UNIVERSITY FOR DEVELOPMENT STUDIES

DESIGN OF PERCEPTUAL VIDEO ENCRYPTION ALGORITHMS FOR CONTENT PROVIDERS

SALAMUDEEN ALHASSAN

BSc (Comp. Sci.), MSc (Comp. Math.)

Thesis Submitted to the Department of Mathematics,
Faculty of Mathematical Sciences, University for Development Studies in
Partial Fulfilment of the Requirements for the Award of Doctor of Philosophy
Degree in Computational Mathematics

2019

UNIVERSITY FOR DEVELOPMENT STUDIES

DESIGN OF PERCEPTUAL VIDEO ENCRYPTION ALGORITHMS FOR CONTENT PROVIDERS

BY

SALAMUDEEN ALHASSAN

(UDS/DMS/0017/14)

Thesis Submitted to the Department of Mathematics,
Faculty of Mathematical Sciences, University for Development Studies in
Partial Fulfilment of the Requirements for the Award of Doctor of Philosophy
Degree in Computational Mathematics

JULY, 2019

DECLARATION

Student

I, Salamudeen Alhassan, hereby declare that this thesis is the result of my own original work and that no part of it has been presented for another degree in this university or elsewhere:

Candidate's Signature:…………………....……   **Date:**…….……..……

Name: Salamudeen Alhassan

Supervisors'

I hereby declare that the preparation and presentation of the thesis was supervised in accordance with the guidelines on supervision of dissertation laid down by the University for Development Studies.

Principal Supervisor's Signature:…………….….…… **Date:**…..…………….

Name: Dr. Mohammed Muniru Iddrisu

Co-supervisor's Signature:……..….……….….…… **Date:**…..……………….

Name: Dr. Mohammed Ibrahim Daabo

Approved by

Head of Department: Dr. Kwara Nantomah

Signature:……………………………………….   **Date:**…………………….

ABSTRACT

There is high demand by content providers of multimedia services such as Pay-TV News, Pay-per-View (PPV) video and Video-on-Demand (VoD) to expand their customer based through advertisement. However, the threats of unauthorized access, reproduction and re-distribution of the high quality product by illegal users' remains high thereby affecting the returns of the content providers/producers. In this regard, perceptual video cryptosystems have become a highly active research area; developing strategies to achieve the aspirations of content providers/producers while maintaining some level of security. Apart from the limitation in scope of existing perceptual video encryption algorithms, the embedded audio signals in the source video are normally not encrypted and transmission efficiency is not also addressed. Consequently, this thesis employs scrambling and encoding techniques, together with residue number system (RNS) to address these limitations. The first proposal is a perceptual video encryption algorithm that combines rotation and unit anti-diagonal matrices to effectively scramble source video into cipher video with varying degradations in visual quality and enhanced security. The second proposal uses specially formulated orthogonal matrices to encrypt the source video through block encoding. Thirdly, a scheme to secure embedded audio signals is proposed using K-shuffle technique. To crown it all, the thesis proposes an efficient scheme that integrates residue number system (RNS) to the proposed video encryption schemes to ensure efficient transmission of cipher video across networks.

ACKNOWLEDGEMENT

I would like to express my deepest appreciation to my supervisors, Dr. Mohammed Muniru Iddrisu and Dr. Mohammed Ibrahim Daabo for their patience, guidance and immersed supports throughout my study and during the course of writing this thesis. Their thoughtful reminders, inspirations, and intuitive advice have helped me to overcome many challenges and guided me to this end.

In addition, I sincerely appreciate the fruitful encouragements from my brothers, Dr. Alhassan Abdul-Baric and Dr. Salifu Katara during my study.

Finally, I am very grateful to my family especially my wife and children, for their great love, support and understanding at all times.

DEDICATION

This thesis is dedicated to my wife and children and the entire Alhassan Musah family at Nyohini in the Northern Region of Ghana.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

LIST OF ABBREVIATIONS

1D	One-dimensional
2D	Two-dimensional
3D	Three-dimensional
AC	Non-DC (coefficient of the transformation)
AES	Advanced Encryption Standard
AVC	Advance Video Coding
C2DVL	Context-based 2D Variable Length Coding
CBC	Cyclic - Redundancy-Check
DC	The lowest frequency (coefficient of the transformation)
DCT	Discrete Cosine Transform
DES	Data Encryption Standard
DPCM	Difference Pulse Coded Modulation
EZW	Embedded Zerotree Wavelet
HDTV	High Definition Television
ISO	International Standards Organization
ITU	International Telecommunications Union
JPEG	Joint Photographic Experts Group
LFSR	Linear Feedback Shift Register

LIS	List of Insignificant Sets
LSP	List of Significant Pixels
LZSS	Lempel-Ziv-Storer-Szymanski
LZW	Lempel-Ziv-Welch
MCPA	Modulo Carry Propagate Adder
MDCT	Modified Discrete Cosine Transform
MPEG	Motion Picture Expert Group
MRA	Multi-Resolution Analysis
MSE	Mean Square Error
PCM	Pulse-Code Modulation
PSNR	Pixel Signal to Noise Ratio
QP	Quantization Paramater
RC4	Ron's Code 4
RGB	Red-Green-Blue
RNS	Residue Number System
RSA	Rivest-Shamir-Adelman
SAQ	Successive-Approximation Quantization
SPIHT	Set Partitioning in Hierarchical Trees
SVC	Scalable Video Coding

CHAPTER ONE

INTRODUCTION

1.1 Background to the Study

In recent times, perceptual video encryption techniques have gain grounds in multimedia security for video communication over networks. Even though naive (fully-layered) video encryption techniques provide a high level security, its implementation on multimedia services such as VoD, Pay-TV and PPV video is considered computationally expensive both at the encryption and decryption ends. This is partly due to the real-time demands of such services and also the limited computing power of technologies (such as mobile device). Also, naive video encryption techniques encrypt/decrypt every bitstream of the source video. This is considered unwise considering that video data often contains huge amount of redundant data that need to be ignored in the encryption/decryption process.

In sharp contrast to naive video encryption techniques, perceptual video encryption techniques are light-weight and low computational complexity techniques that do not encrypt/decrypt every bitstream of the video (only important portions are encrypted/decrypted). They are most profitable when applied to video of financial interest such as those from VoD, Pay-TV and PPV services. Here, the source video is downgraded in visual quality and delivered to potential viewers as a form of try-before-acquisition of the high quality video. These techniques also offer varied levels of visual degradation from which content providers can choose from to deliver their products. Most importantly, these techniques allow content providers to reach out to more potential clients (through advertisement) with less worry on the security of their products with regard to piracy issues.

The first kind of perceptual video encryption technique codename ‘SECMPEG’ was proposed by Meyer and Gadegast (1995) to secure MPEG-1 video stream using Data Encryption Standard (DES) in Cyclic - Redundancy-Check (CBC). Even though the technique achieved high security, the use of DES meant high computational overheads in its implementation.

After ‘SECMPEG’, later research works relied on choas-based techniques to scramble video data (Yahya, 2008; Pande et al., 2010; Babu & Singh, 2013) or transform-based techiques to encode sensitive information in source video (Au-Yeung et al., 2012b; Zeng et al., 2014; Kirthanaa et al., 2015). In all, researchers seek to gain advantage in increase security, high compression ratio, fast transmission and/or low compupational overheads.

Over the years existing perceptual video encryption techniques have paid little attention to the security of embedded audio signals and the need for enhance transmission of cipher video over networks. Consequently, this thesis will focus on proposing efficient perceptual video encryption algorithms that integrate embedded audio security and transmission efficient of video over networks. The thesis will formulate and implement the algorithms and also conduct some data analysis to measure their performance.

1.2 Problem Statement

Recent advancement in communication and internet technologies has resulted in the generation, collection and transmission of huge amount of multimedia data (images, videos, audios, etc.) over networks. These multimedia data may contain private or confidential information or may be associated with some financial interest that need some form of security. Multimedia data security is thus important for multimedia systems. Conventional cryptographic techniques (Data Encryption Standards (DES), Rivest-Shamir-Adelman (RSA), Advanced Encryption Standard (AES), etc.) previously designed for text data may not be suitable for multimedia applications due to their large data size, data redundancy and real time constraint(Bhargava, Shi, & Wang, 2004).

Security is a compromise between the cost of the content being secured and the cost intruders incur to break the protection. The costs of a multimedia security system include the amount of investment by content providers/producers and the payment required for customer service. As much as content providers do not want to spend so much on their security equipment, data receivers on the other hand do not want to acquire extra hardware just to decode the content. Consequently, encryption algorithms which require less computation and low deployment cost become very attractive (Bhargava et al., 2004).

Owing to the limited processing power, memory and bandwidth, wireless mobile systems cannot handle heavy encryption process. Therefore, video encryption schemes for real-time application must take into consideration various performance parameters such as security, speed, computational efficiency and format compliance in their design. Consequently, light-weight and cost-effective encryption techniques are attractive for multimedia services (Kulkarni et al., 2013).

For videos of financial interest (e.g. movies), content providers want to reach more customers as possible in order to expand their revenue base while keeping the quality products secured from illegal users. Perceptual video encryption algorithms fit perfectly well to bridge this gap. With this technique, video data is degraded in quality (via encryption) and delivered to customers as a form of try-before-acquisition of the original product. Unfortunately, existing works in this area do not address computational complexity, transmission efficiency of cipher video and security of embedded audio signals. Also, most of them are limited to certain video standards (e.g. MPEG-4). Thus, this thesis explores new perceptual video encryption techniques that focuses on the security of audio/video data, computational and transmission efficiencies and applicable to a wide range of multimedia systems. With the proposed techniques, video content providers can deploy low quality versions of their video in any format to the general public for free viewing without worry of any reproduction of the high quality versions by unauthorised users. Consequently, viewers who develop interest in these video will contact the providers for the high quality ones at a fee.

1.3 Research Questions

In order to address the research problem, this thesis seeks to find answers to the following questions;

1) What perceptual video encryption techniques are applicable to a wide range of video standards?
2) What efficient ways can audio signals be secured in perceptual video encryption techniques?
3) What efficient ways can RNS be incorporated into perceptual video encryption techniques to enhance transmission of video across networks?

1.4 Objectives of the study

The general objective of the study is to propose efficient perceptual video encryption/decryption algorithms that integrate audio security and RNS for content providers of financial video.

The specific objectives are to:

- propose efficient perceptual video encryption/decryption algorithms via scrambling and encoding techniques.
- propose efficient audio encryption/decryption algorithms that can be embedded in perceptual video encryption techniques.
- propose an efficient technique that integrates RNS into perceptual video encryption techniques.
- implement and analyze the performance of the proposed algorithms using MATLAB.

1.5 Significance of the Study

The astronomical increase in multimedia services, such as confidential video conferencing, pay-TV News, PPV video and VoD require reliable and cost-effective security in both storage and transmission of multimedia data. Also recent advancement in internet technology has raised concern on the security of multimedia data. For instance, the increase in the use of varied video surveillance technology to monitor traffic and public places has raised concerns regarding the privacy and security of the captured subjects. These security and privacy needs can be fulfilled through encryption of multimedia data so as to discourage malicious attacks from adversaries.

Some multimedia services, (PPV, Pay- TV and VoD), requires the quality of multimedia data to be only partially degraded by encryption. Such perceptibility makes it possible for potential customers to view low-quality forms of the video before acquiring the high quality product. On the other hand, other videos such as those from the military, surveillance camera and medicine, require total concealment of their substance from attackers. In all cases, appropriate video encryption algorithms should be designed to meet the specific needs of these multimedia services without any additional cost in terms of hardware to content providers and legitimate receivers.

Unfortunately, existing perceptual video encryption algorithms are limited in scope and do not address issues on transmission efficiency of cipher video, computational complexity of the algorithm and the security of embedded audio signals in the source video. Therefore, the design of perceptual video encryption algorithms that meet the needs of wide range of multimedia services and real time demands and also maintain acceptable levels of security is paramount.

1.6 Scope and Limitation of the study

This thesis is restricted to the development of perceptual video encryption/decryption algorithms. It also entails the encryption/decryption of embedded audio data in the video. These algorithms are applicable to video with financial interest and which can be delivered to consumers in their low quality forms through encryption as form of advertisement.

1.7 Thesis Outline

The subsequent chapters of the thesis are organised as follows; Chapter Two examines some existing video cryptosystems focusing on perceptual video encryption and decryption schemes. The chapter also discusses some existing works on audio cryptography. In Chapter Three various methodologies that will be employed in this thesis (i.e. images and video data, cryptography, digital audio, the K-shuffle technique and the RNS) are discussed. Results and presentations are captured in Chapter Four where the various proposals together with their detail simulations and analyses are presented. Finally, conclusions and recommendations on the thesis are captured in Chapter Five.

CHAPTER TWO

LITERATURE REVIEW

Advancement in multimedia technology has led to the mass production and distribution of multimedia content across the globe. One of the needs of commercial content providers is reaching wider audience and at the same time hiding the products from unauthorized viewers. The essence of cryptography is to conceal the content of data from illegal persons at the same time allow full access to intended/legitimate viewers. In this chapter, detailed discussions on video data and digital audio are presented in Section 2.1 and Section 2.2 respectively while Section 2.3 and Section 2.4 discusses the K-shuffle technique (which is adapted for the proposed audio encryption algorithm) and the Residue Number System respectively. Furthermore, the chapter examines cryptography in Section 2.5 where a survey of existing video and audio encryption techniques together with their shortcoming are exposed. Finally the chapter looks at cryptanalysis in Section 3.6.

2.1 Video Data

Video data consist of a sequence of images over time. The binary or intensity format of video is a sequence of individual images. However, the Red-Green-Blue (RGB) format of video is a sequence of matrices grouped into sets of three, where each matrix denotes an R, G, or B plane. Digital visual data is typically structured in rectangular arrays represented as frames; the elements of these arrays are represented as pixels (or picture elements). Every pixel presents a numerical value; the weight of the value denotes the intensity of this pixel. The weight of the pixels differs within a predefined range which is typically represented as “bitdepth”, i.e. if the bitdepth is 8 bit, the weight of the pixel span between 0 and 2^8-1. Typical cases are binary images with 1 bit-per-pixel (bpp) only or gray value images with 8 bpp where the gray values span between 0 and 255 (Uhl & Pommer, 2005).

Color is defined by means of several pictures (frames), one for each color channel. The prominent example is the RGB representation, where a full resolution frame is dedicated to each of the colors; red, green, and blue. Video adds a temporal dimension to the purely spatially oriented image data. A video consists of individual frames which are temporally ordered one after the other. An individual video frame may comprise of several frames for different color channels. Visual data embodies huge amounts of data to be kept. Consequently, visual data is typically subjected to compression algorithms after capturing (or digitization). Two classes of compression algorithms exist:

- Lossless compression: these compression algorithms reproduce the original data with same numerical values after decompression.
- Lossy compression: the recovered data is an approximation of the original values after decompression when these compression algorithms are applied.

When compared with lossless algorithms (no data is lost during compression), lossy algorithms (data is lost during compression) achieve much higher compression. The constraints enforced by some application services makes lossless algorithms significant as well (e.g. in the area of medical imaging lossless compression is compulsory in many countries due to legislative reasons). Nevertheless, lossy compression algorithms are more significant in multimedia area; the most typical classification principle is whether the underlying integral transform is the wavelet transform or the discrete cosine transform (DCT) (Uhl & Pommer, 2005).

2.1.1 Discrete Cosine Transform (DCT)-based Systems

- **Joint Photographic Experts Group (JPEG)**

The benchmark arrangement of the JPEG standard operates on 8 × 8 pixels blocks onto which a DCT is utilized. The standardised quantisation matrices are used to quantize the resulting data. The quantised coefficients are subsequently scanned along a zig-zag order where the resulting vector is Huffman and runlength encoded. The JPEG standard also comprises an extended system where several progressive modes are defined and a lossless codes which uses pulse coded modulation (DPCM) based (Uhl & Pommer, 2005).

- **MPEG Video Coding (MPEG-1, 2, 4)**

MPEG motion compensated video coding uses the temporal and spatial correlation among frames in a video sequence to predict the current frame from previously (de)coded ones. The I-frames are compressed as random points in the sequence much similar to JPEG compression.

Every other frames are predicted from decoded I-frames; when a bidirectional temporal prediction is performed the corresponding frames are designated B-frames while simple unidirectional prediction yields P-frames. This prediction fails in some regions (e.g. due to occlusion), consequently the residual amongst this prediction and the current frame being processed is computed and additionally stored after lossy compression. This compression is also like that of JPEG compression however a different quantisation matrix is applied. Block-matching algorithms are commonly used to remove temporal correlation due to its simplicity and effectiveness. In block-matching motion compensation, the scene (i.e. video frame) is typically separated into nonoverlapping “block” regions. In order to estimate the motion, every block in the current frame is compared against the blocks in the search area in the reference frame (i.e. previously encoded and decoded frame) and the motion vector (d_1, d_2) corresponding to the best match is returned. The “best” match of the blocks is identified to be that match giving the minimum mean square error (MSE) of all blocks in search area which is defined as

$$MSE(d_1, d_2) = \frac{1}{N_1 N_2} \sum_{(n_1,n_2)\in\beta} [s_k(n_1, n_2) - \hat{s}_{k-1}(n_1 + d_1, n_2 + d_2)]^2, \tag{2.1}$$

where β denotes a $N_1 \times N_2$ block (the “search area”) for a set of candidate motion vectors (d_1, d_2), s is the current frame and $\hat{s}$ the reference frame. The full search algorithm visits all blocks in the search area to compute the minimum. Many techniques have been proposed to reduce the number of candidate blocks so as to speed up the search process. The primary thought is to present a particular specific search pattern which is recursively applied at the position of the minimal local error. The famous algorithm of this type is called “Three Step Search” which decreases the computational amount considerably at the cost of a suboptimal solution (and therefore a residual with slightly more energy). The block giving the minimal error is stored describing the prediction in term of a motion vector which describes the displacement of the block. The pool of all motion vectors of a frame is called motion vector field.

MPEG-1 has been first defined for storing video on CD-ROM, therefore the data rate and subsequently the video quality is rather low. From the algorithmic viewpoint MPEG, MPEG-2 is very similar; nevertheless the scope is shifted to TV broadcasting and even High Definition Television (HDTV). When compared to MPEG-1 the quality is much higher, moreover methodologies have been standardised to enable scalable video streams and error resilience functionalities.

MPEG-4 expands the scope of the MPEG standards series to natural and synthetic (i.e. computer generated) video and offers technologies for interactive video (i.e. object-based video coding). The main compression engine is also similar to MPEG-2 to provide backward compatibility to some extent. Lastly, MPEG-4 AVC (also denoted H.264 in the International Telecommunications Union (ITU) standards series) increases compression efficiency significantly as compared to MPEG-4 video at an enormous computational cost (Uhl & Pommer, 2005).

- **ITU H.26X Video Conferencing**

The Video conferencing standards of the ITU series are fundamentally the same as the MPEG standards; however, there exists a key difference: video conferencing needs to meet real-time needs. Hence, the most costly piece of video coding (i.e. motion compensation) should be confined. As a result, H.261 characterizes no B-frames as opposed to MPEG-1 and H.263 is additionally less intricate when contrasted with MPEG-2. H.261 and H.263 specifically, offer better quality at low bitrates when contrasted with their MPEG partners. H.261 has been characterized to help video conferencing over ISDN, H.263 over PSTN which suggests the interest for even lower bitrates in

H.263. The most recent standard in this series is H.264 which has been planned by the JVT (joint video group) and is indistinguishable from MPEG-4 AVC. This algorithm utilizes a 4 × 4 pixels integer and multi-frame motion compensation. Along these lines, this algorithm is extremely difficult from a computational perspective. (Uhl & Pommer, 2005)

2.1.2 Wavelet-based Systems

Wavelet transform depends on multi-resolution analysis (MRA) and the backbone of picture compression techniques. Picture compression techniques have been productive in giving high compression proportions while safeguarding great picture quality, and have ended up being great contenders to DCT based compression techniques. Second generation wavelet compression plans endeavor to exploit inter-subband relationship – the most conspicuous algorithm around there are zerotree encoding and hybrid fractal wavelet codecs. In the vast majority of these techniques, compression is achieved by applying a quick wavelet transform to decorrelate the picture information, quantising the subsequent transform coefficient and coding the quantised qualities considering the high inter-subband relationships.

The quick wavelet transform can be productively executed by a couple of suitably structured Quadrature Mirror Filters (QMF). Along these lines, wavelet-based picture compression can be seen as a type of subband coding. A 1-D wavelet transform of a signal is done by convolving with both QMF's and down examining by 2; since it is limited, one must settle on some decision about what esteems to cushion the augmentations with. This activity breaks down the original signal into two subbands, which are frequently meant as coarse scale approximation (lowpass subband) and detail signal (highpass subband). Iteratively, the same procedure is performed on the coarse scale approximations a few times. (Uhl & Pommer, 2005).

- **Set Partitioning in Hierarchical Trees (SPIHT)**

It is detected that the coefficients calculated by wavelet decomposition include a high degree of spatial self-similarity across all sub-bands. By concentrating on this similarity, a more effective coefficient representation can be found which is exploited by all second generation wavelet coding schemes. SPIHT employs a spatial orientation tree. This data structure is very like the zerotree structure employ by the Embedded Zerotree Wavelet (EZW) algorithm where each value in the wavelet multi-resolution pyramid is allocated to a node of the tree.

Three lists are employed to represent the image information: The LIS (list of insignificant sets), the LIP (list of insignificant pixels), and the LSP (list of significant pixels). The latter list contains the sorted coefficients which are stored. The SPIHT codec generates an embedded bitstream and is enhanced for encoding speed. Even though SPIHT is not a standard it has been the state of the art codec by which new image compression system was compared. (Uhl & Pommer, 2005)

- **JPEG 2000**

Taubman's coding scheme "Embedded Block Coding with Optimised Truncation" (EBCOT) serves as a base from which the JPEG 2000 image coding standard is designed. The main difference between earlier proposed wavelet-based image compression algorithms (such as EZW or SPIHT) is that EBCOT as well as JPEG 2000 work on independent, nonoverlapping blocks which are coded in numerous bit layers to create an embedded, scalable bitstream. Rather than zerotrees, the JPEG 2000 standard relies upon a per-block quad-tree structure since the entirely autonomous block coding system blocks structures crosswise over subbands or even code-squares. These free code-blocks are passed down a "coding pipeline" and produce separate bitstreams (Tier-1 coding). Transmitting each bit layer matches specific mutilation level. The dividing of the accessible bit budget between the code-blocks and layers ("truncation focuses") is resolved utilizing a complex optimisation technique for ideal rate/twisting performance (Tier-2 coding). The fundamental structural objectives behind EBCOT and JPEG 2000 are versatility and flexibility which are accomplished to a large degree by the autonomous processing and coding of picture blocks, and obviously to give a codec a superior rate-distortion execution than the broadly utilized JPEG, particularly at lower bitrates. In JPEG 2000, the default is to do five-level wavelet segmentation with 7/9-biorthogonal channels and after that fragment the changed picture into non-covering code-squares of close to 4096 coefficients which are passed down the coding pipeline (Uhl & Pommer, 2005).

