REPUBLIC OF TURKEY

AKSARAY UNIVERSITY

GRADUATE SCHOOL OF NATURAL AND APPLIED SCIENCE

DEPARTMENT OF ELECTRICAL ELECTRONIC AND COMPUTER ENGINEERING

MODIFIED AES BASED ALGORITHM FOR MPEG-4 AND H.264 VIDEO ENCRYPTION

MASTER OF SCIENCE THESIS

FARHAD M. HAJ

SUPERVISOR

Prof.Dr. Mehmet R. TOLUN

AKSARAY,2017

REPUBLIC OF TURKEY

AKSARAY UNIVERSITY

GRADUATE SCHOOL OF NATURAL AND APPLIED SCIENCE

DEPARTMENT OF ELECTRICAL ELECTRONIC AND COMPUTER ENGINEERING

MODIFIED AES BASED ALGORITHM FOR MPEG-4 AND H.264 VIDEO ENCRYPTION

MASTER OF SCIENCE THESIS

FARHAD M. HAJ

SUPERVISOR

Prof.Dr. Mehmet R. TOLUN

AKSARAY,2017

AKSARAY UNIVERSITY GRADUATE SCHOOL OF NATURAL AND APPLIED SCIENCES

APPROVAL

FARHAD MUSTAFA HAJ, M.Sc. with thesis student No. 152335804, successfully presented the M.Sc. thesis titled *"*MODIFIED AES BASED ALGORITHM FOR MPEG-4 AND H.264 VIDEO ENCRYPTION*",* prepared after satisfying all the requirements identified in the concerned bylaws, before the jury whose signatures are affixed below.

Thesis Advisor: Prof. Dr. Mehmet R. TOLUN **…….**
Aksaray University

Jury Member: Asst.Prof.Dr. Özlem AKGÜN **.....**
Aksaray University

Jury Member: Asst.Prof.Dr. Abdül Kadir GÖRÜR **.....**
Çankaya University

Date of Submission: 21 July 2017

ate of Defence: 14 September 2017

DECLARATION

The idea for this present study was taken from the literature regarding the topics. I here by advert that the whole work offered in this thesis has been composed and originated by myself unless otherwise specified. The study was conducted completely under the supervision of Professor Dr. Mehmet R. TOLUN.

FARHAD MUSTAFA HAJ

PREFACE

This thesis was prepared as a fulfillment of the degree of master science; the study focused on modifying AES in video encryption give more attention the objectives of Security analysis, PSNR analysis, Encryption Time analysis. The thesis proposed and analysed the performance of AES security enhancement algorithm for MPEG-4 and H.264/AVC videos.

ACKNOWLEDGEMENT

In the name of Allah, the Most Compassionate and Most Merciful. I thank the Almighty for His blessings throughout my research period.

First of all, the ideal thesis advisor Prof. Dr. Mehmet R. Tolun I would like to make a special mention. without his wisdom of the thesis project and patience this study would not be completed.

Many thanks are extended to my family especially my parents, my wife, all my children, my brothers and my sister for providing me all support that I need it and helping me in all possible way.

I implore to express appreciation to Dr. Shavan Askar , Sardar O. Salih and Nashwan S. Fares for them guidance, support and help.

With profound feeling of thank, I want to spacial my hearty thanks to the head of Duhok Industry School and all of my friends in computer department.

TABLE OF CONTENTS

ABSTRACT

Modified AES Based Algorithm for MPEG-4 and H.264 Video Encryption

Nowadays data security has become the most global issue after global warming. Expectations about a world war in the field of data seem on the horizon.
The internet connects all the people through many ways, such as e-mails or voice massages or even video calls. However, the flow of data between the source and destination which is the basic operation of the Internet, can by interrupted by another user or intruder. In order to prevent this intervention there are security algorithms meant to encrypt all the bits from starting point of flow up to the end point. The most widely used flow of data in the Internet is video streaming which takes about 80% of the data flow. In this thesis AES algorithm is used to encrypt the video streaming data. The present research aims to maintain a secure path for a video between the source and the destination by modifying and optimizing the existing AES data securtity algorithm. Amongst the analysed parameters are, PSNR, encryption time, and overhead bit through encryption with compression.

Keywords : H.264/AVC, Encryption/Decryption, Video Encryption, Advanced Encryption Standard (AES), Video Compression, JM19.0 Software Refrences, CABAC.

ÖZET

MPEG-4 ve H.264 Video Şifrelemesi için AES Tabanlı Algoritma

Günümüzde veri güvenliği, küresel ısınmadan sonra en küresel meseledir. Veri alanında bir dünya savaşıyla ilgili beklentiler ufukta görünüyor.
İnternet, e-postalar veya sesli masajlar veya hatta video görüşmeleri gibi birçok yolu kullanarak tüm insanları birbirine bağlar. Bununla birlikte, İnternetin temel işlemi olan kaynak ve hedef arasındaki veri akışı, başka bir kullanıcı veya saldırgan tarafından kesilebilir. Bu müdahaleyi önlemek için, akışın başlangıç noktasından bitiş noktasına kadar tüm bitleri şifrelemek üzere güvenlik algoritmaları bulunur. İnternette en çok kullanılan veri akışı, veri akışının yaklaşık% 80'ini alan video akışıdır. Bu tez çalışmasında AES algoritması video akışı verilerini şifrelemek için kullanılmıştır. Mevcut araştırma, mevcut AES veri güvenliği algoritmasını değiştirerek ve optimize ederek, kaynak ile hedef arasında bir video için güvenli bir yol sağlamayı amaçlamaktadır. Analiz edilen parametreler arasında, PSNR, şifreleme süresi ve sıkıştırma ile şifreleme yoluyla üst bit bulununaktadir.

Anahtar Kelimeler: H.264 / AVC, Şifreleme / Şifre Çözme, Video Şifreleme, Gelişmiş Şifreleme Standardı (AES), Video Sıkıştırma, JM19.0 Yazılım Refrence, CABAC

LIST OF FIGURES

Page

LIST OF TABLE

Page

LIST OF ABBREVAITIONS

2D DWT	two-dimensional discrete wavelet transforms
AC	Arithmetic Coding
AES	Advanced encryption algorithm
AVC	Advanced Video Coding
CABAC	Context-Adaptive Binary Arithmetic Coding
CAVLC	Context-Adaptive Variable Length Coding
CIF	Common Intermediate Format
DC	Discrete Cosine
DCT	Discrete Cosine Transform
DES	Data Encryption Standard
EMP	Electromagnetic Pulse
HD	High-Definition
HDTV	High-Definition Television
HEVC	High Efficiency Video Coding
ISDN	Integrated Services Digital Network
ISO	International Organization for Standardization
ITU	International Telecommunications Union
JM	Joint Model
JVT	Joint Video Team
MB	Macroblock
MPEG	Moving Picture Expert Group
MSE	Mean Square Error
MV	Motion vector
MVEA	Motion Vector Encryption Algorithm
NAL	Network Abstraction Layer
NIST	National Institute of Standards and Technology
PPS	Picture Parameter Set
PSNR	Peak Signal to Noise Ratio
QCIF	Quarter Common Intermediate Format

QP	Quantization Parameter
RBSP	Raw Byte Sequence Payload
RGB	Red Green Blue
SODB	String of Data Bits
SPS	Sequence Parameter Set
SVC	Scalable Video Coding
VCL	Video Coding Layer
VEA	Video encryption algorithm
VOD	Video on Demand
YUV	Luminance, Chrominance blue, Chrominance red

1. PREFACE RESEARCH

1.1 Introduction

Modern advancements in computers technology and communications have brought in a huge bazar for the repartition of digital multimedia via the Internet. Actually,the multimedia are used widely in applications such as video conferencing, video-on demand,and broadcasting. this increase in digital documents, processing tools of multimedia and the global Internet connectivity have too construct a perfect stage for copyright fake and irrepressible repartition of multimedia bringing to the forefront the challenging issue of multimedia content security. Hence, Hence, the knot of affective multimedia data encipher have newly achieve rather notice in both industry and academia.

Although traditional cryptographic cryptography provides the encipher yields a satisfactory level of security, it has several shortcomings. First of all, it is usually high because of the computational cost associated with encrypting all media content. Second, encryption and decryption operations increase the level of complexity in the system. In addition, hardware or additional software application is required.. This is particularly unfavourable in applications such as mobile communications and embedded systems, where devices such as cellular phones and portable equipment's are resource constrained due to size limitation and power consumption considerations. Hence, it is desirable to develop efficient yet secure multimedia encryption techniques.

An interface been created with C# and JM19.0 software reference in visual studio for implementing selective videos to be encoding\decoding with H.264/AVC method, then be encrypted is been simulated. Encryption of the I, P, B-frames using the proposed algorithm for use with H.264 leads to an understandable bit stream, thus, shielding the video stream from intruders. I, P, B-frames encryption provides a fine exchange over encryption robustness and flexibility. It gives enough security for video stream.

1.2 Research motivation

Nowadays, the usage of multimedia data and contents is common and used every day throughout work, home and for other faces of our live. In the lack of a credible security system to keep multimedia data, Multimedia users on general networks such as the Internet may be at high risk of data losing. In this situation, an adequate security for such information is provided so that reliable services offered for different types of business transactions.

Therefore, Companies and organizations in the world are in need of extra security measurements to prevent theft and corrupt of multimedia systems. For example, a high security system is needed when head of a state army wants to alive connection with their officials and establish a dialogue of the video conferencing. Reveal secrets will be important to uncover the secrets. To avoid expose of these secrets. We need a very strong security system.

Many in November 2015, the Sony Entertainment agency NSA Discovery Your Kaspersky skin spy software data robbed in the security lab which was a year-round impugned attack victim of the cyber data output Western Digital hard drives, Seagate, Toshiba And other major manufacturers within.

1.3 Problem Statement

We will discuss in terms of security and multimedia. Therefore, the safety of the encryption system is in the mystery of the key and sub-keys instead of the encryption algorithm, because crackers attempt to return the encryption key. This requires the constraint for investigation a securer mechanism for store data. Asymmetric encryption has the function to use two different keys for a secure communication and update the key periodically. However, the main distribution process increases the run time Also, symmetric encryption has a high speed function, where no major distribution. Therefore, it is proposed to combine the advantages of symmetric encryption (like high speed) and asymmetric (like continuous updating key algorithms) (Fares and Askar, 2016).

H.264 is one type of video that is widely used in internet since, it is the standard that can be used to build mp4 file. Because internet is the most widely used utility, securing and prevention video degradation are the most important requirements in video encryption (Rana et al.,2015).

1.4 Thesis Achievement

A perfect and credible video broadcasting system must be capable of broadcasting videos on a communication medium in a scalable, effective and secure technique. The aims of this research are:

1. build credible and practical simulation from the beginning to measure standard AES processing.

2. To focus on modifying and optimizing existing cryptosystems for MPEG-4 and H.264 video in order to provide enhanced security with lower encryption time.

3. To test how cipher and decipher mechanisms can be execute for multimedia applications.

4. Propose algorithms that offer provide optimum security according to application requirements and resolve the execution of suggest algorithms to achieve multifod objectives.

1.5 Organization of the Thesis

Chapter 2 (A background theory) Referring to the significance of information security and basic goals of information security. We describe the different types of attacks. in addition, explains all video coding standard and describes evaluation criteria for video coding and security.

Chapter 3 (A literature Review) presents an exhaustive literature review on the security requirements for MPEG-4 , H.264/AVC video and AES algorithm. A wide study on the existing cryptographic schemes is used in varied applications.

Chapter 4 (Research Methodology) demonstrate The materials and methods used in this investigate. It process detailed description of the system review, system concepts, system specifications and proposed system structure.

Chapter 5 (Processing Result and Discussion) discusses the implementation and evaluation of the system.

Chapter 6 (Conclusion and future work) A summary of the research contribution and scope for further work has been presented.

2. BACKGROUND THEORY

2.1 Introduction

İn this chapter, we can provide detailed description of basic goals of information security, Cryptanalysis, Video Coding Standard and Evaluation Criteria.

2.2 Basic Goals of Information Security

The basic goals of information security, the force to connect securely between sender and receiver rely on: Confidentiality, Non-repudiation, Data integrity and Authentication.

2.2.1 Confidentiality

Privacy is almost equivalent confidentiality. This means holding has been transferred and hiding sensitive information from unauthorized access to prevent illegal listening by a third party. Hiding fact is to stop or delay free access to information is encrypted. Only authorized recipients must be able to extract the contents of the message from its encrypted form. Data confidentiality ensures that only intended recipients and prevent free access is available. The most common aspects of information security. Individuals, organizations, businesses, etc. Must provide protection against malicious actions that risk the confidentiality of information.

Sometimes maintaining the confidentiality of the data can be a special training for those who are essential to these documents. This training usually involves security risks that may threaten it. Education can help authorized people know the risk factors and how to protect against them.

multitude procedures can be used to certify the confidentiality, like account numbers for e-banking user ID and password (authentication), biometric verification, security

signs, etc. The purpose of a common method of data encryption to ensure confidential. Figure 2.1 transmitter and receiver security risk that unprotected in an environment exposed to Consider.

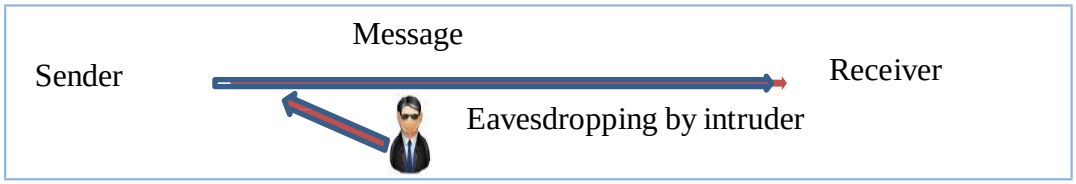


Figure 2.1: Unprotected Communication, loss of privacy

2.2.2 Integrity

Data integrity, the recipient must be able to check a message during the transition period has been changed or modified by a third party. This prevents unauthorized changes, so it can not handle a nuisance to a legitimate message. This also means that changes, it should be allowed to authorized users and mechanisms are provided. For example, the data that is constantly updated, such as online banking, where customers deposit and withdraw money, you must change accounts. The integrity of your data is not large: it does not allow the recipient to know that the message does not change, unless the sender is right. Therefore, it should always be combined with data origin authentication.

Many measures include providing specific data, providing more cryptography to verify integrity, for example, to provide integrity. Others include file permissions and user access controls. When an intruder can change a hidden message, even when it is corrupted, altering or sending, the sender and receiver in an unprotected place cannot cope with the security risks listed in Figure 2.2. If unauthorized detection is not possible, there is a lack of integrity.

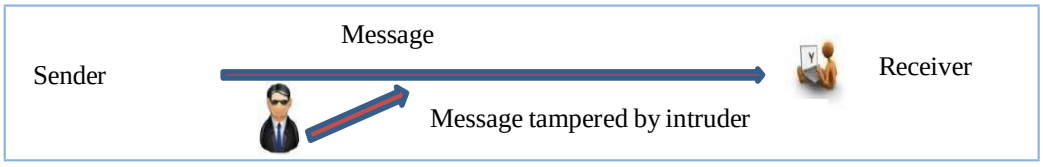


Figure 2.2: Unprotected Communication Loss of Integrity.

2.2.3 Authentication

The authentication of the data you need to be the receiver is able to check the source of the message as well as the identity of the sender to verify the sender's requests. Intrudes and prevents stuffing allegations like any other person. He said he should not pretend to be the sender of the message. In some cases, the authentication data is divided into two parts: the fact that the data had not been modified (data integrity or entity authentication), and the fact that the recipient knows the identity of the Sender authentication (the origin of the data).

Security risks in the 2.3 format, which cannot protect the environment transceivers, when the sender sends the message to the recipient should consider, but intercept the intruder and send to the recipient and the sender and the sender pretends. A variation of this theme: the intruder can send a message to the recipient that the sender is pretending. If one source of the message can not be verified by the receiver, it lacks credibility.

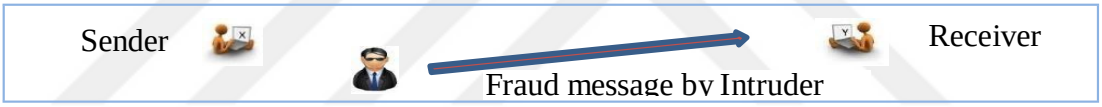


Figure 2.3: Unprotected Communication, lack of authentication.

2.2.4 Non-repudiation

Non-repudiation together confirmation of source protects against rejection via one of the entities of the other participants in a communication of having participated in all or part of the communication. Non-repudiation with proof prevents the sender Any attempt by the sender to falsely deny sending the message. At a time when non-repudiation the receiver to prevent any attempt by the recipient refused to receive the wrong.

2.3 Cryptanalysis

Cryptanalysis is a science of decrypting an enciphered plaintext in part or in whole, while the deciphering without knowing key. Consist on the quantity of recognized

information and the quantity of control upon the system via the enemy or cryptanalyst; we have four main kinds of cryptanalytic attacks.

2.3.1 Ciphertext-only attack

It is an attack that has encrypted to work with the opponent. Plain text, without the knowledge of cryptanalyseur analysis, scrambled by looking for statistical similarities between different ciphers or arrays emerging more than others. If it is a random statistic without cryptanalysis, then the opponent must use the brute-force navigational comprehensive attack.

2.3.2 Brute force attack

This type of attacks depend on tough any possible key series on the chosen ciphertext upto an understandable interpretation of the cipher text into plaintext is done, so we can also be considered text as a kind of cipher text. These well-designed cryptographic systems should be mathematically out of the equation. For example, the AES key is considered safe against today's attacks with a 128-bit size.

2.3.3 Known Plaintext Attack

The attacker knows at least one sample of both the plaintext and the cipher text. In most cases, it records real communication. When using XOR example, this will detect the key as Plaintext XOR Cipher text This one can help you identify the key or part of the key.

2.3.4 Chosen-cipher text attack

cryptanalysis attack selected text code is a template to analyze can collect information through obtaining the Plaintext of selected cipher text. In the attack, enemy has opportunity to enter one or more known cipher texts into the system and get the resulting plaintexts from these parts of information the enemy can try to recover the hidden secret key used for decryption.

2.4 Video Coding Standard

In video compression, there are several international bodies composed of academic and industrial researchers whose main goal is to develop video compression standards as efficient as possible to meet the growing needs of multimedia.

- ITU-T VCEG (International Telecommunications Union Coding Expert Group): an organization that has developed a series of video compression standards such as H261, H263.
- ISO / IEC (International Organization for Standardization / International Electrotechnical Commission) is an international organization whose best-known group is the Moving Experts Group (MPEG). Founded in 1988 to develop video compression standards, this group has developed MPEG1, MPEG2 and MPEG4 standards.
- JVT (Joint Video Team) which is none other than the association of the first two groups that created the famous H264 / AVC. Note that there are six different names for this standard: H.264, H.26L, ISO / IEC 14496-10, JVT, MPEG-4 AVC and MPEG-4 Part 10.

video compression technologies have evolved in the series of MPEG-1, MPEG-2, MPEG3, MPEG-4, MPEG7 et MPEG21, H.261, H.263, H.264 and HEVC (H.265).

2.4.1 MPEG-1

Finalized in 1992, MPEG1 is the first standard MPEG group to compress a digital video. The result is more than 5 publications, which form the MPEG1. In 1993, the first three parts were accepted by ISO, and dealt with binary streams of video (Part1), audio (Part2) and their multiplexing (Part3). Part 4 (1995) describes a test platform to verify compatibility on all media, and Part 5 (1998) is a reference implementation of the algorithms. The MPEG1 uses a resolution of 352 pixels by 240 pixels, or 84,480 pixels. Its typical bit rate is of the order of 1.5Mbps (Sikora, 1997).

2.4.2 MPEG-2

The limitations of MPEG1 were soon apparent. Its limited resolution and low quality did not make it a standard that could last for a long time in the face of the rapid

evolution of computer and digital media. MPEG2 then appeared and was finalized in November 1994 by introducing a wide range of choices regarding resolution and bit rate control. Initially developed for the transmission of TV quality video having a throughput of between 4 and 9Mb / s. The standard now allows the progressive or interlaced compression of video at speeds ranging from 1.5Mb / s to 100Mb / s (Haskell et al, 1996).

2.4.3 MPEG-3

The MPEG3 was originally intended for very high bit rates but did not live as a whole since it was assimilated to the MPEG2 standard.

2.4.4 MPEG-4

As early as 1995, the MPEG4 standard began to emerge from the theoretical point of view with the aim of achieving a bit rate of less than 64 kbit / s. The first official document was written in early 1998, it is ISO 14496. The aim of the MPPEG4 is to incorporate the existing MPEG codecs and to add a new dimension allowing a much more flexible and much more efficient standard. The standard allows the encoding of a wide variety of video formats (size, resolution, frame rate) as well as the encoding of arbitrary video objects, still images and 3D synthetic objects. As a result, this standard addresses a wide range of audiovisual applications ranging from videoconferencing to audiovisual production and streaming over the Internet. MPEG-4 Visual (Part 2) was finalized in 1999. MPEG-4 AVC (Advanced Video Coding or Part 10) was developed in collaboration with ITU-T as H.264 recommendations (Richardson, 2004).

2.4.5 MPEG-7 and MPEG-21

The MPEG7 is the latest addition to the MPEG codec family, and received its first official sketch in September 2000. The MPEG7 is no longer really about video compression but mainly deals with multimedia content and interactivity. The MPEG21 is an extension of MPEG7.

2.4.6 H.261

The H.261 standard (Recommendation, 1990) is the first video standard (made in 1990). It has been developed for video telephony applications at bit rates px64 kb / s or p is between 1 and 30 for the ISDN (Integrated Services Digital Network) network. The processed image format is QCIF (144x176 pixels) and optionally CIF (288x352pixels) and Sub-QCIF (96x128). Each MB undergoes a TCD of size 8x8. The coefficients thus obtained are then quantized and then coded by a variable length coding. The prediction is done INTER image and can be improved by motion compensation and filtering. Images can be I or P coded but not B.

2.4.7 H.263

H.263 (Rijkse, 1996) is a video coding standard for very low bit rate video communication, the first version of which was adopted in 1995. It covers videoconferencing and videoconferencing applications. This standard is based on the principles set out in H.261. The formats supported by this standard are the Sub-QCIF (96 × 128 pixels) and the QCIF (144x176 pixels) and optionally the CIF (288x352 pixels), the 4CIF (576x704 pixels) and the 16CIF (1152x1408 pixels). The use of image B is possible. In addition to this basic configuration, six encoding options have been added to increase encoding performance and functionality. Version 2 of Recommendation H.263 (1998), often referred to as H.263 + uses twelve additional options and now allows to define customized image formats and frequencies. The added options improve quality and error robustness. The latest version of H.263 (2000) called H.263 ++ adds three options. In addition to the improvement in quality and compression ratio, it takes better account of real-time video transmission over a network with an unsecured quality of service (loss of packets, etc.).

2.4.8 H.264

Developed by the JVT group, the H.264 also called MPEG-4 Part 10 or AVC (Advanced Video Coding) is a high-performance standard for video compression. Compared to MPEG-2, the H.264 delivers high-definition video with only half the bit rate and even a quarter. The H.264 standard is used in applications such as Blu-ray disc players, You Tube and iTunes videos, web software such as Adobe Flash Player

and Microsoft Silverlight. H.264 is also twice as efficient as the MPEG-4 Part 2 standard for compressing natural videos. We will limit ourselves to this brief introduction of the H.264 since chapter two is entirely devoted to it.

2.4.9 HEVC (H.265)

Successor to the H.264, the H.265 video coding format is well on its way to be launched in products by 2013. In January 2013, many representatives of the telecom, computer, electronic industry and the television world have approved the publication of a draft for the High Efficiency Video Coding (HEVC) standard. It is the H.265 video encoding format that must follow the current H.264 / AVC standard. If H.264 is used for HD and Full HD video encoding and decoding, H.265 has been designed to accompany 4K (4,096 x 2,160 pixels) and Ultra HDTV or 8K (7,680 x 4,320 pixels). The tour de force is that the H.265 promises to improve the performance of the H.264 and that it would double the level of compression without compromising the quality of the image. It is said that H.265 can make 35 to 67% better than H.264 in compression.

H.265 is being developed as part of collaboration between MPEG and ITU-T. Its finalization is scheduled for the beginning of 2013 with a first adoption in the field of mobility. For TV services, this should take longer. The MPEG group is also working on a new compression format for 3D video to dispense with glasses. A technology that could be standardized in 2014. This solution can provide more than the left and right views currently offered by the 3D Blu-ray format. Users will be able to move their heads to the left or right to obtain 3D effects.

