, the details of test images used during the experiments are shown. As shown in the table, the test images have different resolutions.

The encryption algorithm is implemented in Matlab version R2015a. For cloud and cloud-based applications, we use the same version of Matlab for

[3]https://media.xiph.org/video/derf/

[4]http://www.hlevkin.com/TestImages/classic.htm/.

Test Sequence	Total Frames	Encoded Frames	Frame Rate	Resolution
PeopleOnStreet	150	150	30	2560×1600
Kimono	240	150	24	1920×1080
Cactus	500	150	50	1920×1080
Rush_Hour	500	150	25	1920×1080

Table 2: Details of Test Video Sequences

Test Image	Resolution
barbara	720×576
baboon	500×480
airplane	512×512
BoatsColor	787×576
goldhill	720×576
lenna	512×512
pepper	512×512

Table 3: Details of Test Images

implementation. The current version of Matlab provides Amazon-EC2 cloud facility with an ease for the researchers to use built-in functions to perform cloud-based computing. As our aim is real-time processing, we mainly focus on processing time and size of the test video sequences. The computational time and change of size of the encoded videos are also noted down after applying our proposed scheme, AES-256, AES-192, AES-128, and DES. In Table 4, various test sequences are compared in terms of bit rate, size after encoding and before encryption, and encoding time. These parameters form the base for Table 5, where we have compared our proposed scheme with the existing schemes in terms of size (in MB) for various test sequences. In Table 5, unlike the existing encryption schemes, the test sequences perform efficiently and require less

Test Sequence	Bit Rate (Mbps)	Size after Encoding and Before Encryption (MB)	Encoding Time (Minutes)
PeopleOnStreet	121.22	45.71	380.19
Kimono	23.41	18.29	168.56
Cactus	156.26	28.56	215
Rush_Hour	64.92	30.72	203.77

Table 4: Comparison of Various Test Sequences

Test Sequence	DES	AES-128	AES-192	AES-256	Proposed Scheme
PeopleOnStreet	49.36	50.14	50.15	50.24	45.86
Kimono	20.48	18.73	18.73	18.74	18.72
Cactus	31.99	36.56	36.56	36.57	28.57
Rush_Hour	30.96	16.46	16.47	16.47	30.76

Table 5: Size (in MB) Comparison of various Test Sequences

memory using our proposed scheme.

The average computational time required to apply the above mentioned encryption techniques is shown in Table 6. Each test sequence requires a specific amount of time using these encryption techniques. All these test sequences, i.e., PeopleOnStreet, Kimono, Cactus, and Rush_Hour require less computational time to process, when our proposed scheme is applied. DES, on the other side, requires ample amount of time to process. In case of AES with different key sizes, the computational time increases with an increase in size of the key.

Performance graph of Figure 6 is based on the calculations of Tables 4 and 5. As shown in these two tables and Figure 6, our proposed scheme keeps almost

Test Sequence	DES	AES-128	AES-192	AES-256	Proposed Scheme
PeopleOnStreet	2.14	0.75	0.86	0.95	0.92
Kimono	0.86	0.299	0.34	0.38	0.34
Cactus	1.34	0.47	0.54	0.59	0.58
Rush_Hour	1.44	0.51	0.58	0.64	0.62

Table 6: Average Computational Time (Sec)

the same size of encoded video stream as that before the encryption, and other schemes produce significant increase in the size. Performance graph of Figure 7 is based on the calculations of Table 6. As our proposed scheme is based on AES-256, it is obvious from Table 6 and Figure 7 that it produces slightly less time as compared to AES-256 but slightly increased time as compared to AES-128 and AES-192 due to large keys.