- **Lossless Formats: JBIG, GIF, PNG**

A single compression technique in its pure form is often employed by lossless techniques. This is in direct contrast to most lossy compression techniques where several algorithms (e.g., transformation, quantisation, coding) are combined. DPCM codec is applied in Lossless JPEG. Howerver, PNG and GIF together employ dictionary coding as the base technique; Lempel-Ziv-Welch (LZW) coding is applied in GIF while Lempel-Ziv-Storer-Szymanski

(LZSS) coding is used in PNG. For compressing bitplanes, JBIG employs context-based binary arithmetic coding (Uhl & Pommer, 2005)

2.2 Digital Audio

Digital audio is sound signal that is recorded or converted into digital form. In digital audio, sound wave of audio signal is encoded as numerical samples in continuous sequence, notably at compact disc (CD) audio quality. Pulse Code Modulation (PCM) is a quantizing/digitizing algorithm which converts analog signals to digital samples (Shahid, Chaumont & Puech, 2010).

Digital audio technologies record, manipulate, produce and distribute sound such as songs, instrumental pieces, podcasts, and others. The quality of audio sample can be determined by three parameter; sample resolution (the number of bits per sample), sampling rate (the number of times per second the analog wave form was read to collect data) and number of audio channel sampled (one or two channels).

Audio file formats exist for both compressed (reduce file size) and uncompressed (raw) form. Audio file formats can be identified by the file extensions. Examples include; .wav (Microsoft waveform), .au (sun microsystems), .mp3 (MPEG Layer III Audio), e.t.c.

In order to protect and preserve the content of audio data, various schemes such as steganography (https://en.wikipedia.org/wiki/Digital_audio), watermarking (https://en.wikipedia.org/wiki/Audio_file_format) and cryptography (Farkash et al., 1991; Borujeni, 2000; Raghunandhan et al., 2013; Tamimi & Abdalla, 2014; Makwana & Parmar, 2014; Pitale et al., 2015) have been advanced by researchers.

2.3 K-shuffle

A deck of 52 cards when broken into two piles of 26 cards each can be reordered to get back its original arrangement by alternatively taking cards from each pile in turns in 8 different repetitions. This is known as a perfect 2-shuffle and often referred to as a faro shuffle. Two types of faro or perfect 2-shuffles exist; the first shuffle also called an out shuffle leaves the top card at the top and the second shuffle (in shuffle) is where the top card becomes the second card (Diaconis et al., 1983; Packard & Packard, 1994; Ramnath & Scully, 1996; Madain et al., 2014;Lima & da Silva Neto, 2016).

Let $n, m \in \mathbb{Z}$ and $n, m > 1$. Given $l = nm$ cards are orderly numbered from 1 to l. Place the cards in m piles of n cards each in order as follows; the first pile have cards from 1 through n, the second pile containing cards $n + 1$ through $2n$, the third pile containing cards $2n + 1$ through $3n$, etc., and the last pile containing cards $(m - l)n + l$ up to nm. A perfect $k -$shuffle as described by Packard and Packard (1994) re-arrange the cards as follows; take the first card in each pile, follow by the second, then the third, etc., and ending with the last card in each pile. After the re-arrangement, the first and the last card remain at their respective positions. Packard & Packard described the order of the $k -$shuffle; $d_k(n)$, to be the minimum number of times the $k -$ shuffle needs to be recycled to return the cards to their original arrangement (Persi et al., 1983; Packard & Packard, 1994; Ellis, Fan & Shallit, 2002; Steve, Persi & Ron, 2014).

Consider $m = 4$ and $n = 5$, the 4-piles of numbers from 1 to 20 after one 4-shuffle is shown in Figure 2.3.1.

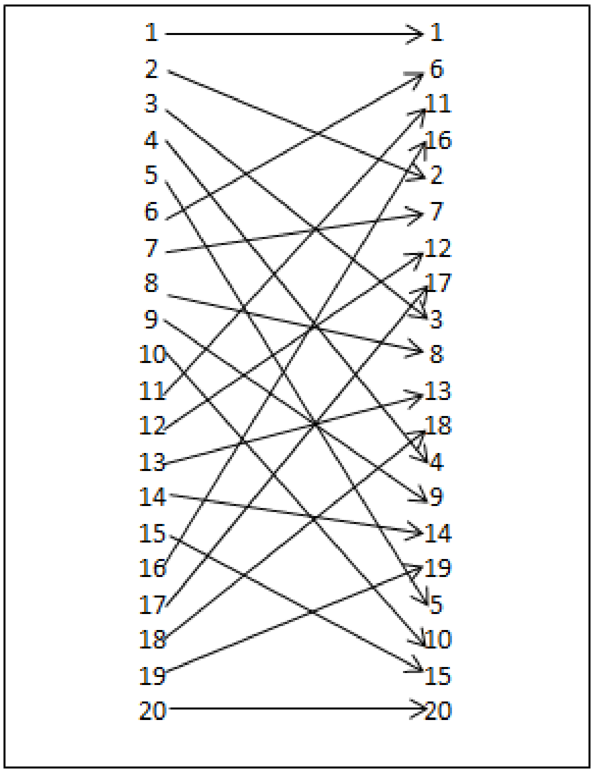


Figure 2.3.1 One 4-Schuffle of numbers from 1 to 20

Beyond traditional card shuffling, perfect shuffle algorithms have been employed in areas such a cryptography (Diaconis et al., 1983; Medvedoff & Morrison, 1987), networking (Ernastuti, 2014; Sultana & Shubhangi, 2017) and parallel processing (Duato et al., 2002).

2.4 Residue Number System

Residue number system is described as a non-weighted number system having numerous benefits in numerical computations. The inherent features in RNS such as the digit-to-digit computations, parallelism, fault tolerance, high computational speed (Jarvinen et al., 2001) and low power dissipation make it ideal for implementation in fields of communication (Stone, 1971; Chervyakov et al., 2015; Daabo et al., 2016), Digital Signal Processing (DSP) (Kosek et al., 1989; Yang et al., 2011; Singh, 2014), intensive computations such as digital filtering, correlations, convolutions, direct digital frequency synthesis (Baraniecka & Jullien, 1978), Discrete Fourier Transform (DFT) computations (Ramirez et al., 2001), Fast Fourier Transform (FFT) computations, image processing (Mehrin et al., 2011; Omondi & Premkumar, 2007; Alhassan & Gbolagade, 2013; Wang et al., 2014; Daabo et al., 2016) and cryptography (Taylor, 1990; Shetty et al., 2017; Yao, 2018).

RNS are based on the congruence relation. Two integers x and y are said to be congruent modulo m if m divides exactly the difference of x and y; Mathematically (Omondi & Premkumar, 2007),

$$x \equiv y \ (mod\ m). \qquad (2.4.1)$$

If q and r are the quotient and remainder, respectively, of the integer division of Y by m that is, $Y = q.m + r$ then, by definition $Y \equiv r \ (mod\ m)$. The relationship between r, Y and m is given by (Omondi & Premkumar, 2007)

$$r = |Y|_m. \qquad (2.4.2)$$

RNS is determined by the set $S = \{m_1, m_2, \dots, m_N\}$ of N positive and pairwise relatively prime moduli where the dynamic range M is the product of the moduli m_i. Thus,

$$M = \prod_{i=1}^{N} m_i. \qquad (2.4.3)$$

With this, every integer Y in $[0, M-1]$ has a unique representation $(y_1, y_2, \dots, y_N)$ in S. The set S and the numbers y_i are respectively called the moduli set and residue of Y modulo m_i (Omondi & Premkumar, 2007; Mehrin et al., 2011; Alhassan & Gbolagade, 2013; Daabo et al., 2016; Bello & Gbolagade, 2017).

2.4.1 RNS to Binary Conversion

Two traditional techniques are widely used for RNS to binary reverse conversion; the Chinese Remainder Theorem (CRT) and the Mixed Radix Conversion (MRC).

- Chinese Remainder Theorem (CRT)

Consider the moduli set $\{m_1, m_2, m_3, \ldots, m_N\}$ with $\gcd(m_i, m_j) = 1$ for $i \neq j$ and dynamic range $M = \prod_{i=1}^{N} m_i$, the CRT for an integer Y having RNS representation $(x_1, x_2, x_3, \ldots, x_N)$ is reversed converted by

$$Y = \left| \sum_{i=1}^{N} M_i \left| {M_i}^{-1} x_i \right|_{m_i} \right|_M , \tag{2.4.4}$$

where $M_i = \frac{M}{m_i}$ and ${M_i}^{-1}$ is the multiplicative inverse of M_i with respect to m_i (Wang, 2000; Wang et al., 2002; Lu, 2004; Omondi & Premkumar, 2007; Mehrin et al., 2011).

- The Mixed Radix Conversion (MRC)

Suppose the moduli compose of positive pairwise relative prime integers is being ordered as $m_N > m_{N-1} > \cdots > m_1$. A nonnegative operand Y in the range of $[0, M-1]$ in mixed radix representation is given by

$$Y \leftrightarrow < \hat{y}_N, \hat{y}_{N-1}, \ldots, \hat{y}_1 >,$$

where

$$Y = \hat{y}_N \prod_{i=1}^{N-1} m_i + \hat{y}_{N-1} \prod_{i=1}^{N-2} m_i + - \cdots + \hat{y}_2 m_1 + \hat{y}_1, \tag{2.4.5}$$

and $0 \leq \hat{y}_i < m_i$ for all i (Meyer-Base et al., 1997; Lu, 2004; Omondi & Premkumar, 2007; Mehrin et al., 2011).

Apart from the traditional techniques, other researchers have proposed much faster and efficient RNS to binary reverse convertor for particular moduli sets (Jenkins, 1978; Ibrahim & Saloum, 1988; Gallaher et al., 1997; Bhardwaj et al., 1998; Mathew et al., 1999; Zarei & Askarzadeh, 2010; Low & Chang, 2011; Mehrin et al., 2011; Banhanfar & Zarei, 2013; Kaushik & Srivastava, 2013; Hiasat, 2018). However, most of them are simple modification of the CRT or MRC techniques (Piestrak, 1994; Piestrak, 1995; Persson, 2009).

2.5 Cryptography

Cryptography (practiced by cryptographers) is the art and science of keeping messages secure. On the other hand, the art and science of breaking ciphertext to reveal its substance is called cryptanalysis (carried out by cryptanalysts). A system for enciphering and deciphering data is a cryptosystem. Enciphering involves an algorithm for combining the plaintext with one or more cipher keys (which are secret numbers or strings of characters known only to the sender and/or recipient). The branch of mathematics encompassing both cryptography and cryptanalysis is cryptology.

Two main classes of cryptosystems are; public key methods that use two different keys for encryption and decryption and secret key methods that use the same key for encryption and decryption (Schneier, 1996; Menezes et al., 1996; Stallings, 2006; Raphael & Sundaram, 2010).

“Plaintext” is an original message and its encoded form is called the “ciphertext”. The process of disguising a “plaintext” in such a way as to conceal its substance is encryption (encoding or enciphering). On the other hand the process of turning “ciphertext” back into “plaintext” is decryption (decoding or deciphering). The encryption and decryption process is shown in Figure 2.5.1.

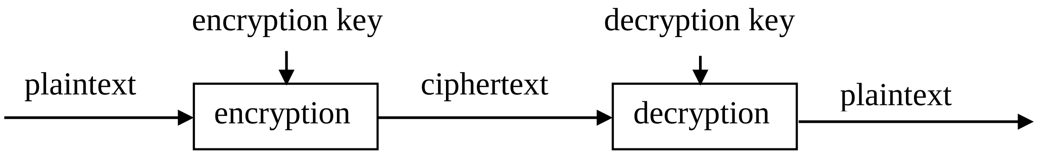


Figure 2.5.1 Encryption and Decryption

In cryptographic algorithms, plaintext is denoted by ***M***, for message, or ***P***, for plaintext. This can be a stream of bits, a text file, a bitmap, a stream of digitized voice, a digital video or image. The plaintext can be intended for either transmission or storage. ***M*** is the message to be encrypted. Ciphertext is denoted by ***C*** which may be the same size as ***M*** or larger or smaller (that is combining encryption with compression). The encryption function ***E***, operates on ***M*** to produce ***C***. Mathematically

$$E(M) = C. \tag{2.5.1}$$

Also, the decryption function ***D*** operates on ***C*** to produce ***M*** as

$$D(C) = M. \tag{2.5.2}$$

The goal of encrypting and decrypting a message is to recover the original plaintext, hence the following identity holds true, ***D*(*E*(*M*)) = *M***.

A cryptographic algorithm is the mathematical function used for enciphering and deciphering. Two functions are generally considered: one for encryption and the other for decryption. The security of a restricted algorithm is largely based on keeping the way it works secrete as such a big or varying group of users cannot use it, since every time a user leaves the group everyone else must change to a different algorithm. Quality control or standardization is not permitted in restricted algorithms and also, every group of users must have their own distinctive algorithm. Modern cryptography resolves this problem with a cipher key, denoted by **K**. The cipher key might be any one of a large number of values. The collection of possible values of the key is called the key space and both encryption and decryption techniques use it. A cryptosystem encompasses an algorithm, all possible plaintexts, ciphertexts, and cipher keys. Two general types of key-based algorithms and known: symmetric and asymmetric (public-key) (Schneier, 1996; Stallings, 2006).

2.5.1 Symmetric Algorithms

Also called conventional algorithms, symmetric algorithms are those algorithms where the enciphering key can be computed from the deciphering key and vice versa. Most symmetric algorithms have the same key for enciphering and deciphering and the sender and receiver must agree to this key before they can communicate. The security of a symmetric algorithm resides in the key; exposing the key means that anyone could encipher and decipher messages. The key must remain secret as long as the communication needs to remain secret. Enciphering and deciphering with a symmetric algorithm are denoted by

$$E_K\,(M) = C, \tag{2.5.3}$$

$$D_K(C) = M. \tag{2.5.4}$$

Symmetric algorithms can be categorized into two. Stream algorithms (stream ciphers) work on the plaintext a single bit (or sometimes byte) at a time while block algorithms (block ciphers) work on the plaintext in groups of bits (Schneier, 1996).

2.5.2 Public-key Algorithms

The design of public-key algorithms (also called asymmetric algorithms) is such a way that the key used for enciphering is different from the key used for deciphering. Additionally, the deciphering key cannot (at least in any reasonable amount of time) be computed from the enciphering key. These algorithms are called “public-key” because the enciphering key can be made public: anyone can use the enciphering key to encrypt a message, but only a specific person with the corresponding deciphering key can decrypt the message. In these systems, the enciphering key is normally referred to as the public key, and the deciphering key is called the private key (secret key).

Encryption using public key **K** is denoted by

$$E_K(M) = C. \tag{2.5.5}$$

Even though the public key and private key are different, decryption with the secret key is denoted by

$$D_K(C) = M. \quad (2.5.6)$$

Sometimes, messages will be encrypted with the private key and decrypted with the public key (Schneier, 1996).

Menezes et al. (1996) broadly classified the objectives of cryptography into four main goals:

- Confidentiality (secrecy or privacy) is a facility used to obscures the content of information from all except for intended users. This can range from physical protection to mathematical algorithms so as to render the data incomprehensible.
- Data integrity is a service that resolves the illegal alteration of data by having the capacity to detect data manipulation by such groups (unauthorized parties).
- Authentication is a facility that consents identification and its function applies to both entities (entity authentication) and information (data origin authentication). Two parties entering into a communication should identify each other and information transmitted over a channel should be authenticated as to the origin, date and time sent, data content, among others.
- Non-repudiation is a service that avoids an entity from refusing previous obligations. A means to resolve disputes arising from a party denying certain actions were previously taken is necessary. This may require a process involving a trusted third party in order to resolve such disputes.

2.5.3 Video Encryption Algorithms

Video encryption algorithms are broadly classified into three (3) methods (Uhl & Pommer, 2005; Wong & Bishop, 2006; Pande & Zambreno, 2013). These are;

2.5.3.1 Completely-Layered Encryption (Full-encryption)

This class of algorithm encrypts the entire video bitstream. The naive technique is a kind of full encryption technique in which a conventional cryptosystem (DES, AES and RSA) is used in the encryption process. This technique is not applicable in real-time video application due to its heavy computation needs and low speed (Bhargava et al., 2004; Pande et al., 2010).

2.5.3.2 Selective/Partial Encryption.

The selective/partial encryption technique consists of all algorithms that perform encryption on selected bitstream in order to save computational complexity. Spatially selective encryption is a special type of selective enciphering approach where the bits are selected based on spatial data (Bergeron & Lamy-Bergot, 2005; Amir & Shahrokh, 2006; Dubois et al., 2014).

2.5.3.3 Perceptual Encryption

After encryption using the perceptual encryption technique, low-quality perceptual data of the video content is conserved. The plain video of original visual quality is accessible after decrypting. This technique allows the audio/video quality to be controlled. Perceptual encryption techniques are of low-security in terms of content confidentiality (Watanabe, 2015; Au-Yeung et al., 2011a, 2011b, 2012a, 2012b; Wang et al., 2013; Bernatin et al., 2016; Ding et al., 2017).

2.5.3.4 Features of Video Encryption Algorithms

Some performance parameters that every good video encryption technique should possess are (Uhl & Pommer, 2005; Wong & Bishop, 2006);

- Format-compliance: The ability of the enciphered image/video to be decodable at the receiver's end without the knowledge about the deciphering key. Enciphered bit stream must be compliant with the compressor used and also, a standard decoder should be able to decode the encrypted image/video.
- Scalability: This means multi-level security for varied services with flexible parameter settings. The basic idea to realize scalability is to encipher partial layers and/or partial data in selected layers. Scalability can be viewed as a control mechanism for the visual quality of the enciphered image/video.
- Perceptibility: Partial encryption of visible content of the source image/video, which is suitable for pay-after-trial services of digital multimedia, such as Pay-TV and VoD services.
- Error-tolerability: The original image/video should be recoverable when some bit errors are introduced into the cipher.
- Security: The algorithm should resist attacks such as brute-force and known-plaintext attack.
- Speed: This is very important in image/video cryptosystem especially for real-time applications. Encryption and decryption time should be as small as possible.

2.5.3.5 Application Areas of Video Encryption Algorithms

Video is widely used in many areas of human Endeavour as stated in Section 2.1. Examples of systems which need some form of encryption support to accomplish the desired respective functionalities and securities:

- **Telemedicine**: Protection of confidential patient medical image records when stored in databases and transmitted over networks of any kind.
- **Video Conferencing**: Protecting of video conferences between individuals and companies from potential attackers.
- **Surveillance**: The astronomical growth of surveillance systems which typically record and store visual data may contain critical private information of the persons being recorded and need to be protected from eavesdroppers in order to preserve basic citizens' rights.
- **Video-on-Demand (VoD)**: This is an entertainment application in which movies are broadcasted from a VoD server to a client after it has been demanded by the client. The client normally pays a monthly subscription rate or a pay-per-view basis in order to access the video server. In any case, to secure the revenue for the investments of the VoD company, the transmitted movies have to be secured throughout transmission to guard them from non-paying attackers. Furthermore, some ways are needed to disable a legitimate consumer to transfer the movies to a non-paying friend or, even worse, to record the flicks, burn them onto video versatile disc (DVD) and sell these products in massive quantities.

- **Pay-TV News**: As oppose to Free-TV, Pay-TV is funded by the subscription payments of the clients. This means that Pay-TV channels are consumed by only consumers who have paid their subscription fees. The broadcasted content is normally encrypted by the content provider and decrypted by the clients' set-top box using smartcard technology (Uhl & Pommer, 2005).

2.5.3.6 The need for Video Encryption Algorithms

Furht and Kirovski (2014) pointed out that the simplest way to encrypt an image or a video is to consider the 2-dimensional/3-dimensional (2D/3D) stream as a 1-dimensioanl (1D) data stream, and then encrypt the 1D stream with any available cipher. They called this simple idea of encryption as naive encryption. Naive encryption is adequate to protect multimedia content in some civil applications. Furht and Kirovski (2014) outlined issues to be considered in the design of advanced encryption algorithms for multimedia content as follows:

- **Tradeoff between bulky data and slow speed**: Even in the compressed form digital video are generally huge in size. Hence, it is extremely difficult to achieve both fast and secure real-time encryption simultaneously with traditional cryptosystems on huge data.
- **Tradeoff between encryption and compression**: The randomness of ciphertexts can drastically reduce the compression strength if encryption is performed before compression. Thus, one need to apply encryption after compression, however the special and numerous image/video structures make it difficult to insert an encryption algorithm into the integrated system. As an example, some common

compression standards (such as MPEG-x) are hostile to selective encryption. Thus, there exist remarkable tradeoffs between the compression and also the encryption.

- **Dependence of encryption on compression**: Lossy compression technique is largely implemented for images/videos to dramatically scale back the size of the information for enciphering. It is therefore natural to expect that the design and implementation of quick enciphering schemes are needed.
- **Inability of compression to reduce data size**: Due to legal issues in some application such as medical imaging, lossy compression of images/videos is unacceptable. In such situations, lossless compression or leaving the images/videos uncompressed may be the only choice.
- **Intractable high redundancy**: High redundancy exists in uncompressed images/videos which may hinder ciphers running in Electronic Code Book (ECB) mode to hide all inherent information in the source images/videos. Suppose an image contains an area of fixed color (i.e. large redundancy) then the edge will be approximately preserved after enciphering. This is because those successive identical pixels lead to the same repeated patterns when a block cipher is implemented in ECB mode.
- **Loss of the avalanche property**: Seemingly, a straight implementation of traditional ciphers in image and video.