2.5 Evaluation Criteria

cryptography is the essential condition for encryption the security of multimedia content. As multimedia encryption is various by binary/text encryption, multimedia needs two kinds of security namely, cryptographic security and perceptual security. Cryptographic security refers to the security against cryptographic attacks, while perceptual security focuses to make the enciphered multimedia contents unintelligible to human perception. Perceptual security is a measure of the perceptual deformation of the encryption video with respect to the normal video. Hence an efficient encryption

scheme has to provide perceptual security. For this, the encryption scheme should be designed to meet the following objective evaluation criteria:

2.5.1 Peak Signal to Noise Ratio (PSNR).

Whatever the performance of the technical compression, there will always be distortions in the reconstructed image and video, the quality of which can be assessed objectively and subjectively. (PSNR), compression ratio (CR) and structural similarity (SSIM). The higher the PSNR value, the closer the compressed image is to the image source. Images / videos with the same PSNR values may have different perceptual qualities (Hands et al, 2004). The evaluation by the SSIM has proved its correspondence with perceived quality visually. In fact, this method involves measuring the visibility error between a reconstructed image and a reference image using a variety of human visual system (SVH) property. The value of SSIM is between 0 and 1. The higher the value of SSIM, the more the images resemble each other (Wang et al, 2004).

With the subjective evaluation, observers judge directly and visually the quality of the image / video (Ghrare et al, 2008).

The PSNR value is expressed in decibels (dB) and is given by the following equationis given by eqn. 2.1.

$$PSNR = 10\log_{10}\left(\frac{255^2}{MSE}\right) \tag{2.1}$$

Where MSE is defined as

$$MSE = \frac{1}{|A|} SSD \tag{2.2}$$

and SSD is the sum of the squared differences between the reconstructed (s) and the original (s) macro block pixels given by eqn.(2.3) and A represents the reference macroblock.

$$SSD = \sum_{(x,y)\in A} \left|s[x,y,t] - s[x,y,t]\right|^2 \tag{2.3}$$

2.5.2 Low encryption time

It is imperative that the encipher algorithms must be actives such that the operations of transmission or recieption could not be delay. Hence the encryption time must not exceed an acceptably small portion of the total computation time of compression.

3. RELATED WORK

3.1 Introduction

An extensive literature related, with video encryption for the International Telecommunications Union (ITU) and the International Standards Organisation (ISO) and known by several names: ‘H.264’, ‘MPEG-4 Part 10’ and ‘Advanced Video Coding’. A general related work on evolution of selective encryption for MPEG-4 part10 and H.264 so as to maximize video quality with low encryption time and minimum overhead bits is also presented.

3.2 Literature Review

Internet video streaming has common application in world like video on demand, video conferencing, digital television and mobile TV. One major agent to assume is that data security. The protection for video streaming can be perform in various ways via cryptography. video appear encipher and decipher again that it can not be read by authorized recipients.

At the end of 2000, the NIST after fifteen satisfied the winner of the five finalists advertised and approved as Rijndael algorithm. Algorithm designed two Belgian researchers was published in 2002, the algorithm AES Rijndael, NIST selected, because most of the criteria specified NIST technology. In addition, there are data encipher algorithms(Fares and Askar, 2016).

The correct way to encrypt each byte in the Moving Picture Experts Group (MPEG) on the naive algorithm, streaming video using standard encryption,like Advanced Encryption Standard (AES) or encryption standard Data) (Shi and Bhargava, 1998). The basic obstacle with encryption algorithms that enhancement encryption for real-time applications like Pay Per View, Video On Demand (VOD), etc. The large video algorithm is inappropriate because it is very slow, when using

triple DES. Due to cryptographic operations, increased delays and overheads will not adopt for real-time video encoding.

In 2004, Slagell explained that the pure permutation algorithm is assailable to known-plaintext attack, and therefore, its use must be carefully take into consideration. It should be famed that percipiet one I-frame of an MPEG stream is enough to decrypt the permutation list, based on Shannon's Theorem. all frames could be easily decrypted.The permutation list is evaluate out, or becomes known, (Shi and Bhargava, 1998).

In Zig-Zag permutation (Shi and Bhargava, 1998; Tang, 1997) the Zig-Zag permutation use symmetric key, the researcher proposed random permutation to encrypt DC coefficients by DES to scramble AC coefficients. the problems of the MPEG video encryption algorithm by using random permutation list instead of zigzag order within the MPEG compression process and shows that this encryption method causes a significant increase in the size of the MPEG video stream and cannot withstand the known-plaintext attack..

The design perceptual encryption proposed in (Li et al., 2007) In many applications perceptual encryption is helpful, like pay-per-view videos, pay-TV and video-on-demand (VoD). The author as well comapered this security with deblocking attacks and highlighted some measures with a known or chosen clear text attack. The suggested perceptual encryption projects ability as well be extended to supply non-perceptual encryption by merely adding a VLC encryption portion.

(Shi and Bhargava, 1998; Li et al., 2007), proposed a light-weight encryption algorithm, known as the Video Encryption Algorithm (VEA). It uses simple XOR of sign bits of the DCT coefficients of and I frame using a secret m-bit binary key. The Video Encryption Algorithm (VEA), consist of four algorithms, Algorithm I, Algorithm II (VEA), Algorithm III (MVEA), and Algorithm IV (RVEA).

In 2009, Elkilani & Abdul-Kader in (Elkilani and Abdul-Kader, 2009) have proposed the performance of AES is compared to two symmetric encryption techniques called; RC4 and XOR.

In this paper, implement AES for MPEG-4 in a real time secure video transmit system. Compared the performance of the AES with respect to two major encryption techniques over a peer to peer channel. Evaluating the difference between the overhead resulting from different data types in multimedia to the three encryption techniques (Xor, RC4, AES). Quality of server is very important in multimedia networking; we have measured this system performance based on the delay where it is visible awhile in the transmission and reception of data.

With the development of internet and multimedia applications in divided circumference. For this reason, multimedia security of the communication aspect against the continuous adding when we use of digital data transmission. Multimedia security is usually supported by a combination of methods or methods used to protect multimedia content. These methods are largely based on cryptography. In addition, using the same encryption, using the same private key again between the relying party or to protect the confidential information by converting an unreadable (encryption) other special keys (decryption) art. There are three key ciphering schemes, public key (or asymmetric) ciphering, hash functions (Abomhara, 2010) in the modern secret ciphering field.

In 2011, (Karthigaikumar and Rasheed, 2011) have proposed a technique to simulation of image encryption using AES algorithm to keep the confidential image data from unauthorized access.in this research for image encipher the Author layout the 128 bit encoder using AES Rijndael Algorithm. Optimized and Synthesizable VHDL code is developed for the implementation of 128- bit data encryption and process. AES encryption is an active projects for both software and hardware implementation. We compare to software implementation, hardware implementation prepared higher speed and greater physical security. Hardware implementation is useful in wireless security like mobile telephony and military communication where there is a greater assertion on the speed of communication.

In 2013, the authors in paper, (Shi and Bhargava, 1998; Karthigaikumar and Rasheed, 2011) The proposed algorithm was compared with the currently known methods of cryptography. The two different types of the encryption methods (Symmetric key encryption and Asymmetric key encryption) were focus on and evaluated with respect to their security level and encryption speed. The proposed encryption algorithm and

crisscross permutation algorithm are fast. Simple Permutation algorithm and the proposed video encryption algorithm are the most secure algorithms. Simple Permutation is very slow while applying DES on entire video stream. The proposed algorithm will be most suitable as it provides high level of security with a good computation speed.

In 2014, proposed the method of protect an MPEG-4 file is taked out via the encipher of data on the transmitter end and decipher of the enciphered data on the recieptors end (Pai et al,2014). The researcher comparative between video only encryption and audio only encryption and also both of video/audio encrypted with AES/DES algorithm. This method applied for protecting multimedia information hold invent to be active and effective. Major these techniques may be apply for protecting chosen the portion of audio or video data.

(Bahrami and Naderi, 2014) proposed encryption video stream algorithm A5 / 1 and W7. In this aspect, the A5 / 1 and W7 Frames for Home Video Stream ciphers encrypt individual factors used to convert DC. In addition, the method for selecting the encryption algorithm change each DCT coefficient was proposed. Based on histogram correlation and statistical tables and peak signal to noise image is encrypted. speed algorithms for video applications in real time is very important.

In 2014, Kotel and others proposed the preface of configuration devices and high-level programming language designed hardware fast encryption technology in FPGA(Kotel et al., 2014). The author's implementation and hardware platform of a real time video encryption processing. The processing encrypts videos in real time using the AES algorithm. The researcher proposed a computationally efficient architecture for AES. The system is optimized in terms of execution speed and hardware utilization. The design as know AES encryption processor is developed in Xilinx System Generator and integrated as a dedicated hardware peripheral to the Micro blaze 32-bit soft RISC processor with the EDK embedded system and implemented targeting a Spartan3A DSP 3400 device (XC3SD3400A-4FGG676C). The video encryption processing has been verified successfully.

(Deshmukh and Kolhe, 2014) The author use the process encipher of image and video use in deferent field of applications such as medical imaging,multimedia systems and

military communication.. The researcher apply comparative between standard AES and modified AES to decrease the computation of the algorithm and for enhancing the encryption effeciency. The standard AES encryption algorithm can be used effectively to encrypt. The performance of AES encryption frames is sufficient to display the received Frames on time

In 2014, Gupta and others proposed different algorithm to performance is analysed for streaming two video formats MPEG and AVI. While streaming two sample video in cloud environment are encrypted by three different techniques which are Naïve Video Encryption, Layered Video Encryption & Selective Video Encryption(Gupta et al., 2014).

Following conclusions have been deduced:

i. The selective takes the least time while naïve encryption takes the most.
ii. Selective video encryption is quicker.
iii. The stream rate in cloud increases as the data size increases.
iv. Layered Video encryption is best.

(Tayde and Siledar, 2015) the author using AES algorithm for encipher, decipher file in android phone. The smart phone have to arrive long way in terms of security. Encryption is used for security of information in transmission process and data storage. The author used advanced encryption algorithm. AES algorithm is not just for security as well as great speed. The small devices such as mobile phone can be implemented especially on different platforms. also data is split, transmitted, stored for many purpose like production, banking, research and development. The user experiment quicker file encipher and decipher. The AES encryption and decryption algorithm in android phone show run quicker.

(Abaas and Shibeeb, 2015) proposed the method by modified AES to make it more suitable for encrypting digital video. The Modification focuses on the slowest transformations in original AES which is mix columns transformations and replace them with new Henonmap chaotic based mask and one mix columns transformation. The proposed scheme provides high key space, high key sensitivity and less time for

encryption and decryption processes than original AES as well as it offers high resistance against differential and statistical attacks.

In 2015, Mishra and others New techniques to provide video data security by using the matrix affine cipher(MAC) associated with two-dimensional discrete transform (2D DWT) waves. Cryptographic encryption and proposed decoding by MAC video with the 2D-DWT algorithm is very safe and robust and suitable for video security. Introduced cryptographic security to provide two layers of RGB video data: The first layer of security provided by the security agent and the second layer structure parameters developed by Mars video encryption methods are common but this method can only succeed When the decryption key and Mac is a set of parameters. The statistical analysis of the encrypted frame RGB video display of the proposed technique offers a very high RGB video security and statistical analysis of the open RGB video frame representing a proposed cryptographic security to provide video data without loss of sensitive information. The cryptographic set is robust and adapted to ensure the delivery of video data over an insecure network (Mishra et al., 2015).

In 2015, Asghar and others, Provides a sufficient encryption scheme (SE) for H.264 (Scalable SVC) video coding to keep the efficiency of the compliancy stream compression and decoding format, without prejudice to confidentiality. SE for H.264 scalable layers with sensibly chosen codewords and bin-strings of the CAVLC and CABAC entropy coders respectively. acheivement system has been approved in sequence with a different spatial resolution, which performs the advantages of this system compared to stand by mechanism (Asghar et al.,2015). This advantages consist of :

- By encrypting partial data obtain minimal computational delay.
- By keeping the compression ratio without changing the bit rate will not be any escalation.
- Conformance with bit stream decoder will be unchanged.

(Yi et al., 2016), proposed a Fast H.264 video decoding process by altering the algorithm dealing with the Variable-length (VLC) code bit stream analysis. The author explains by using the speedup the proposed ideas for the real-time application of H.264

video encryption. In a video processing system, according to our observation, the time part of video decoding accounts more than all processing time. Here we choose an existing video encryption system as a reference to illustrate the importance of reducing the decoding time and to demonstrate the impact of our proposed fast decoding method.

In 2016, Yadav and others proposed the algorithm is hybridizing with random process. Also this algorithm is random process- Advanced Encryption Standard (RP-AES). In order, cryptography can be applied for the data safety however in state of video all cryptography algorithms are not compatible as the video encoding schema is different for different file formats. In this algorithm is (RP-AES) process to improve the security mechanism due to this random process, the random process can efficient in generating runtime passwords every time(Yadav et al., 2016).

(Memos and Psannis, 2016) proposed encryption algorithm for HEVC, It is eminent that H.265/MPEG-H or high efficiency video coding (HEVC) standard presents specific complexity and implementation and it is being integrated into multimedia systems and protocols, While model the current codec for resolutions beyond HDTV for real-time streaming of video files .Moreover, HEVC presents more effective rate–distortion(R–D) performance, using specific algorithms. This algorithm combine two algorithms proposed for previous standards and it is modified so as to be correctable to the new video compression standard. However, this suggested algorithm is more secure and effective compared to prior algorithms used.

4. RESEARCH METHODOLOGY

4.1 Introduction

The research will include two parts; first a YUV video sequence will be utilized and encoded using the H.264 encoder. The encoded video data will be fed into an encryption system using the pre-mentioned novel security algorithm and then will be decrypted . The resulting decrypted data will be decoded using the H.264 decoder and we can show the YUV format by Yuviewer application..

4.2 H.264 Stream Format

H.264 encoded data transmitted or stored as a set of packets recognized as the Network Abstraction Layer Unit. The binary stream is made and broken up into packets which are shown in Figure 4.1. NAL unit is placed in the hierarchy of syntactical structures. A NAL unit contains either parameters set, describing properties of the stream, or slices video data. In a NALs there are two parameter group, the first Sequence Parameter Set (SPS) which typically holds information about color coding and resolution, the second Picture Parameter Set (PPS) containing facts about picture encoding, partitioning of picture into slices and entropy coding. there are only one PPS and one SPS in the stream. the figure 4.1 shows the stracture of H.264/AVC.

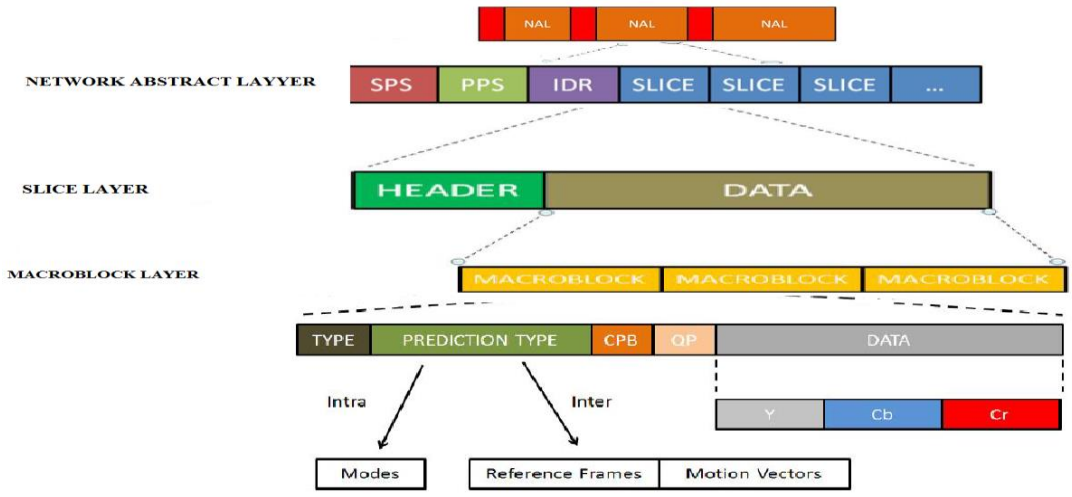


Figure 4.1: H.264 Structure

4.2.1 NAL layers

The first byte of a NAL packet is a header that holds information about the packet type. The second NAL-packet payload is known as Raw Byte Sequence Payload (RBSP). RBSP consists of a series of bits specified order of String of Data Bits (SODB). The first byte of RBSP(most is important, (far left)) and holds the eight bits SODB; next byte of RBSP shall hold the following eight SODB and so on, until there are fewer than eight bits SODB.

4.2.2 Slice Layers

We have two types of slice layer, header slice and data slice. The title of the slice transfer information corresponding to all the macroblocks in the slice, such as the type of slice which determines the type of macroblock is allowed, a match of corresponding frame numbers, setting of the reference image and Default quantization parameter (QP). the slices data consists of a series that make up the macroblock. Macroblock which contains no data, a skip macroblock, is a very common phenomenon in several coded sequence.

4.2.3 Macroblock Layers

Macroblocks are the major holder of information because they contain groups which consist of one 16×16 luminance block and two 8×8 chrominance blocks. The macroblock coding mode is transmitted. Most macroblocks I-frame are encoded in intra mode, which P-picture macroblocks can be encoded in intra or inter mode. In the B-frame of the inter-mode prediction signal for motion compensation can be formed with forecasts, backwards, or offers are interpolated. Motion compensation is usually based on a 16 x 16 blocks and the motion vector predicts from a previous motion vector encoded in the same slice.

4.3 H.264 Video Codec (encoder\decoder)

H.264 video codec It is an industry standard for video compression, digital video conversion process into a format that takes less capacity when stored or transmitted. Video compression (or video encoding) is the important technologies for apps such as digital television, DVD video, mobile TV, video conferencing and streaming video on

the Internet. Video compression uniformity allows products from different manufacturers (e.g., encoders, decoders, and storage devices) to interact. The encoder converts the video into a compressed format and the decoder converts compressed video into incompressible format.Figure 4.2 shows the process encoding and decoding and highlighting elements covered by H.264.

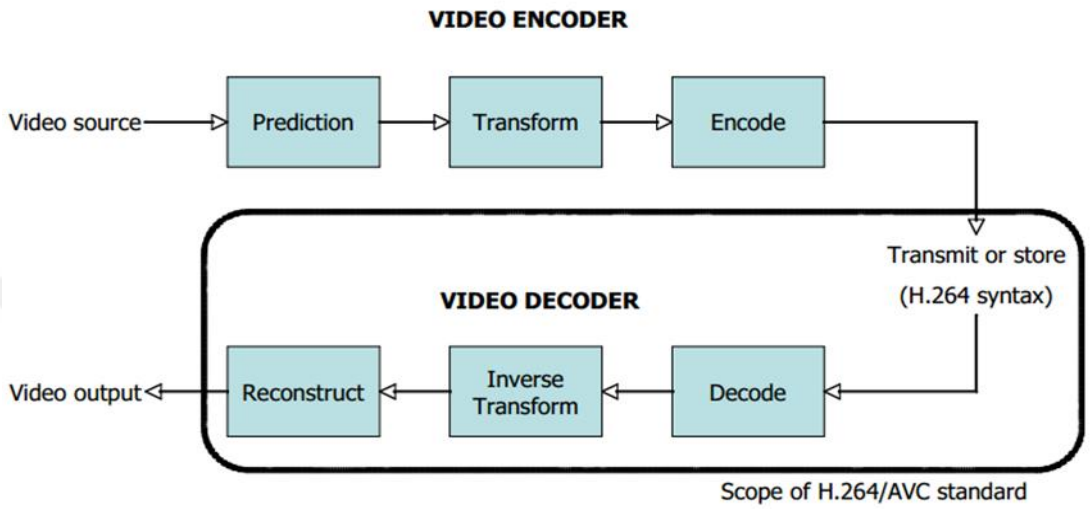


Figure 4.2: The H.264 video encoding and decoding process.

4.3.1 Encoding and Decoding Process

An H.264 video encoder implements a

prediction. Transformation and encoding processes for sending / storing generate a compressed H.264 bitstream. An H.264 video decoder implements the supplemantary operations of decoding, inverse transform, and reconstruction to predict a decoded video sequence.

The real time analog video is sampled and digitized into digital frames. All digital frames contains YUV models, wherever Y is a luminance components and UV representing chrominance components in the frame. Block coding approach divides images into macroblocks each for example 16x16 pixels. In 4: 2: 0, each MB (macroblock) consists of 16x16 = 256 Y components, and 8x8 = 64 U and 64 V, respectively.

4.3.2 Prediction

This mechanism makes it possible to establish a set of predicted values on the samples of the input signal. Generally, this estimate is based on information from the already encoded samples of the signal. The difference between the samples of the original signal and their prediction makes it possible to obtain the prediction residues which are more efficient to compress.

4.3.2.1 Intra-frame prediction (I-Frame)

Intra prediction uses 16*16 and 4*4 block sizes to predict the macroblock from the surrounding, previously-coded pixels within the same frame Figure 4.3.

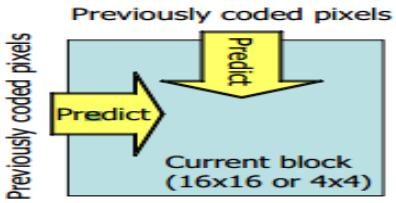


Figure 4.3: Intra frame prediction (I-frame)

The prediction is subtracted from the recent macroblock to produce a residual data by encoder Spatial redundancies are reduced by intra prediction. Intra estimation tries to predict the recent block by examining the neighboring pixels from adjacent blocks in a defined set of different directions. The variance between the actual block and the predicted block is then coded.

The nine modes of prediction are described in Figure (4.4)

Mode 0 (Vertical): Samples are predicted from the top samples by a simple vertical interpolation.

Mode 1 (horizontal): The left samples I, J, K and L are extrapolated horizontally.

Mode 2 (DC): Samples are obtained by averaging between vertical and horizontal.

Mode 3 (Down-Left Diagonal): The examples are incorporated at a 45 ° angle between the top right and bottom-left.

Mode 4 (Direct Diagonal): Examples are extrapolated at an angle of 45 ° downwards and to the right.

Mode 5 (Vertical-Left): Extrapolation at an angle of approximately 26.6 ° from the top left to the bottom right.

Mode 6 (Horizontal-Down): Extrapolation at an angle of approximately 26.6 ° below the horizontal.

Mode 7 (Vertical-Right): Interpolation at an angle of approximately 26.6 ° to the right of the vertical.

Mode 8 (Horizontal-Up): Interpolation at an angle of approximately 26.6 ° above the horizontal. There are 9 directions of intra prediction mode as shown in Figure 4.4.

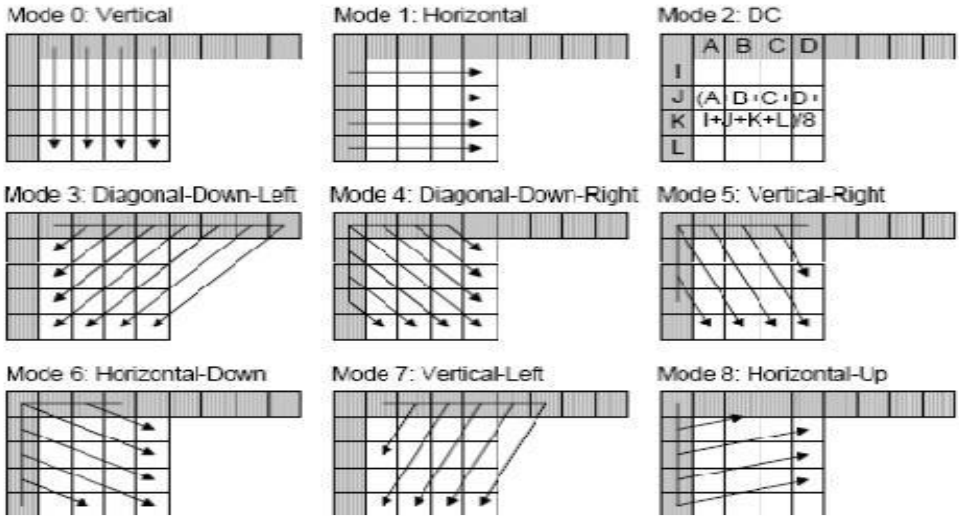


Figure 4.4: Nine modes 4x4 intra prediction H.264AVC.

4.3.2.2 Inter-frame Prediction (P-Frame and B-Frame)

Inter-prediction is based on estimating and compensating motion by taking advantage of the temporal redundancies that exist between the consecutive images. The efficiency of H.264 derives from the estimation of the motion. The standard has kept most of the methods used in previous standards and it has added flexibility and other features. There are several motion estimation and compensation techniques, the most well-known of which are block matching algorithms (BMA). These techniques are widely

used in standardized video encoders. Indeed, they guarantee the best compromise between complexity and coding efficiency, and a greater ease of implantation. Therefore, we consider only these methods. In addition, used to determine and remove the temporal redundancies that exist between individual images.

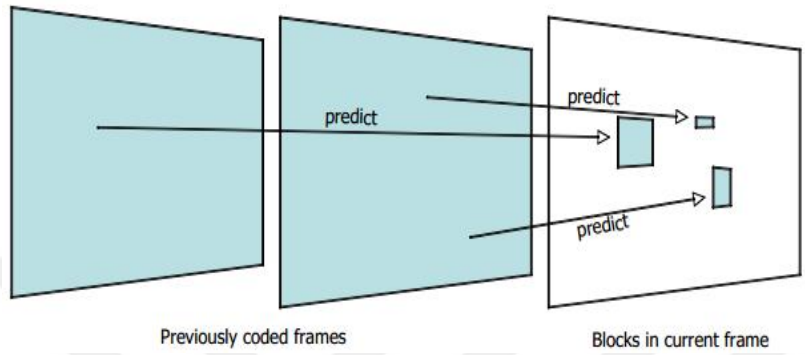


Figure 4.5: Inter-frame Prediction (P-frame and B-frame).