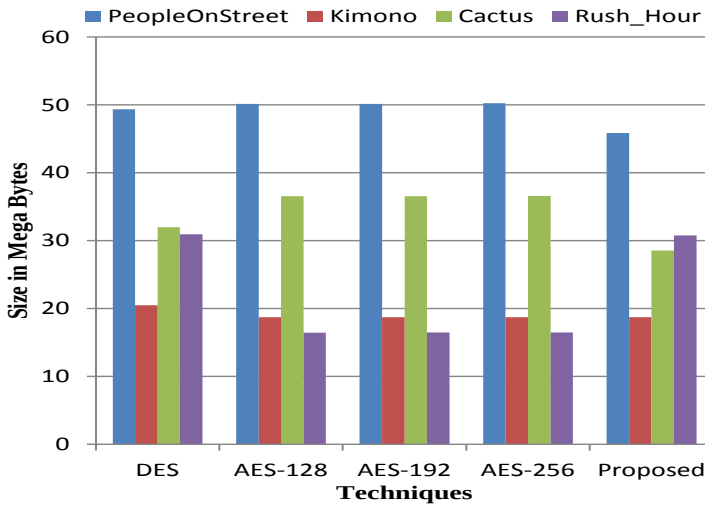


Figure 6: Size Comparison

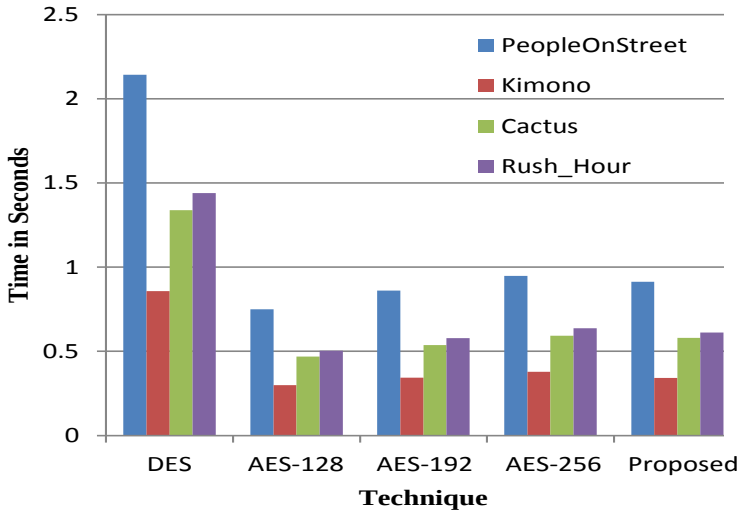


Figure 7: Time Comparison

5. Conclusion

Encrypting secret data in compressed video streams is a relatively new research area which is attracting the attention of researchers. This is primarily due to privacy and security issues concerned with the public clouds. In this article, a secured scheme has been presented which hides the secret data in HEVC encoded video stream, i.e., in compressed domain. The proposed scheme consists of three major phases, which are video encoding, data encryption, and decryption with/without decoding. The proposed scheme tries to maintain the original video stream size after encryption without affecting the visual quality of video data. Thus, it produces an ideal platform for real-time video applications.

The secret data is distributed in encoded video stream so it is difficult for hackers to extract entire secret data. This is due to the fact that hackers do not know the exact locations and patterns of the hiding scheme, even if they steal the secret key. Another major advantage is that our proposed scheme fully supports the encoding and decoding structure of HEVC standard. The video stream with encrypted secret data can easily be decoded without getting corrupted or showing any sign of extra hidden information. Experimental results

have shown that the proposed scheme maintains the visual quality with a slight compromise on increasing the size of the encoded video stream.

References

[1] M. Ali, S. U. Khan, A. V. Vasilakos, Security in cloud computing: Opportunities and challenges, Information Sciences 305 (2015) 357–383.

[2] M. N. Asghar, M. Ghanbari, An efficient security system for CABAC bin-strings of H. 264/SVC, IEEE Transactions on Circuits and Systems for Video Technology 23 (3) (2013) 425–437.

[3] F. Bossen, Common HM test conditions and software reference configurations. JCT-VC I1100, in: 9th meeting of Joint Collaborative Team on video coding (JCT-VC) of ITU-T SG16 WP3 and ISO/IEC JTC1/SC29/WG11, Geneva, 2012.

[4] B. Bross, W.-J. Han, J.-R. Ohm, G. J. Sullivan, T. Wiegand, High efficiency video coding (HEVC) text specification draft 8, JCTVC-J1003, July .

[5] A. Celebi, M. Kenkel, T. T. Wong, Bandwidth Enhancement of a Compact Transverse Bilateral Helical Antenna With Parasitic Element for Mobile Device Applications, IEEE Transactions on Antennas and Propagation 63 (3) (2015) 937–945.