2.5.4 Related Works on Perceptual Video Encryption Algorithms

Perceptual video encryption techniques retain some comprehensive content of the source video after encryption. To this end, research works (Yahya et al., 2008; Pande et al., 2010; Babu & Singh, 2013) rely on choas-based techniques to scramble video data. Others (Au-Yeung et al., 2012b; Zeng et al., 2014; Kirthanaa et al., 2015) achieved this by utilising transform-based techiques to encode sensitive information in source video. This subsection presents a survey of some exiting perceptual video encryption algorithms and seeks to highlight some of the inherent problems.

Meyer and Gadegast (1995) presented the concept of selective encryption technique named 'Secure MPEG' (SECMPEG) to secure high-volume video-signal of MPEG-1 video standard (Furht et al., 2005). Their method uses Data Encryption Standard (DES) in Cyclic - Redundancy-Check (CBC) mode to encrypt MPEG video stream. Four levels of security are implemented; (1) Entire stream headers encryption. (2) Entire stream headers and all DC and lower AC coefficients of intra-coded blocks encryption. (3) Encrypting I-frames and all I-blocks in P- and B frames. (4) Encrypting entirely the bit streams. The number of I blocks in P or B frames can be of the same order as the number of I blocks in I frames. It is indicative to not that encryption ratio may vary based on which parameters are encrypted. A very less encryption ratio occurs when only headers are encrypted. On the other hand, 100% encryption ratio results when all bitstreams are encrypted. The number of parameters encrypted and the use of traditional algorithms have a great impact on the speed of the technique. The method ensures varied security levels but encrypting only stream headers is easily predictable. Also, encrypting all the bitstreams can results in high security at the expense of a special encoder and decoder. Moreover, the audio component of cipher video is not encrypted and efficient transmission of cipher video across networks is not also considered.

Spinsante et al. (2005) proposed a partial encryption scheme that utilizes a Linear Feedback Shift Register (LFSR) to generate cipher key that are EX-ORed with the source video for H.264/AVC coding standard in order to degrade the quality of video. The effects of selectively encryption of Quantization Paramater (QP), the Deblocking Filter coefficients and the information on the selected Intra prediction mode of the luma (Y) components of video produce were tested. The LFSR generates a 56 bytes long cipher key for encryption. The parameter chosen for encryption at the i-th coding step is EX-ORed with a numerical value, depending on the i-th bit of the cipher key. The scheme achieved good results with increased degradation of the source video when all the parameters are combined into a single ciphering process. Nevertheless, this scheme also failed to address transmission efficiency of cipher video and mechanisms to secure embedded audio content.

Wong and Bishop (2006) proposed a selective video encryption algorithm that utilizes multiple stream ciphers and a unique multi-key mechanism to enhance security while retaining efficiency and format compliance which is ideal for consumer devices. Computational overhead is only reduced when partial video encryption is performed. This algorithm is well suited for parallel hardware implementation. However, apart from its limited scope (applicable to only video block-transform based compression technique), the algorithm lacks support for embedded audio encryption and efficient means of transmission of cipher video.

Socek et al. (2007) in their correlation-preserving encryption (CPE) scheme proposed a video encryption technique of reasonable security and compression compliant. The scheme has support for both lossless and lossy low-motion spatial-only video codecs. They extended the approach to cover digital video steganography to disguise a given video with another video. The drawbacks of the scheme are similar to those outlined above. The remaining review works contain the same challenges.

Li et al. (2007) proposed a perceptual video encryption algorithm which selectively encrypts fixed-length codewords (FLC) in MPEG-video bitstreams under the control of three perceptibility factors (P_{sr}, P_{sd} and P_{mv}). Intra DC coefficient are encrypted with probability P_{sr}, sign bits of non-zero discrete cosine transform (DCT) coefficients and ESCAPE DCT coefficients are encrypted with probability P_{sd} and sign bit and residual of motion vectors are encrypted with probability P_{mv}. Their scheme provides useful features such as strict size-preservation, on-the-fly encryption and multiple perceptibilities which can be used to support more applications with different requirements.

Wang et al. (2008) proposed a perceptual video encryption scheme by exploiting the special feature of entropy coding in H.264. The enciphering algorithm consists of coded block pattern permutation, sign of trailing ones scrambling and levels of nonzero coefficients encryption. The scheme chooses important syntax elements and sensitive coded elements to encrypt, using mathematical operations, permutation and stream ciphers. Their test results and analyses revealed that the scheme can provide some reasonable level of security with low computational complexity and running cost, and have a little effect on the compression ratio and transmission bandwidth.

Au-Yeung et al. (2009) proposed a novel partial video encryption scheme that embeds encryption at the transform stage of video encoding. A number of new unitary transforms which are equally efficient and can substitute DCT are developed and used as alternates to DCT during the encoding process. Partial encryption is achieved by applying in turns the new transform to individual blocks using pre-designed cipher key. They conducted experimental results on H.264 codec and showed that their proposed algorithm provides a sufficient level of encryption without affecting the overall video quality and requires a few overhead-bits.

Au-Yeung et al. (2010a) extended the work on partial video encryption using alternative transforms (Au-Yeung et al., 2009) to integer-based alternative transforms by modifying floating-point based transforms to be compatible with H.264 standard. The proposed encryption technique is divided into three parts: design of alternative transforms, random key generation, and the application of alternative transforms according to the cipher key. Design of integer-based alternative transforms is achieved using rotation angles at $(7\pi/24, 8\pi\ /24)$ to yield the transform matrix shown in Equation (2.5.7),

$$\begin{bmatrix} 0.4305 & 0.5610 & 0.5610 & 0.4305 \\ 0.6124 & 0.3536 & -03536 & -0.6124 \\ 0.5610 & -0.4305 & -0.4305 & 0.5610 \\ 0.3536 & -0.6124 & 0.6124 & -0.3536 \end{bmatrix} = \begin{bmatrix} d & e & e & d \\ f & g & -g & -f \\ e & -d & -d & e \\ g & -f & f & -g \end{bmatrix}. \tag{2.5.7}$$

RC4 is used as the random key generator. The security of the proposed scheme is enhanced by applying different transforms for horizontal rows and vertical columns where Step 3 of their previous work (Au-Yeung et al., 2009) is modified.

Au-Yeung et al. (2010b) further proposed a perceptual video encryption algorithm using alternative unitary transforms to solve the problem of sign-flip of the DC component in each block that seems to overwhelm the impact of applying alternative transforms in their previous work. A more detailed study of the selection of rotation angles is made which yielded selected angles that make the sign-flip of each DC component unnecessary. The proposed encryption algorithm consists of two steps: random (secrete) key generation and alternating transforms according to the secrete key. Their experimental results and analyses using H.264 codec showed that the coding efficiency of using the newly-selected rotation angles remains unchanged, the R-D performance increases slightly when the key is known and had a better encryption capability.

Shahid et al. (2010) in their paper addressed a problem of compression and selective encryption (SE) for AVS part-2 Jizhun profile. The method is implemented in the context-based 2D variable length coding (C2DVLC) module of video codec. SE is implemented by the use of AES algorithm with Cipher Feedback (CFB) mode on a subset of codewords while C2DVLC acts as the encryption step without affecting the coding efficiency of AVS. Real-time constraints are effectively handled by having exactly the same bitrate when the method is used. Their experimental results showed that their method accomplishes the anticipated level of encryption in each frame, while retaining the full AVS video coding standard compliance under a least set of computational requirements. However, the use of AES is considered computation expensive in both encryption and decryption techniques.

Magli et al. (2011) proposed three transparent video encryption algorithms for H.264/AVC and H.264/SVC compressed video. Two of the proposals employ the knowledge of generating controlled drift to regulate the visual quality of H.264/AVC video while the third utilizes scalable video coding to encrypt H.264/SVC video. In proposed algorithm 1, encryption is performed on each I-coded luminance macro-block (MB) separately in both I- and P-frames according to the quantized DCT values of a 4×4 block. The proposed algorithm 2 creates I-frames with two quality levels and creates drift using the mismatch between referenced and decoded I-frames for encryption. The proposed algorithm 3 handles any combination of temporal, spatial and quality scalability of the H.264/SVC standard and employs AES to encrypt the layers. Their test results showed that proposed algorithm 1 provides the less overhead (an advantage) and require external compression tools (a drawback), algorithm 2 provides full syntax-complaint after encryption (an attractive choice) but will require an external arithmetic coder to achieve a very small overhead which will complicate the system design. Performance of algorithm 3 is good especially for low-motion and low-frame-rate sequences.

Chandra et al. (2012) proposed an effective and generalized scheme for video encryption and decryption of video-frame. Their method is based on a matrix computation scheme which uses a concept of video-frame and XOR ($\oplus$) operation. The scheme collects all video frames (I-frame) and a key image frame for encryption and decryption. In order to improve the security of the scheme the I-frame blocks in P- and B-frames are encrypted. In addition to the limitations outlined initially, this method is applicable for only video on storage and not for real time video applications.

Kulkarni et al. (2013) proposed a selective algorithm for video compressed with H.264 format. In their algorithm, video frames along with audio are shuffled and AES is used to selectively encrypt the sensitive video codewords. A video with its audio information is first divided into frames (using a frame cutter) and then shuffled with a shuffling block. The shuffled frames are passed on to a frame stitching block to construct the cipher video. The use of AES algorithms in their proposed method provided it a high level of security and it could prevent unauthorized users from viewing the video file. A comparison of their method and two other methods; Simple Permutation algorithm and Crisscross Permutation algorithm where conducted. They showed that, their method and Simple Permutation algorithm have advantage over Crisscross Permutation algorithm in terms of security. Also, in terms of speed Crisscross Permutation algorithm and their method have advantage over Simple Permutation algorithm. Even though embedded audio is encrypted, this scheme is applicable to only video on storage and third-party software is needed to perform the frame cutting/dividing process.

Zeng et al. (2014) generalized sign-flipping technique by randomly embedding sign-flips into all stages of the DCTs butterfly structure to increase the encryption space for higher security. Their work on perceptual encryption of H.264 videos extended the study on integer DCT of size 4 × 4. The work first examined separable application of the 4 × 4 2D DCT in which different sign-flipping strategies are employed along its horizontal and vertical dimensions. The next step converts the 4 × 4 2D DCT into a 16-point 1D butterfly structure so that more sign-flips can be inserted at its various stages. In the last step, different schemes are selected to pair the node-variables in the 16-point 1D butterfly structure to further widen the encryption space. Their experimental results showed that the scheme achieve larger key space and can resist various cryptographic attacks.

Kirthanaa et al. (2015) proposed an improved perceptual video encryption and decryption using S-transform and extended the work by analyzing the selection of rotation of blocks to be used to generate the S- transforms. In the encryption process each block of video frame from a video file is extracted and encrypted using a cover image and applying DCT to each block. The DCT encrypted blocks are rotated using S-transformation in a swap function and transmitted. Their experimental results and analyses with the standard H.264 codec showed that the proposed scheme has a higher encrypting capability and does not affect the coding efficiency of the codec.

Bernatin et al. (2016) proposed a perceptual video encryption scheme using Set Partitioning in Hierarchical Tree (SPIHT) transform. The SPIHT scheme depends on wavelet coding method and all frames (images) are converted using discrete wavelet transform (DCT). At the start of the encryption process, video frame (image) is divided into four sub bands (three sub bands and one low frequency sub band) and cascade into horizontal and vertical sections. The sub bands are then embedded using successive-approximation quantization (SAQ) into the cipher image. Their analyses using H.264 standard showed that the technique achieves good Pixel Signal to Noise Ratio (PSNR) values and it is more secured to transfer video data.

Ding et al. (2017) designed six new scan orders for H.264/AVC encoder by analysing the energy distribution of discrete cosine transform (DC) coefficients that enhances the scrambling effect and encryption space for perceptual video encryption. Perceptual video encryption is achieved by randomly selecting one of the scan orders with a security key, and the sign bit flipping of DC coefficients is also incorporated to increase the encryption space. Their experimental results showed that the proposed scheme provides better scrambling effect, higher security and low computational complexity.

2.5.5 Related Works on Audio Encryption Algorithms

As noted in Section 2.2, audio cryptosystems also employ chaos (Sadkhan & Mohammed, 2015), transform (Sridharan et al., 1991) and encoding techniques to secure audio content.

In 2009, Gnanajeyaraman et al. used the properties of sensitivity to initial conditions in higher dimensional chaotic maps to encrypt audio signals. In their technique, variables are used as cipher keys and because chaotic maps are sensitive to initial conditions and also chaotic trajectory is unpredictable, their method achieved higher security, high key space and could withstand chosen/known-plaintext attacks. However, the processing time, error rate are not measure to ascertain the applicability to real-time processing of audio signals.

Manpreet and Sukhpreet (2014) surveyed various audio encryption technques by comparing their performance against basic symmetric encryption standards. They compared known cryptographic techniques such as DES, 3DES, RC2, RC4, RC6, BLOWFISH and outlined the strengths and weaknesses in terms of the key size, block size, cipher type, attacks and security when applied in audio data for Network security purpose. They concluded that making modifications to those techniques will lead to more secured audio data.

Khalil (2016) proposed a symmetric key encryption algorithm to encrypt real-time audio signal using logarithm operation. This method uses two real numbers (a, b) as cipher keys to encrypt audio signals p into cipher signals s by using the formula $s = \log_b(ap) = \log_b(a) + \log_b(p)$. The decryption function $p = \frac{b^s}{a}$ uses the same cipher keys to recover the plain audio signal p. Khalil method has a better error rate of 0.00017 and yields audio signal exactly as the original than the RSA method which yields low quality audio signal (with error rate = 0.9737). However, no explanation is offered on how this method threat negative values in the encryption process given that logarithm of negative numbers yields complex value and audio signal contains only real value.

Khalil (2017) proposed a novel encrypting/decrypting technique for audio signal by using digital image as cipher key and cover for audio signal. In this technique, each sample of the audio signal is combined with values of three color components of a pixel fetched from the cover image to produce a quaternion number. The absolute value of the quaternion number is transmitted and decrypted when received into the original audio using quaternion mathematics. Simulation results show that the proposed method is robust and more secure against common signal processing (e.g. Gaussian noise) attacks without affecting the used bandwidth of the communication channel. No enough security and performance analysis have been provided to measure the strength of the method on those scores.

Kwon et al. (2012) achieved partial encryption of audio data by using watermarking and scrambling in MP3. Their proposed algorithm employs the magnitude information of Modified Discrete Cosine transform (MDCT) to scramble and protect the content of audio against eavesdropping and illegal mass distribution after descrambled. The encryption algorithm first decomposed the Pulse-Code Modulation (PCM) data of the source audio into 32 subbands after it is dealt with Analyser of Filterbank, MDCT is performed and watermark embedded into the MDCT coefficients. Finally, the watermarked MDCT coefficients are scrambled synchronously using the watermark sequence and cipher key.

Sharma et al. (2013) proposed a partial encryption technique to secured wav audio signal by selective encryption and decryption of important audio information. In this technique, time domain audio signal is transformed to frequency domain audio signal by use of Discrete Fourier Transform (DFT). RSA technique is applied to encrypt and decrypt the lower frequency bands since all the frequency regions do not participate equally in the communication. They observe that, the encryption on the lower frequency band (phase values) is more effective than the higher one.

2.6 Cryptanalysis

The central idea in cryptography is to keep the plaintext (or the key, or both) secret from all attackers. It is presumed that attackers have full access to the communications between the sender and receiver (Schneier, 1996).

Schneier (1996) defined cryptanalysis as the science of recovering the plaintext of a message without access to the cipher key. Every successful cryptanalysis may recover the plaintext or the cipher key or may find weaknesses in the cryptosystem that ultimately lead to the previous results. The loss of a key through non-cryptanalytic means is referred to as compromise (Schneier, 1996).

An attempted cryptanalysis is called an attack. Some types of cryptanalytic attacks elaborated by Schneier (1996) include:

- **Ciphertext-only attack**: In this method, the cryptanalyst has the ciphertext of numerous messages that have been encrypted using the same encryption algorithm. The task of the cryptanalyst is to recover as many as possible the plaintext of several messages.
- **Known-plaintext attack**: In this attack, the cryptanalyst has possession of the ciphertext of numerous messages and their accompanying plaintext. With this, he/she can infer the key (or keys) used to encrypt the messages or an algorithm to decrypt any new messages that are encrypted using the same key (or keys).
- **Chosen-plaintext attack**: The cryptanalyst in this attack has possession of the ciphertext and accompanying plaintext of several messages and can also choose the plaintext that can be encrypted.
- **Chosen-ciphertext attack**: The cryptanalyst can choose different ciphertexts to be decrypted and has access to the decrypted plaintext.
- **Brute-force attack**: In this attack, the cryptanalyst thoroughly try a set of keys until a valid decryption is achieved. This method is proven to be impractical for huge data size or large key spaces.

2.7 Chapter Summary

The chapter examined the various components that will be employed in the proposed techniques of perceptual video encryption. Detailed discussions have been made on video data (the main data) and cryptography (the main methodology). The k-shuffle technique widely used in card shuffling and applied in various fields such as networking and processor designs; and also applied in encryption is also covered. For transmission purposes residue number system has also been covered extensively that will be adopted in the encoding and transmission of cipher video. The chapter also, examined existing perceptual video and audio encryption algorithms that offer varied degradations in audio/visual qualities. The surveyed algorithms employed various techniques such as chaos (scrambling), transforms and encoding in order to achieve their individual objectives. Table 2.7.1 summarises then the strengths and weaknesses of the surveyed algorithms.

Table 2.7.1 Summary of Identified Strengths and Weaknesses Surveyed Algorithms.

Author	Video Standard	Key space	Support for Audio Security	Support for Real-Time Encryption
Meyer and Gadegast	MPEG-1	256 bit	No	No
Spinsante et al.	H.264/AVC	56 byte	No	No
Kulkarni et al.	Block-based video	128, 192, 256 bit	Yes	No
Li et al.	MPEG	≥ 100 bit	No	Yes
Wang et al.	H.264			
Wong and Bishop	MPEG	64, 256 bit	No	Yes
Zeng et al.	H.264	128 bit	No	Yes

Some of the drawbacks identified in these algorithms include;

- most of them are designed for particular video codecs/standards and are therefore limited in implementation
- none of the video encryption algorithms considered the ease of transmission of cipher video across networks
- no (except one) video encryption algorithm considered the audio aspect of the source video which also needs some form of degradation in quality
- the audio techniques are all fully-layered encryption and as such integration with perceptual video encryption algorithms will require extra work

Having identified these drawbacks, the next chapter will concentrate on the proposed alternative perceptual video encryption techniques.

CHAPTER THREE

METHODOLOGY

It has been determined from literature that perceptual encryption can be achieved by use of chaos (scrambling) and encoding techniques. Chaos based techniques are effective when randomness is used in the selection of blocks of a video for encryption. While scrambling techniques rearranges values of plain video in a chaotic manner, encoding techniques ensure that the values of plain video are altered from the original. When rotation matrix is used with initial conditions (coordinates), random points are obtained within a frame. This feature is integrated with the unit anti-diagonal matrix to formulate a new chaos-based perceptual video encryption algorithm in Section 3.1. In order to directly affect the pixel values of video frame and to enhance security, Section 3.2 presents a new pixel encoding technique that uses specially formulated orthogonal matrices for perceptual encryption of video data. In order to secure the audio aspect of commercial video, the K-shuffle technique is used in Section 3.3 to formulate a new audio encryption algorithm with varying degradation in the audio quality. Transmission efficiency of cipher video across networks is considered in Section 4.4 where RNS is adopted to encode cipher video into two residual video (with smaller value) for easy transmission. In Section 3.5 the integrated perceptual video cryptosystem is discussed. The various data analysis procedures used to evaluate the performances of the proposals are explained in Section 6.

3.1 Proposed Perceptual Video Encryption Algorithm via Unit Anti-Diagonal Matrix

In this section, a new symmetric perceptual video encryption algorithm that uses rotation matrix and unit anti-diagonal matrix to visually degrade video data with varying perceptibility is proposed. Blocks of pixels in video frames are identified and extracted using rotation matrix and encrypted/decrypted using unit anti-diagonal matrix.

3.1.1 Two-Dimension (2D) Rotation of an Arbitrary Point around the Origin

Consider a point P and P' in 2D coordinate system as shown in Figure 3.1.1. Then

$$x = r \times \cos a, \tag{3.1.1}$$

$$y = r \times \sin a, \tag{3.1.2}$$

and

$$x' = \ r \times cos(\alpha + \beta), \tag{3.1.3}$$

$$y' = r \times sin(\alpha + \beta). \tag{3.1.4}$$

From Equation (3.1.3) and using trigonometric identities

$$x' = r \times \cos(\alpha + \beta)$$

$$= r * (\cos a \times \cos \beta - \sin \alpha \times \sin \beta)$$

$$= x \times \cos \beta - y \times \sin \beta. \tag{3.1.5}$$

Similarly, Equation (3.1.4) gives

$$y' = r \times sin(\alpha + \beta)$$

$$= r * (\sin\alpha \times \cos\beta + \cos\alpha \times \sin\beta)$$

$$= y \times \cos\beta + x \times \sin\beta. \qquad (3.1.6)$$

Thus, in matrix notation

$$\begin{pmatrix} x' \\ y' \end{pmatrix} = \begin{pmatrix} \cos\beta & -\sin\beta \\ \sin\beta & \cos\beta \end{pmatrix} \begin{pmatrix} x \\ y \end{pmatrix}. \qquad (3.1.7)$$

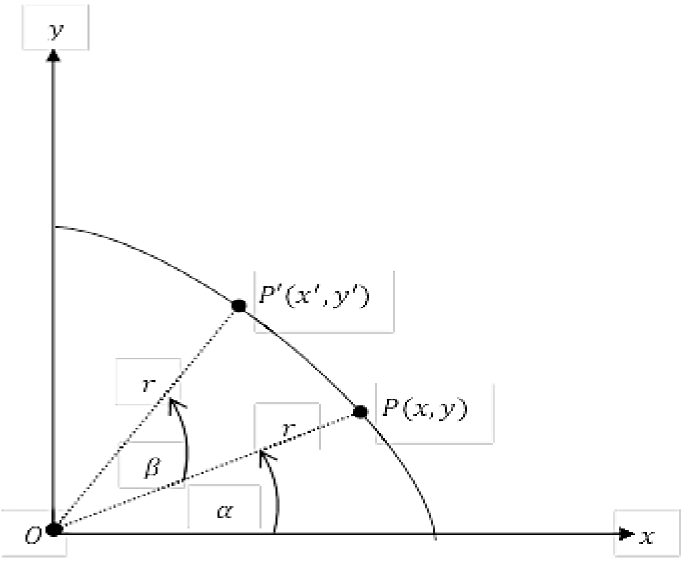


Figure 3.1.1 2D Coordinate System

3.1.2 Block Flipping Operation

Blocks of pixels in video frames can be flipped over a dimension (dimension is one (1) for column (vertical) flipping and two (2) for row (horizontal) flipping) using a unit-anti-diagonal matrix.