To enhance the efficiency of coding, the macroblock is divided into smaller blocks that attempt to contain and isolate the motion. Motion vectors do the block prediction process. H.264/MPEG-4 AVC has smaller block sizes, and motion vectors precision, block shape flexible .

4.3.3 Transform

A video sequence is input to the transform and quantization processor in 4:2:0 format, where the Y, Cb and Cr components are intra-predicted, transformed and quantized. The results which come from the intra estimation or motion estimation stages are converted from the spatial domain to the frequency domain. The purpose of the transformation step in an image or video encoder is to convert the image from one domain to another. The choice of a transformation depends on a number of criteria.

1. Data in the transformed domain must be decorrelated (separated into components with a minimum of interdependence) and compact (most of the energy of the transformed data must be concentrated in a small number of coefficients).

2. The process must be reversible.

3. The transform must have no computational complexity (requires a small memory, feasible using limited-precision arithmetic, requires a small number of arithmetic operations, etc.).

Many transformations have been proposed for the compression of images and videos. The best known can be classified into two categories: based on blocks and based on images. In the first class, there is the very popular DCT which operates on blocks of size NxN. Block-based transforms do not require large memory capacity and are well suited for compression of block-based motion-compensated residuals, but their major drawback is the appearance of artifacts at the contours of the blocks.

Image-based transformations process the entire image or the entire image (case of videos). The best known transform in this category is the DWT or simply the wavelet transform which has shown, thanks to its multi-resolution analysis capabilities, a better efficiency than the bulk transformations but has the disadvantage of being very greedy in memory.

Figure (4.6) shows the differences between transformation process of H.264 and its Predecessors.

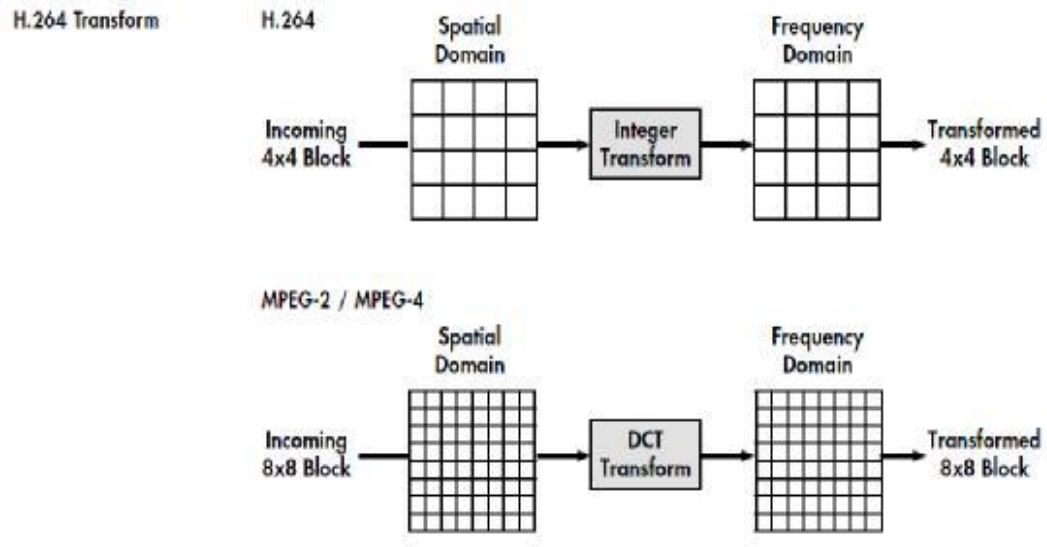


Figure 4.6: H.264 Transform.

4.3.4 Quantization

Quantification aims to discretize the amplitudes of the coefficients resulting from the whole transform. The simplest quantification is the uniform quantification which consists in dividing all the coefficients by the same value. However, it is advisable to adapt this quantizer to the values obtained after the transform. Quantization thresholding tables are then used which preserve the importance of the coefficients of the transform. It is sufficient to divide the most energetic coefficients (low frequencies) by low thresholds and the least energy (high frequency) coefficients by high thresholds. Having a wide range of quantizer values allows the encoder to control the balance between flow and quality. The QP values may be different for the luminance component and the chrominance component. The two parameters vary from 0 to 51 and, by default, QP chrominance (denoted QPc) can be determined from QP luminance (denoted QPy) so that QPC is less than QPy for QPy> 30. However, Between QPc and QPy can be defined by the user and reported in the parameters of the image Figure 4.7Shows Quantization process reduces the accuracy of the transform coefficients according to a Quantization Parameter (QP).

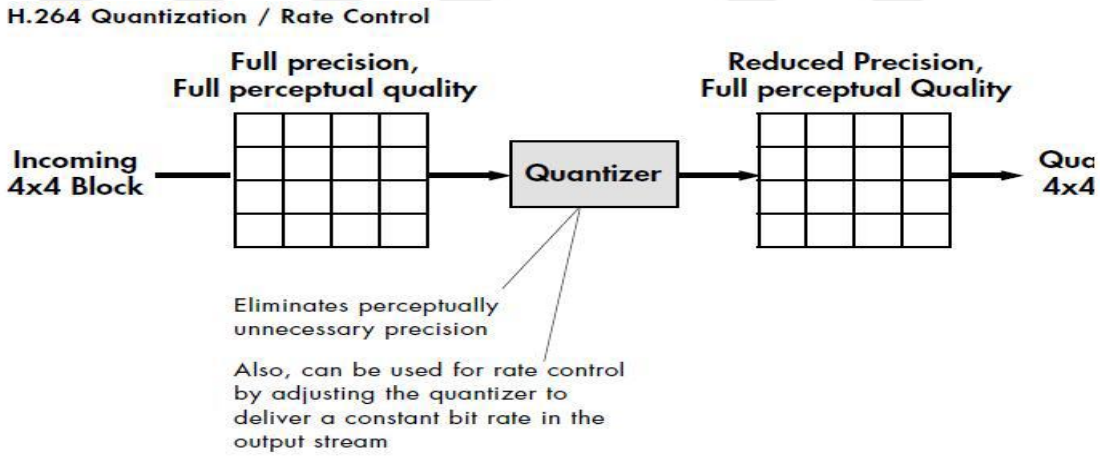


Figure 4.7: H.264 Quantization Stage.

4.3.5 Loop Filter

It is known that block processing such as prediction, transform and quantization introduces distortions in the contours of the blocks. This degrades the objective and subjective quality of the video stream. To eliminate this type of artifact, two anti-block filtering schemes have been proposed: post-filtering and filtering performed in the in-

loop filtering. The anti-block filter is applied after the inverse transform in the encoder (before reconstruction of the macroblock for future predictions), and before reconstruction of the macroblock in the decoder. The encoder may, however, disable the filtering operation. Experimental results have shown that filtering in the ball reduces the bit rate by 5 to 10% while producing the same objective quality as the unfiltered video. However, the computational complexity of the filter in the H.264 / AVC standard is very large due mainly to data access patterns and high adaptability. It easily consumes one third of the computational complexity of the decoder.

4.3.6 Entropy Coding

Entropy coding can be achieved in three different ways in the H.264 standard. A first method uses a Universal Variable Length Coding (UVLC) table. This table is used to encode most synchronization elements like headers. The other two methods are used to encode almost all other syntactic elements (coefficients, motion vectors). On the one hand, it is a Context Adaptive Variable Length Coding (CAVLC) and Context Adaptive Binary Arithmetic Coding (CABAC) coding. Compared to CAVLC, CABAC encoding generally guarantees a 10 to 15% bit rate reduction for the same image quality.

4.4 H.264 Profiles

The standard includes the following 6 sets of characteristics, which are called profiles, each targeting a class of specific applications and using a particular set of coding algorithms.

• **Baseline Profile (BP)**: Mainly for low-cost, low-resource applications, this profile is highly sought-after in mobile and videoconferencing applications.

• **Main Profile (MP)**: Originally intended for consumer and broadcast applications, this profile has lost importance when the High profile has been added for the same purpose.

• **Extended Profile (XP)**: Intended for Streaming Videos, this profile has robust capabilities for data loss and streaming.

• **High Profile (HP)**: The main profile for broadcast and disk storage, especially for high-definition television (this profile was adopted for HD DVD and Blu-ray discs as well as for high-definition French digital TV).

• **High 10 Profile (H10P)**: This profile goes beyond consumer applications and relies on the High profile, adding over 10 bits of accuracy per pixel.

• **High 4: 2: 2 Profile (H422P)**: The main profile for professional applications, it relies on the High 10 profile, adding support for 4: 2: 2 quantitation up to 10 bits per pixel.

• **High 4: 4: 4 Profile (H444P)**: This profile is based on the High 4: 2: 2 profile, adding support for 4: 4: 4 quantization, up to 12 bits per pixel and Support for an efficient lossless mode. Note: The High 4: 4: 4 profile is in the process of removing the standard in favour of a new 4: 4: 4 profile being developed.

Table 4.1: H.264 Profiles Application

Profile	Application
Baseline	Mobile applications Videoconferencing Wireless communications
Main	Television broadcasting (SDTV) Video storage
Extended Profile	Streaming Video
High	Television broadcasting (HDTV) Content distribution Studio editing Content contribution Post processing

4.5 Overview of AES Algorithm

In this chapter, the theory will be backlit for the Advanced Encryption Standard (AES) planner have an idea about the frame and performance of the algorithm that is, how it works, and the resulting.

At the beginning, a message is entered into AES scheme to conduct the enciphering process before it is sent. Because AES is a block cipher algorithm, the message is sub-divided into fixed block size of 128 bits. each block is converted of two matrix dimensions called State. In this case, the AES is able to perform diverse operations on using the algorithm procedures.

The AES system is iterative process, i.e., It contains a number of rounds utilized to state; the number of rounds depends on the key size. It is 10, 12 and 14 repetition to the key size 128-bit, 192-bit and 256-bit, respectively. Any round contains four transformations added to state values these are: Byte- Substitution, Shifting-Rows, Mixing Columns and Add Round Key. The latest round in each is exclusion in order to it does not have Mixing Columns.

The encryption key of the length (16, 24 or 32 bytes) is expanded in to a key schedule that contains a linear array of 44, 52 or 60 words of 32 bits each, respectively. Any key of the four sub-keys is utilized by AddRoundKey procedure to all rounds.

4.6 Message to Blocks Conversion

Algorithm of AES is an encrypt/decrypt stationary -size block of data of 128-bit (16 bytes) length as well. Typically, the size of messages arbitrary length, Message length from one to another could different (for example 1MB).

To transform a message into blocks of plaintext, at the beginning should be divided into a number of constant-size blocks of 128-bit length. Then, Encryption of these blocks and decryption based on the mode of operation utilized.

AES block cipher algorithm, which demand the input of messages to be an exact multiple block size of 128 bits (16 bytes). If a plaintext input that should be encrypted is not an exact multiple of the block size of 128-bit before the encryption operation, It must be supplemented by the addition of padding characters to the end of the message.

Figure 4.8 explain transmutation message to a number of blocks. Typically, the characters utilized for the stuffing are (zero (0) value, space or character x) and called phantom characters.

Figure 4.8: Message to Block Conversion.

4.7 Block to State Conversion and Vice Versa

Through encryption/decryption the blocks are needed to diversion to state and at the finish of processing it transforms back to blocks.

4.7.1 Block to State Conversion

The input block is a series of 16 bytes can be changed to an intermediate state could be represented as a matrix of bytes, consisting of four columns and four rows, called state array. Figure 4.9 shows the way of how a block is converted to a state. The first four bytes (0, 1, 2 and 3) become the first column of the state array arranged from up to down such that $S_{0,0}$ = b0, $S_{1,0}$ = b1, $S_{2,0}$ = b2 and $S_{3,0}$ = b3. The second four bytes (4, 5, 6 and 7) become the second column of the state array arranged from up to down such that $S_{0,1}$ = b4, $S_{1,1}$ = b5, $S_{2,1}$ = b6 and $S_{3,1}$ = b7. The bytes (8, 9, 10 and 11) become the third column of the state array arranged from up to down such that $S_{0,2}$ = b8, $S_{1,2}$ = b9, $S_{2,2}$ = b10 and $S_{3,2}$ = b11. The bytes (12, 13, 14 and 15) become the fourth column of the state array arranged from up to down such that $S_{0,3}$ = b12, $S_{1,3}$ = b13, $S_{2,3}$ = b14 and $S_{3,3}$ = b15.

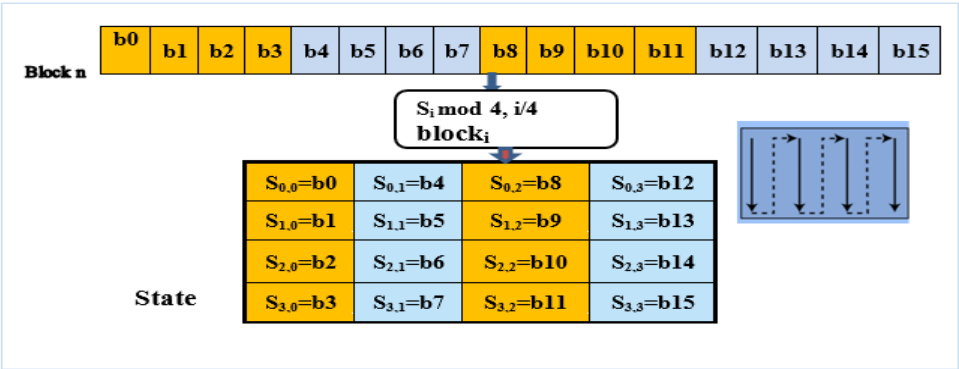


Figure 4.9: Block to State Conversion.

4.7.2 State to Block Conversion

After the encryption or decryption, then it should set an intermediate state for a series of 16 bytes can be represented as a set of a series of 16-byte returns called output block. Figure 4.10 shows the mechanism of how a state is converted to a block. The first column of the sate becomes the bytes(0,1,2 and 3) of the output block such that b0=$S_{0,0}$, b1=$S_{1,0}$, b2=$S_{2,0}$ and b3=$S_{3,0}$. The second column of the sate becomes the bytes (4,5,6 and 7) of the output block such that b4=$S_{0,1}$, b5=$S_{1,1}$, b6=$S_{2,1}$ and b7=$S_{3,1}$. The third column of the sate becomes the bytes(8,9,10 and 11) of the output block such that b8=$S_{0,2}$, b9=$S_{1,2}$, b10=$S_{2,2}$ and b11=$S_{3,2}$. The fourth column of the sate becomes the bytes(12,13,14 and 15) of the output block such that b12=$S_{0,3}$, b13=$S_{1,3}$, b14=$S_{2,3}$ and b15=$S_{3,3}$.

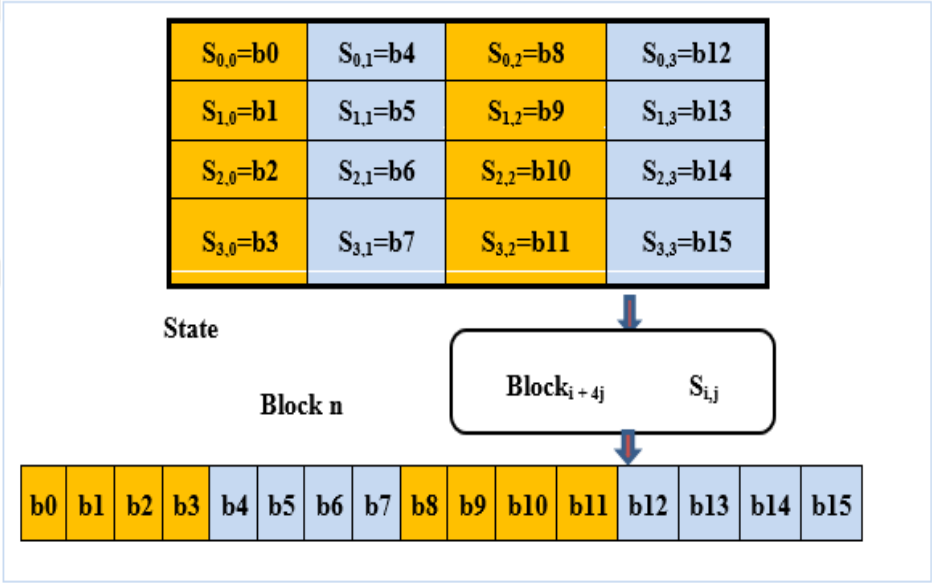


Figure 4.10: State of Block Conversion

4.8 Encryption/Decryption

In the AES algorithm, the input block length, the output block length and the medium value as it is called the State are limited to 16 bytes. The state is a matrix of 4 rows and 4 columns. This state is represented by Nb = 4, that reflects the number of 32-bit words in the state. While the cipher key length can be independently determined to 16 byte, 24 byte or 32 byte, the length of the key is represented by a matrix of 4 rows and

4, 6 or 8 columns that is represented by Nk = 4, 6, or 8, reflecting the number of 32-bit words in the Cipher Key.

The AES algorithm is repeated operations. i.e., it contains a number of rounds applied to block of data for the encrypting process. The number of rounds to be performed on the single block of data during the execution of the AES algorithm depends on the key size. For the key size with 128-bit length, the number of rounds (Nr) is 10 iterations. It is called AES_128. Whereas the key size with 192-bit length, the number of rounds (Nr) are 12 iterations. It is called AES_192. As for the key size with 256-bit length, the number of rounds (Nr) is 14 iterations. It is called AES_256.

The relation between the cipher key length, data block size and number of rounds required for each AES-type are given in Table 4.2.

Table 3.2: Relation Between Key, Data and Rounds

Key-type	Key Length (Nk words)	Block Size (Nb words)	Number of Rounds (Nr)
AES-128	**4**	**4**	**10**
AES-192	**6**	**4**	**12**
AES-256	**8**	**4**	**14**

Each round for encryption and decryption performs four transformations on blocks of data. These functions (and inverses) are described in sections 4.11.1 and 4.12 below.

4.9 Key Expansion

The procedure for a key expansion of the AES algorithm takes as input the length of the encryption key is equal to key [4*Nk] (16,24 or 32 bytes) that could be seen as array of inputs Nk-word of 32 bits each where Nk=4, 6, 8 count on key size.. And as output, it generates a key schedule that contains a linear array of 44/52/60 words of 32 bits each. The total words that the key expansion routine generates count on the key size which is Nb(Nr+1) words. Figure 4.11 shows an example of 16 bytes key, how a

Key Expansion schedule generated from a cipher key. The expansion of the encryption key into the key schedule proceeds according to the pseudo code in Figure 4.12.

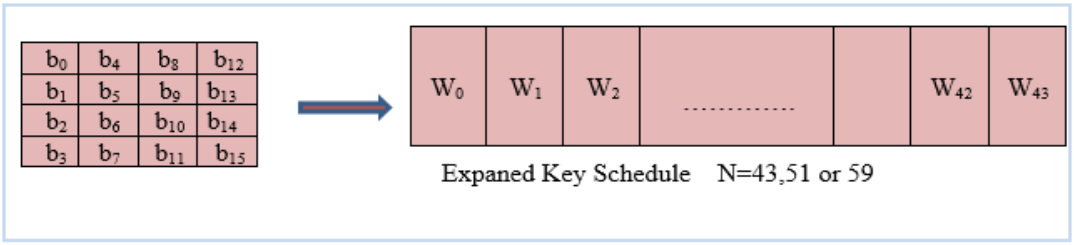


Figure 4.11: Key Expansion from Cipher Key.

```
KeyExpansion (byte key[4*Nk], word W[Nb*(Nr+1)],
Nk) begin
    word tmp ,      i = 0
    while (i < Nk)
            W[i] = word (key[4*i], key[4*i+1], key[4*i+2],
            key[4*i+3])  i = i+1
    end while
    i = Nk
    While (i < Nb * (Nr+1)]
                        tmp = W[i-1]
                        if (i mod Nk = 0)
                                tmp = SubWord(RotWord(tmp)) □Rcon[i/Nk]
                        else
                            if (Nk > 6 and i mod Nk = 4)   then   tmp = SubWord(tmp)
                        end if
        W[i] = W [i-Nk] □tmp
        i = i + 1
    end while
end
```

Figure 4.12: Pseudo Code for Expansion Procedure.

For the initial Round and for all of the Nr rounds (where Nr=10, 12 or 14). Figure 4.13 show an example of how a round key is discrete over all rounds.

Round	Words
Initial –Round	W_0 W_1 W_2 W_3
Round 1	W_4 W_5 W_6 W_7
Round 2	W_8 W_9 W_{10} W_{11}
.....	.....
Round Nr	W_{4Nr} W_{4Nr+1} W_{4Nr+2} W_{4Nr+3}

Figure 4.13: Round Key.

The key expansion scheme contains building the initial round key (4 words of 32bits) directly from the key by dividing it to four words, as shown in Figure 3.14. Then, building the 1 to Nr round keys (each round is 4 words of 32bits), while all word (W_i) in the round key is created based on the previous word (W_{i-1}) and on the word (W_{i-Nk}). Except for the words in positions that are a multiple of Nk, when a special transformation is applied to the word (W_{i-1}) by rotating W_{i-1}, its value is substituted and XOR-ed with a constant value. The output is called (t_i) which is finally XOR-ed with the word (W_{i-Nk}). See Figures 4.14 and 3.15 for key expansion. In the 256-bit key (Nr= 14), if Nk= 8 and i-4 is a multiple of Nk, then SubWord is applied to w[i-1] prior to the XOR.

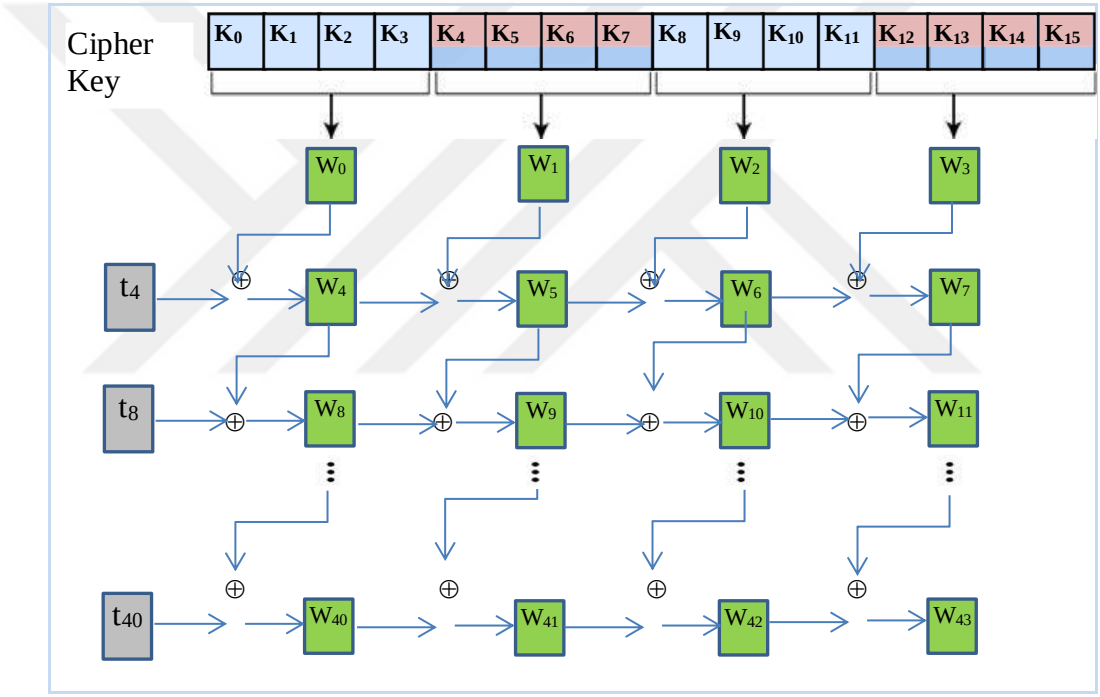


Figure 4.14: Key Expansion Construction.

The following are functions used in Key Expansion routine.

RotWord procedure is a function that performs a one-byte circular left shift on a word.

Example: RotWord[b0,b1,b2,b3] = [b1,b2,b3,b0].

SubWord procedure is a function that performs a byte substitution on each byte of its input word using the S-box table.

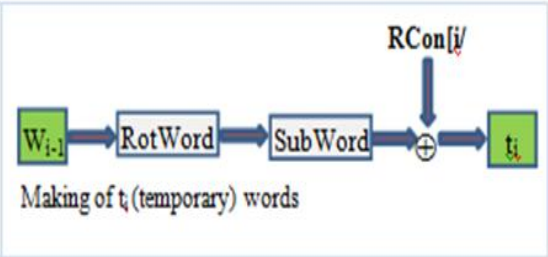
Rcon[i] is a round constant word array in which the three rightmost bytes are zero, while the leftmost byte is different for each round and defined as:

RCon[j] = (RCon[j],0,0,0)

Where RCon[1] =1 , RCon[j] = 2 * RCon[j-1] .