[6] V. Chang, M. Ramachandran, Towards achieving data security with the cloud computing adoption framework, IEEE Transactions on Services Computing 9 (1) (2016) 138–151.

[7] C.-A. Chen, M. Won, R. Stoleru, G. G. Xie, Energy-efficient fault-tolerant data storage and processing in mobile cloud, IEEE Transactions on Cloud Computing 3 (1) (2015) 28–41.

[8] X. Chen, L. Jiao, W. Li, X. Fu, Efficient Multi-User Computation Offloading for Mobile-Edge Cloud Computing, IEEE/ACM Transactions on Networking .

[9] A. Corradi, M. Destro, L. Foschini, S. Kotoulas, V. Lopez, R. Montanari, Mobile Cloud Support for Semantic-enriched Speech Recognition in Social Care .

[10] J. Daemen, V. Rijmen, AES proposal: Rijndael .

[11] K. Elgazzar, P. Martin, H. Hassanein, Cloud-Assisted Computation Offloading to Support Mobile Services, IEEE Transactions on Cloud Computing .

[12] F. Guo, Y. Mu, W. Susilo, H. Hsing, D. Wong, V. Varadharajan, Optimized Identity-Based Encryption from Bilinear Pairing for Lightweight Devices, IEEE Transactions on Dependable and Secure Computing .

[13] F. Han, J. Qin, J. Hu, Secure searches in the cloud: a survey, Future Generation Computer Systems 62 (2016) 66–75.

[14] T. Jung, X.-Y. Li, Z. Wan, M. Wan, Control cloud data access privilege and anonymity with fully anonymous attribute-based encryption, IEEE Transactions on Information Forensics and Security 10 (1) (2015) 190–199.

[15] B. Juurlink, M. Alvarez-Mesa, C. C. Chi, A. Azevedo, C. Meenderinck, A. Ramirez, Understanding the application: An overview of the h. 264 standard, in: Scalable Parallel Programming Applied to H. 264/AVC Decoding, Springer, 5–15, 2012.

[16] B. Kaliski, PKCS# 1: RSA encryption version 1.5 .

[17] J. Lainema, F. Bossen, W.-J. Han, J. Min, K. Ugur, Intra coding of the HEVC standard, IEEE Transactions on Circuits and Systems for Video Technology 22 (12) (2012) 1792–1801.

[18] J. Liu, M. Au, X. Huang, R. Lu, J. Li, Fine-grained Two-factor Access Control for Web-based Cloud Computing Services, IEEE Transactions on Information Forensics and Security .

[19] J. K. Liu, M. H. Au, W. Susilo, K. Liang, R. Lu, B. Srinivasan, Secure sharing and searching for real-time video data in mobile cloud, IEEE Network 29 (2) (2015) 46–50.

[20] S. Ma, Identity-based encryption with outsourced equality test in cloud computing, Information Sciences 328 (2016) 389–402.

[21] D. Marpe, H. Schwarz, T. Wiegand, Context-based adaptive binary arithmetic coding in the H. 264/AVC video compression standard, IEEE Transactions on Circuits and Systems for Video Technology 13 (7) (2003) 620–636.

[22] A. Ostaszewska, R. Kłoda, Quantifying the amount of spatial and temporal information in video test sequences, in: Recent Advances in Mechatronics, Springer, 11–15, 2007.

[23] I. C. Paar, I. J. Pelzl, Introduction to Public-Key Cryptography, in: Understanding Cryptography, Springer, 149–171, 2010.

[24] U. L. Puvvadi, K. D. Benedetto, A. Patil, K.-D. Kang, Y. Park, Cost-Effective Security Support in Real-Time Video Surveillance, IEEE Transactions on Industrial Informatics 11 (6) (2015) 1457–1465.

[25] Z. Shahid, M. Chaumont, W. Puech, Fast protection of H. 264/AVC by selective encryption of CAVLC and CABAC for I and P frames, IEEE Transactions on Circuits and Systems for Video Technology 21 (5) (2011) 565–576.

[26] Z. Shahid, W. Puech, Visual protection of HEVC video by selective encryption of CABAC binstrings, ieee transactions on multimedia 16 (1) (2014) 24–36.