An $n \times n$ matrix A is anti-diagonal if the (i, j) elements are zero for all $i, j \in \{1, \dots, n\}$ with $i + \ j \neq n + 1$. Let A be a unit anti-diagonal matrix defined by

$$A = \begin{bmatrix} 0 & 0 & - & - & 0 & 1 \\ 0 & 0 & - & - & 1 & 0 \\ - & - & - & - & - & - \\ - & - & - & - & - & - \\ 0 & 1 & - & - & 0 & 0 \\ 1 & 0 & - & - & 0 & 0 \end{bmatrix}.$$

The properties of an anti-diagonal matrix include:

1. The product of two anti-diagonal matrices is a diagonal matrix.
2. The product of a diagonal matrix and anti-diagonal matrix is anti-diagonal.

3.1.2.1 Illustration on Unit Anti-Diagonal Matrix.

Let $X(n, m)$ be a matrix with entries as

$$X = \begin{bmatrix} x_{11} & x_{12} & - & - & x_{1(m-1)} & x_{1m} \\ x_{21} & x_{22} & - & - & x_{2(m-1)} & x_{2m} \\ - & - & - & - & - & - \\ - & - & - & - & - & - \\ x_{(n-1)1} & - & - & - & - & x_{(n-1)m} \\ x_{n1} & x_{n2} & - & - & - & x_{nm} \end{bmatrix}.$$

The elements of X are flipped column wise by multiplying A and X as

$$X_c = AX = \begin{bmatrix} \mathrm{x}_{n1} & \mathrm{x}_{n2} & - & - & \mathrm{x}_{n(m-1)} & \mathrm{x}_{nm} \\ \mathrm{x}_{(n-1)1} & \mathrm{x}_{(n-1)2} & - & - & \mathrm{x}_{(n-1)(m-1)} & \mathrm{x}_{(n-1)m} \\ - & - & - & - & - & - \\ - & - & - & - & - & - \\ \mathrm{x}_{21} & \mathrm{x}_{22} & - & - & \mathrm{x}_{2(m-1)} & \mathrm{x}_{2m} \\ \mathrm{x}_{11} & \mathrm{x}_{12} & - & - & \mathrm{x}_{1(m-1)} & \mathrm{x}_{1m} \end{bmatrix}.$$

Also, XA will flip X row wise as follow

$$X_r = XA\begin{bmatrix} \mathrm{x}_{1m} & \mathrm{x}_{1(m-1)} & - & - & \mathrm{x}_{12} & \mathrm{x}_{11} \\ \mathrm{x}_{2m} & \mathrm{x}_{2(m-1)} & - & - & \mathrm{x}_{22} & \mathrm{x}_{21} \\ - & - & - & - & - & - \\ - & - & - & - & - & - \\ \mathrm{x}_{(n-1)m} & \mathrm{x}_{(n-1)(m-1)} & - & - & \mathrm{x}_{2(m-1)} & \mathrm{x}_{(n-1)1} \\ \mathrm{x}_{nm} & \mathrm{x}_{n(n-1)} & - & - & \mathrm{x}_{1(m-1)} & \mathrm{x}_{n1} \end{bmatrix}.$$

Example 3.1.1

Consider a 4 × 4 unit anti-diagonal matrix A as follows

$$A = \begin{bmatrix} 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 \end{bmatrix}.$$

Also, let X be a 4 × 4 matrix with entries as follows

$$X = \begin{bmatrix} 1 & 2 & 4 & 5 \\ 6 & 7 & 8 & 9 \\ 10 & 11 & 12 & 13 \\ 14 & 15 & 16 & 17 \end{bmatrix}.$$

To flip X column wise (vertically), perform the operation AX to give

$$AX = \begin{bmatrix} 14 & 15 & 16 & 17 \\ 10 & 11 & 12 & 13 \\ 6 & 7 & 8 & 9 \\ 1 & 2 & 4 & 5 \end{bmatrix}.$$

Similarly, flipping X row wise (horizontally) can be done by using XA

$$XA = \begin{bmatrix} 5 & 4 & 2 & 1 \\ 9 & 8 & 7 & 6 \\ 13 & 12 & 11 & 10 \\ 17 & 16 & 15 & 14 \end{bmatrix}.$$

3.1.3 Encryption Algorithm

The inputs to this algorithm are: the video file, number of iteration, block size, angle of rotation and the unit anti-diagonal matrix. The output is the cipher video. The encryption algorithm is presented in Listing 3.1.1.

Listing 3.1.1 Encryption algorithm

a. Input video file V, number of iteration I, block size n , angle of rotation θ and unit-anti-diagonal matrix K
b. For each video frame;
    1. For each iteration
    2. Compute point of rotation and extract block of pixels V_b
    3. Multiple block of pixels by anti-diagonal matrix (Equation (3.1.7)) to obtain cipher frame $V_b \times K$
    4. Add cipher frame to cipher video V_E
c. Transmit cipher video V_E

Figure 3.1.2 shows the block diagram of the encryption algorithm. In the figure, video frames are extracted from plain video (V) and then encrypted into cipher video (V_E) using the cipher key supplied.

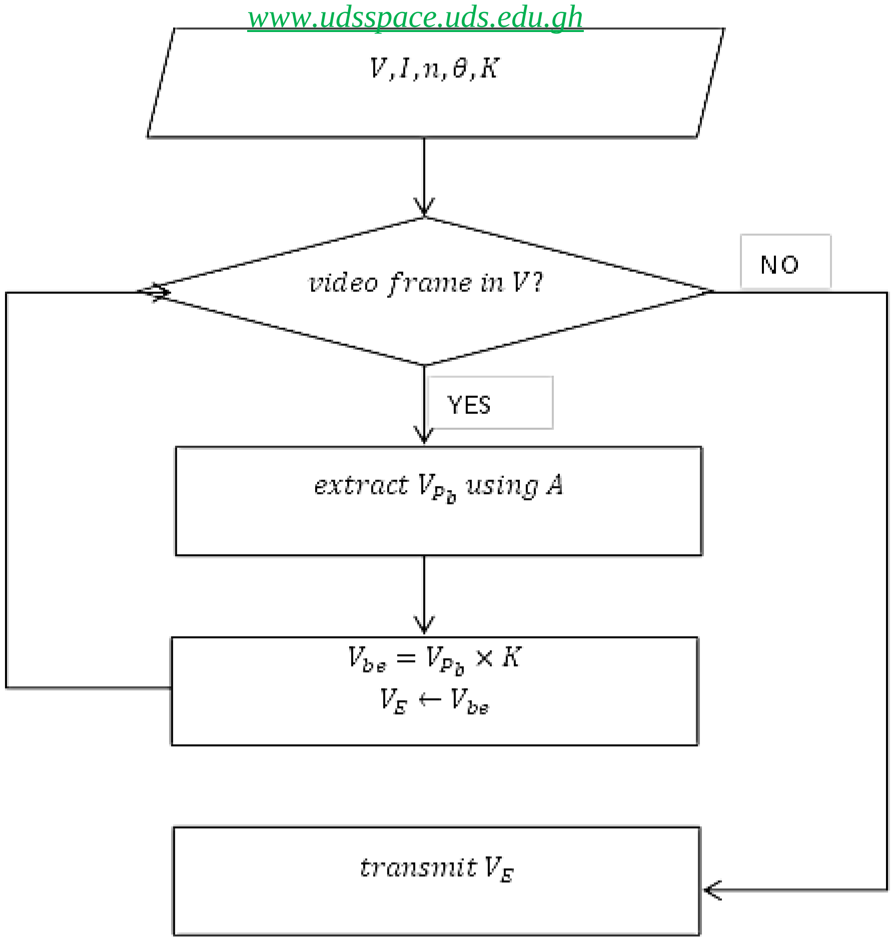


Figure 3.1.2 Block diagram of proposed encryption algorithm

3.1.4 Decryption Algorithm

This is a reverse of the encryption algorithm. The decryption algorithm takes as inputs the cipher video and cipher key and then decrypts the cipher video into its plain form. This algorithm is outlined in Listing 3.1.2.

Listing 3.1.2 Decryption Algorithm

a. Input cipher video V_E, number of iteration I, block size n, angle of rotation θ and anti-diagonal matrix K

b. For each frame in cipher video

    1. For each iteration

2. Compute point of rotation and extract cipher block of pixels V_{E_b}
3. Multiply block of pixels by anti-diagonal matrix to obtain plain frame $V_{E_b} \times K$
4. Add plain frame to plain video V

c. Transmit plain video V

Figure 3.1.3 shows the MATLAB Simulink system for the proposed encryption and decryption of real-time video using the proposed technique. The encryption algorithm is embedded in the sub-block “EncryptionAlgorithm1” while the sub-block “DecryptionAlgorithm1” contains the decryption algorithm. Sub-blocks “Cipher Video” and “Recovered Video” display the cipher video (from “EncryptionAlgorithm1”) and the decrypted video (from “DecryptionAlgorithm1”) respectively (see Appendix 2 for the MATLAB Simulink source codes).

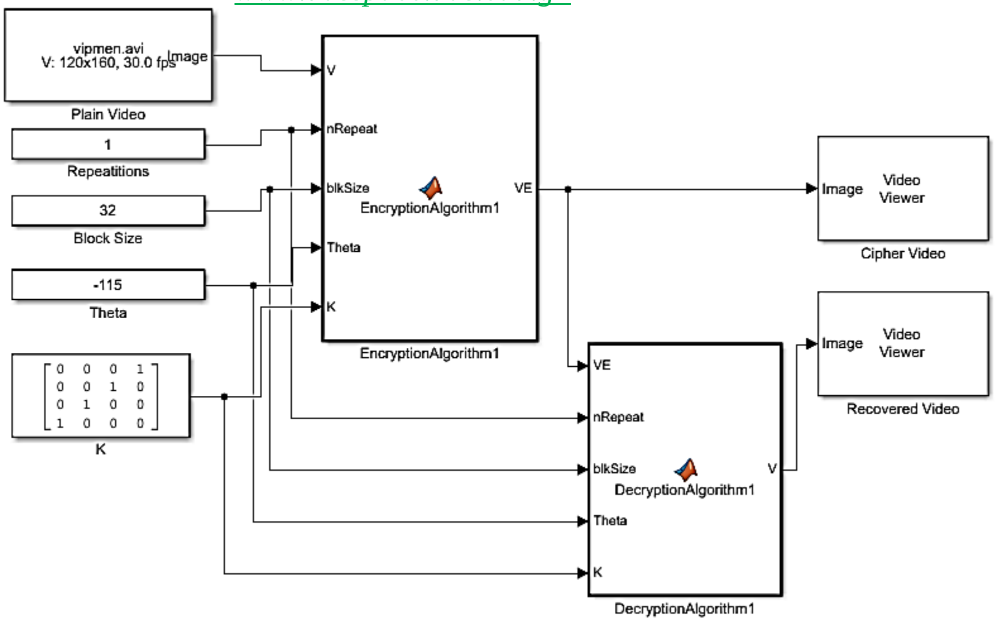


Figure 3.1.3 MATLAB Simulink system for the proposed encryption and decryption technique one (1) of real-time video data

3.1.5 Experimental Results

Figure 3.1.4 to Figure 3.1.6 show simulated results of the proposed algorithm using three (3) different video formats: ‘carphone.mp4’, ‘news.avi’ and ‘xylophone.mpg’. The simulations were done using different cipher key values. The first frame (named ‘a’) in all the results presented are unencrypted.

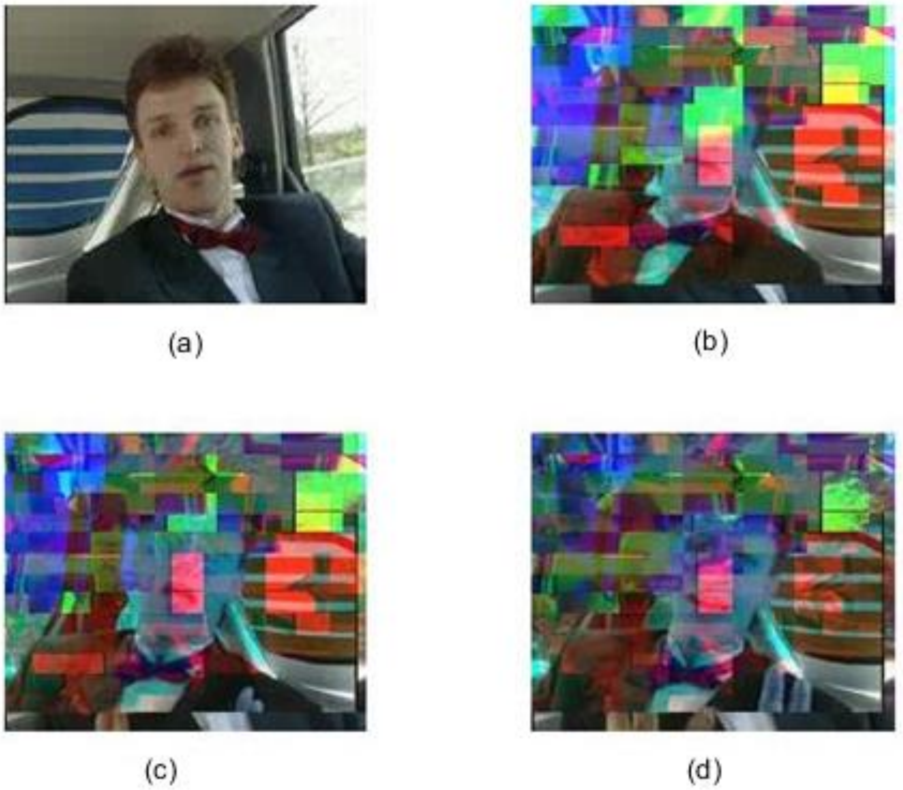


Figure 3.1.4 'Carphone.mp4' – Encrypted results of frame 1(a), 15(b), 270(c) and 315(d) with I =1, n=32, θ=60, Ixy =(10,1)

In Figure 3.1.4, frame 1 of "Carphone.mp4" is not encrypted. However, frames 15, 270 and 315 are encrypted yielding different degradation in visual quality. The inputs to this simulation process were, "Carphone.mp4" as the plain video, 1 as the number of repetition, 32 as block size, 60 as θ, (10, 1) as the initial coordinate and a 32×32 unit anti-diagonal matrix k. The Simulink system was run and frames 15, 270 and 315 were extracted from the cipher video. Also, frame 1 of the plain video was extraction and combined with the other frames to obtained Figure 3.1.4.

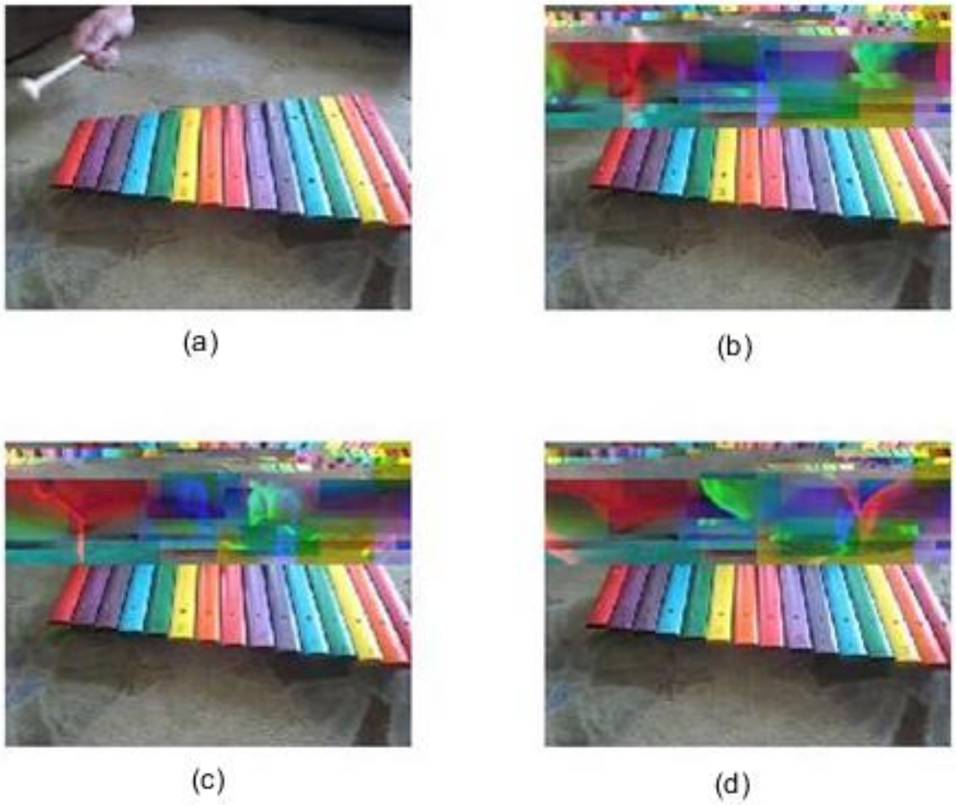


Figure 3.1.5 ‘Xylophone.mpg’ – Encrypted results of frame 1(a), 28(b), 70(c) and 98(d) with I=1, n=62, θ=113, ixy=(5,5)

Frame 1 of “Xylophone.mpg” in Figure 3.1.5 is unencrypted. However, frames 28, 70 and 98 are encrypted yielding different degradation in visual quality. A simulation process similar to the one described in Figure 3.1.4 was carried out to obtain Figure 3.1.5 using the parameters specified in the caption.

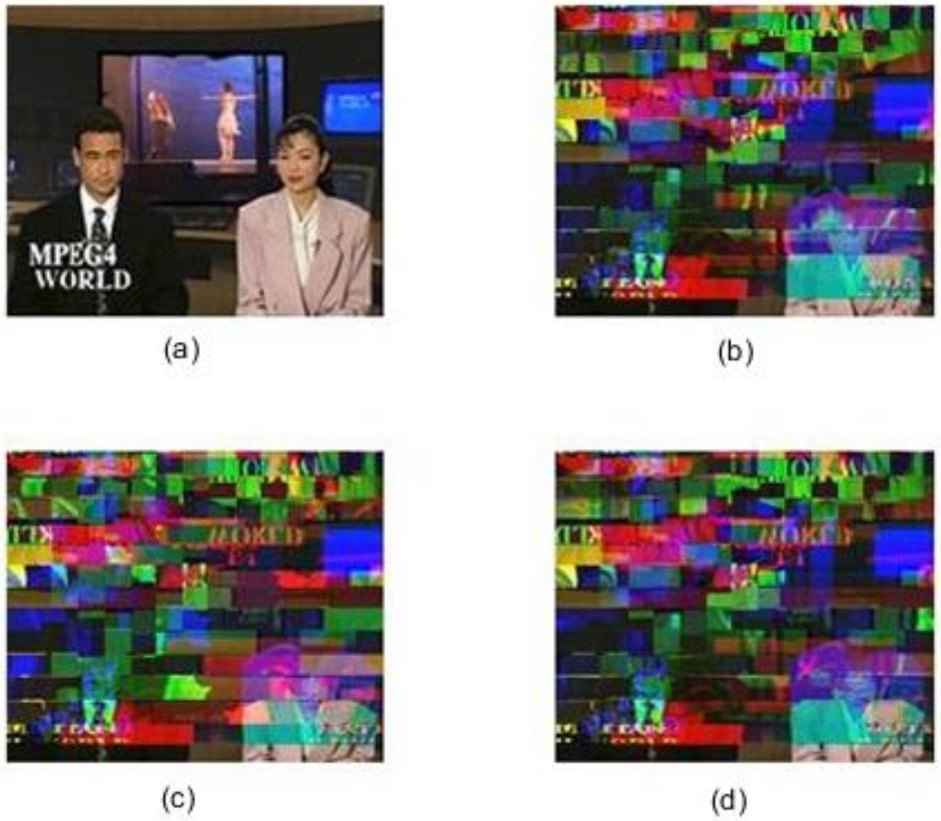


Figure 3.1.6 ‘news.avi’ – Encrypted results of frame 1(a), 45(b), 105(c) and 165(d) with I=1, n=16, θ= -15, ixy=(1,1)

Frame 1 of “news.avi” in Figure 3.1.6 is unencrypted. However, frames 45, 105 and 105 are encrypted yielding different degradation in visual quality. To obtain this result, the simulation process was repeated using the parameters specified in the caption. Here the plain video was “news.avi” and number of repetition, block size, θ and initial conditions were 1, 16, -15 and (1, 1) respectively. A 16×16 unit anti-diagonal matrix was also constructed and input for k.

3.2 Proposed Perceptual Video Encryption Algorithm Using Orthogonal Matrix

In order to directly change the values of pixels, the properties of rotation matrices are employed to construct special orthogonal matrices that are used to encrypt and decrypt video. The encryption function multiples the matrices (cipher keys) with blocks of the plain video to obtain the cipher video. In the decryption process, the transpose of the cipher key is multiplied with encrypted blocks of the cipher video to retrieve the original video without any error. The new idea in the method is the use of augmented integer rotation (orthogonal) matrices to perform the encryption and decryption of video efficiently.