Figure 4.15 shows an example of constructing temporary words t_i , when i =4Nr.

t_i = **SubWord(RotWord(temp))** ⊕ RCon[j] – the round constant, Figure 4.16.



Round	Constant(RCon)	Round	Constant(RCon)
1	(01 00 00 00)16	6	(20 00 00 00)16
2	(02 00 00 00)16	7	(40 00 00 00)16
3	(04 00 00 00)16	8	(80 00 00 00)16
4	(08 00 00 00)16	9	(1B 00 00 00)16
5	(10 00 00 00)16	10	(36 00 00 00)16

Figure 4.15: Temp t_i Constraction.

Figure 4.16: Round Constant.

4.10 Cipher(Encryption)

Before entering data encryption unit, it is converted to a state of the group, as previously described. After adding the round to the state group in the first-round key, it is applied to the number of rounds for the state. Number applies to the state of rounds depends on the size of the key. This number may be 10, 12 or 14 iterations main 128,192 and 256 bits, respectively. After wards, in addition to the final round of a group of countries, which differ from the number -1 rounds because he has no job Mix columns. Finally, the state is transmitted to the output array. All cipher (Nr) rounds perform four transformation functions on the state and they are identical with the exception of the last round, because it does not include the Mix Columns transformation function: -

1) Sub Bytes (substitution).
2) Shift Rows (permutation).
3) Mix Columns (mixing).
4) Add Round Key (round key addition).

The Cipher is described in the cipher structure in Figure 4.17 and in the pseudo code in Figure 4.18.

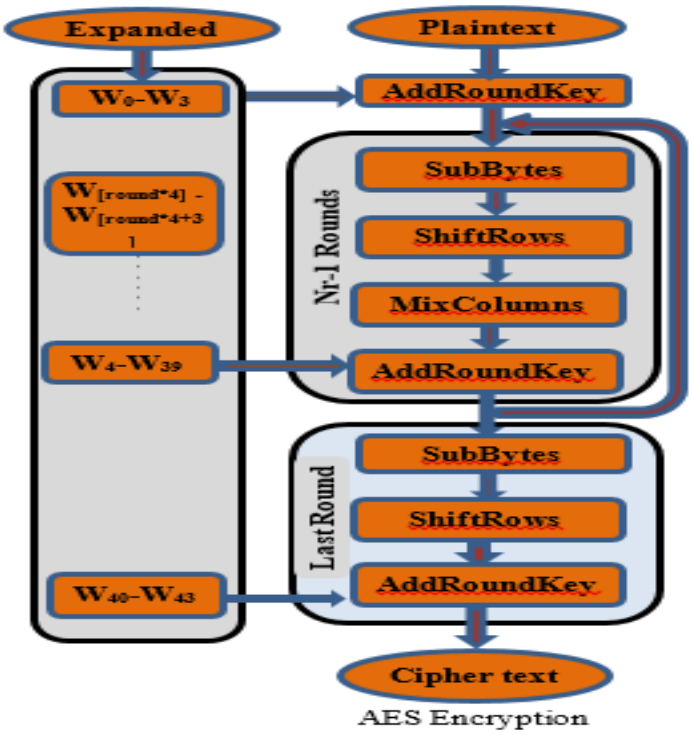


Figure 4.17: Cipher Stracture.

```
AES_Cipher(byte input[4*Nb], byte output[4*Nb], word  W[Nb*(Nr+1)])
begin
    byte state_Matrix[4,Nb]
    state_Matrix = input
    AddRoundKey(state_Matrix, W[0, Nb-1])                    // See Sec 4.10.4.
    for round = 1 step 1 to Nr–1                             // round 1 to 9,11 or 13
   SubBytes(state_Matrix)
AES_Decipher(byte input[4*Nb], byte output[4*Nb], word W[Nb*(Nr+1)])
Begin
    byte state_Matrix[4,Nb]
     state_Matrix = input
     AddRoundKey(state_Matrix, W[Nr*Nb, (Nr+1)*Nb-1])  // See Sec4.11.3
     for round = Nr-1  step -1 down to 1                      // round  9,11 or 13 down to 1
   InvShiftRows(state_Matrix)                                 // See Sec 4.11.1
            InvSubBytes(state_Matrix)                         // See Sec 411.2
   AddRoundKey(state_Matrix, W[round*Nb, (round+1)*Nb-1])
             InvMixColumns(state_Matrix)                      // See Sec 4.11.4
     end for
      InvShiftRows(state_Matrix)                              // last round 10 , 12 or 14
      InvSubBytes(state_Matrix)
     AddRoundKey(state_Matrix, W[0, Nb-1])
     output = state_Matrix
end

end
```

Figure 4.18: Pseudo Code for Cipher

4.10.1 SubBytes Transformation Function

The Sub Bytes or substitution transformation is only non-linear byte substitution that treats each byte of the state independently; by using S-Box table which consists of 16 rows indexed from 0 to F in hexadecimal values and 16 columns indexed from 0 to F in hexadecimal, this table contains values ranging from 0 to FF in hexadecimal values. The substitution operation takes input data of 8 bits, uses first nibble of byte as index to row, while the last nibble is use as index to column, then substitutes the value with intersection between row and column, as a result, the output data of 8 bits is different. The goal of this mission is to create a state of confusion. Table S-Box is the most expensive component in a complex and important AES. It is regarded as the heart of the AES algorithm. The S-box is invertible table and it is constructed by composing two transformations:

- Compute the multiplicative inverse over finite field GF (2^8).
- Apply Affine Transformation (over GF (2)).

The substitution function used the S-box; this box is presented in hexadecimal format as illustrated in Figure 4.19. And it is works as follows: -
For example, if input value S0,1 = {12}, then the new value would be determined by the intersection of the row with index '1' and the column with index '2' (see Figure 3.20). This would produce the output value S'1, 2 = {C9}.

The substitution function applies the S-box to each byte of the State. Figure 4.20 shows an example of how state values are substituted to the new values using the S-Box technique.

x \ y	0	1	2	3	4	5	6	7	8	9	a	b	c	d	e	f
0	63	7c	77	7b	f2	6b	6f	c5	30	01	67	2b	fe	d7	ab	76
1	ca	82	c9	7d	fa	59	47	f0	ad	d4	a2	af	9c	a4	72	c0
2	b7	fd	93	26	36	3f	f7	cc	34	a5	e5	f1	71	d8	31	15
3	04	c7	23	c3	18	96	05	9a	07	12	80	e2	eb	27	b2	75
4	09	83	2c	1a	1b	6e	5a	a0	52	3b	d6	b3	29	e3	2f	84
5	53	d1	00	ed	20	fc	b1	5b	6a	cb	be	39	4a	4c	58	cf
6	d0	ef	aa	fb	43	4d	33	85	45	f9	02	7f	50	3c	9f	a8
7	51	a3	40	8f	92	9d	38	f5	bc	b6	da	21	10	ff	f3	d2
8	cd	0c	13	ec	5f	97	44	17	c4	a7	7e	3d	64	5d	19	73
9	60	81	4f	dc	22	2a	90	88	46	ee	b8	14	de	5e	0b	db
a	e0	32	3a	0a	49	06	24	5c	c2	d3	ac	62	91	95	e4	79
b	e7	c8	37	6d	8d	d5	4e	a9	6c	56	f4	ea	65	7a	ae	08
c	ba	78	25	2e	1c	a6	b4	c6	e8	dd	74	1f	4b	bd	8b	8a
d	70	3e	b5	66	48	03	f6	0e	61	35	57	b9	86	c1	1d	9e
e	e1	f8	98	11	69	d9	8e	94	9b	1e	87	e9	ce	55	28	df
f	8c	a1	89	0d	bf	e6	42	68	41	99	2d	0f	b0	54	bb	16

Figure 4.19: S-Box: substitution value

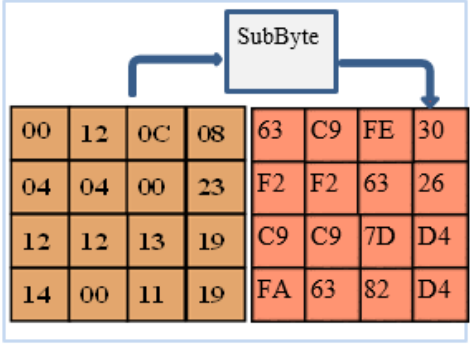


Figure 4.20: An Example of SubByte.

4.10.2 ShiftRows Transformation Function

The ShiftRows function provides a simple permutation of the data, whereas the other transformations involve substitutions. The goal of this function is to provide diffusion of values between columns to the state. This type of permutation has the effect of moving bytes to lower positions in the row, while the lowest bytes wrap around into the top of the row. See Figures 4.19 and 4.20.

This transformation processes the State by cyclically shifting each row of 0, 1, 2 and 3 of the state with different numbers of bytes (offsets) such as:-

- 1st row is unchanged
- 2nd row circular shifted 1 byte to left
- 3rd row circular shifted 2 bytes to left
- 4th row circular shifted 3 bytes to left

The shift rows operation replaces an individual byte from one column to another, which is a linear way of a multiple of 4 bytes. This diffusion ensures that the 4 bytes that belong to one column spread out over all columns in the state. See Figures 4.21 and 4.22.

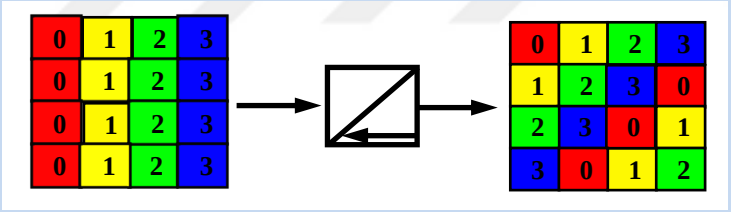
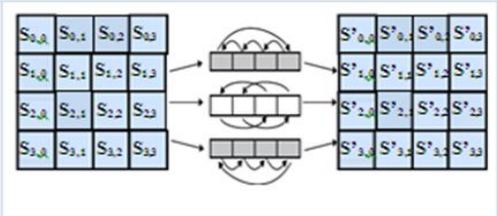


Figure 4.21: ShiftRow.

Figure 4.23 shows an example of how state values are permuted to create a new state using ShiftRow method. Each column is separated over all columns.



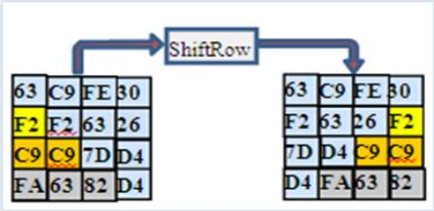


Figure 4.22: ShiftRow Transformation. **Figure 4.23**: An example of ShiftRow.

4.10.3 MixColumns Transformation Function

The MixColumns transformation function provides a simple "substitution" of the data. The purpose of this function is to provide diffusion of values between columns to the state. This leads the diffusion to the cipher values.Every byte of a column in the state is processed separately and mapped into a new value that depends on all four bytes in that column in combination with constant matrix, as shown in Figure 4.24.

It is designed as a matrix multiplication where each byte is treated as a polynomial that makes use of arithmetic over GF (2^8) using prime polynomial m(x) =$x^8+x^4+x^3+x+1$, the output is a constant matrix, as shown in Figure 4.25.

The constant matrix used is based on a linear code with maximal distance between code words – this gives good mixing of the bytes within each column.

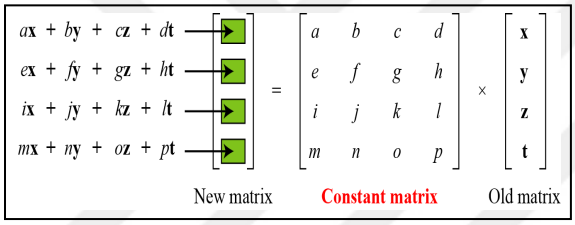


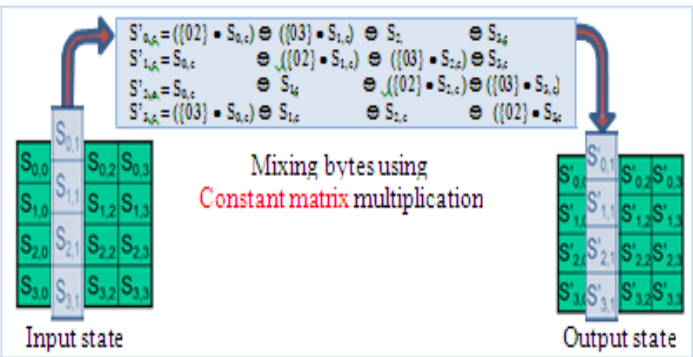
Figure 4.24: Mixing Bytes Using MatrixMultiplication.

02	03	01	01
01	02	03	01
01	01	02	03
03	01	01	02

Figure 4.25: Constant Matrix as value.

The MixColumns transformation function adds an inter-byte transformation that changes the bits inside a byte, based on the bits inside the neighboring bytes. This provides diffusion at the bit level as shown in Figure 4.26. Where dot (.) operator represent multiplication over Finite field GF (2^8).

Figure 4.27 shows an example of how state values are substituted to the new values using the MixColumns technique.



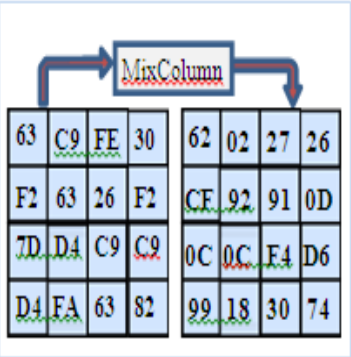


Figure 4.26: Apply MixColumn to each Column.**Figure 4.27**: An Example of MixColumn.

4.10.4 AddRoundKeyTransformation Function

The Add Round Key transformation function provides a simple "substitution" of the data. This transformation is a simple bitwise XOR operation that added a Round Key to the current block of the state, like that

$[S_{0,c},S_{1,c},S_{2,c},S_{3,c}]$ XOR $[W_{round*4+c}] = [S'_{0,c},S'_{1,c},S'_{2,c},S'_{3,c}]$

Where $0 <= c < 4$, $0 <= round <= Nr$ and [Wi]= key word

All Round Key contains of four words of 32-bit each provided by the key schedule. All of those four words of the round key XOR-ed with the columns of the State, AddRoundKey operates one column of the state at a time independently, as shown in Figure 4.28. The operation is matrix addition.

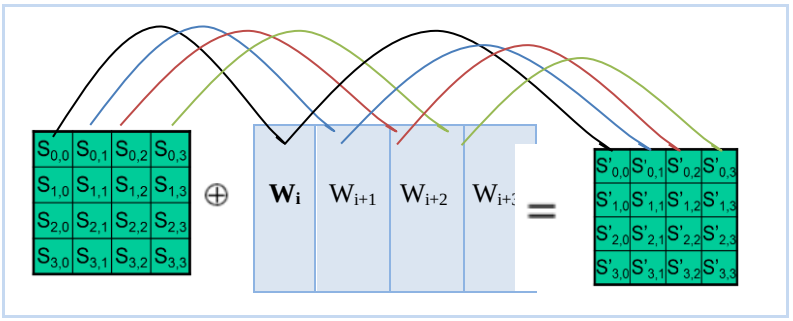


Figure 4.28: Add Round Key Function.

It is the only transformation function that uses the round key and obscures the output state. And it is an identical function which means its inverse for decryption since bitwise XOR operation is its own inverse, with reversed keys.

4.11 Decipher(Decryption)

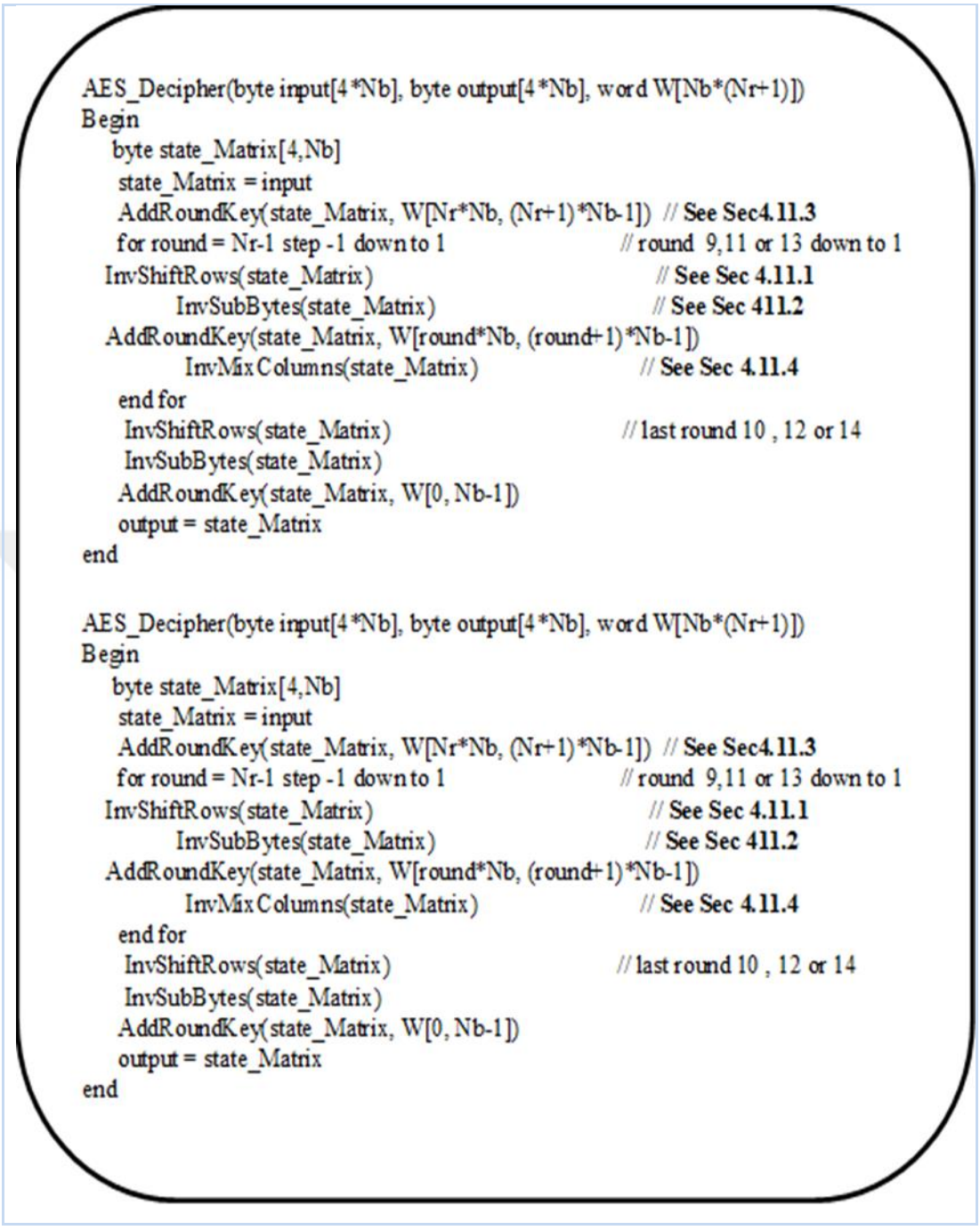
In the AES algorithm, the decryption step is not conformable to the encryption step, that means not the same transformation procedures and structures are used for encryption and decryption. However, all transformation procedures for ciphering are easily reversible and their inverse format is used in deciphering. However, in case of the cipher key, the expanded key used in reverse order in decipher algorithm.

Before decrypting the encrypted block, it converted to the state array as It formerly explained. After adding the Round key to the state array in the initial round, a number of rounds applied to the state array, the number of rounds that applied to the state depends on the key size, and can be 10, 12, or 14 iterations for the key 128,192 and 256 bits respectively just like cipher procedure. After that the final round added to the state array which is different from the Nr-1 rounds because it has not Inverse Mix Columns function. Finally, the state converted to the output array which is the plaintext.

All decipher (Nr) rounds perform four transformation functions on the state and they are identical with the exception of the last round, because it does not include the InvMixColumns transformation function, these functions are:-

1) InvShiftRows (permutation).
2) InvSubBytes (substitution).
3) Add Round Key (round key addition).
4) InvMixColumns (mixing).

The decipher (decryption) is described in the pseudo code in Figure 4.29 and the decipher (decryption) structure is described in Figure 4.30.



```
AES_Decipher(byte input[4*Nb], byte output[4*Nb], word W[Nb*(Nr+1)])
Begin
    byte state_Matrix[4,Nb]
    state_Matrix = input
    AddRoundKey(state_Matrix, W[Nr*Nb, (Nr+1)*Nb-1])  // See Sec4.11.3
    for round = Nr-1 step -1 down to 1                  // round 9,11 or 13 down to 1
  InvShiftRows(state_Matrix)                              // See Sec 4.11.1
          InvSubBytes(state_Matrix)                       // See Sec 411.2
  AddRoundKey(state_Matrix, W[round*Nb, (round+1)*Nb-1])
          InvMixColumns(state_Matrix)                     // See Sec 4.11.4
    end for
     InvShiftRows(state_Matrix)                         // last round 10 , 12 or 14
     InvSubBytes(state_Matrix)
    AddRoundKey(state_Matrix, W[0, Nb-1])
    output = state_Matrix
end

AES_Decipher(byte input[4*Nb], byte output[4*Nb], word W[Nb*(Nr+1)])
Begin
    byte state_Matrix[4,Nb]
    state_Matrix = input
    AddRoundKey(state_Matrix, W[Nr*Nb, (Nr+1)*Nb-1])  // See Sec4.11.3
    for round = Nr-1 step -1 down to 1                  // round 9,11 or 13 down to 1
  InvShiftRows(state_Matrix)                              // See Sec 4.11.1
          InvSubBytes(state_Matrix)                       // See Sec 411.2
  AddRoundKey(state_Matrix, W[round*Nb, (round+1)*Nb-1])
          InvMixColumns(state_Matrix)                     // See Sec 4.11.4
    end for
     InvShiftRows(state_Matrix)                         // last round 10 , 12 or 14
     InvSubBytes(state_Matrix)
    AddRoundKey(state_Matrix, W[0, Nb-1])
    output = state_Matrix
end
```

Figure 4.29: Pseudo Code for the Decipher.

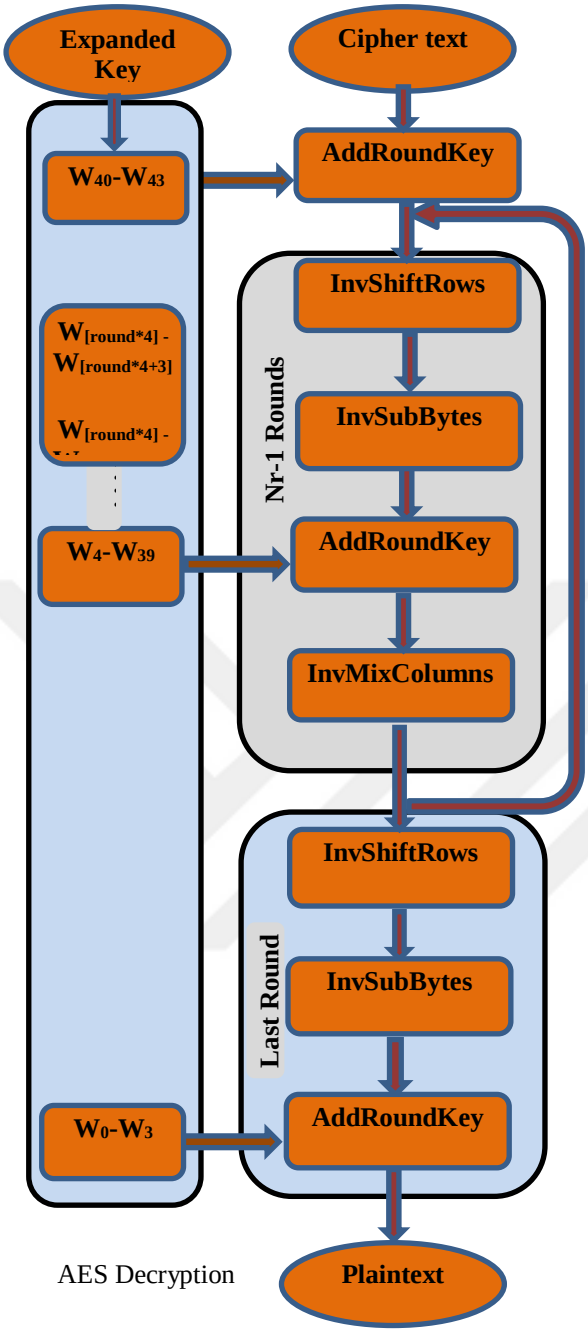


Figure 3.30: Decipher Struct.

4.11.1 InvShiftRows Transformation Function

InvShiftRows is the opposite of shiftRow, which returns the permutated bytes to their original position. The purpose of this function is to rotate diffused values between the columns of the state. Such a function has the effect of moving the byte to the top of the sequence and shifting the highest byte to the bottom of the row. See Figures 4.31 and 4.32.