[27] G. J. Sullivan, J.-R. Ohm, W.-J. Han, T. Wiegand, Overview of the high efficiency video coding (HEVC) standard, IEEE Transactions on Circuits and Systems for Video Technology 22 (12) (2012) 1649–1668.

[28] G. J. Sullivan, P. N. Topiwala, A. Luthra, The H. 264/AVC advanced video coding standard: Overview and introduction to the fidelity range extensions, in: SPIE 49th Annual Meeting Optical Science and Technology, International Society for Optics and Photonics, 454–474, 2004.

[29] V. Sze, M. Budagavi, G. J. Sullivan, High Efficiency Video Coding (HEVC), in: Integrated Circuit and Systems, Algorithms and Architectures, Springer, 1–375, 2014.

[30] S. Tanwir, H. Perros, A survey of VBR video traffic models, IEEE Communications Surveys & Tutorials 15 (4) (2013) 1778–1802.

[31] J.-L. Tsai, A New Efficient Certificateless Short Signature Scheme Using Bilinear Pairings, IEEE Systems Journal, .

[32] M. Usman, X. He, K.-M. Lam, M. Xu, S. M. M. Bokhari, J. Chen, Frame Interpolation for Cloud-Based Mobile Video Streaming, IEEE Transactions on Multimedia 18 (5) (2016) 831–839.

[33] M. Usman, X. He, M. Xu, K. M. Lam, Survey of Error Concealment techniques: Research directions and open issues, in: Picture Coding Symposium (PCS), 2015, IEEE, 233–238, 2015.

[34] G. Van Wallendael, A. Boho, J. De Cock, A. Munteanu, R. Van de Walle, Encryption for high efficiency video coding with video adaptation capabilities, IEEE Transactions on Consumer Electronics 59 (3) (2013) 634–642.

[35] H. Wang, S. Wu, M. Chen, W. Wang, Security protection between users and the mobile media cloud, IEEE Communications Magazine 52 (3) (2014) 73–79.

[36] Y. Wen, X. Zhu, J. J. Rodrigues, C. W. Chen, Cloud mobile media: Reflections and outlook, IEEE Transactions on Multimedia 16 (4) (2014) 885–902.

[37] J. Wu, C. Yuen, N.-M. Cheung, J. Chen, C. W. Chen, Streaming Mobile Cloud Gaming Video over TCP with Adaptive Source-FEC Coding .

[38] C. Xiao, L. Wang, M. Zhu, W. Wang, A resource-efficient multimedia encryption scheme for embedded video sensing system based on unmanned aircraft, Journal of Network and Computer Applications 59 (2016) 117–125.

[39] D. Xu, R. Wang, Y. Q. Shi, Data hiding in encrypted H. 264/AVC video streams by codeword substitution, IEEE Transactions on Information Forensics and Security 9 (4) (2014) 596–606.

[40] D. Xu, R. Wang, Y. Q. Shi, An improved scheme for data hiding in encrypted H. 264/AVC videos, Journal of Visual Communication and Image Representation 36 (2016) 229–242.

[41] Y. Yao, W. Zhang, N. Yu, Inter-frame distortion drift analysis for reversible data hiding in encrypted H. 264/AVC video bitstreams, Signal Processing 128 (2016) 531–545.

[42] Z. Yin, F. R. Yu, S. Bu, Z. Han, Joint cloud and wireless networks operations in mobile cloud computing environments with telecom operator cloud, IEEE Transactions on Wireless Communications 14 (7) (2015) 4020–4033.

[43] J. Zhang, Q. Dong, Efficient ID-based public auditing for the outsourced data in cloud storage, Information Sciences 343 (2016) 1–14.

[44] Y. Zhang, X. Chen, J. Li, D. S. Wong, H. Li, I. You, Ensuring attribute privacy protection and fast decryption for outsourced data security in mobile cloud computing, Information Sciences .

[45] Y. Zhang, D. Niyato, P. Wang, Offloading in mobile cloudlet systems with intermittent connectivity, IEEE Transactions on Mobile Computing 14 (12) (2015) 2516–2529.

[46] J. Zhou, Z. Cao, X. Dong, N. Xiong, A. V. Vasilakos, 4S: A secure and privacy-preserving key management scheme for cloud-assisted wireless body area network in m-healthcare social networks, Information Sciences 314 (2015) 255–276.