3.2.1 Rotation Matrix

For 2 × 2 rotation matrix

$$Let\ A = \begin{bmatrix} \cos\theta & \sin\theta \\ -\sin\theta & \cos\theta \end{bmatrix}\ and\ A^T = \begin{bmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{bmatrix}.$$

It follows that

$$AA^T = A^TA = \begin{bmatrix} \cos\theta & \sin\theta \\ -\sin\theta & \cos\theta \end{bmatrix}\begin{bmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{bmatrix} = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} = I.$$

Also,

$$detA = detA^T = 1.$$

Given that $P = \begin{bmatrix} a \\ b \end{bmatrix}$ then $AP = \begin{bmatrix} \cos\theta & \sin\theta \\ -\sin\theta & \cos\theta \end{bmatrix}\begin{bmatrix} a \\ b \end{bmatrix} = \begin{bmatrix} a\cos\theta + b\sin\theta \\ -a\sin\theta + b\cos\theta \end{bmatrix}.$

Thus, to obtain P from AP, perform the operation $A^TAP = IP = \begin{bmatrix} a \\ b \end{bmatrix}.$

For 3 × 3

$$A = \begin{bmatrix} \cos\theta & \sin\theta & 0 \\ -\sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{bmatrix} \; and \; A^T = \begin{bmatrix} \cos\theta & -\sin\theta & 0 \\ \sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{bmatrix}.$$

Thus

$$AA^T = \begin{bmatrix} \cos\theta & \sin\theta & 0 \\ -\sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \cos\theta & -\sin\theta & 0 \\ \sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} = I,$$

and

$$detA = detA^T = 1.$$

Also for 4 × 4

$$A = \begin{bmatrix} \cos\theta & \sin\theta & 0 & 0 \\ -\sin\theta & \cos\theta & 0 & 0 \\ 0 & 0 & \cos\theta & \sin\theta \\ 0 & 0 & -\sin\theta & \cos\theta \end{bmatrix} \; and$$

$$A^T = \begin{bmatrix} \cos\theta & -\sin\theta & 0 & 0 \\ \sin\theta & \cos\theta & 0 & 0 \\ 0 & 0 & \cos\theta & -\sin\theta \\ 0 & 0 & \sin\theta & \cos\theta \end{bmatrix}.$$

Thus

$$AA^T = \begin{bmatrix} \cos\theta & \sin\theta & 0 & 0 \\ -\sin\theta & \cos\theta & 0 & 0 \\ 0 & 0 & \cos\theta & \sin\theta \\ 0 & 0 & -\sin\theta & \cos\theta \end{bmatrix} \begin{bmatrix} \cos\theta & -\sin\theta & 0 & 0 \\ \sin\theta & \cos\theta & 0 & 0 \\ 0 & 0 & \cos\theta & -\sin\theta \\ 0 & 0 & \sin\theta & \cos\theta \end{bmatrix}$$

$$= \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} = I,$$

and $detA = detA^T = 1$.

3.2.2 Extension to Integer

Video frames contain pixel values in the range of 0-255. The goal is to construct square matrices (cipher key matrices) with similar properties as rotation matrices.

Let $a, bbe\ integers\ and\ Dbe\ a\ domain\ of\ interest$

Construct a 2×2 cipher key matrices A and A^T as:

$$A = \begin{bmatrix} a & b \\ -b & a \end{bmatrix}, \qquad A^T = \begin{bmatrix} a & -b \\ b & a \end{bmatrix},$$

where a, b are chosen such that:

$$|a^2 + b^2|_D = 1,$$

and

$$|detA|_D = |detA^T|_D = 1,$$

$$\forall\ a \neq 0, b \neq 0\ and\ 1 < (a, b) < D,$$

$$A\ is\ the\ encrytion\ key(matrix), \qquad A^T is\ the\ decrytion\ key(matrix).$$

For every a, b chosen, matrix A can be written in any of the following forms:

$$\begin{bmatrix} a & -b \\ b & a \end{bmatrix}, \begin{bmatrix} -a & b \\ -b & -a \end{bmatrix}, \begin{bmatrix} -a & b \\ b & a \end{bmatrix}, \begin{bmatrix} a & b \\ b & -a \end{bmatrix}, \begin{bmatrix} -a & -b \\ b & -a \end{bmatrix}, \begin{bmatrix} a & -b \\ -b & -a \end{bmatrix}, \begin{bmatrix} -a & -b \\ -b & a \end{bmatrix}.$$

MATLAB function to generate a, b given the domain D is presented in Listing 3.2.1.

Listing 3.2.1 Cipher key values generator

```
function get_cypher = aug(D)
   tb=[];
   for a=2:1:(D-1)
      for b=i:1:(D-1)
         if $mod(a^2 + b^2, D)$==1
            tb=[tb; [a b]];
         end
      end
   end
```

3.2.3 Augmented Matrix

The general equation to augment the 2-dimensioinal and 3-dimensional form of matrix A by a factor of 2^i is to introduce the zero matrix (**0**) as shown in Equation (3.2.1)

$$A_{n\times n} = \begin{bmatrix} A^* & \mathbf{0} \\ \mathbf{0} & A^* \end{bmatrix}, \tag{3.2.1}$$

where $n = \begin{cases} 2.2^i\ , i = 1,2,3, \dots for\ 2 - dimensional\ matrix \\ 3.2^i\ , i = 1,2,3, \dots for\ 3 - dimensional\ matrix \end{cases}$

$A^* = A_{(\frac{n}{2}\times\frac{n}{2})}\ matrix$, and **0** (zero matrix) has the dimension as A^*.

Example 3.2.1 Let $i = 1$,

The augmented matrix for 2-dimensional matrix is the 4×4 matrix

$$A_{4\times 4} = \begin{bmatrix} a & b & 0 & 0 \\ -b & a & 0 & 0 \\ 0 & 0 & a & b \\ 0 & 0 & -b & a \end{bmatrix}.$$

The augmented matrix for 3-dimensional matrix is the 6×6 matrix

$$A_{6\times6} = \begin{bmatrix} a & b & 0 & 0 & 0 & 0 \\ -b & a & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & a & b & 0 \\ 0 & 0 & 0 & -b & a & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix}.$$

For dimensions which are not multiples of 2×2^i *or* 3×2^i reformulate Equation (3.2.1) into Equation (3.2.2).

$$A_{n\times n} = \begin{bmatrix} A^{**} & \mathbf{0} \\ \mathbf{0}^T & 1 \end{bmatrix}, \qquad (3.2.2)$$

where $n = \begin{cases} (2.2^i) + 1\,, i = 1,2,3, \dots for\ 2 - dimensional\ matrix \\ (3.2^i) + 1\,, i = 1,2,3, \dots for\ 3 - dimensional\ matrix \end{cases}$

$A^{**} = A_{(n-1)\times(n-1)}$ square matrix obtained from Equation (3.2.1), and $\mathbf{0}$ is a zero column vector which has the same number of rows as A^{**}.

Example 3.2.2 Let $i = 1$,

The augmented matrix for 2×2 rotation matrix is the 5×5 matrix

$$A_{5\times5} = \begin{bmatrix} a & b & 0 & 0 & 0 \\ -b & a & 0 & 0 & 0 \\ 0 & 0 & a & b & 0 \\ 0 & 0 & -b & a & 0 \\ 0 & 0 & 0 & 0 & 1 \end{bmatrix}.$$

The augmented matrix for 3×3 rotation matrix is the 7×7 matrix

$$A_{7\times7} = \begin{bmatrix} a & b & 0 & 0 & 0 & 0 & 0 \\ -b & a & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & a & b & 0 & 0 \\ 0 & 0 & 0 & -b & a & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix}.$$

Encryption and decryption functions

Given plain text p, encryption key matrix A, decryption key matrix A^T and domain D,

- Encryption function

$$E_p = |A \times p|_D. \tag{3.2.3}$$

- Decryption function

$$D_{Ep} = \left|A^T \times E_p\right|_D = p. \tag{3.2.4}$$

Example 3.2.3

To encrypt $\begin{bmatrix}11\\16\end{bmatrix}$ in modulo 26.

Let (a, b) be (13, 14) respectively. Thus using 2x2 matrix the plain text p and encryption key A are

$$p = \begin{bmatrix}11\\16\end{bmatrix}\ and\ A = \begin{bmatrix}13 & 14\\-14 & 13\end{bmatrix}.$$

From Equation (3.2.1) encrypt p as

$$E_p = |A \times p|_{26} = \left|\begin{bmatrix}13 & 14\\-14 & 13\end{bmatrix}\begin{bmatrix}11\\16\end{bmatrix}\right|_{26} = \begin{bmatrix}367\\54\end{bmatrix}_{26} = \begin{bmatrix}3\\2\end{bmatrix}.$$

Using Equation (3.2.2), decrypt $E_p$as

$$D_{Ep} = \left|A^T \times E_p\right|_{26} = \left|\begin{bmatrix}13 & -14\\14 & 13\end{bmatrix}\begin{bmatrix}3\\2\end{bmatrix}\right|_{26} = \begin{vmatrix}11\\68\end{vmatrix}_{26} = \begin{bmatrix}11\\16\end{bmatrix}.$$

Example 3.2.4

Consider the cipher key, plain text and domain as

$$A = \begin{bmatrix} 15 & -20 & 0 \\ -20 & -15 & 0 \\ 0 & 0 & 1 \end{bmatrix}, p = \begin{bmatrix} 9 \\ 13 \\ 4 \end{bmatrix}, D = 26.$$

For encryption

$$E_p = |A \times p|_{26} = \left| \begin{bmatrix} 15 & -20 & 0 \\ -20 & -15 & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 9 \\ 13 \\ 4 \end{bmatrix} \right|_{26} = \begin{vmatrix} -125 \\ -375 \\ 4 \end{vmatrix}_{26} = \begin{bmatrix} 5 \\ 15 \\ 4 \end{bmatrix}.$$

Using (2), decrypt $E_p$as

$$D_{Ep} = \left|A^T \times E_p\right|_{26} = \left| \begin{bmatrix} 15 & -20 & 0 \\ -20 & -15 & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 5 \\ 15 \\ 4 \end{bmatrix} \right|_{26} = \begin{vmatrix} -225 \\ -325 \\ 4 \end{vmatrix}_{26} = \begin{bmatrix} 9 \\ 13 \\ 4 \end{bmatrix}.$$

3.2.4 Proposed Algorithms

3.2.4.1 Encryption Algorithm

The inputs to the encryption algorithm are the plain video V and the encryption key A, and the output is the cipher video V_E. This is illustrated in Listing 3.2.2.

Listing 3.2.2 Encryption algorithm

```
a. Input cipher key matrix $A$, video file $V$ and block interval $jump$
b. For each randomly selected frame $F$ of $V$
    i. Encrypt $F$ as follows
        1.Compute $blksize = size(A, 1) - 1$
        2.For each $i$ row of $F$ with increment $jump$
        3.For each $j$ column of $F$ with increment $jump$
```

```
                4.$temp = mod(A \times F\big(i:(i + blksize), j:(j + blksize)\big), 256)$
                5.$F_E(i:(i + bsize), j:(j + bsize)) = temp$
        ii.  Save encrypted frame $F_E$ into cipher video $V_E$
    c.  Transmit $V_E$
```

Figure 3.2.1 show the block diagram of the encryption process in which the original video (together with the cipher key and block size) is processed into cipher video and transmitted.

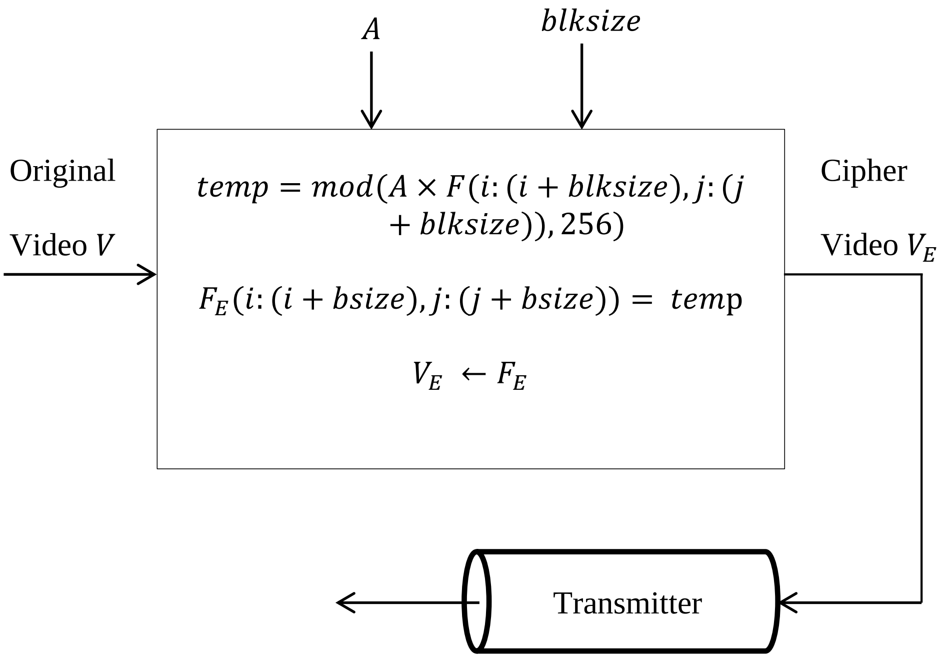

A
$blksize$
Original
Video V
$temp = mod(A \times F(i:(i + blksize), j:(j + blksize)), 256)$
$F_E(i:(i + bsize), j:(j + bsize)) = temp$
$V_E \leftarrow F_E$
Cipher
Video V_E
Transmitter

Figure 3.2.1 Block diagram of encryption algorithm

3.2.4.2 Decryption Algorithm

Inputs to the decryption algorithm are the cipher video $V_E$and the decryption key A^T, and the output is the plain video V. The complete algorithm is presented in Listing 3.2.3.

Listing 3.2.3 Decryption algorithm

```
a. Input cipher video $V_E$ , cipher key matrix $A^T$ and block interval $jump$
b. For each encrypted frame $F_E$ of $V_E$
    i. Decrypt $F_E$ as follows
        1.Compute $blksize = size(A^T, 1) - 1$
        2.For each $i$ row of $F_E$ with increment $jump$
        3.For each $j$ column of $F_E$ with increment $jump$
        4.$temp = mod(A^T \times F_E\big(i:(i + blksize), j:(j + blksize)\big), 256)$
        5.$F(i:(i + bsize), j:(j + bsize)) = temp$
    ii. Save decrypted frame $F$ into plain video V
c. Transmit V
```

The block diagram of the decryption process is shown in Figure 3.2.2. In the figure, cipher video from the receiver (together with cipher key) is decrypted into original video and then transmitted.

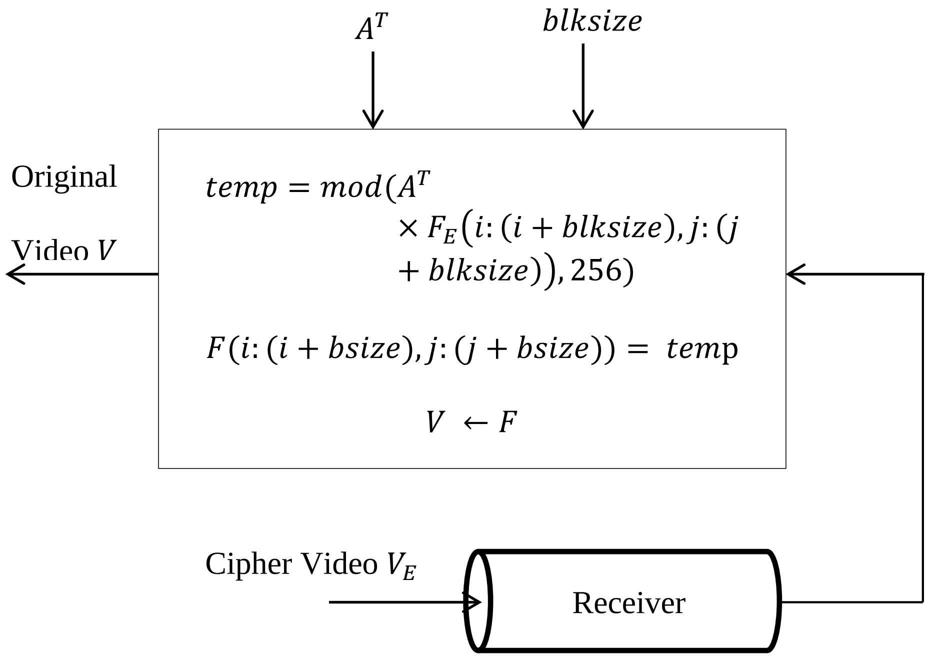

A^T
$blksize$
Original
Video V
$temp = mod(A^T \times F_E(i:(i + blksize), j:(j + blksize)), 256)$
$F(i:(i + bsize), j:(j + bsize)) = temp$
$V \leftarrow F$
Cipher Video V_E
Receiver

Figure 3.2.2 Block diagram of encryption algorithm

Figure 3.2.3 show the MATLAB Simulink system for the proposed encryption and decryption of real-time video data using the proposed technique. Sub-block “EncryptionAlgorithm2” runs the encryption algorithm while sub-block “DecryptionAlgorithm2” processes the decryption algorithm. Sub-blocks “Cipher Video” and “Recovered Video” display the cipher video (from “EncryptionAlgorithm2”) and the decrypted video (from “DecryptionAlgorithm2”) respectively.

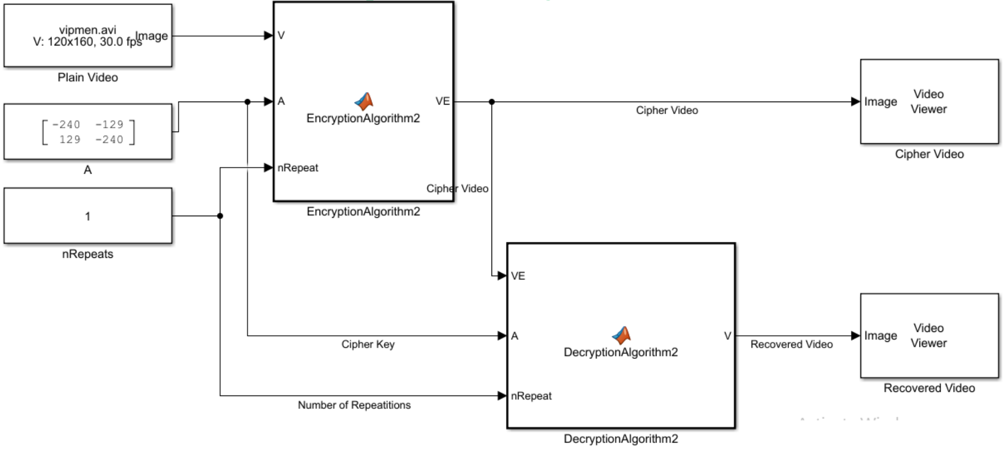


Figure 3.2.3 MATLAB Simulink system for the proposed encryption and decryption technique two (2) of real-time video file.

3.2.5 Experimental Results

3.2.5.1 Cipher Keys

Different sizes of the cipher keys were used for simulation in MATLAB. The integer values for the cipher keys are: 39 and 252. The cipher key constructs for $2 \times 2, 3 \times 3, 4 \times 4, 5 \times 5, 8 \times 8$ and $n \times n$ are:

i. 2×2 cipher key matrix

$$A = \begin{bmatrix} 39 & 252 \\ -252 & 39 \end{bmatrix}.$$

ii. 3×3 cipher key matrix

$$A = \begin{bmatrix} 39 & 252 & 0 \\ -252 & 39 & 0 \\ 0 & 0 & 1 \end{bmatrix}.$$

iii. 4×4 cipher key matrix

$$A = \begin{bmatrix} 39 & 252 & 0 & 0 \\ -252 & 39 & 0 & 0 \\ 0 & 0 & 39 & 252 \\ 0 & 0 & -252 & 39 \end{bmatrix}.$$

iv. 5 × 5 cipher key matrix

$$A = \begin{bmatrix} 39 & 252 & 0 & 0 & 0 \\ -252 & 39 & 0 & 0 & 0 \\ 0 & 0 & 39 & 252 & 0 \\ 0 & 0 & -252 & 39 & 0 \\ 0 & 0 & 0 & 0 & 1 \end{bmatrix}.$$

v. 8 × 8

$$A = \begin{bmatrix} 39 & 252 & 0 & 0 & 0 & 0 & 0 & 0 \\ -252 & 39 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 39 & 252 & 0 & 0 & 0 & 0 \\ 0 & 0 & -252 & 39 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 39 & 252 & 0 & 0 \\ 0 & 0 & 0 & 0 & -252 & 39 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 39 & 252 \\ 0 & 0 & 0 & 0 & 0 & 0 & -252 & 39 \end{bmatrix}.$$

vi. 16 × 16

Let

$$\mathbf{0} = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

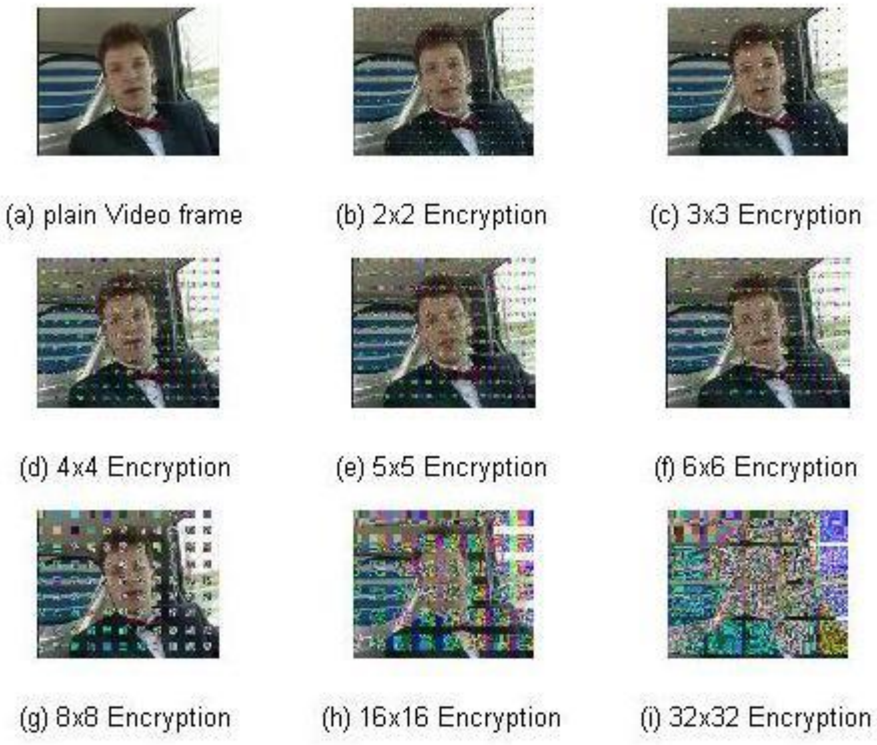
$$\therefore A = \begin{bmatrix} A_{8\times 8} & \mathbf{0} \\ \mathbf{0} & A_{8\times 8} \end{bmatrix}.$$

vii. $n \times n$

Equations (4.2.1) and (4.2.2) are used to construct the $n \times n$ cipher keys.