This conversion cyclically shifts each line of state 0, 1, 2, and 3 to a different number of bytes (offsets):

- 1st row is unchanged
- 2nd row circular shifted 1 byte to right.
- 3rd row circular shifted 2 bytes to right.
- 4th row circular shifted 3 bytes to right.

The inverse shift rows operation replaces 4 bytes of a column that are distributed in a column of all the columns in the state (return bytes to their original position), as shown in Figures 4.31 and 4.32.

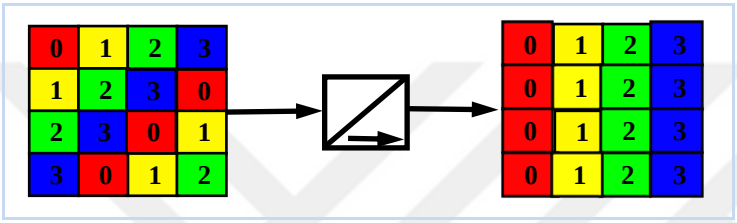


Figure 4.31: Inverse Shift Row.

Figure 3.33 Shows an example of how the permutation state value is returned to its corresponding state using the InvShiftRow technique (returning to the original values).

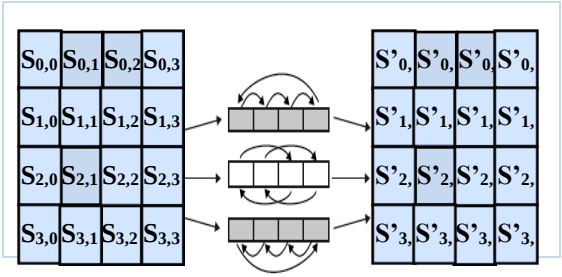


Figure 4.32: InvShiftRows.

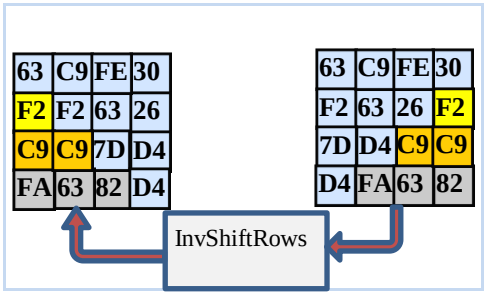


Figure 4.33: An Example of InvShiftRows.

4.11.2 InvSubBytes Transformation Function

The InvSubBytes function is the opposite of the Substitution Byte function applied to each byte of the inverted S-Box.

The InvSubBytes function is also a non-linear byte substitution that treats each byte of the state independently; by using Inverse S-Box table which also consists of 16 rows indexed from 0 to F in hexadecimal values and 16 columns indexed from 0 to F in

hexadecimal values. This table contains values ranging from 0 to FF in hexadecimal values.

The InvSubBytes operation which takes input data of 8 bits uses the first nibble as index to row, while the last nibble is used as index to column, then substitutes the current value with intersection between row and column; as a result, the output is one byte. The aim of this function is to get back the confused state.

The Inverse S-box is constructed by composing two transformations:

- Apply the inverse of Affine Transformation (over GF (2)).
- Compute the multiplicative inverse over finite field GF (2^8).

The InvSubBytes function uses the inverse S-box; this box is presented in hexadecimal format as illustrated in Figure 3.34 and it works as follows:-

For example, if input value S1,2 = {4d}, then the new value would be determined by the intersection of the row with index '4' and the column with index 'd' (see Figure (4.34). This would produce the output value S'1, 2 = {65}, this means it gets back to original value.

The function applies the Inverse S-box to each byte of the State. Figure 4.35 shows an example of how state values are substituted back to the original using the inverse S-Box technique.

x \ y	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0	52	09	6A	D5	30	36	A5	38	BF	40	A3	9E	81	F3	D7	FB
1	7C	E3	39	82	9B	2F	FF	87	34	8E	43	44	C4	DE	E9	CB
2	54	7B	94	32	A6	C2	23	3D	EE	4C	95	0B	42	FA	C3	4E
3	08	2E	A1	66	28	D9	24	B2	76	5B	A2	49	6D	8B	D1	25
4	72	F8	F6	64	86	68	98	16	D4	A4	5C	CC	5D	65	B6	92
5	6C	70	48	50	FD	ED	B9	DA	5E	15	46	57	A7	8D	9D	84
6	90	D8	AB	00	8C	BC	D3	0A	F7	E4	58	05	B8	B3	45	06
7	D0	2C	1E	8F	CA	3F	0F	02	C1	AF	BD	03	01	13	8A	6B
8	3A	91	11	41	4F	67	DC	EA	97	F2	CF	CE	F0	B4	E6	73
9	96	AC	74	22	E7	AD	35	85	E2	F9	37	E8	1C	75	DF	6E
A	47	F1	1A	71	1D	29	C5	89	6F	B7	62	0E	AA	18	BE	1B
B	FC	56	3E	4B	C6	D2	79	20	9A	DB	C0	FE	78	CD	5A	F4
C	1F	DD	A8	33	88	07	C7	31	B1	12	10	59	27	80	EC	5F
D	60	51	7F	A9	19	B5	4A	0D	2D	E5	7A	9F	93	C9	9C	EF
E	A0	E0	3B	4D	AE	2A	F5	B0	C8	EB	BB	3C	83	53	99	61
F	17	2B	04	7E	BA	77	D6	26	E1	69	14	63	55	21	0C	7D

Figure 4.34: Inverse S-Box: Value.

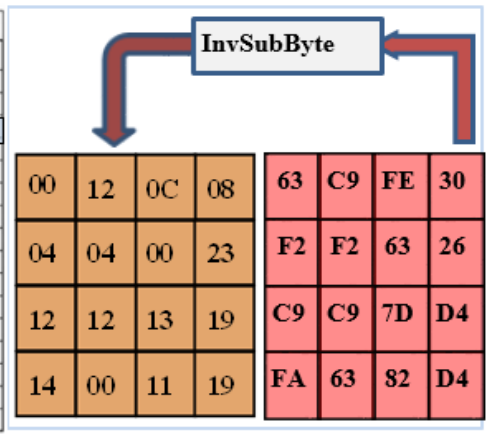


Figure 4.35: An Example of InvSubByte.

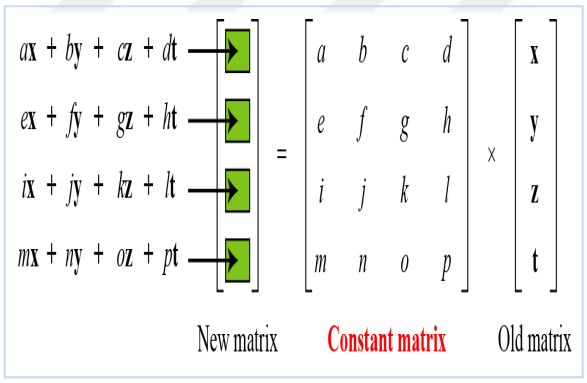
4.11.3 AddRoundKey Transformation Function

It is a conformable function, i.e., it is Special inverse, ago this function includes just the XOR process, while the XOR process is an inverse of itself; it is only a bitwise process. It is described earlier in this chapter.

4.11.4 InvMixColumns Transformation Function

The InvMixColumns function is the inverse of the Mix Columns function which returns the permutated bytes to their original position.. The purpose of this function is to return the diffused values between columns in the state. Each byte of a column in the state is processed separately and mapped back into its original value that is dependent on all four bytes in that column in combination with the inverse constant matrix, as shown in Figure 4.36.

It is designed as a matrix multiplication where each byte is treated as a fixed inverse of polynomial $a^{-1}(x)$ that makes use of arithmetic over GF (2^8), the output is a constant inverse matrix as shown in Figure 4.37. The constants inverse matrix used is based on a linear code.



0E	0B	0D	09
09	0E	0B	0D
0D	09	0E	0B
0B	0D	09	0E

Figure 4.36: Mixing Byte Using Inv Matrix **Figure 4.37**: Constant Invers Matrix.

The InvMixColumns added an inter-byte transformation that brings back the bits inside a byte, based on the bits inside the neighboring bytes. This brings back the diffusion at the bit level as shown in Figure 4.38. Where dot (.) operator: represent multiplication over GF (2^8).

Figure 4.39 shows an example of how values are substituted to the original values using the InvMixCo lumn technique.

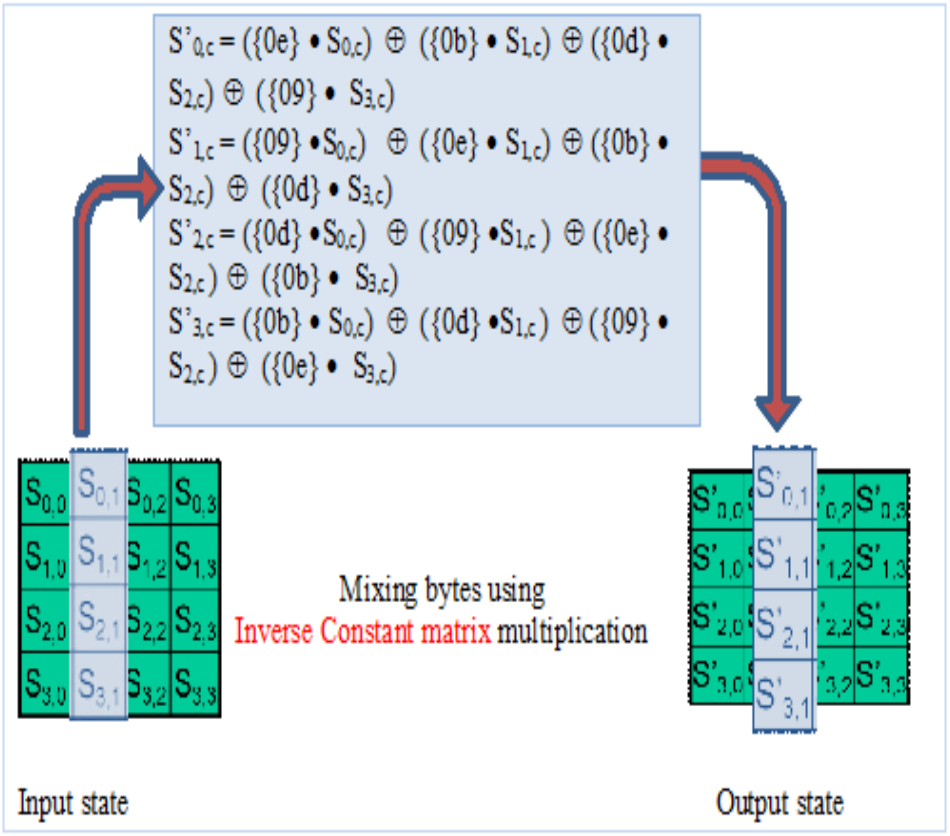


Figure 4.38: Apply InvMixColumn to each Column.

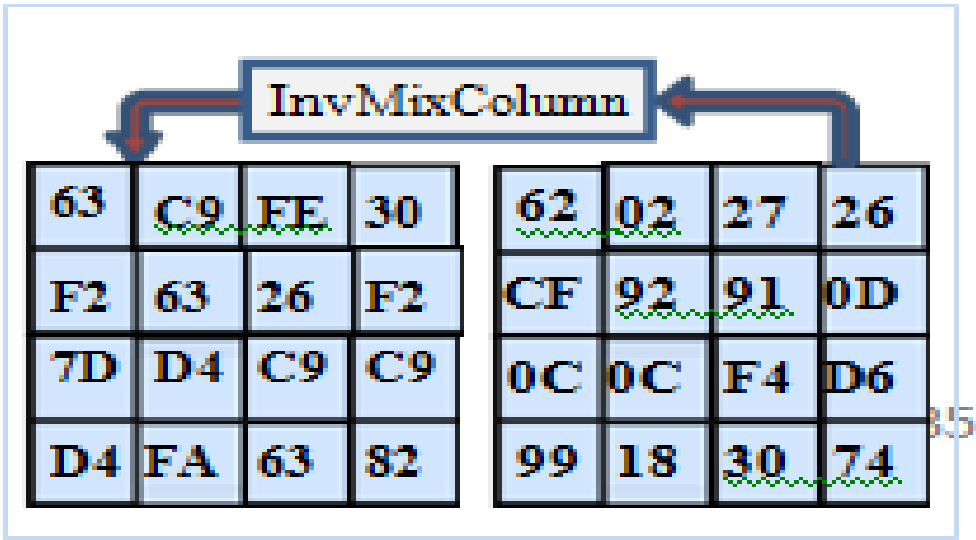


Figure 4.39: An Example of InvMixColumn.

4.12 Proposed Work

The main aims of this standardization effort are to develop a easy and direct video coding design, with enhanced compression performance, and to provide a “network-friendly” video representation which addresses “conversational” (video telephony) and “non-conversational” (storage, broadcast or streaming) applications.

Encrypting all I blocks and P, B frames increases the security but at the cost of high encryption time since the identification of I blocks from B, P frames consumes more time.

This chapter presents detailed description Real Time encryption Algorithm which consists of the follows stages;

I. H.264 analysis Algorithm: to reach I, B,P-slice, luma and chrome , 4x4 block residual data, encode and decode luma and chrome block. In this case, identification and some ideas are taken from Joint Model (JM) which represents standards for encoding and decoding for (H.264) and this idea are used to analysis video (CABAC Stage) in effective way to provide reliable results.

II. Suggested Cryptography Algorithm for video encryption: to add additional security to video encryption.

III. Bitstream Calculation (crypto image) Algorithm: to determine the number of bits will be delayed in video encryption after cryptography.

IV. Adaptively Algorithm: to ensure the embedding process is done with the least changing in video.

V. PSNR Calculation: to calculate performance between the origin and destroyed video.

For the experimental results, a computer has been used Intel (R) Celeron (R) B 820 @ 1.70GHz, 2.70GHz and 4096 Mb (4 GB) RAM in Windows 10 Pro 64-bit operating system has also been run from In order to collect data performance.

Figure 4.40 is Block Diagram explaining how these algorithms are implemented.

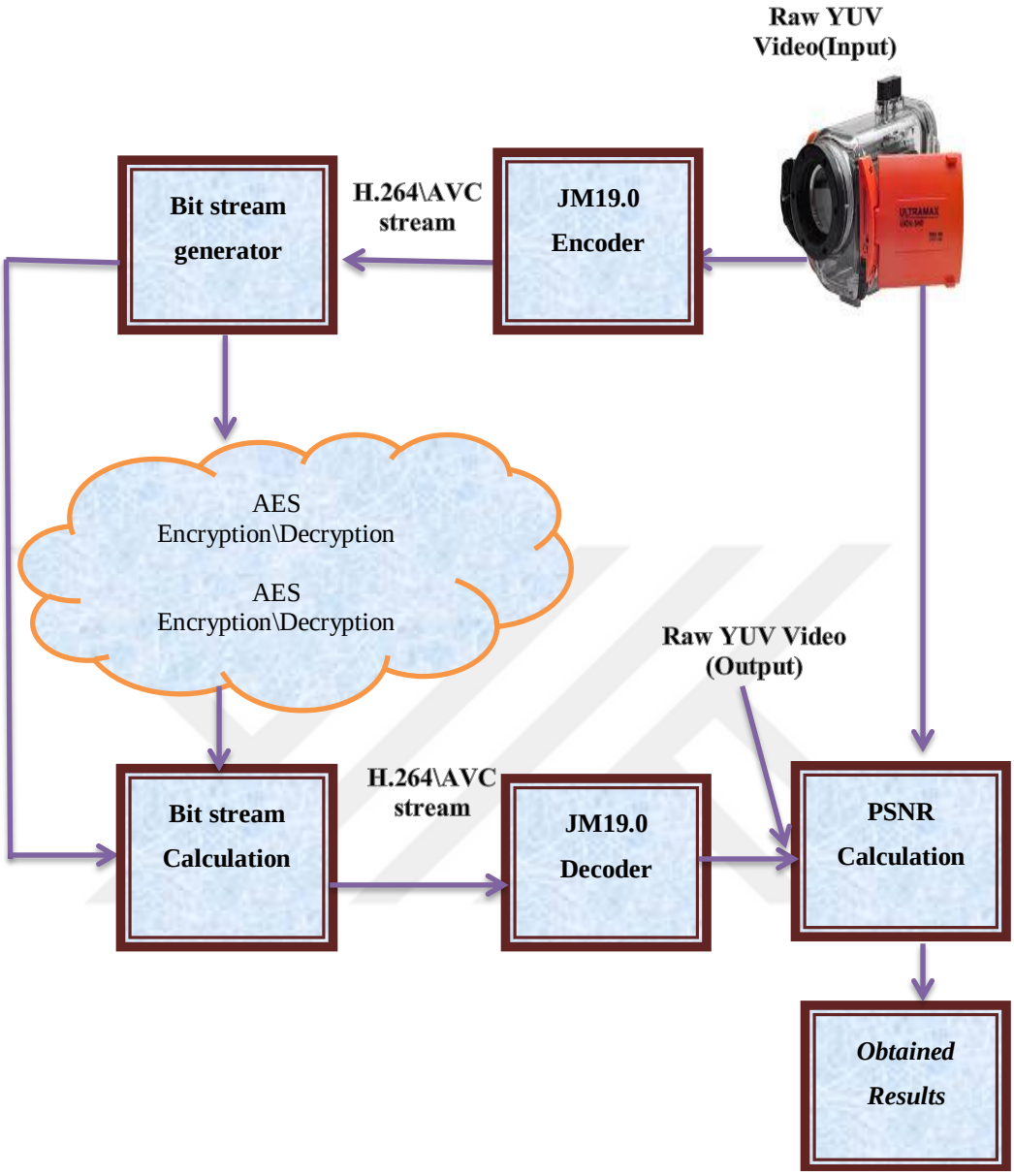


Figure 4.40: The Block Diagram of Proposed Video encryption.

1. The input video to Encode by using JM encoder.
2. The output bitstream of the JM encoder consists of input video, number of frames to be encoded, format of output, period of I frames etc.
3. The output bitstream of the JM encoder is identified by the hexadecimal value 0x65.
4. The output bitstream of the JM encoder is is fragmented into blocks of 128 bits.
5. Each block is given as input to the AES algorithm.
6. The key is obtained from the user.

7. The encoded bitstream is encrypted/decrypted using AES block cipher with key size of 128,192,256 bits.

As well as protecting video streams, the schemes of video encryption algorithm should consider real-time requirements and compression ratio. The encrypted bitstream should also prevent error propagation. In our thesis proposes a new scheme in which the H.264/AVC and the AES encryption algorithms are cAs a In general survey and design in detail proposed method show in Figure 3.41 and 3.42. While encryption is very important to know very well and the directly impact on the compression ratio and security. This encryption algorithm proposed system is as follows:

The fundamental concept of the design is to encrypt I,P,B-Frames bitstream of the H.264/AVC encoder of joint model (JM19.0) software using AES in block cipher mode with a key of 128,192,256 bits. Figure 4.41 presents the main encoding/encryption process. Before encryption algorithm we can encoding process on the H.264/AVC encoder of joint model (JM19.0) software frame by frame , The encoder creates a reconstructed video file, a coded video file(bitstream), and can optionally generate a trace file that records every syntax element of the coded sequence. After encoding the all frame or partial frame of video , We can encoded bitstream is encrypted/decrypted using AES block cipher with key size of 128,192,256 bits . The encrypted bitstream is then written to the encoder data file. The inverse operations are the decryption process of decoding the encoded frames back to their original form. Figure 4.42 represents the main decryption/decoding process.

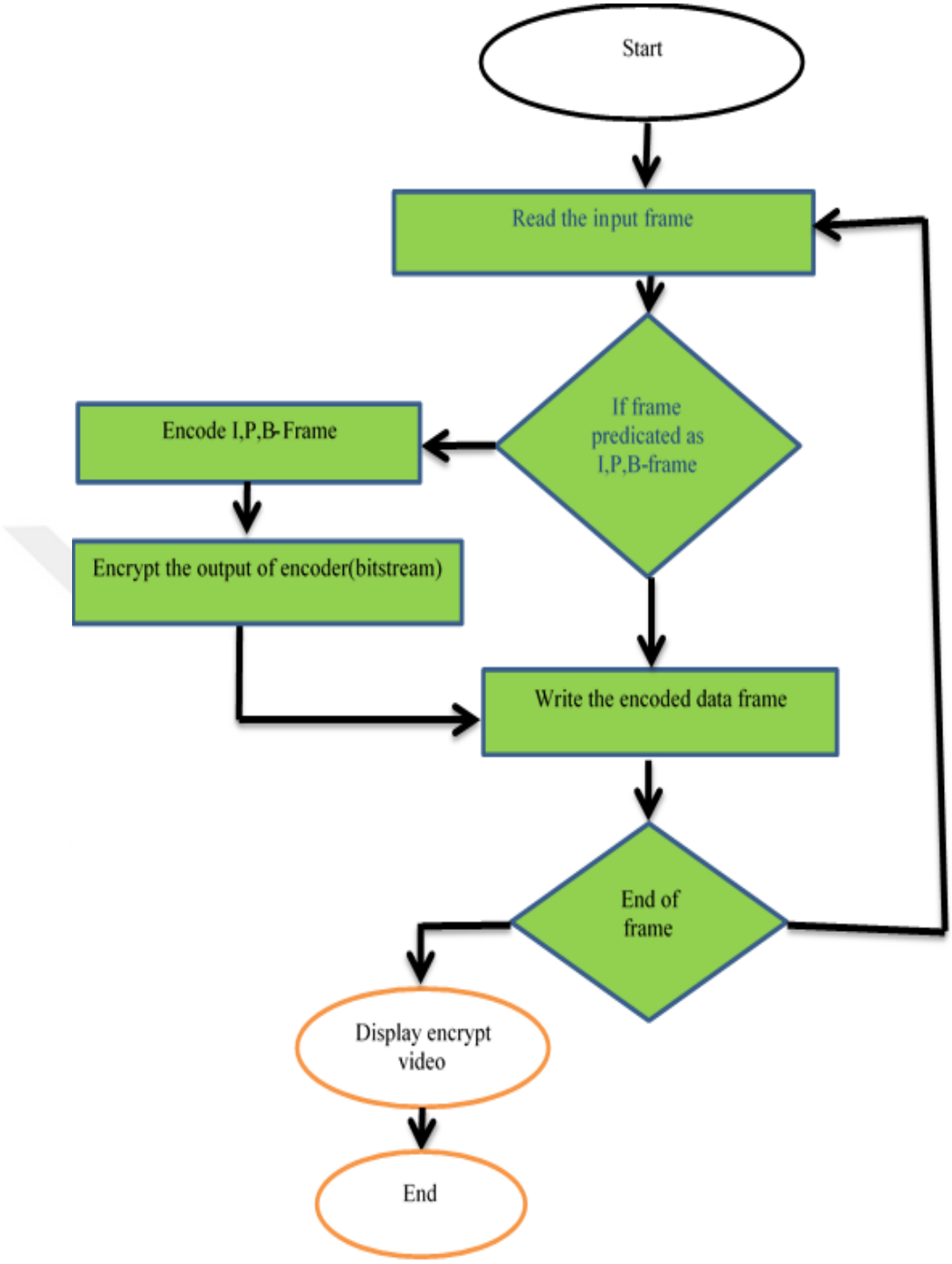


Figure 4.41: Flow Chart of encoding/encryption operation.

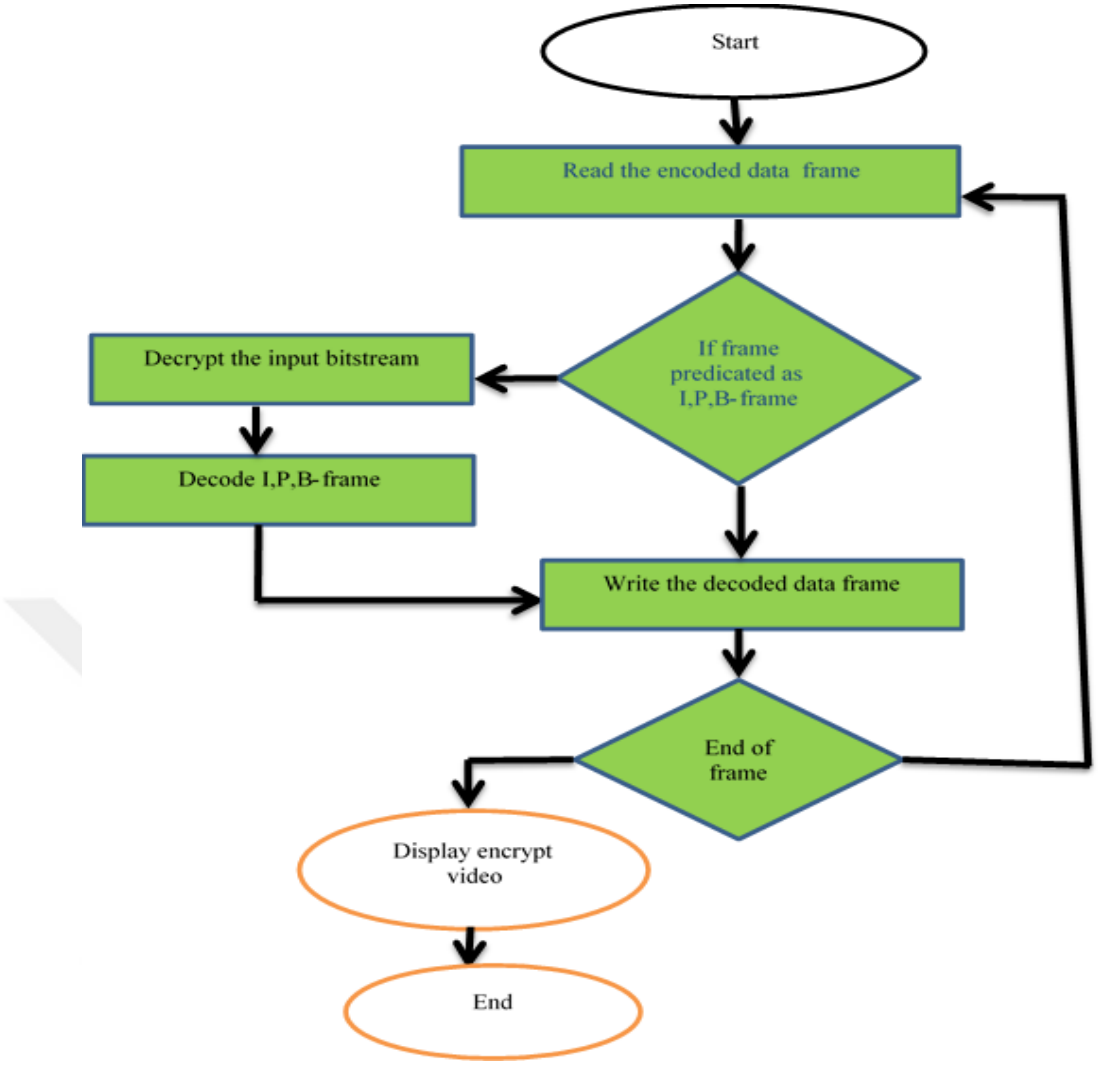


Figure 4.42: Flow Char Of decryption/decoding operation.

4.13 Conclusion

In this chapter, the most important aspects of the H.264 / AVC Video compression standard were presented and detailed. Also a state-of-the-art of the related work for Advanced Encryption Standard (AES) scheme was presented. The two main objectives of encryptions are security and execution time (speed). There are many studies and theses that tried to enhance these objectives to a better merit. Simulation results show that the proposed algorithm satisfies AVC encryption requirements by providing high security at low encryption time and less overhead bits. The proposed scheme is also computationally efficient by encrypting domains selectively according to each layer and offers high security through the use of mutually different keys

5. PROCESSING RESULT AND DISCUSION

5.1 Introduction

In this chapter, an evaluation for the proposed scheme is conducted. In the beginning, a comparison among AES_128, AES_192 and AES_256 schemes is conducted to have an insight into the performance of all of them in terms of their speed\delay differences, compares the PSNR value.

As simulator, a JM19.0 software reference in visual studio for encoding\decoding and C# is designed. It runs to assess encryption\decryption by AES algorithm. The application is developed by using the count mode of operation. It has processed and displayed many kinds of encryption information, for instance, encryption time in seconds/fractions, quantum of encryption data in bits, changed bits for avalanche effect case, etc. Tests have been conducted to the AES scheme and the proposed scheme to calculate a time difference in seconds and fractions that the laptop conducts, whereas the attack resistance has been calculated on the basis of key length.

5.2 Simulation Analysis

Behold a single video frame (picture) consist of M×N pixels (where M is the width and N is the height of the picture) where each picture is in the YUV colour space. Normally, the YUV format consists of several formats. We used the 4:2:0 format in our work. The purpose for using YUV 4:2:0 is that YUV 4:2:0 is a planar format, this meaning the Y, U, and V values are grouped together instead of interspersed. The aim for this by grouping the U and V values jointly, the frame becomes much more compressible. When given an array of an image in the YUV 4:2:0 format, all the Y values come first, followed by all the U values, followed finally by all the V values. evaluate have been conducted to the AES scheme and the suggested scheme to calculate a time difference in seconds and fractions that the laptop conducts, whereas

the attack resistance has been calculated on the basis of key length. Also we as well applied Peak-Signal-to-Noise-Ratio (PSNR) for our measure of image quality.

We can use various video in our simulation, QCIF video sequences with 176×144 pixels (Foreman),176×144 pixels (Tempete), 176×144 pixels (Hall),176×144 pixels (Container), and 176×144 pixels (Akiyo).as appear in figures 5.1-5.5 The video is compressed using the H.264/AVC encoder/decoder, discussed in the previous chapters. Our video encryption algorithm is used to encrypt/decrypt an I,P,B-frame bitstreams.

Figure 5.1: Original video Sequence YUV 4:2:0 With 176x144 pixel(Foreman).

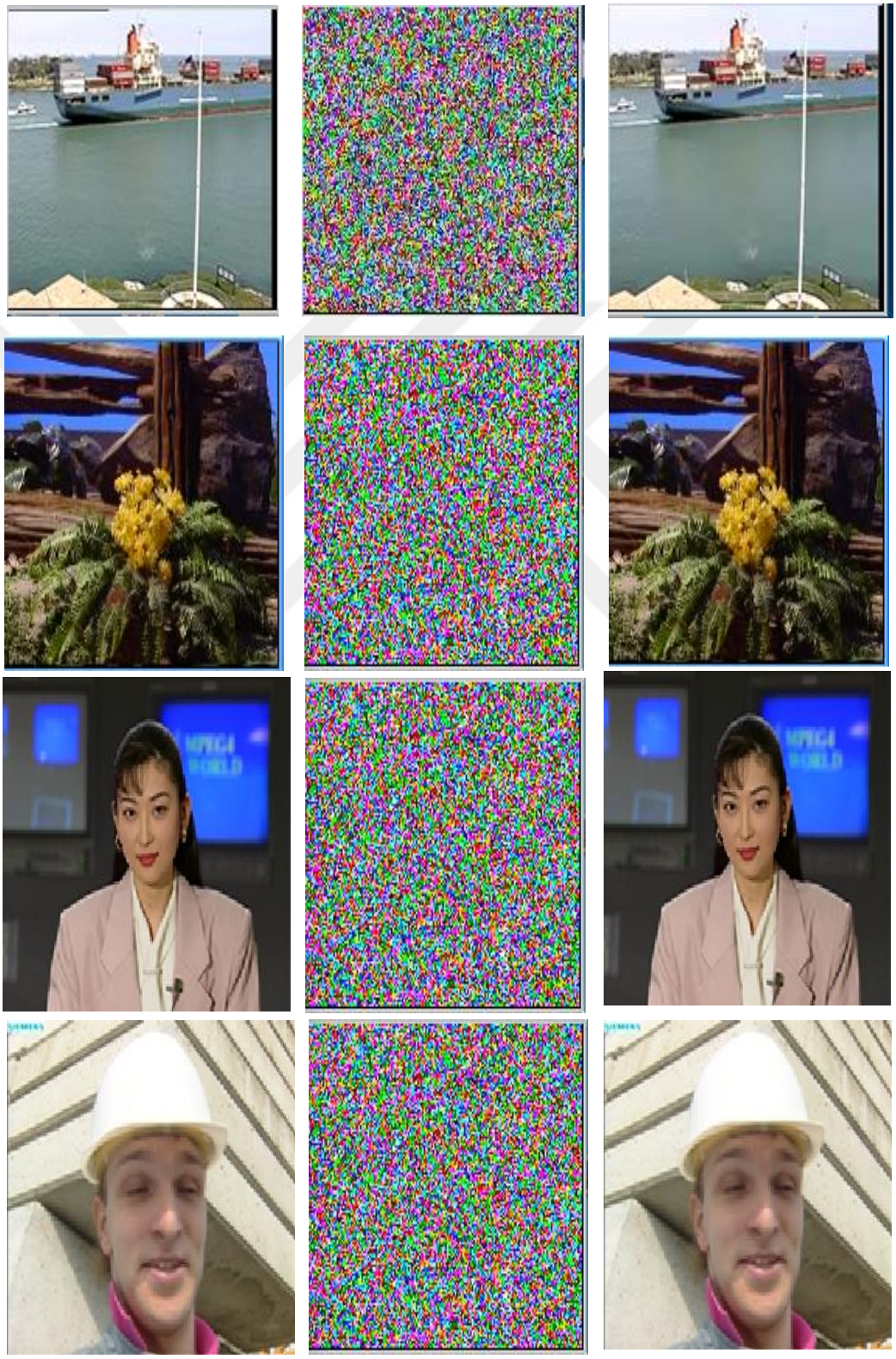
Figure 5.2: Original Video Sequence YUV 4:2:0 With 176x144 pixel (Tempete).

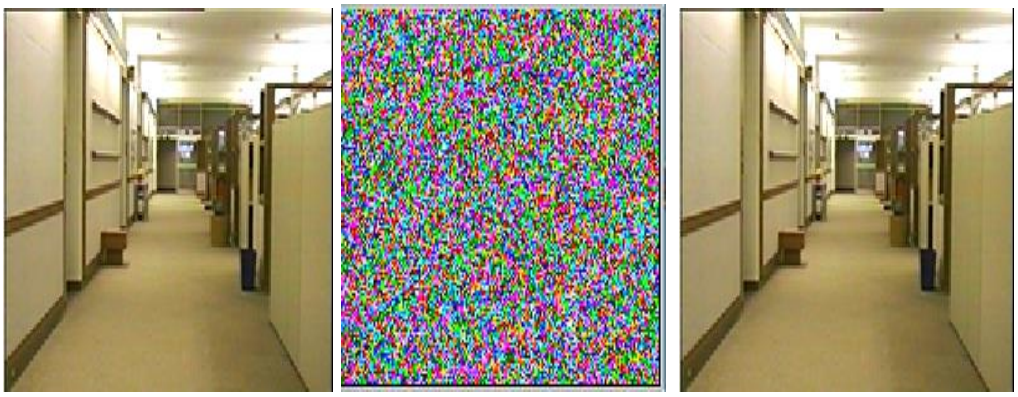
Figure 5.3: Original Video Sequence YUV With 176x144 Pixel(Hall).

Figure 5.4: Original Video Sequence YUV With176x144 Pixel(Container)

Figure 5.5: Original Video Sequence YUV 4:2:0With 176x144 Pixel (Akiyo).

The input video is first encoded using JM19.0 software. The encoded bit stream is selectively encrypted using AES. Figure 5.6. shows the effect of video encryption.





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

Figure 5.6:Output of video encryption (Container, Tempete, Foreman, Akiyo, and Hall seq.) (a) Original video (b) encrypted video (c) Decryption video

The frame sequence is also viewed in Elecard stream analyser as shown in Figure 5.7 The tallest frame (purple) represents I,P and B frame.

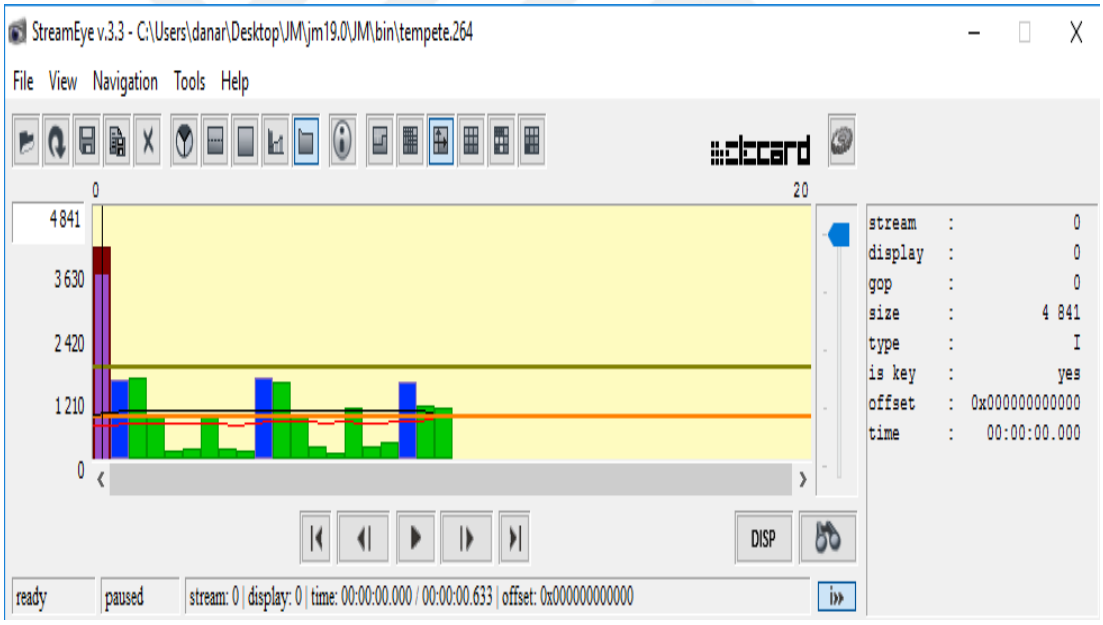


Figure 5.7: Snapshot encoded 20 Frames of Tempete sequence(IPBBB...).

The encrypted I,P and B frame is denoted by colour change as shown in Figure 5.8

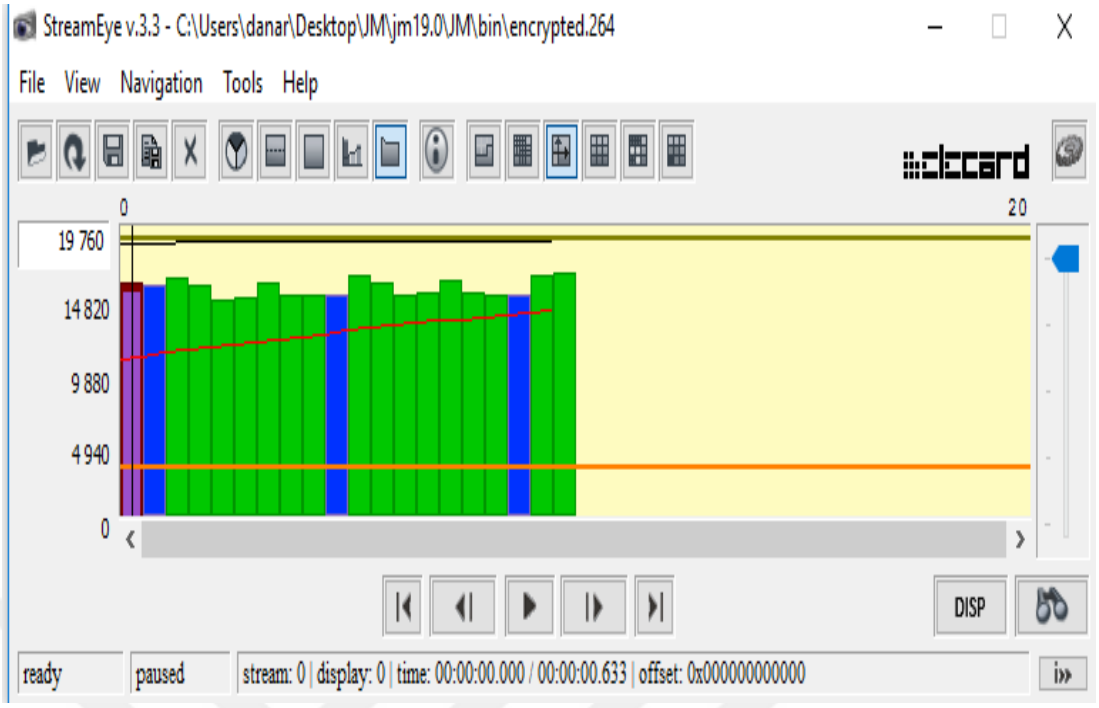


Figure 5.8: Snapshot of encrypted I, P and B frame.

5.2.1 Processing Time Analysis

To compute an average time that AES_128, AES_192 and AES_256 need to encrypt/decrypt the message, an input data that ranges from 1 MB to 30 MB is used. In this data range, thirty samples of video of grandma (tests) have been made to each of the processing time for encryption/decryption by AES_128, AES_192 and the AES_256. An approximate processing time for executing the input data of some size is appear in the Table 5.1and 5.2 for each of these approaches. The average of each approach in comparison with others is calculated as follows:

The average processing time to calculate AES_192 = AES_128 time * 1.18
The average processing time to calculate AES_256 = AES_192 time * 1.17

Table 5.1: AES_128_AES_192, AES_256 Processing Encryption Time

Time / Data Size	AES_128	AES_192	AES_256
1 MB	0.077	0.106	0.130
2 MB	0.139	0.159	0.162
3 MB	0.175	0.186	0.229
4 MB	0.213	0.238	0.276
5 MB	0.259	0.284	0.338
6 MB	0.303	0.389	0.464
7 MB	0.353	0.45	0.456
8 MB	0.398	0.442	0.534
9 MB	0.440	0.497	0.576
10 MB	0.482	0.542	0.683
11 MB	0.527	0.614	0.683
12 MB	0.596	0.662	0.745
13 MB	0.619	0.698	0.798
14 MB	0.664	0.751	0.910
15 MB	0.707	0.802	0.939
16 MB	0.741	0.907	0.963
17 MB	0.812	0.971	1.23
18 MB	0.898	1.11	1.133
19 MB	0.907	1.34	1.203
20 MB	0.949	1.79	1.275
21 MB	0.997	1.139	1.334
22 MB	1.65	1.235	1.360
23 MB	1.100	1.290	1.427
24 MB	1.113	1.297	1.473
25 MB	1.158	1.328	1.528
26 MB	1.212	1.402	1.584
27 MB	1.392	1.426	1.687
28 MB	1.469	1.639	1.725
29 MB	1.475	1.667	1.787
30 MB	1.491	1.798	1.805

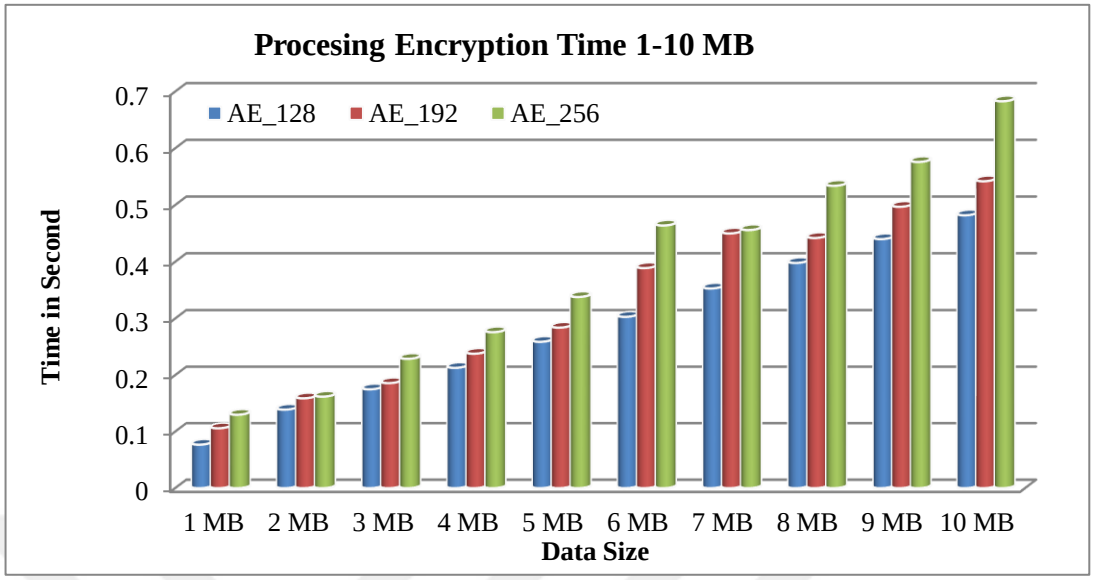


Figure 5.9: A Sample of Excute Time of Data Size (1-10 MB).

Figures 5.9, 5.10, 5.11 and 5.12 illustrate the time difference for video encryption between AES_128, AES_192 and AES_256 tests for data ranges from 1 MB to 30 MB. Tests have been conducted to have an insight into the speed\delay time differences by showing thirty different video samples, as a result AES_128 is 1.18 faster than AES_192, whereas AES_192 is 1.17 faster than AES_256.

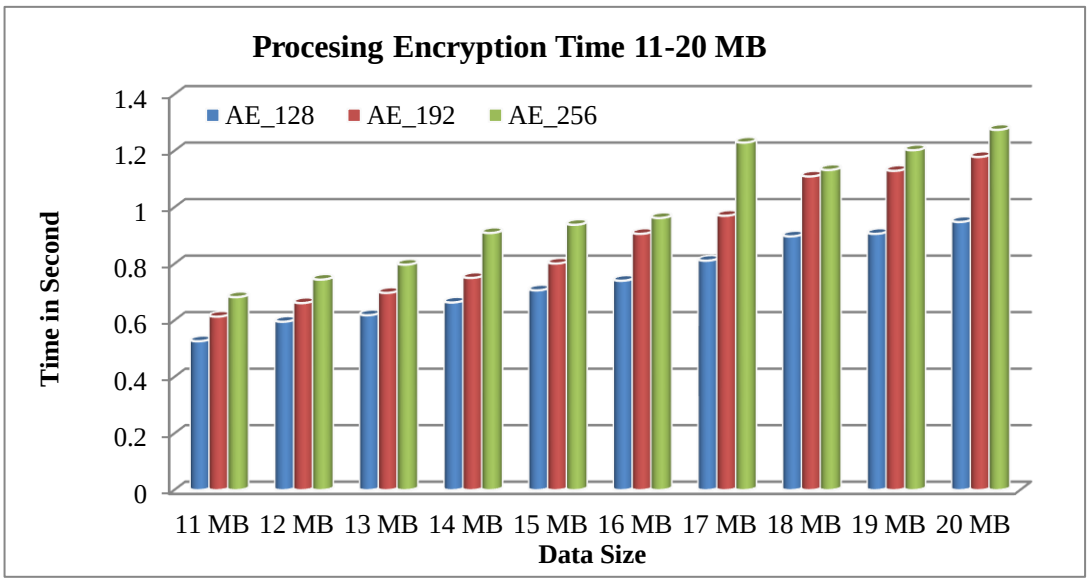


Figure 5.10: A Sample of Excute Time of Encryption for Data Size (11-20 MB).

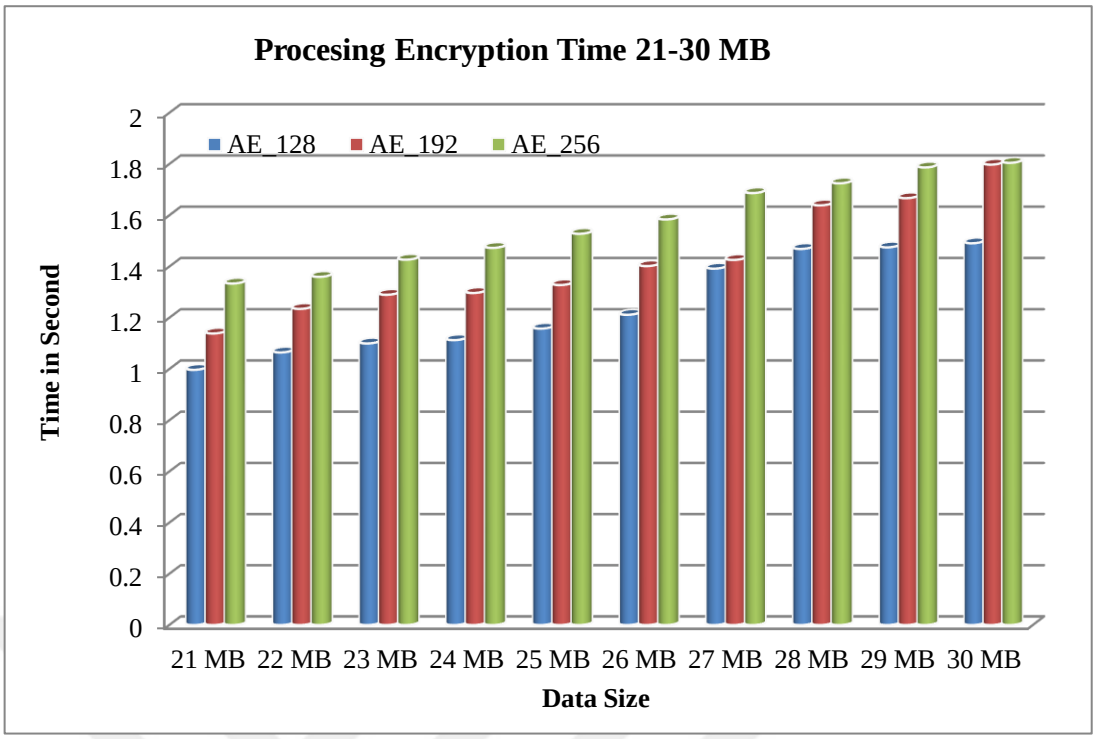


Figure 5.11: A Sample of Excute Time for Data Size (21-30 MB).