3.2.5.2 Encryption

Simulation on the encryption algorithm was done on ‘carphone.mp4’ video file using different cipher keys. Figure 3.2.4 shows the cipher frames for the different cipher keys used to encrypt the first frame of ‘carphone.mp4’ video.



(a) plain Video frame

(b) 2x2 Encryption

(c) 3x3 Encryption

(d) 4x4 Encryption

(e) 5x5 Encryption

(f) 6x6 Encryption

(g) 8x8 Encryption

(h) 16x16 Encryption

(i) 32x32 Encryption

Figure 3.2.4 Cipher video frames

The results in Figure 3.2.4 show that perceptibility decreases as the dimension of the cipher key increases. Thus; the blocks of pixels encrypted increases as the dimension of the cipher key increases.

3.2.5.3 Decryption

After encryption, simulations were carried out to find out the recovery efficiency of the decryption algorithm. The recovered frames of Figure 3.2.4 are shown in Figure 3.2.5.

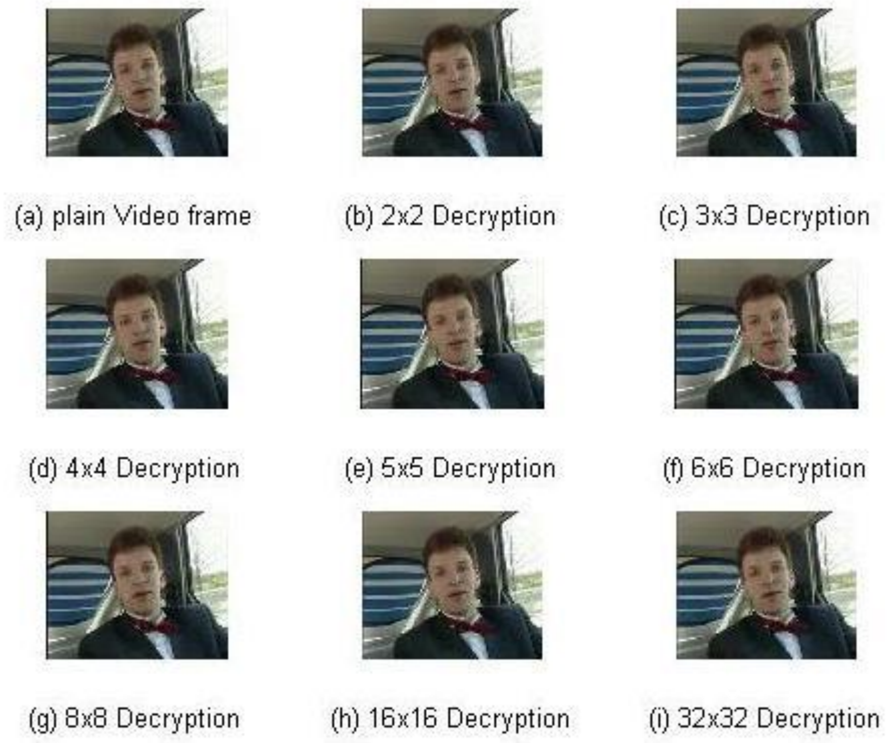

(a) plain Video frame (b) 2x2 Decryption (c) 3x3 Decryption
(d) 4x4 Decryption (e) 5x5 Decryption (f) 6x6 Decryption
(g) 8x8 Decryption (h) 16x16 Decryption (i) 32x32 Decryption

Figure 3.2.5 Recovered video frames

The results in Figure 3.2.5 depict that the decryption algorithm recovered exactly the plain video frames without loss of any data. The recovery efficiency is examined further in the following.

3.3 Proposed Audio Encryption Algorithm Using K-Shuffle Technique

In this technique audio data emanating from video is encrypted/decrypted using K-shuffle technique. Sound data will be extracted and the values arranged in m −piles (i.e $n \times m$ matrix). The encryption procedure involves re-arranging the data by taking the 1st values of each of the m −piles, then the 2nd values, up to the n^{th}values. Whiles maintaining the first and last values at their respective positions, this technique scramble the data in-between into a chaotic form. Experimental results using MATLAB Simulink simulation tools show that the cipher audio is completed different from the original audio data. Encryption efficiency increases with larger m and as the number of re-shuffling increases. Also the recovered sound data when compared with the original data has a zero error rate.

3.3.1 Cryptographic Functions

The cryptographic functions scramble original/cipher audio into cipher/recovered audio using the k-shuffle technique. These functions compute the index of x^{th} value to include in the next re-arrangement.

3.3.1.1 Encryption Function

Let $n, m, l, x \in \mathbb{Z}$, the encryption function is given by

$$E(x) = \begin{cases} x & if\ x = 1\ or\ x = l \\ |(n(x-1)+1)|_l & otherwise \end{cases}, \qquad (3.3.1)$$

where $l = nm, x = 1,2,3, \dots l.$

$$Note: if\ |(n(x-1)+1)|_{l-1} = 0\ then\ E(x) = l.$$

3.3.1.2 Decryption Function

Let $n, m, l, x \in \mathbb{Z}$, the decryption function is given by

$$D(x) = \begin{cases} x & if\ x = 1\ or\ x = l \\ |(m(x-1)+1)|_{l-1} & otherwise \end{cases}, \qquad (3.3.2)$$

$$where\ l = nm, x = 1,2,3, \dots l.$$

$$Note: if\ |(m(x-1)+1)|_l = 0\ then\ E(x) = l.$$

3.3.1.3 Cipher Key

The cipher keys of the proposed cryptosystem are $(n, m, i \in \mathbb{Z})$. n and m are chosen so that $l = nm$ is very close to the length of audio data to be encrypted/decrypted. Also, since k-shuffle is periodic for a given l (periodicity is not greater than l), i should be chosen such that it does not compromise the technique.

3.3.1.4 Encryption Algorithm

The encryption algorithm employs the encryption function to scramble original audio (p) into cipher audio (s). The cipher keys are $n, m, i \in \mathbb{Z}$. n is the number of rows (signal values), m the number of columns (piles) and i is the number of re-shuffling to perform. Listing 3.3.1 details the encryption algorithm.

Listing 3.3.1 Encryption algorithm

```
1. Input $p, n, m, i$
2. Compute
        $l = nm$
        $for\ k = 1\ to\ i$
                $for\ x = 1\ to\ length(p)$
                        $if\ (x = 1\ or\ x = l\ or\ x = length(p))$
                                $s(x) = p(x)$
                        $else$
                                $temp = |(n(x-1)+1)|_{l-1}$
                                $if\ (temp == 0)$
                                        $temp = l - 1$
                                $end$
                                $s(x) = p(temp)$
                        $end$
                $next\ x$
                $p = s$
        $next\ k$
3. Transmit $s$
```

3.3.1.5 Decryption Algorithm

The decryption algorithm uses the decryption function to recover plain audio (p) from cipher audio (s). The cipher keys are $n, m, i \in \mathbb{Z}$. Listing 3.3.2 details the decryption process.

Listing 3.3.2 Decryption algorithm

```
1. Input $s, n, m, i$
2. Compute
        $l = nm$
        $for\ k = 1\ to\ i$
            $for\ x = 1\ to\ length(p)$
                $if(\ x = 1\ or\ x = l\ or\ x = length(p))$
                    $p(x) = s(x)$
                $else$
                    $temp = |(m(x-1)+1)|_{l-1}$
                    $if\ (temp == 0)$
                        $temp = l - 1$
                    $end$
                    $p(x) = s(temp)$
                $end$
            $next\ x$
            $s = p$
        $next\ k$
3. Transmit $p$
```

Figure 3.3.1 shows the block diagram of the proposed technique. The original audio data are acquired and transformed (encrypted) into cipher audio using the secret key in the encryption sub-block. The cipher audio data are transmitted through a noise-free communication channel into the receiver. At the decryption sub-block, the received cipher audio is deciphered (decrypted) using the same cipher (secret) key.

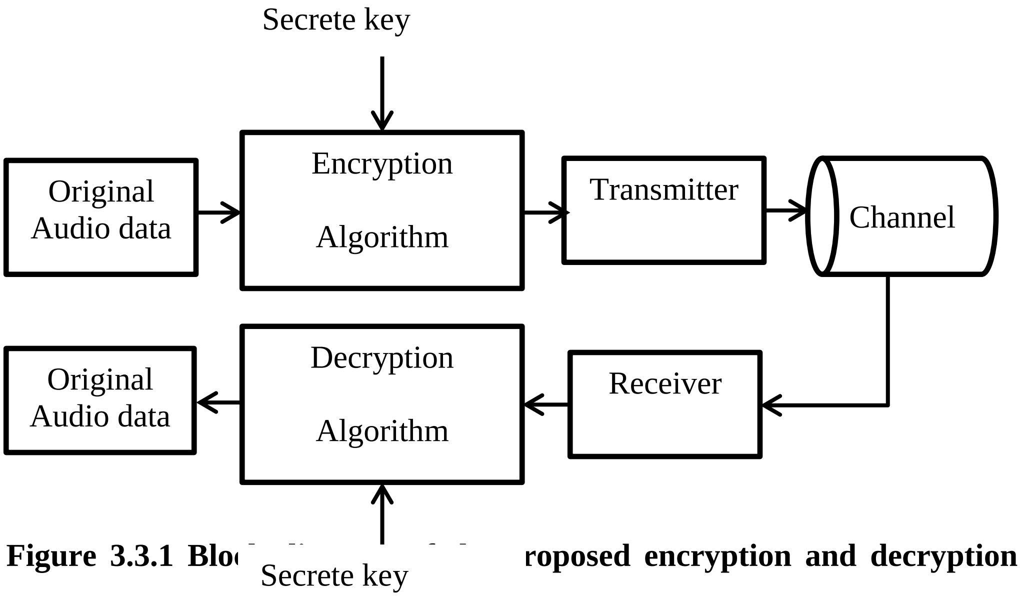


Figure 3.3.1 Blo roposed encryption and decryption process

For elaboration purposes, the proposed encryption and decryption functions for the proposed method are shown in Figure 3.3.2 and Figure 3.3.3.

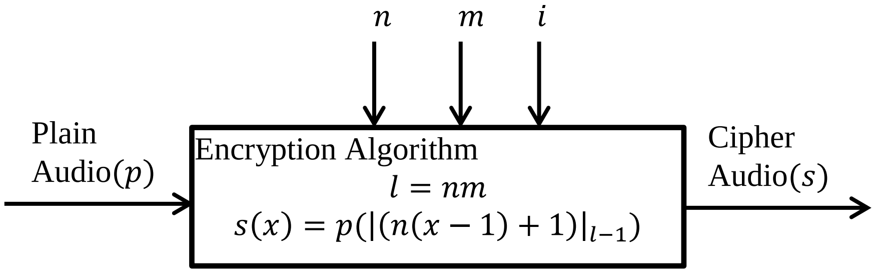


Figure 3.3.2 Encryption Algorithm

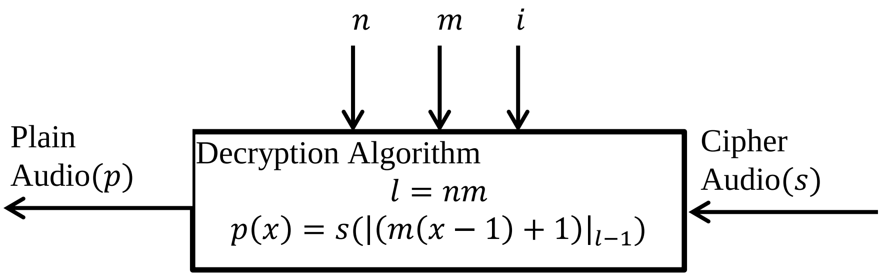


Figure 3.3.3 Decryption algorithm

3.3.2 Experimental Results

The proposed algorithms are implemented using MATLAB Simulink tools to encrypt/decrypt real-time audio data from either an audio file or the microphone. Figure 3.3.4 shows the block diagram of the MATLAB Simulink simulation processes.

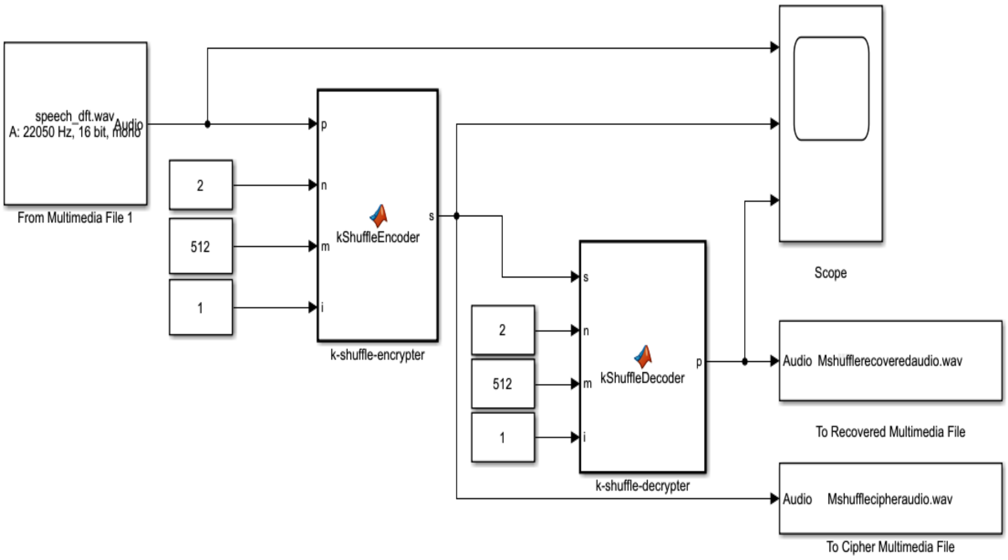
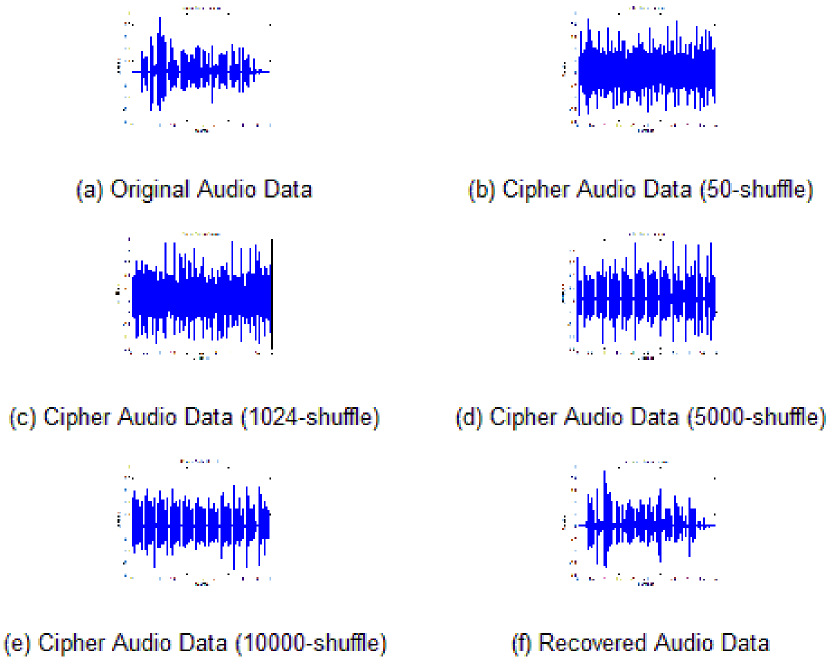

speech_dft.wav
A: 22050 Hz, 16 bit, mono
Audio
From Multimedia File 1
2
512
1
p
n
m
i
kShuffleEncoder
s
k-shuffle-encrypter
2
512
1
s
n
m
i
kShuffleDecoder
p
k-shuffle-decrypter
Scope
Audio Mshufflerecoveredaudio.wav
To Recovered Multimedia File
Audio Mshufflecipheraudio.wav
To Cipher Multimedia File

Figure 3.3.4 MATLAB Simulink system for the proposed encryption and decryption of real-time audio data

In Figure 3.3.4, the encryption function (kShuffleEncoder) takes the input audio data p (a real mono audio file with sampling rate of 22050 Hz, 16 bit) together with the cipher key (n, m, i) and encrypts it into the cipher audio signal s using Listing 1. The decryption function (kShuffleDecoder) receives the cipher audio s and the same cipher key (n, m, i) to recover the original audio signal p using Listing 2. The output of both encryption (cipher audio signals) and decryption (recovered audio signals) functions are store to multimedia files for analysis. As an example, the figure shows the cipher key (n, m, i) as $(2, 512,1)$ and the input audio signal is a block size of 1024.

Figure 3.3.5 shows the graphs of the audio signals obtained from the simulation process. Different k- (i.e. 50-, 1024-, 5000-, 10000-) shuffles of audio signals are simulated for the encryption and decryption methods.



(a) Original Audio Data

(b) Cipher Audio Data (50-shuffle)

(c) Cipher Audio Data (1024-shuffle)

(d) Cipher Audio Data (5000-shuffle)

(e) Cipher Audio Data (10000-shuffle)

(f) Recovered Audio Data

Figure 3.3.5 Graph of MATLAB simulation results of Real-Time audio data using proposed method

Table 3.3.1 summarizes the parts of the graph in Figure 3.3.5. The simulated results show that the graph of each of the cipher audio signal is different from the original audio signal. This implies that different cipher keys will yield different cipher audio signal (results) that are completely different from the original audio signal.

Table 3.3.1 Summary of simulation properties of Figure 4.3.5

Part	Cipher key $(\boldsymbol{n}, \boldsymbol{m}, \boldsymbol{i})$
a) Original audio signals	None
b) Cipher audio signals	(2200, 50, 1)
c) Cipher audio signals	(107, 1024,1)
d) Cipher audio signals	(22,5000,1)
e) Cipher audio signals	(11,10000,1)
f) Recovered audio signals	(2200, 50, 1)

3.4 Proposed RNS Decoder for Cipher Video using the Moduli Set $\{\mathbf{2^n - 1, 2^n, 2^n + 1}\}$

In this section, a new reverse converter with smaller dynamic range is proposed using the moduli set $\{2^n - 1, 2^n, 2^n + 1\}$ for the fast decoding of cipher video. The section will show how the quotient technique is used to convert numbers in RNS to their equivalent binary (weighted) number system. Two techniques are proposed; The technique for the dynamic range [0, $m_3 m_2 - 1$] and the technique for the dynamic range [0, $m_2 m_1 - 1$]. In each technique, two moduli in the moduli set are used in the reverse conversion process and one modulus is made redundant. The proposed techniques are integrated into the proposed perceptual video encryption schemes.

3.4.1 Proposed Reverse Converter for Dynamic Range $[\mathbf{0}, \widehat{\boldsymbol{M}} = \boldsymbol{m_3 m_2} - \mathbf{1}]$

Given the pairwise relative prime moduli set $\{m_1, m_2, m_3\} = \{2^n - 1, 2^n, 2^n + 1\}$ and residue (x_1, x_2, x_3) , X can be represented as in the following three different ways;

$$X = m_1 q_1 + x_1, \tag{3.4.1}$$

$$X = m_2 q_2 + x_2, \tag{3.4.2}$$

$$X = m_3 q_3 + x_3, \tag{3.4.3}$$

where $q_i, i = 1, 2, 3$ is the quotient when X is divided by the modulus $m_i, i = 1, 2, 3$.

Also, q_2 can be represented by

$$q_2 = q_3 + d, \tag{3.4.4}$$

where $d \in \mathbb{Z}\ and\ d \geq 0$.

Illustration 3.4.1

Consider $n = 3, \{m_1, m_2, m_3\} = \{7, 8, 9\}, \widehat{M} = 72$ and using Equations (3.4.2) and (3.4.3)

For $X = 17$,

$$17 = 8(2) + 1 = 9(1) + 8$$

$$\Rightarrow 2 = 1 + 1, where\ d = 1$$

For $X = 32$,

$$32 = 8(4) + 0 = 9(3) + 5$$

$$\Rightarrow 4 = 3 + 1, where\ d = 1$$

For $X = 63$,

$$63 = 8(7) + 7 = 9(7) + 0$$

$$\Rightarrow 7 = 7 + 0, where\ d = 0$$

Hence $q_2 = q_3 + d$

Thus Equation (3.4.3) becomes

$$X = m_2q_3 + m_2d + x_2. \tag{3.4.5}$$

From Equation (3.4.4)

$$q_3 = \frac{X - x_3}{m_3},$$

and substituting gives

$$X = m_2\left(\frac{X - x_3}{m_3}\right) + m_2d + x_2,$$

$$Xm_3 = Xm_2 - m_2x_3 + m_3m_2d + m_3x_2,$$

$$X = \frac{m_3m_2d + m_3x_2 - m_2x_3}{(m_3 - m_2)},$$

but $(m_3 - m_2) = 1$, hence

$$X = m_3m_2d + m_3x_2 - m_2x_3.$$

Eliminate the term in d by taking both sides modulo $\widehat{M}$

$$|X|_{\widehat{M}} = |m_3m_2d + m_3x_2 - m_2x_3|_{\widehat{M},}$$

$$X = |m_3x_2 - m_2x_3|_{\widehat{M},}. \tag{3.4.6}$$

3.4.2 Hardware Implementation

The binary representations of the respective residues are as follows

$$x_{1,n-1}x_{1,n-2}\dots x_{1,1}x_{1,0}, \tag{3.4.7}$$

$$x_{2,n-1}x_{2,n-2}\dots x_{2,1}x_{2,0}, \tag{3.4.8}$$

$$x_{3,n}x_{3,n-1}\dots x_{3,1}x_{3,0}. \tag{3.4.9}$$

Equation (3.4.6) can be simplified as

$$X = |\tau_1 + \tau_2|_{2^n(2^n+1)}$$

$$= \left|\tau_{1,2n-1}\tau_{1,2n-2}\dots\tau_{1,1}\tau_{1,0} + \tau_{2,2n-1}\tau_{2,2n-2}\dots\tau_{2,1}\tau_{2,0}\right|_{2^n(2^n+1)}$$

$$= X_{2n-1}X_{2n-2}\dots X_1X_0, \tag{3.4.10}$$

where

$$\tau_1 = 2^n x_2 + x_2$$

$$= x_{2,n-1}x_{2,n-2}\dots x_{2,1}x_{2,0}\overbrace{00\dots00}^{n-bits} + x_{2,n-1}x_{2,n-2}\dots x_{2,1}x_{2,0}$$

$$= x_{2,n-1}x_{2,n-2}\dots x_{2,1}x_{2,0} \bowtie x_{2,n-1}x_{2,n-2}\dots x_{2,1}x_{2,0}$$

$$= \tau_{1,2n-1}\tau_{1,2n-2}\dots\tau_{1,1}\tau_{1,0}, \tag{3.4.11}$$

and

$$\tau_2 = |-2^n x_3|_{2^n(2^n+1)}$$

$$= \left|\bar{x}_{3,n}\bar{x}_{3,n-1}\dots\bar{x}_{3,1}\bar{x}_{3,0}\overbrace{11\dots11}^{n-bits}\right|_{2^n(2^n+1)}$$

$$= \tau_{2,2n-1}\tau_{2,2n-2} \dots \tau_{2,1}\tau_{2,0}. \qquad (3.4.12)$$

3.4.3 Hardware Realization

The hardware of the proposed scheme applicable to values of $n \leq 4$, can be realized with a simple modulo carry propagate adder (MCPA) and an inverter. Equation (3.4.11) is a concatenation of bits, which would not require any hardware resource for that operation. Equation (3.4.12) would only require a bit inverter, which is not expensive and at the same time, does not impose undue delay on the scheme. It is only in Equation (3.4.10) that an MCPA of length $2n$-bits would be required to add the results of (3.4.11) and (3.4.12). Thus, the hardware resources in this regard are $2n$-bits wide whiles the delay imposed by such an adder is $4n$-bits. The delay for an inverter is usually unity; therefore, the total delay that would be imposed on the scheme is $(4n + 1)$-bits. The block diagram for the reverse conversion is shown in Figure 3.4.1.