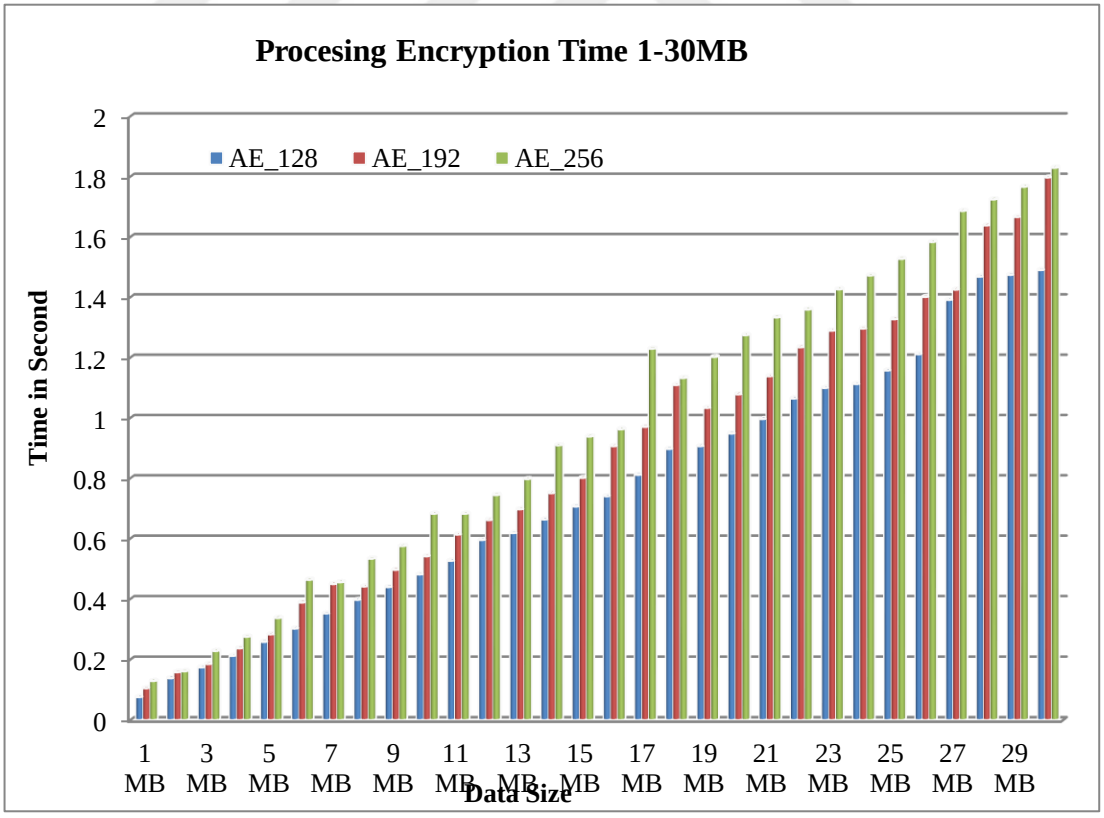


Figure 5.12: A Sample of Excute Time for Data Size (1-30 MB).

Table 5.2: AES_128,AES_192,AES_256 Processing Decryption Time

Time / Data Size	AES_128	AES_192	AES_256
1 MB	0.081	0.087	0.105
2 MB	0.135	0.144	0.172
3 MB	0.183	0.199	0.244
4 MB	0.233	0.269	0.302
5 MB	0.280	0.309	0.360
6 MB	0.328	0.370	0.432
7 MB	0.386	0.424	0.491
8 MB	0.436	0.479	0.484
9 MB	0.485	0.536	0.622
10 MB	0.521	0.592	0.682
11 MB	0.577	0.661	0.735
12 MB	0.632	0.718	0.806
13 MB	0.700	0.778	0.864
14 MB	0.743	0.832	0.948
15 MB	0.785	0.908	1.13
16 MB	0.846	0.975	1.63
17 MB	0.875	1.12	1.112
18 MB	0.973	1.58	1.202
19 MB	0.985	1.100	1.269
20 MB	1.114	1.173	1.335
21 MB	1.113	1.221	1.403
22 MB	1.153	1.285	1.432
23 MB	1.259	1.341	1.535
24 MB	1.264	1.397	1.577
25 MB	1.268	1.434	1.683
26 MB	1.371	1.515	1.677
27 MB	1.423	1.526	1.763
28 MB	1.454	1.688	1.873
29 MB	1.517	1.699	1.940
30 MB	1.527	1.767	1.960

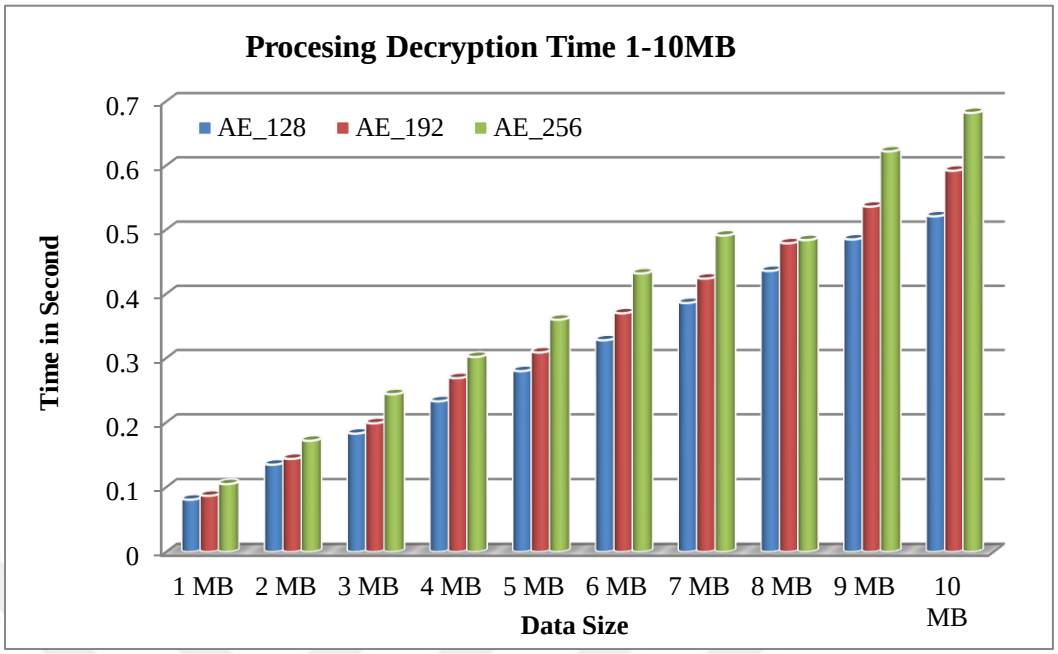


Figure 5.13: A Sample of Excute Time for Data Size (1-10 MB).

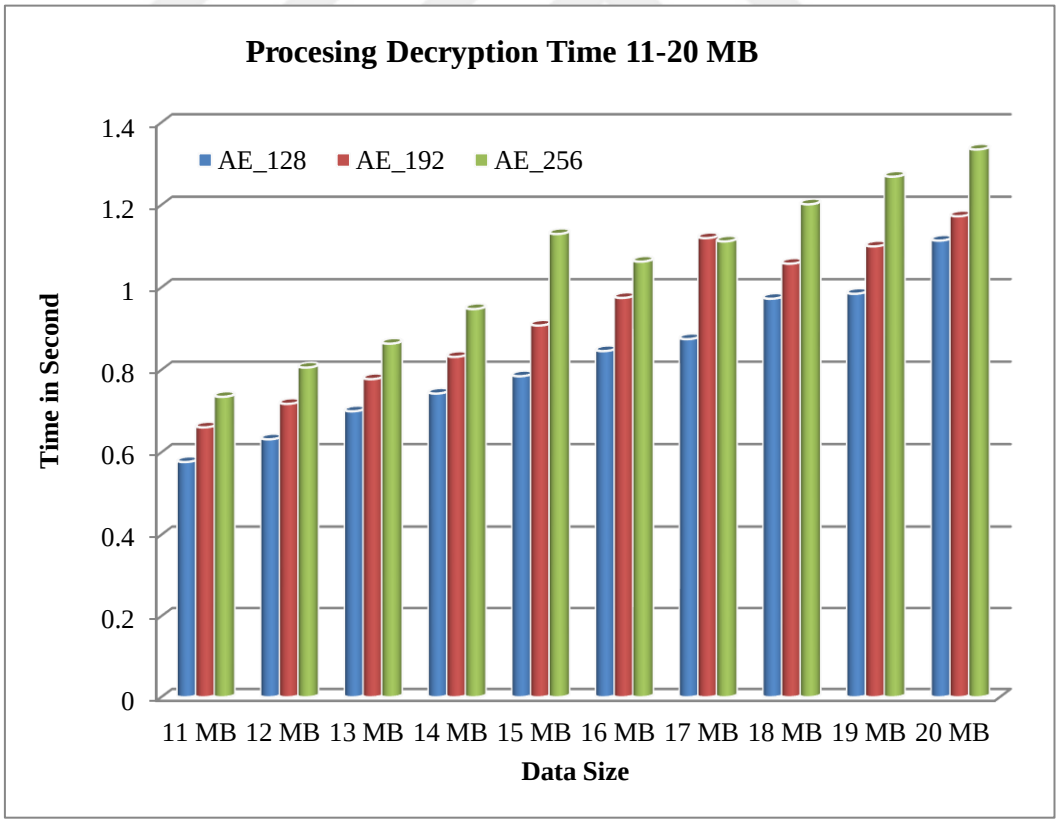


Figure 5.14: A Sample of Excute Time for Data Size (11-20 MB).

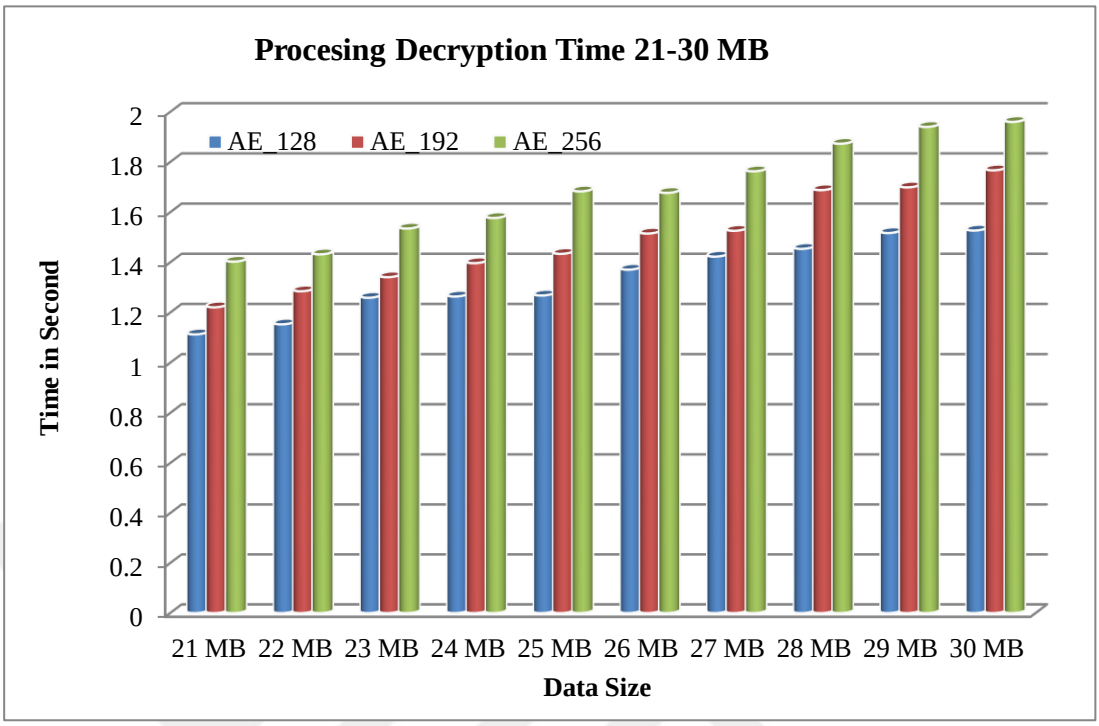


Figure 5.15: A Sample of Excute Time for Data Size(21-30 MB).

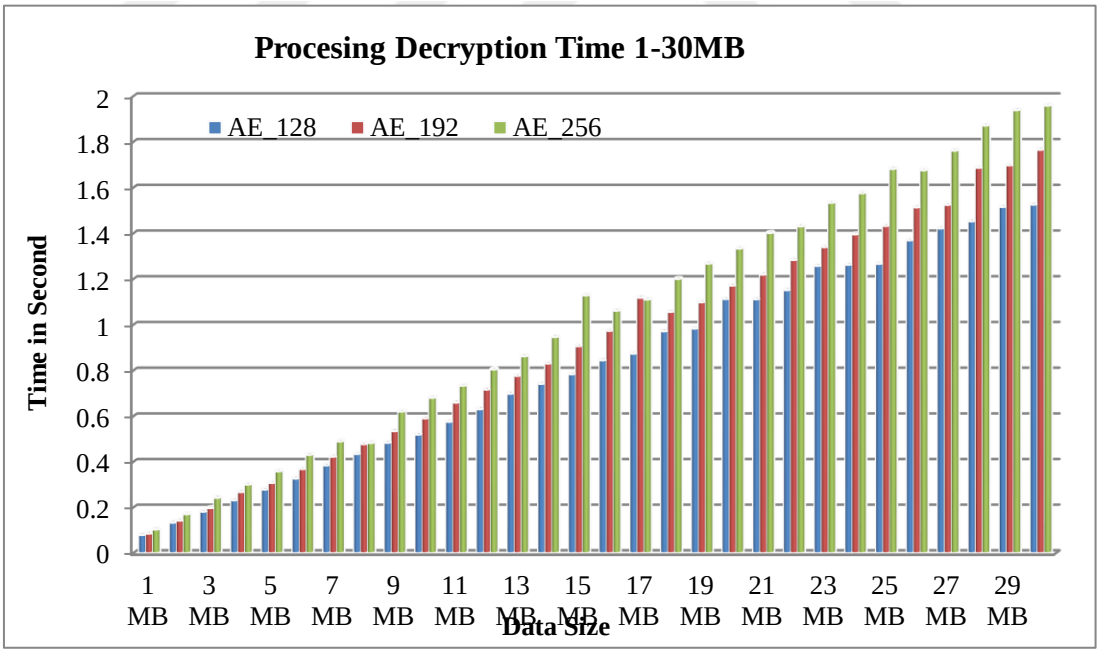


Figure 5.16: A Sample of Excute Time for Data Size (1-30 MB).

In the both process encryption/decryption, as a result shows that data processing time for the AES_128 technique is less than AES_192 and AES_256.

Encryption/decryption process AES_256 technique is more secure than AES_192 and AES_128.

5.2.3 Peak Signal to Noise Ratio (PSNR) analysis

Whatever the performance of the compression technique, there will always be distortions in the reconstructed image and video, the quality of which can be assessed objectively and subjectively. The objective measurement of quality is made in terms of signal to noise ratio (PSNR), compression ratio (CR) and structural similarity (SSIM). The higher the PSNR value, the closer the compressed image is to the source image. In order to measure the quality of the video data at the output of the decoder, Mean Square Error (MSE) and Peak Signal to Noise Ratio (PSNR) are often used. The PSNR in terms of decibels (dB) between two frames having 8 bits per pixel.

To evaluate the PSNRs (YUV) video of original video, encoding with encryption and decryption with decoding for different QP parameter shown in table 4.3, we can taking several videos e.g foreman, container, tempete, hall and akiyo for 100 frame all videos by entropy coding method of CABAC and frame rate is 30 fps.

Table 5.4: PSNR for original video, encoding with encryption video ,decryption with decoding video.

Video	**QP**	**PSNR**		
		Original Video	**Encoding with Encryption video**	**Decryption with Decoding video**
Container	24	43.55	9.14	36.6
	30	38.8	9.20	34.51
	36	32.81	9.33	31.19

Table (5.4 Cont.): PSNR for original video, encoding with encryption video, decryption with decoding video.

Video	QP	PSNR		
		Original Video	Encoding with Encryption video	Decryption with Decoding video
Foreman	24	43.82	8.45	35.35
	30	38.47	8.5	33.93
	36	32.41	8.61	30.62
Hall	24	43.90	9.15	34.34
	30	39.95	9.21	33.53
	36	33.1	9.34	30.73
Tempete	24	42.24	8.05	31.77
	30	35.5	8.09	30.18
	36	28.77	8.13	26.82
Akiyo	24	45.32	8.66	35.72
	30	42.29	8.71	35.1
	36	35.34	8.82	32.40

However, the PNSRs in the encrypted video are slightly lower cause it shows the strength of data security. In these encrypted videos, the PNSRs for the encrypted I,P,B-Frames are all lower than those of the decrypted I,P,B-Frames, showing a difference of at least 30 dB. Likewise, it was noticed in the encrypted videos that, at every I, P, B-Frames Frame there is a huge decrease in the PNSR.

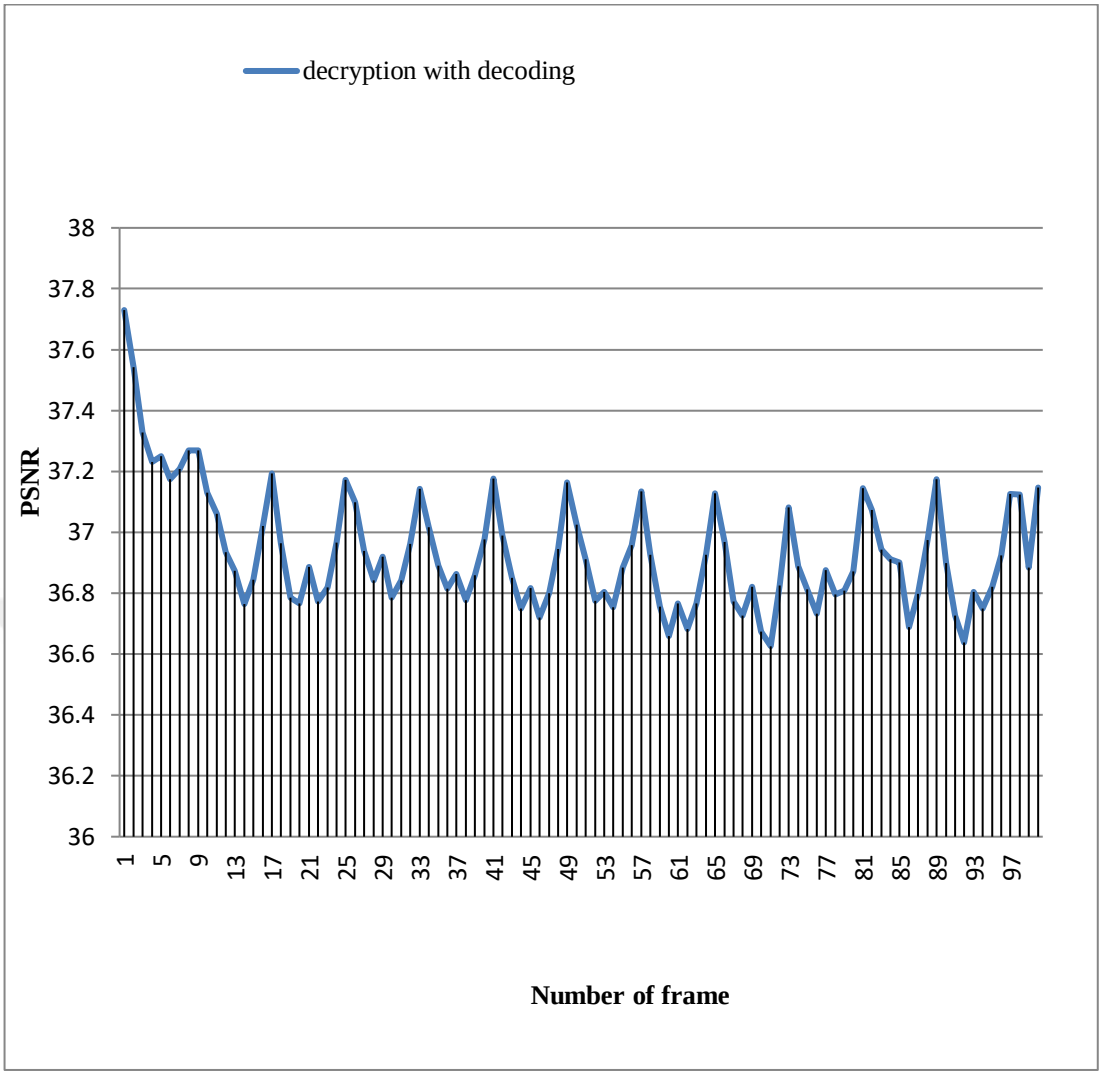


Figure 5.17: PSNR of 100 Frames Taken from the 'Container' decryption with decoding.

In figure 5.18-19 shows that experimental of 100 frames for container video when we increase decoding with decrypted in farmers it causes to decrease decoding with encryption.

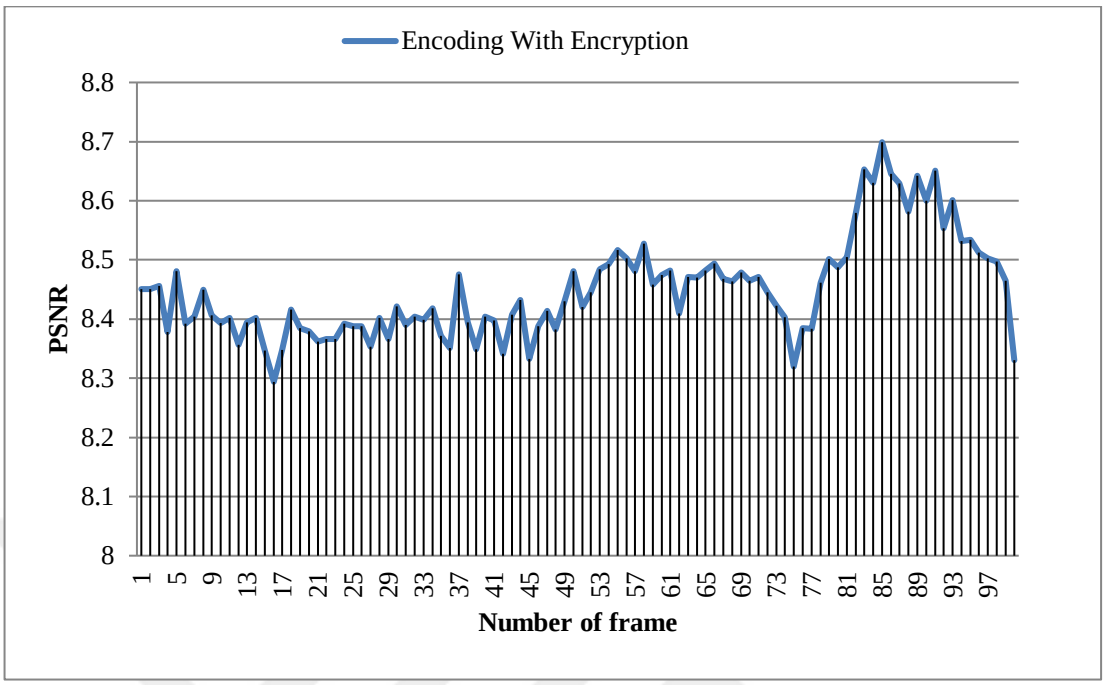


Figure 5.18: PSNR of 100 frames taken from the 'Container' encoding with encryption.

5.3 Discussion

In comparing results in this research with a research called novel semi-symmetric encryption it has compared the estimated time for both process and which was high. Unlike the results that I got which was much lower that show in the above research.

In addition to that PSNR ratio in previous researches was I a standard ratio range if been compared with my research which was higher.

5.4 Conclusion:

An interface been created with C# and JM19.0 software reference for implementing selective videos to be encoded with H.264/AVC method, then be encrypted is been simulated. Encryption of the I, P, B-frames using the proposed selective algorithm for use with H.264 leads to an understandable bit stream, thus, shielding the video stream from intruders. I, P, B-frames encryption provides a fine exchange over encryption robustness and flexibility. It gives enough security for video stream

6. CONCLUSION AND FUTURE WORK

6.1 Conclusion

In this research digitized video encryption techniques for safe transmission and secure storage of multimedia data been discussed. Observing the importance of the multimedia security and encryption on the World Wide Web, then reviewing the newest technologies used to image and video encryption, an enhanced selective video encryption of H.264/AVC using AES was proposed.

An interface been created with C# and JM19.0 software reference for implementing selective videos to be encoded and decoded with H.264/AVC method in details, then be encrypted is been simulated. Encryption of the I, P, B-frames using the proposed selective algorithm for use with H.264 leads to an understandable bit stream, thus, shielding the video stream from intruders .I,P,B-frames encryption provides a fine exchange over encryption robustness and flexibility. It gives enough security for video stream. It has processed and displayed many kinds of encryption information, for instance, encryption time in seconds/fractions, quantum of encryption data in bits. Tests have been conducted to the AES scheme and the proposed scheme to calculate a time difference in seconds and fractions that the laptop conducts, whereas the attack resistance has been calculated on the basis of key length.

An evaluation for the proposed scheme is conducted. In the beginning, a comparison among AES_128, AES_192 and AES_256 schemes is conducted to have an insight into performance all of them in terms of speed\delay differences, compares the PSNR value, overhead bit and security level for each of them.

6.2 Future work

1. The proposed scheme will be utilized with the rest of operation modes such as Electronic Code Book (ECB)**,** Cipher Block Chaining (CBC)**,** Cipher Feedback (CFB) and Output Feedback (OFB).

2. Apply modifications on AES to Voice, video and text encryption, Also comparative between all of them.

3. Enhancement in compression performance by introduction of new Functionalities which also improves security as encryption is combined with compression.

4. Another technique can be added to the proposed selective schemes which are selection criteria.in these criteria, Encryption techniques can be chosen dynamically as the content will be distributed and the selection criteria can be changed as needed by the application.

5. Apply watermarking technology in h.265\HEVC file.

REFERENCES

Abaas, S. A., and Shibeeb, A. 2015. A New Approach for Video Encryption Based on Modified AES Algorithm. IOSR Journals (IOSR Journal of Computer Engineering), 1(17), 44-51.

Abomhara, M. A. S. 2010. Enhancing selective encryption for H. 264/AVC using advanced encryption standard. International Journal of Computer Theory and Engineering, Vol. 2(2), pp. 223-229.

Asghar, M. N., Ghanbari, M., Fleury, M., and Reed, M. J. (2015). Sufficient encryption based on entropy coding syntax elements of H. 264/SVC. Multimedia Tools and Applications, 74(23), 10215-10241.

Bahrami, S., and Naderi, M. 2014. Encryption of Video Main Frames in the Field of DCT Transform Using A5/1 and W7 Stream Encryption Algorithms. Arabian Journal for Science & Engineering (Springer Science and Business Media BV), 39(5), 4077-4088.

Deshmukh, P., and Kolhe, V. 2014. Modified aes based algorithm for mpeg video encryption. In Information Communication and Embedded Systems (ICICES), 2014 International Conference on (pp. 1-5). IEEE.

Elkilani, W. S., and Abdul-Kader, H. M. 2009. Performance of encryption techniques for real time video streaming. In Networking and Media Convergence, 2009. ICNM 2009. International Conference on (pp. 130-134). IEEE.

Fares, N. M., and Askar, S. A Novel Semi-symmetric Encryption Algorithm for Internet Applications 2016.

Ghrare, S. E., Ali, M. A. M., Ismail, M., and Jumari, K. 2008. Diagnostic quality of compressed medical images: Objective and subjective evaluation. In Modeling and Simulation, 2008. AICMS 08. Second Asia International Conference on (pp. 923-927). IEEE.

Gupta, S., Kishor, L., and Goyal, D. 2014. Comparative Analysis of Encrypted Video Streaming in Cloud Network. IJCSIT, 5(4), 5470-5476.

Hands, D. S., Huynh-Thu, Q., Rix, A. W., Davis, A. G., and Voelcker, R. M. 2004. Objective perceptual quality measurement of 3G video services. In 3G Mobile Communication Technologies, 2004. 3G 2004. Fifth IEE International Conference on (pp. 437-441). IET.

Haskell, B. G., Puri, A., annd Netravali, A. N. 1996. Digital video: an introduction to MPEG-2. Springer Science and Business Media, 7(1), 265-266.

Karthigaikumar, P., and Rasheed, S. 2011. Simulation of image encryption using AES algorithm. IJCA Special Issue on "Computational Science-New Dimensions & Perspectives" NCCSE, pp. 166-172.

Kotel, S., Zeghid, M., Baganne, A., Saidani, T., Daradkeh, Y. I., and Rached, T. 2014. FPGA-Based Real-Time Implementation of AES Algorithm for Video Encryption. Recent Advances in Telecommunications, Informatics and Educational Technologies, 27-36.

Kulkarni, A. K. 2012. Implementation of fast inter-prediction mode decision in H. 264/AVC video encoder.

Li, S., Chen, G., Cheung, A., Bhargava, B., and Lo, K. T. 2007. On the design of perceptual MPEG-video encryption algorithms. IEEE Transactions on Circuits and Systems for Video Technology, 17(2), 214-223.

Memos, V. A., and Psannis, K. E. 2016. Encryption algorithm for efficient transmission of HEVC media. Journal of Real-Time Image Processing, 12(2), 473- 482.

Mishra, D. C., Sharma, R. K., Dawar, M., and Hanmandlu, M. 2015. Two Layers of Security for Color Video by Matrix Affine Cipher with Two-Dimensional Discrete Wavelet Transform. Fractals, 23(04), 1550037.

Pai, T. P., Raghu, M. E., and Ravishankar, K. C. 2014. Video Encryption for Secure Multimedia Transmission-A Layered Approach. In Eco-friendly Computing and Communication Systems (ICECCS), 2014 3rd International Conference on (pp. 127-132). IEEE.

Rana J.S. Al-Janabi, Abdul Monem S. Rahma and Matheel E.Abdulmunim, 2015, " A New Digital Watermarking Algorithm Based on Image Comparison Techniques", Int.J.Computer Technology and Applications,Vol 6(1) ,PP.7-11.

Rao, K. R., Kim, D. N., and Hwang, J. J. 2013. Video coding standards: AVS China, H. 264/MPEG-4 PART 10, HEVC, VP6, DIRAC and VC-1. Springer Science and Business Media, Netherland.

Richardson, I. E. 2004. H. 264 and MPEG-4 video compression: video coding for next-generation multimedia. 2rd Edition, Wiley ISBN 0-470-84837-5, United Kingdom.

Rijkse, K. 1996. H. 263: Video coding for low-bit-rate communication. IEEE Communications magazine, 34(12), 42-45.

Shi, C., and Bhargava, B. 1998. A fast MPEG video encryption algorithm. In Proceedings of the sixth ACM international conference on Multimedia (pp. 81-88). ACM.

Sikora, T. 1997. MPEG digital video-coding standards. *IEEE signal processing magazine*, *14*(5), 82-100.

Tang, L. 1997. Methods for encrypting and decrypting MPEG video data efficiently. In Proceedings of the fourth ACM international conference on Multimedia (pp. 219-229). ACM.

Tayde, S., and Siledar, S. 2015. File Encryption, Decryption Using AES Algorithm in Android Phone. International Journel of Advanced Research in computer science and software engineering, Vol. 5(5), pp. 550-554.

Turletti, T., and Huitema, C. 1996. Videoconferencing on the Internet. IEEE/ACM Transactions on networking, 4(3), 340-351.

Varalakshmi, L. M. 2015. *Studies on Security Enhancement Schemes For MPEG-2And H. 264/AVC Video* (Doctoral dissertation).

Wang, Z., Bovik, A. C., Sheikh, H. R., and Simoncelli, E. P. 2004. Image quality assessment: from error visibility to structural similarity. *IEEE transactions on image processing*, *13*(4), 600-612.

Yadav, A., Verma, M., and Patidar, K. 2016. An efficient video data security mechanism based on RP-AES. International Journal of Advanced Technology and Engineering Exploration, 3(16), pp. 36-42.

Yi, K., Lee, Y. H., and Joo, J. H. 2016. Fast H. 264 Video Decoding by Bit Order Reversing and Its Application to Real-time H. 264 Video Encryption. International Journal of Multimedia and Ubiquitous Engineering, 11(3), 45-56.

APPENDIX

A: Appendix of the proposed code for ecoding video.

B: Appendix of the proposed code of interface for video encryption.

APPENDIX A: Appendix of the proposed code for ecoding video.

```
###############################################################################
#########
# Files
###############################################################################
#########
InputFile                    = "foreman_qcif.yuv"       # Input sequence
InputHeaderLength            = 0         # If the inputfile has a header, state it's length in
StartFrame                   = 0         # Start frame for encoding. (0-N)
FramesToBeEncoded            = 100       # Number of frames to be coded

FrameRate                    = 30.0      # Frame Rate per second (0.1-100.0)
Enable32Pulldown             = 0         # Enable 'hard' 3:2 pulldown (modifying the inpur data)
                                         #0 = disabled
                                         #1 = A, B, Bt|Cb, Ct|Db, D
                                         #2 = A, B, C, Ct|Db, D
SEIVUI32Pulldown             = 0         # Enable 3:2 pulldown through VUI and SEI metadata
signaling. Five methods are supported:
                                        # 0 = disabled
                                        # 1 = A, Bt|Bb, Bt|Cb, Ct|Cb, D
                                        # 2 = A, B, C, C, D
                                        # 3 = At|Ab, Bt|Bb, Bt|Cb, Ct|Cb, Dt|Db
                                        # 4 = A, Bt|Bb, Bt|Cb, Ct|Db, Dt|Db
                                        # 5 = At|Ab, Bt|Bb, Bt|Cb, Ct|Db, Dt|Db
SourceWidth          = 176              # Source frame width
SourceHeight         = 144              # Source frame height
SourceResize         = 0                # Resize source size for output
OutputWidth          = 176              # Output frame width
OutputHeight         = 144              # Output frame height
ProcessInput         = 0                # Filter Input Sequence
Interleaved          = 0                # 0: Planar input, 1: Packed input
PixelFormat          = 0                # Pixel Format for 422 packed inputs
                                        # 0: UYVY
                                        # 1: YUY2/YUYV
                                        # 2: YVYU
                                        # 3: BGR (Unsupported)
                                        # 4: V210 (Video Clarity)
TraceFile            = "trace_enc.txt"          # Trace file
ReconFile            = "foreman_qcif_rec.yuv"   # Reconstruction YUV file
OutputFile           = "foreman_qcif.264"       # Bitstream
StatsFile            = "foreman_qcif.dat"       # Coding statistics file
```

Figure A.1: JM19.0 encoder configuration file

We can use set of parameter in encoder configuration file to encoded the YUV file by H.264 bitsream.

1. Input and output file names.
2. Frame size (Y component).
3. Number of frames to be encoded;

4. Source frame rate, necessary to correctly set rate control parameters.
5. Pixel format.

```
Input YUV file                  : foreman_qcif.yuv
 Output H.264 bitstream         : foreman_qcif.264
 Output YUV file                : foreman_qcif_rec.yuv
 YUV Format                     : YUV 4:2:0
 Frames to be encoded           : 100
 Freq. for encoded bitstream    : 30.00
Frame    Bit/pic   QP   SnrY    SnrU    SnrV    Time(ms) MET(ms) Frm/Fld Re
=========================================================
00000  (NVB)    320
00000 (IDR)   91   36   36  32.241  39.187  39.822   493      0    FRM    3
00008 ( P )   4480   36  32.056  38.596  38.871     1352    578    FRM    2
00093(B )    936     38  30.539  38.501  38.454     4213    3139    FRM    0
00095( B )    352     38  31.317  38.528  38.663     3250    2431    FRM    0
00099( P )   2384     36  32.652  38.986  39.029     2707    1968    FRM    2
00097( B )    536     36  31.795  38.838  38.955     3247    2405    FRM    0
00098( B )    432     36  32.207  38.854  38.944     3177    2375    FRM    0
-------------------------------------------------------------------------------
 Total Frames:  100
 Leaky BucketRateFile does not have valid entries.
  Number Leaky Buckets: 8
    Rmin     Bmin     Fmin
   37230    13486    13486
   46530    12418    12418
------------------ Average data all frames  ----------------------------------
 Total encoding time for the seq.  : 348.103 sec (0.29 fps)
 Total ME time for sequence        : 258.804 sec
 Y { PSNR (dB), cSNR (dB), MSE }   : {  31.267,  31.231,  48.97020 }
 U { PSNR (dB), cSNR (dB), MSE }   : {  38.541,  38.520,   9.14342 }
 V { PSNR (dB), cSNR (dB), MSE }   : {  39.173,  39.141,   7.92485 }
 Total bits                        : 124424 (I 9136, P 52824, B 62144 NVB 320)
 Bit rate (kbit/s)  @ 30.00 Hz     : 37.33
 Bits to avoid Startcode Emulation: 732
 Bits for parameter sets           : 320
 Bits for filler data              : 0
```

Figure A.2: JM19.0 encoder output display

Also in encoder control we can use set of parameter to encode the YUV file by H.264 bitstream.

1. Basic control parameters.
2. Profile and Level.
3. I-slice control; I and P slice quantization parameters (QP.
4. motion estimation control.
5. search range.
6. reference frames.
7. partition sizes.

APPENDIX B: Appendix of the proposed code of interface for video encryption/deryption.

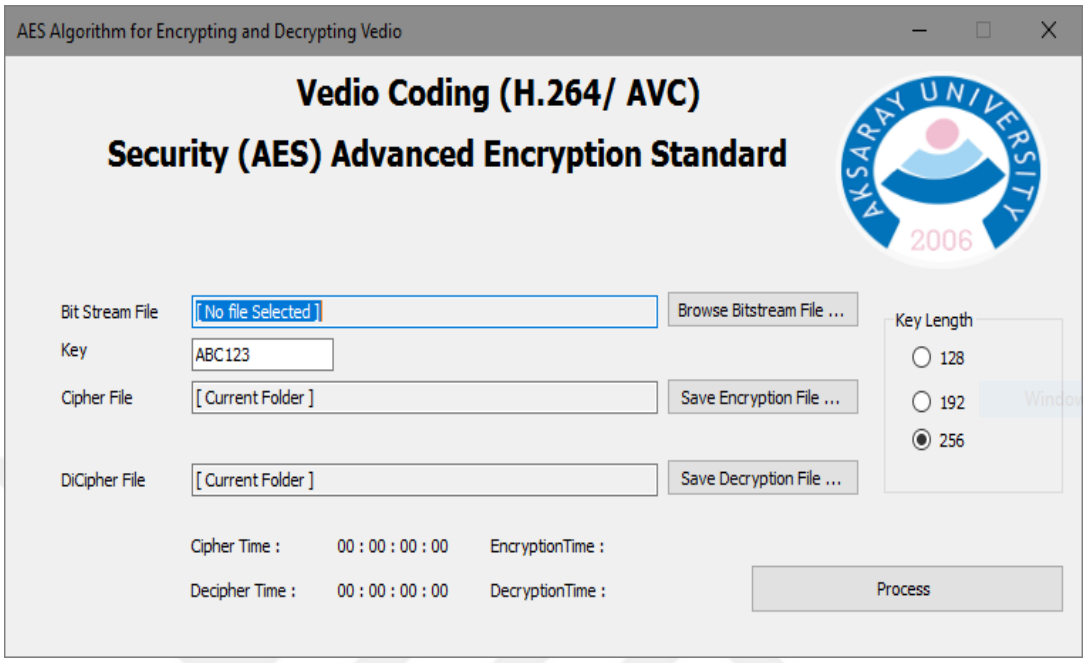


Figure B.1: User Interface (UI) to video encryption\decryption

As shown in above Figure 6.3 User Interface (U), to process video file user should do the following Bit Stream Box: is a plain file which user intends to do encryption and decryption on it.

Cipher File Box: is an encrypted file.
Decipher File Box: is a decrypted file.
Process command: is a command when clicked it process encryption and decryption on bit stream file.
Cipher Time: is an elapsed time. time takes to do encryption.
Decipher Time is an elapsed time. time takes to do decryption.

Code Explaining

The following is segment of codes which are used to do encrypt and decrypt on plain file.
When user click on process the following code in figure 6.4 is executed

```
61  private void btnProcess_Click(object sender, EventArgs e)
62  {
63      if (openFileD1 != null && saveFileD2 != null && saveFileD3 !=null)
64      {
65          int keyLen = 256;
66          if(Radio128.Checked)
67          {
68              keyLen = 128;
69          }else if(Radio192.Checked)
70          {
71              keyLen = 192;
72          }
73          AESClass.EncryptFile(openFileD1.FileName, saveFileD2.FileName, txtKey.Text, keyLen);
74          AESClass.DecryptFile(saveFileD2.FileName, saveFileD3.FileName, txtKey.Text, keyLen);
75          lblEncryptionTime.Text = AESClass.encryptTime;
76          lblDecryptionTime.Text = AESClass.decryptTime;
77
78      }
79  }
```

Figure B.2: process to encrypted file

Code from line 65 to 72 check which key is selected to use for encryption and decryption algorithm, then the code in line 73 call static method (Encrypt File) of class AESClass which takes four parameters:

```
public static void EncryptFile(string originfileName, string encryptrdFileName, string key, int keyLen)
```

The following is a body this function

```
99  public static void EncryptFile(string originfileName, string encryptrdFileName, string key, int keyLen)
100 {
101     string file = originfileName;
102     string password = key;
103     byte[] bytesToBeEncrypted = File.ReadAllBytes(file);
104     //Create bit stream
105     File.WriteAllBytes("allBytes.txt", bytesToBeEncrypted);
106
107     byte[] passwordBytes = Encoding.UTF8.GetBytes(password);
108
109     // Hash the password with SHA256
110     passwordBytes = SHA256.Create().ComputeHash(passwordBytes);
111     byte[] bytesEncrypted = AES_Encrypt(bytesToBeEncrypted, passwordBytes, keyLen);
112 }
```

Figure B.3: The body of process encrypted file

Code in line 111 call a function AES_Encrypt, this function is response for encryption process. This function takes three parameters, first Bytes intend to encrypt, second password is used for encryption and decryption process and final is a key length which is selected from UI.

Function signature

```
public static byte[] AES_Encrypt(byte[] bytesToBeEncrypted, byte[] passwordBytes, int keyLen)
```

Body of the function

```
15        public static byte[] AES_Encrypt(byte[] bytesToBeEncrypted, byte[] passwordBytes, int keyLen)
16        {
17            //Start tick time
18            Stopwatch Stopwatch = new Stopwatch();
19            Stopwatch.Start();
20            byte[] encryptedBytes = null;
21            // Set your salt here, change it to meet your flavor:
22            // The salt bytes must be at least 8 bytes.
23            byte[] saltBytes = new byte[] { 1, 2, 3, 4, 5, 6, 7, 8 };
24            using (MemoryStream ms = new MemoryStream())
25            {
26                using (RijndaelManaged AES = new RijndaelManaged())
27                {
28                    AES.KeySize = keyLen;
29                    AES.BlockSize = 128;
30                    var key = new Rfc2898DeriveBytes(passwordBytes, saltBytes, 1000);
31                    AES.Key = key.GetBytes(AES.KeySize/8);
32                    AES.IV = key.GetBytes(AES.BlockSize / 8);
33                    AES.Mode = CipherMode.CBC;
34                    using (var cs = new CryptoStream(ms, AES.CreateEncryptor(), CryptoStreamMode.Write))
35                    {
36                        cs.Write(bytesToBeEncrypted, 0, bytesToBeEncrypted.Length);
37
38                        cs.Close();
39                    }
40                    encryptedBytes = ms.ToArray();
41                }
42            }
43
44              TimeSpan ts = Stopwatch.Elapsed;
45
46            // Format and display the TimeSpan value.
47            encryptTime = String.Format("{0:00}:{1:00}:{2:00}.{3:00}",
48                ts.Hours, ts.Minutes, ts.Seconds,
49                ts.Milliseconds);
50            return encryptedBytes;
51        }
52        public static byte[] AES_Decrypt(byte[] bytesToBeDecrypted, byte[] passwordBytes, int keyLen)...
94        public static void EncryptFile(string originfileName, string encryptrdFileName, string key, int keyLen)...
108       public static void DecryptFile(string encryptrdFileName, string decyptedFileName, string key, int keyLen)...
120   }
121   }
```

Figure B.4: The body of process encryption and decryption with final key length.

The above function is used to encrypt plain file and return array of encrypted byte.

Same as EncrypFile, the is DecrypFile function which is used to decrypt file which encrypted by above function, the following is signature of DecryptFile function

```
public static void DecryptFile(string encryptrdFileName, string decyptedFileName, string key, int keyLen)
```

Body of the function

```
108        public static void DecryptFile(string encryptrdFileName, string decyptedFileName, string key, int keyLen)
109        {
110            string fileEncrypted = encryptrdFileName;
111            string password = key;
112
113            byte[] bytesToBeDecrypted = File.ReadAllBytes(fileEncrypted);
114            byte[] passwordBytes = Encoding.UTF8.GetBytes(password);
115            passwordBytes = SHA256.Create().ComputeHash(passwordBytes);
116
117            byte[] bytesDecrypted = AES_Decrypt(bytesToBeDecrypted, passwordBytes, keyLen);
118            File.WriteAllBytes(decyptedFileName, bytesDecrypted);
119        }
```

Figure B.5: The of process decrypted file.

In line 117 the is a function called AES_Decrypt which is response for decryption process.

This function has three parameters, first byte to decrypt, second password which has been used for encryption and final is key length, the following is signature of the function.

```
52        public static byte[] AES_Decrypt(byte[] bytesToBeDecrypted, byte[] passwordBytes, int keyLen)
```

Body of the function

```
52          public static byte[] AES_Decrypt(byte[] bytesToBeDecrypted, byte[] passwordBytes, int keyLen)
53          {
54              Stopwatch stopWatch = new Stopwatch();
55              stopWatch.Start();
56
57              byte[] decryptedBytes = null;
58
59              // Set your salt here, change it to meet your flavor:
60              // The salt bytes must be at least 8 bytes.
61              byte[] saltBytes = new byte[] { 1, 2, 3, 4, 5, 6, 7, 8 };
62
63              using (MemoryStream ms = new MemoryStream())
64              {
65                  using (RijndaelManaged AES = new RijndaelManaged())
66                  {
67                      AES.KeySize = keyLen;
68                      AES.BlockSize = 128;
69                      var key = new Rfc2898DeriveBytes(passwordBytes, saltBytes, 1000);
70                      AES.Key = key.GetBytes(AES.KeySize / 8);
71                      AES.IV = key.GetBytes(AES.BlockSize / 8);
72
73                      AES.Mode = CipherMode.CBC;
74
75                      using (var cs = new CryptoStream(ms, AES.CreateDecryptor(), CryptoStreamMode.Write))
76                      {
77                          cs.Write(bytesToBeDecrypted, 0, bytesToBeDecrypted.Length);
78                          cs.Close();
79                      }
80                      decryptedBytes = ms.ToArray();
81                  }
82              }
83
84                 TimeSpan ts = stopWatch.Elapsed;
85
86              // Format and display the TimeSpan value.
87                  decryptTime = String.Format("{0:00}:{1:00}:{2:00}.{3:00}",
88                  ts.Hours, ts.Minutes, ts.Seconds,
89                  ts.Milliseconds);
90              return decryptedBytes;
91          }
```

Figure B.6: The body of process decrypted file

Parameters of decoder configuration file can be changed as needed. The above figure 6.8 Contains all parameters of decoder configuration file we can selected and modification of any parameters.

```
###########################################################################################
Files
###########################################################################################
InputFile          = "foreman_qcif.264"       # H.264/AVC coded bitstream
OutputFile         = "foreman_qcif_dec.yuv"   # Output file, YUV/RGB
RefFile            = "foreman_qcif_rec.yuv"   # Ref sequence (for SNR)
WriteUV            = 1            # Write 4:2:0 chroma components for monochrome streams
FileFormat         = 1            # NAL mode (0=Annex B, 1: RTP packets)
RefOffset          = 1            # SNR computation offset
POCScale           = 2            # Poc Scale (1 or 2)
```

Figure B.7: JM19.0 decoder configuration file and decoder control

```
-------------------------------------------------------------------------------
 Input H.264 bitstream                : foreman_qcif.264
 Output decoded YUV                   : foreman_qcif_dec.yuv
 Input reference file                 : foreman_qcif_rec.yuv
-------------------------------------------------------------------------------
Profile IDC  : 100
Image Format : 176x144 (176x144)
Color Format : 4:2:0 (8:8:8)
-------------------------------------------------------------------------------
POC must = frame# or field# for SNRs to be correct
-------------------------------------------------------------------------------
 Frame        POC  Pic#   QP    SnrY     SnrU     SnrV   Y:U:V Time(ms)
-------------------------------------------------------------------------------
00000(IDR)     0    0    36   0.0000   0.0000   0.0000  4:2:0     91
00001( b )     2    4    38   0.0000   0.0000   0.0000  4:2:0     18
00002( B )     4    3    37   0.0000   0.0000   0.0000  4:2:0     28
00003( b )     6    4    38   0.0000   0.0000   0.0000  4:2:0     16
```

Figure B.8: JM decoder output display.

The above figure 6.10 a decoder output display. This figure displays the output of decoder configuration file after we selected the number of frame will be decoded.

CURRICULUM VITAE

Name and Surname: Farhad Mustafa HAJ

Birth Date and Place: 1.1.1984 Dahuk\IRAQ

E-mail address: farhad.hac@gmail.com

EDUCATION:

Technical Diploma Degrees: Dahuk University, 2003-2005

Department of Computer system

Duhok Technical Institute/ IRAQ

Bachelor's Degree: Dahuk University, 2005-2010

Department of Computer science

Duhok/ IRAQ

Master's Degree: Aksaray University, 2015-2017

Department Electrical Electronic and Computer Engineering

Aksaray/ TURKEY