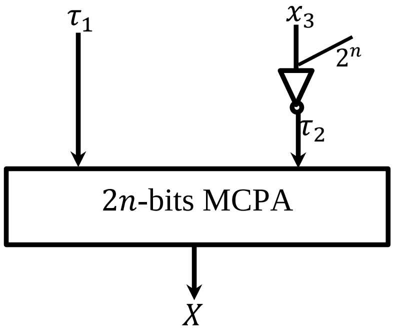


Figure 3.4.1 Block diagram for the reverse conversion

Example 3.4.1

Consider $n = 3, X = 51, \{m_1, m_2, m_3\} = \{7, 8, 9\}, (x_1, x_2, x_3) = (2, 3, 6), \widehat{M} = 72$

$$|m_3x_2 - m_2x_3|_{\widehat{M}} = |9(3) - 8(6)|_{72}$$

$$= |27 - 48|_{72}$$

$$= |-21|_{72}$$

$$51 = X$$

Example 3.4.2

Consider $n = 4, X = 133, \{m_1, m_2, m_3\} = \{15, 16, 17\}, (x_1, x_2, x_3) = (13, 5, 14),\ \widehat{M} = 272.$

$$|m_3x_2 - m_2x_3|_{\widehat{M}} = |17(5) - 16(14)|_{272}$$

$$= |85 - 224|_{272}$$

$$= |-139|_{272}$$

$$133 = X$$

3.4.4 Proposed Reverse Converter for Dynamic Range $[\mathbf{0}, \widehat{\boldsymbol{M}} = \boldsymbol{m_2 m_1} - \mathbf{1}]$

From Equations (3.4.1) and (3.4.2),

$$q_1 = q_2 + d.$$

Illustration 3.4.2

Consider $n = 3, \{m_1, m_2, m_3\} = \{7, 8, 9\}, \widehat{M} = 56$ and using Equations (3.4.1) and (3.4.2)

For $X = 17$,

$$17 = 7(2) + 3 = 8(2) + 1$$

$$\Rightarrow 2 = 2 + 0, where\ d = 0$$

For $X = 32$,

$$32 = 7(4) + 4 = 8(4) + 0$$

$$\Rightarrow 4 = 4 + 0, where\ d = 0$$

For $X = 49$,

$$63 = 7(7) + 0 = 8(6) + 1$$

$$\Rightarrow 7 = 6 + 1, where\ d = 1$$

Hence $q_1 = q_2 + d$.

Thus equation (3.4.2) becomes

$$X = m_1 q_2 + m_1 d + x_1.$$

From equation (3.4.3),

$$q_2 = \frac{X - x_2}{m_2},$$

and substituting gives,

$$X = m_1\left(\frac{X - x_2}{m_2}\right) + m_1 d + x_1$$

$$Xm_2 = Xm_1 - m_1x_2 + m_2m_1d + m_2x_1$$

$$X = \frac{m_2m_1d + m_2x_1 - m_1x_2}{(m_2 - m_1)}.$$

But $(m_2 - m_1) = 1$

$$\therefore X = m_2m_1d + m_2x_1 - m_1x_2. \tag{3.4.13}$$

Eliminate the term in d by taking both sides of Equation (3.4.13) modulus $\widehat{M}$

$$|X|_{\widehat{M}} = |m_2m_1d + m_2x_1 - m_1x_2|_{\widehat{M}}$$

$$\therefore X = |m_2x_1 - m_1x_2|_{\widehat{M}}. \tag{3.4.14}$$

Example 3

Consider $n = 3, X = 51,\ \{m_1, m_2, m_3\} = \{7, 8, 9\},\ (x_1, x_2, x_3) = (2, 3, 6), \widehat{M} = 56.$

$$|m_2x_1 - m_1x_2|_{\widehat{M}} = |8(2) - 7(3)|_{56}$$

$$= |16 - 21|_{56}$$

$$= |-5|_{56}$$

$$51 = X$$

Example 4

Consider $n = 4, X = 133, \{m_1, m_2, m_3\} = \{15, 16, 17\}, (x_1, x_2, x_3) = (13, 5, 14), \widehat{M} = 240.$

$$|m_2 x_1 - m_1 x_2|_{\widehat{M}} = |16(13) - 15(5)|_{240}$$

$$= |208 - 75|_{240}$$

$$= |133|_{240}$$

$$133 = X$$

3.4.5 RNS Integration with Proposed Video Cryptosystems

In Figure 3.4.2, the forward and proposed reverse converters for the moduli set $\{2^n - 1, 2^n, 2^n + 1\}$ are fitted after the encryption algorithm and before the decryption algorithm respectively. Pixels values of encrypted video are passed through the forward converter which yields two residues (x_2, x_3) corresponding to the moduli $\{2^n\}$ and $\{2^n + 1\}$. Continuous bitstream of the residues are transmitted through the transmission channel in fixed length code words. At the receivers' end, the fixed length code words are transformed back to continuous form and the proposed reverse converter is applied to recover the cipher video. The original video is then recovered from the cipher video by applying the decryption algorithm sub-block.

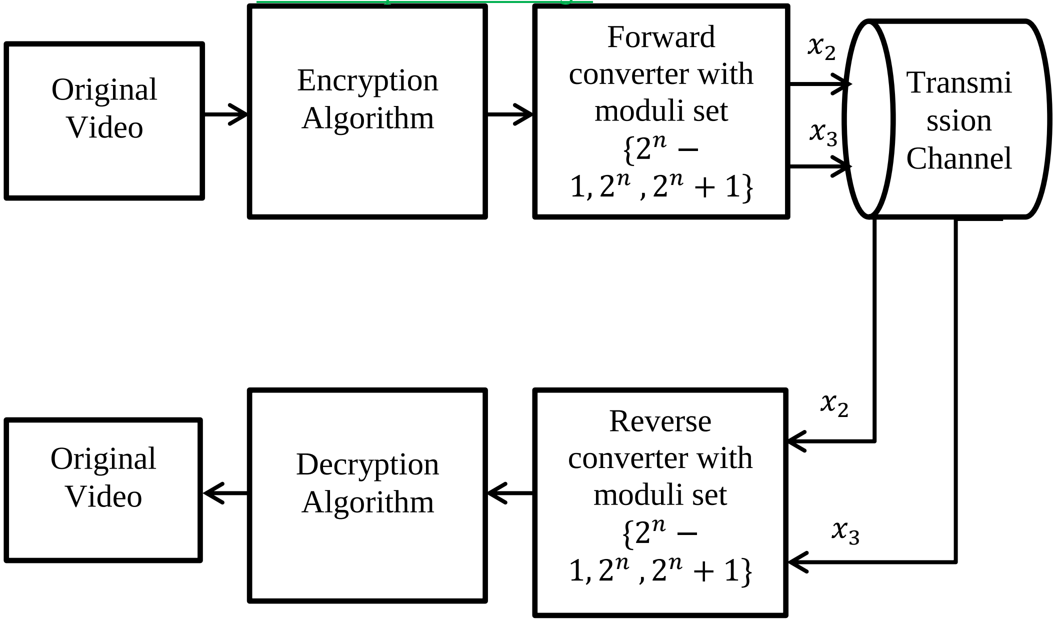

Original Video
Encryption Algorithm
Forward converter with moduli set $\{2^n - 1, 2^n, 2^n + 1\}$
x_2
x_3
Transmission Channel
x_2
x_3
Reverse converter with moduli set $\{2^n - 1, 2^n, 2^n + 1\}$
Decryption Algorithm
Original Video

Figure 3.4.2 Block diagram of Proposed Integration of RNS with perceptual video cryptosystems

The MATLAB Simulink systems to test the proposed integrated scheme are shown in Figure 3.4.3 and Figure 3.4.4. In Figure 3.4.3, RNS is integrated with the proposed perceptual video encryption algorithm via unit anti-diagonal matrix. After encryption with 'EncryptionAlgorithm1' sub-block, the cipher video is passed through the sub-block 'RNSEncoder' to be encoded into two residual videos x_2 and x_3 for transmission. Since residual values are less than that of the original, transmission is much easier and faster. At the receiver's end, residual videos are decoded by the 'RNSDecoder' sub block into the cipher video followed by decryption into the plain video using the 'DecryptionAlgorithm1' sub-block.

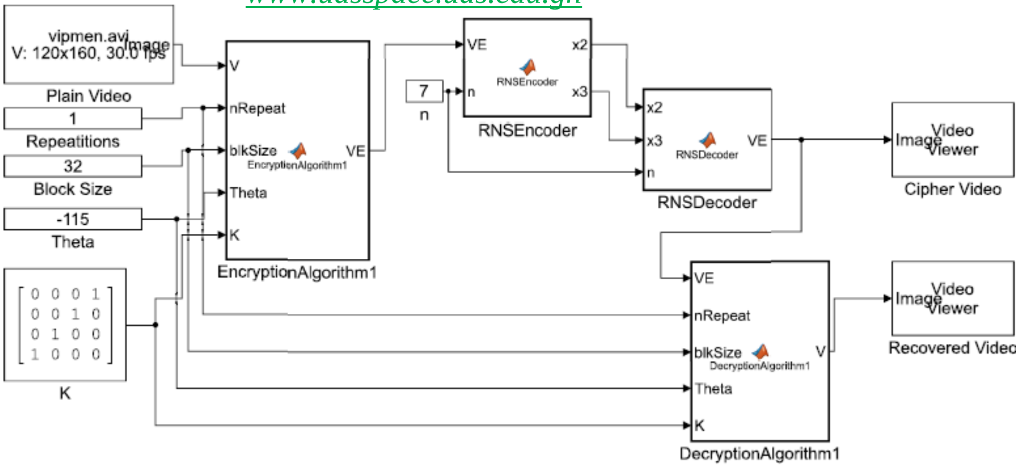


Figure 3.4.3 Integration of RNS with proposed perceptual video via unit anti-diagonal matrix

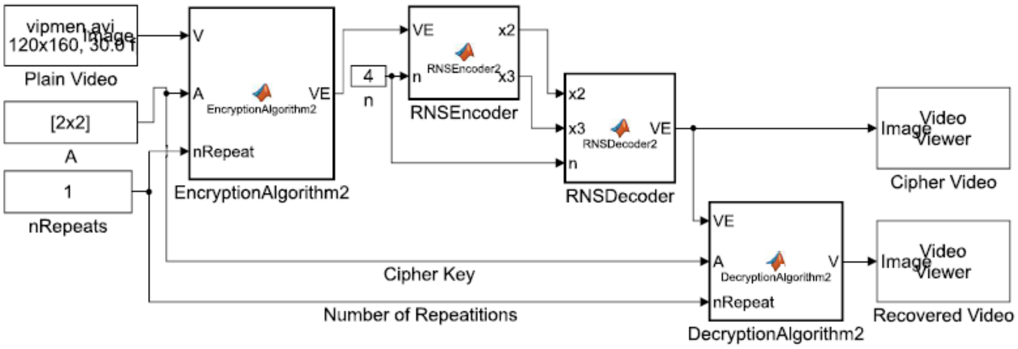


Figure 3.4.4 Integration of RNS with proposed perceptual video using orthogonal matrix

The integration of RNS with the proposed perceptual video encryption and decryption algorithms using orthogonal matrix is presented in Figure 3.4.4 where the encryption (‘EncryptionAlgorithm2’) and decryption (‘DecryptionAlgorithm2’) sub-blocks perform their respective processes explained in Section 3.2. Also, the ‘RNSEncoder’ and ‘RNSDecoder’ sub-blocks perform forward and reverse conversion of cipher video respectively.

Figure 3.4.5 shows experimental results of the residual video after encoding a cipher frame using $n = 4$. The dark pictures of parts **b)** and **c)** confirm the smaller values achieved through the RNS encoder. Any adversary receiving these will require extra efforts to decode them to the cipher state before efficient decryption can occur.

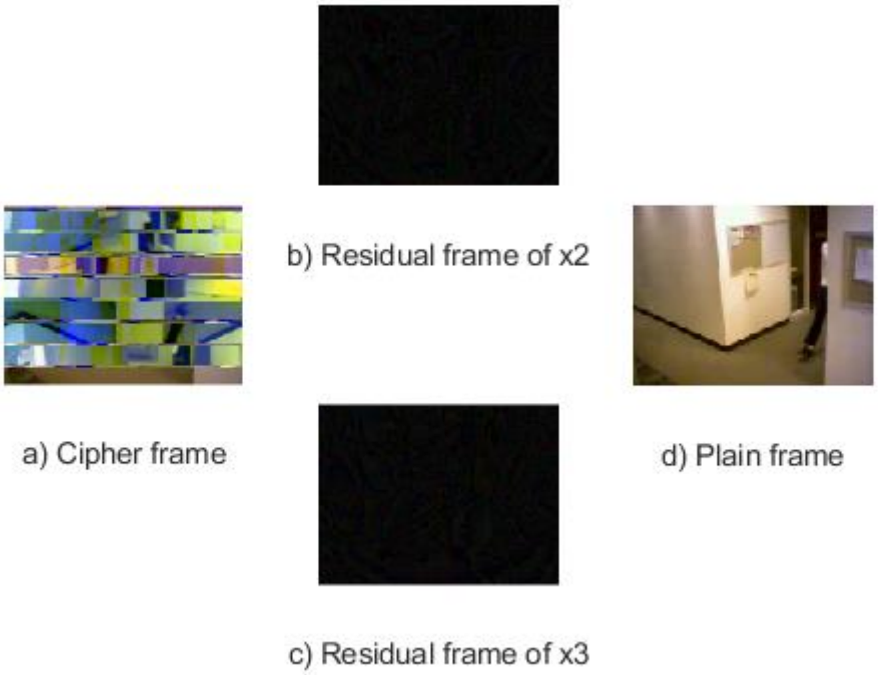


Figure 3.4.5 Residual frames of encoded video using RNS. (a) Cipher frame, (b) Residual frame x2, (c) Residual frame x3 and (d) Recovered Frame

3.5 Proposed Integrated Perceptual Video Encryption Algorithm

Integrating all, the proposed perceptual video encryption algorithm is shown in Figure 3.5.1. Original video containing audio bitstream is split into two; audio bitstream and video frames. The audio bitstream is passed through the K-shuffle encryption sub-block and encrypted into cipher audio for transmission. The video frames on the other hand go through the perceptual video encryption and RNS encoder sub-blocks to be encrypted and encoded into two residual cipher frames x_2 and x_3 which are then transmitted through a communication channel. At the receiver's end, the two residual cipher video frames are decoded into single cipher frame by means of the RNS decoder sub-block followed by decryption by the perceptual video decryption sub-block. Each recover video frame is integrated with the corresponding recovered audio bitsteams (via the K-shuffle decryption algorithm sub-block) into the original video. At the receivers' end, only legitimate users (those with the correct cipher keys) will view and hear the original video and audio.

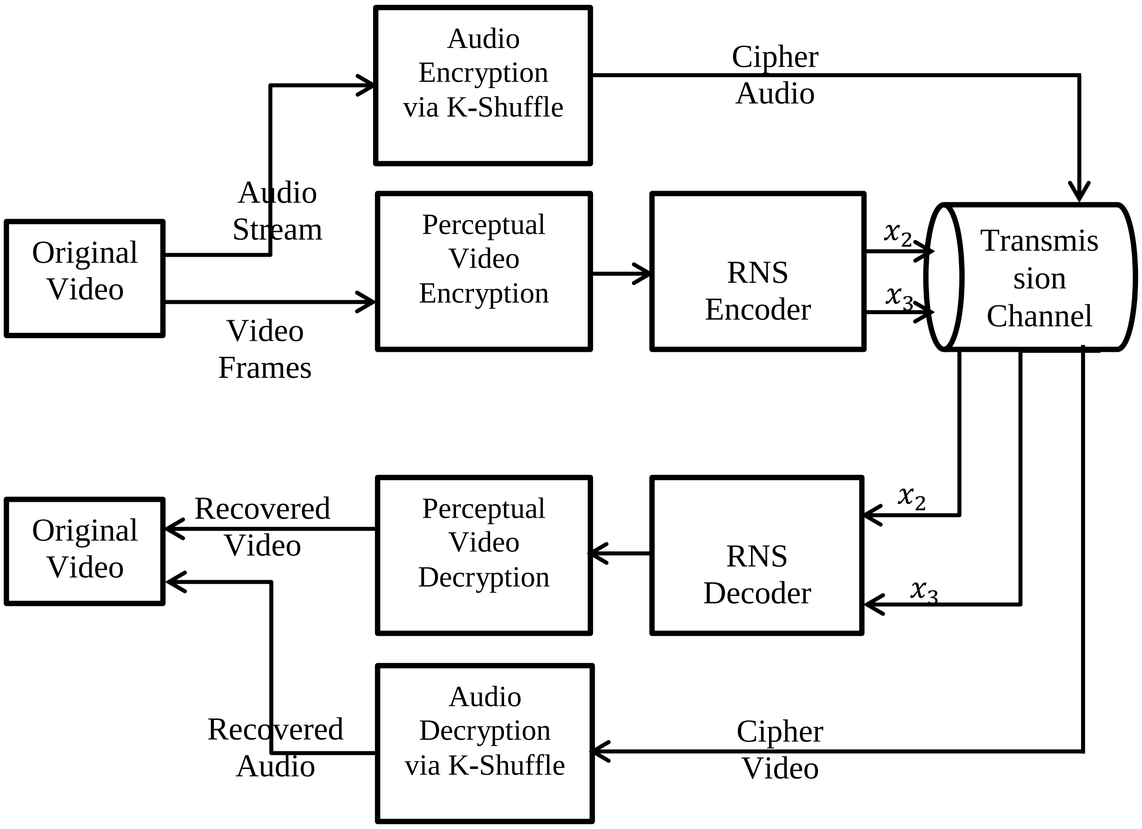


Figure 3.5.1 Proposed integrated perceptual video encryption system

3.6 Data Analysis

Each proposed algorithm was coded in MATLAB Simulink (version 8.0.0.783 (R2012b)) and simulated with different real-time audio/video data using an HP notebook laptop (Windows 8.1 Pro 64-bit operating system, AMD E-300 APU (2 CPUs) 1.3GHz and 4GB memory). The data obtained were analysed on the following;

- **Processing Time.** This is the time taken by MATLAB to process a given audio/video data. In order to measure this, each algorithm was run several times using different inputs and the average processing time computed. The essence of this measure is to determine whether a proposed algorithm and ideal for real-time implementation (see Appendix B).
- **Cipher Key Sensitivity Analysis.** A given cryptographic algorithm should be sensitive to slight changes in the cipher key(s). That is, every distinct cipher key(s) should result to a distinct cipher/recovered video. No two or more cipher keys should produce the same cipher/recovered video. This is essential to thwart any unauthorised users who may want to guess the cipher key(s).
- **Cipher Key Space Analysis.** This analysis is done to determine the domain of allowed cipher keys of the individual algorithms and whether they are large enough to resist any Bruce-force attack. The key space is computed from the individual cipher keys parameters and presented in bits.
- **Visual Degradation.** Visual distortion of video can be measured using the peak signal-to-noise ratio (PSNR). When comparing image

compression quality, the Mean Square Error (MSE) and the Peak Signal to Noise Ratio (PSNR) are the two most used error metrics. While the MSE denotes the cumulative squared error between the compressed and the original image, PSNR is a measure of the peak error. The lesser the value of MSE, the lesser the error, while lesser values of PSNR indicate higher visual degradation. To compute the PSNR, first calculate the mean-squared error using the following equation

$$MSE = \frac{\sum_{M,N}[I_1(m,n) - I_2(m,n)]^2}{M \times N}, \qquad (3.4.15)$$

where $I_1(m,n)$ and $I_2(m,n)$ are the pixel values of the original and cipher frames respectively, M and N represent the number of rows and columns in the source images respectively.

Thus PSNR is computed by the following equation

$$PSNR = 10\log_{10}\left[\frac{R^2}{MSE}\right], \qquad (3.4.15)$$

where R is the maximum fluctuation in the source image data type. For instance, a source image with double-precision floating-point data type has $R =\ 1$. Suppose that it has an 8-bit unsigned integer data type $R = 255$.

To carry out this analysis, a MATLAB function (see Appendix C) was written to compute the MSE and PSNR values between plain and cipher video frames.

- **Chosen/Known-Plaintext Attacks**. Chosen and known-plain message attacks are attacks in which the adversary besides knowing the encryption algorithm, can access or choose a set of plain messages, modify them and observe the corresponding cipher message. These attacks are considered the most prevalent and most threatened in the point of view of content protectors and it is also considered effective against XOR-ed based cryptosystems. One of the requirements of a good cryptosystem is its security against these attacks. For each algorithm, a plain video frame and its corresponding cipher frame area extracted after simulation processes. Also, an unknown cipher frame was extracted from an unknown cipher video. The following processes were then carried out with the three different frames:
    - Obtain the histograms of the plain, cipher and unknown cipher frames
    - XOR the plain frames with their corresponding cipher frames to obtain the fourth frames and their histograms.
    - XOR the unknown cipher frames with the fourth frames and obtain the fifth frames and their histograms.
    - Compare the histograms of the plain frames and their corresponding XOR-ed fifth frames for any resemblance. The levels of resemblances determine the levels susceptibilities to this attack.

3.7 Chapter Summary

The proposed perceptual video cryptosystems have been well expatiated in this chapter. Mention is made of the proposed perceptual video encryption algorithm via unit anti-diagonal matrix where the source video is scrambled into cipher video with varying degradations. Also, perceptual video encryption is achieved by means of pixel encoding using orthogonal matrices. The chapter also presented a proposed audio encryption technique that well integrates with the perceptual video encryption algorithms to secure audio bitstream in video.

For fast transmission of cipher video across networks, RNS is integrated into the proposed integrated cryptosystem to produce residual cipher video that can be accommodated by limited bandwidth. Apart from this function, the introduction of RNS also adds an extra layer of security to the proposed cryptosystem.

In the next chapter, detail analyses of the proposed encryption and decryption algorithms are presented.

CHAPTER FOUR

RESULTS PRESENTATIONS

In this chapter, the software implementations and performance analyses of the various proposed techniques are presented and some conclusions drawn. Analyses of the proposed perceptual video encryption algorithm via unit anti-diagonal matrix are presented Section 4.1. Section 4.2 presence analyses of the simulated results for the proposed perceptual video encryption algorithm using orthogonal matrix while the analyses of the proposed audio encryption algorithm using k-shuffle technique are presented in Section 4.3. Finally, the analyses of the proposed integrated RNS cryptosystem are presented in Section 4.4.

4.1 Analyses of the Proposed Perceptual Video Encryption Algorithm via Unit Anti-Diagonal Matrix

The proposed algorithms were coded in MATLAB Simulink (see Figure 3.1.3 and Sections A.1 – A.2 of Appendix A) in order to simulate and analyse their performances on processing speed, key sensitivity, key space, visual degradation and chosen/known-plaintext attacks. It is revealed after data analyses that this technique has low computational overhead (which is ideal for on-the-fly encryption) and it can resist chosen/known-plaintext attacks. In comparison with others existing techniques, it is also shown in this section that, the technique performs better in terms of the quality of visual degradation, processing time and independent of the video formats.

4.1.1 Processing Time

The processing time used by MATLAB to execute the encryption algorithm are presented in Sections B.1 – B.3 of Appendix B. A summary of the average processing time is shown in Table 4.1.1. All together, the total time needed to encrypt a 26 seconds video is estimated to be 0.487s. Thus, approximately 1.873% of a video length is required to encrypt/decrypt the video. This time is better than that obtained by Kulkarni et al. (2013) and the algorithm is ideal for real-time video processing and for use with low-level consumer devices.

Table 4.1.1 Average Processing Time of Proposed Encryption Algorithm

Video File (length)	Processing Time (sec)
Carphone (12sec)	0.147
Xylophone (4sec)	0.090
News (10sec)	0.250

However, the processing time will increase as the number of iteration (*I*) increases and block size (n) decreases.

4.1.2 Cipher Key Sensitivity Analysis

This analysis test how sensitive the proposed algorithm is to a small change in the cipher key values. Thus, procedure is to try to recover the plain video from its corresponding cipher video with a small change to the cipher key. Experimental results of failed decrypted frames due to slight changes to the cipher keys are presented in Figure 4.1.1.

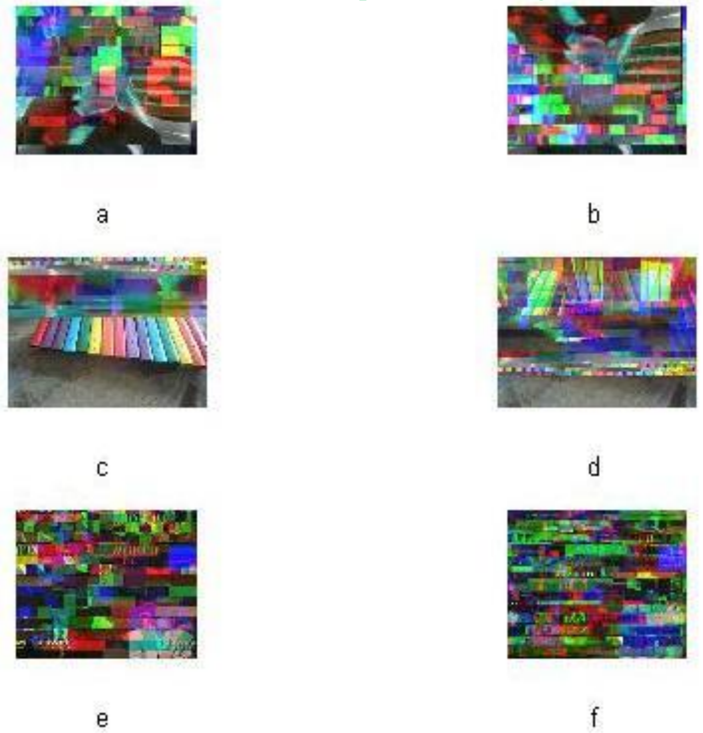


Figure 4.1.1 Key sensitivity analysis. a, c, e are cipher frames, b, d, f are the respective recovered frames with a slight change in cipher key.

The small changes are ‘Carphone’ (Ixy(**10,0**)) , ‘xylophone’ (θ=**112.9**) and ‘news’ (n=**17**) with other parameters held constant. It can be seen that no better perceptual data has be recovered. Hence, the proposed algorithm is highly sensitive to any small change in the cipher keys.

4.1.3 Key Space Analysis

The Bruce-force attack is rendered ineffective when the cipher key space is reasonable large. The security of the proposed method lies on the choice of θ, l, n and K. $\theta \in \mathcal{R}$ and $l, n, k \in \mathbb{Z}$. The length of θ is 2^9 and the other three cipher key parameters are 2^8 each. This gives a total of $2^9 \times 2^8 \times 2^8 \times 2^8 = 2^{33} \approx 8589934592$ key combinations. Obviously this space is good enough to resist Bruce-force attack for video whose financial interest is only for a short period.

4.1.4 Visual Degradation

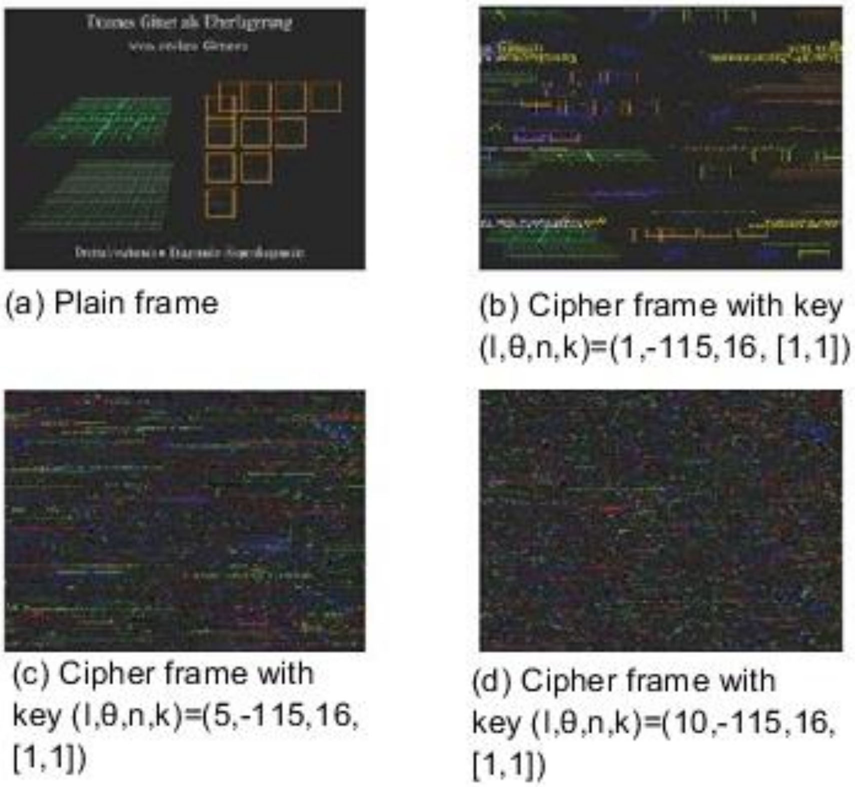
In this analysis, a plain frame (from a plain video) and its corresponding cipher frame (from a cipher video) were extracted and passed to the "**PsnrMse**" function (See Section A.7 of Appendix A) and the outputs were recorded (see Sections C.1 – C.3 of Appendix C). The average MSE and PSNR values of the results are presented in Table 4.1.2.

Table 4.1.2 Average MSE and PSNR values of selected plain and cipher video frames

File (cipher keys)	MSE	PSNR (dB)
Carphone.mp4(1, 32, 60, [10,1])	1.3253e+03	16.9103
xylophone.mpg (1,62, 113, [5,5])	646.0534	20.0564
News.avi(1,6, -15,[1,1])	1.4564e+03	16.8019

From the Table 4.1.2, the higher values of MSE and the lower values of PSNR imply that, there is large difference between cipher videos and their plain counterparts. Hence, the proposed algorithm has high visual degradation and acceptable security. Visual degradation increases as the number of iterations and block size increases and decreases respectively.

Degradation in visual quality of cipher video is independent on the underlying make-up of the source video. The proposed method is better than Li et al. (2007) whose technique is dependent on the amplitudes of the intra DC coefficients. In their proposed algorithms cipher video produced from video whose intra-frame DC coefficients are all zeros is still "partially perceptible" when the highest degradation parameter (i.e. $p_{sr} = 1$) is used. Figure 4.1.2 shows the performance of the proposed method on MPEG-1 video whose intra-frame DC coefficients are zeros. It shows that completely layered encryption is assured even when the intra-frame DC coefficients of the source video are all zeros.



(a) Plain frame

(b) Cipher frame with key (l,θ,n,k)=(1,-115,16, [1,1])

(c) Cipher frame with key (l,θ,n,k)=(5,-115,16, [1,1])

(d) Cipher frame with key (l,θ,n,k)=(10,-115,16, [1,1])

Figure 4.1.2 The encryption results of the 150th frame in an MPEG-1 video

(Video Source: http://www5.in.tum.de/forschung/visualisierung/duenne_gitter/DG_4.mpg)

4.1.5 Chosen/Known-Plaintext Attacks

In Figure 4.1.3, part **c)** is the histogram of XOR-ing the plain video frame **a)** and its corresponding cipher frame **b)**. Part **e)** is obtained by XOR-ing **c)** and an unknown cipher frame **d)**. Since part e) failed in recovering any meaningful histogram (as in a)), the proposed method is safe against chosen/known-plaintext attack.

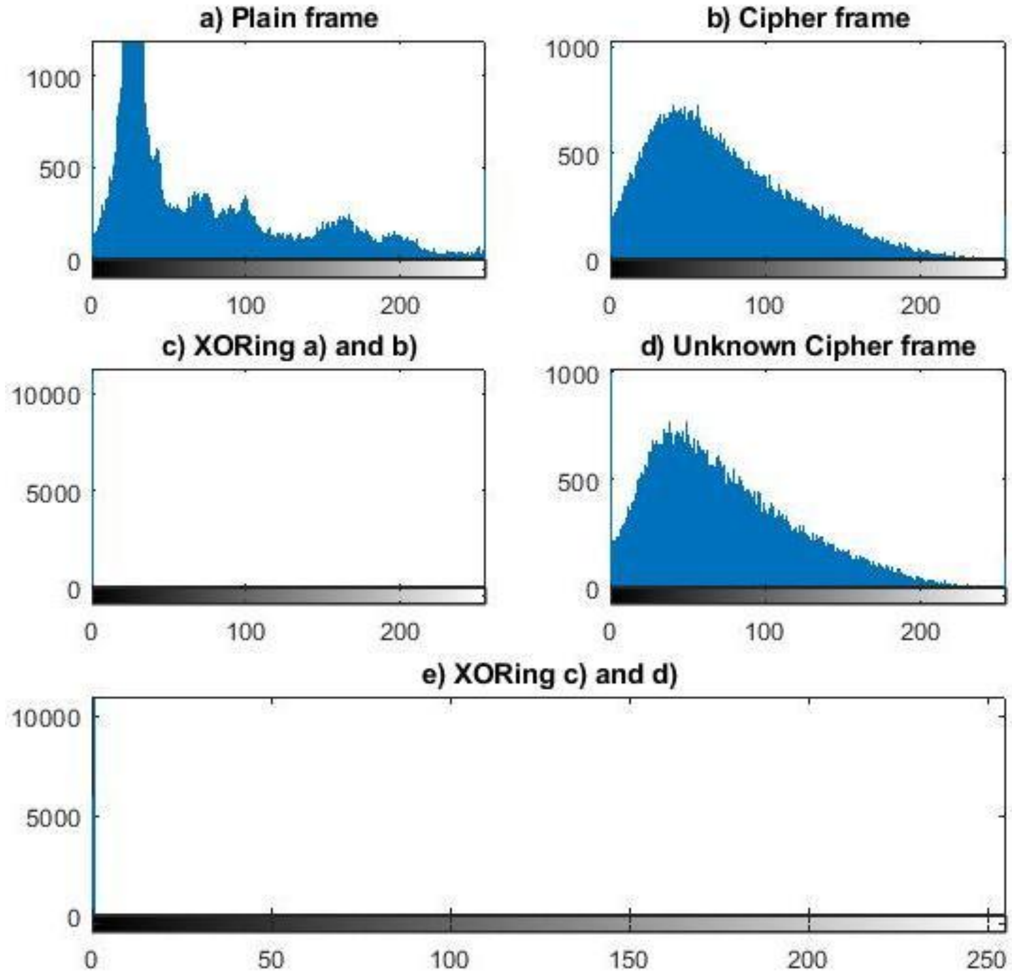


Figure 4.1.3 Histograms of plain and cipher video frames

4.1.6 Conclusion

In this section, a novel perceptual video encryption algorithm that uses unit anti-diagonal and rotation matrices to visually degrade video with varying perceptibility has been proposed. The technique has been extensively simulated using MATLAB and its performance measured.

In terms of processing time, the proposed technique is found to approximately encrypt only 1.873% of the video length which is ideal for real-time video processing. This is better than the 3.7% of video length encrypted using the scheme of Kulkarni et al. (2013).

A good cryptographic algorithm should be sensitive to any slight alteration in the cipher key. On this score, the proposed scheme is found to be highly sensitive to any slight alteration of the cipher key values. This is clearly depicted in Figure 4.1.1 where slight changes in the cipher key values led to unsuccessful retrieval of the original quality video.

It is also found that the scheme has a large enough key combinations (≈ 268435456) to preclude Bruce-force attacks for video whose financial interest is only for a short period.

Visual degradation is the hallmark of perceptual video encryption schemes since perceptible data is controlled by that. The results obtained in Table 4.1.2 show that the proposed scheme offers a high level of visual degradation and acceptable level of security. This is evident in the high values obtained for MSE and low values obtained for PSNR. On this score too, the scheme is found to outperform the work of Li et al. (2007) whose technique depend on the underlying make-up of the source video and could not obtain complete encryption for video whose intra-frame DC coefficients are zero.

One shortcoming of most perceptual video encryption algorithms are their susceptibilities to chosen/known-plaintext attacks. The results obtained in Figure 4.1.3 for the XOR-ing operations reveal that the proposed scheme resists these attacks

In summary, the experimental results and analyses show that the proposed perceptual video encryption algorithm via unit anti-diagonal matrix:

- ✓ produces cipher video whose perceptibility is controlled by varying the block size and the number of iteration,
- ✓ can be applied to all formats of video,
- ✓ is resistant to chosen/known-plaintext attack,
- ✓ faster when compared with other existing techniques.

4.2 Analyses of the Proposed Perceptual Video Encryption Algorithm using Orthogonal Matrix

The MATLAB codes to simulate the proposed encryption and decryption algorithms are presented respectively in Section A.3 and Section A.4 of Appendix A. After simulation, data analyses were carried out on processing speed, key sensitivity, key space, visual degradation, chosen/known-plaintext attacks and decryption efficiency.

4.2.1 Procession Time

The average processing time (in seconds) for ten (10) simulations of the encryption and decryption processes (see Sections B.4 and B.5 in Appendix B) are summaries in Tables 4.2.1 and 4.2.2, and Figure 4.2.1. It can be deduced from Tables 4.2.1 and 4.2.2 that the average processing time spent in the encryption and decryption processes decreases as the size of the cipher key increases. This is attributed to the reduction in the number of iteration needed to completely encrypt/decrypt video frames as the size of the cipher key increases.

Table 4.2.1 Average processing time of encryption process

Size of cipher key	2x2	3x3	4x4	5x5	6x6	7x7	8x8	16x16	32x32
Average Encryption time	0.3563	0.2052	0.1370	0.1216	0.1204	0.1106	0.0938	0.0400	0.0242

Table 4.2.2 Average processing time of decryption process

Size of cipher key	2x2	3x3	4x4	5x5	6x6	7x7	8x8	16x16	32x32
Average Decryption time	0.3670	0.2151	0.1451	0.1301	0.1201	0.1129	0.0868	0.0411	0.0246

The decreases in processing time is clearly seen in Figure 4.2.1 as the curves slop downward from a smaller (2×2) cipher key to a larger (32×32) on the plot of the average CPU time against the dimension of the cipher keys for the encryption and decryption processes.

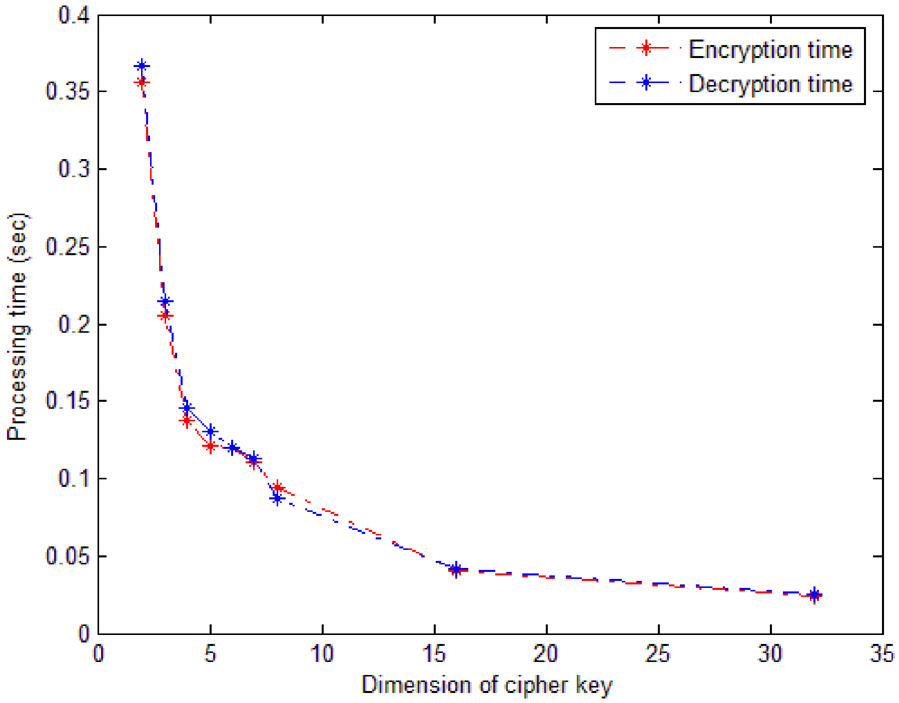
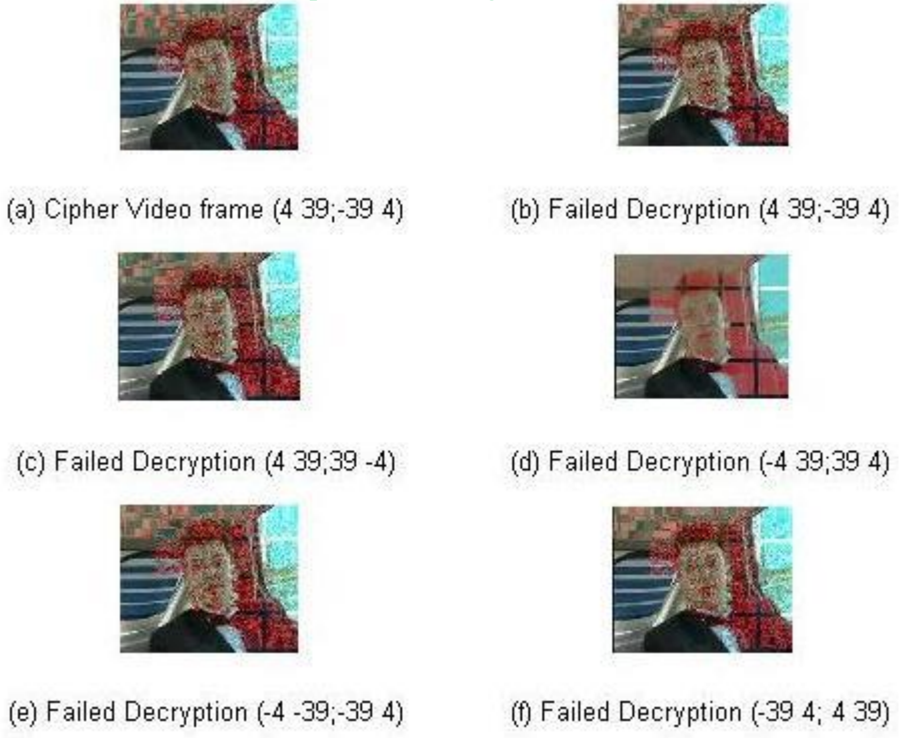


Figure 4.2.1 Plot of average CPU time of encryption and decryption processes

4.2.2 Cipher Key Sensitivity Analysis

The behaviour of the proposed algorithms to slight changes in the cipher key was measured using a 32×32 cipher key matrix. In the simulation, the encryption key was constructed from $\begin{bmatrix} 4 & 39 \\ -39 & 4 \end{bmatrix}$ while the decryption keys defer only in sign or position of the value. Figure 4.2.2 show the results obtained. Slight changes in the decryption key lead to unsuccessful recover of plain video from cipher video. This indicates that the proposed algorithms are highly sensitive to changes in the cipher keys. Thus, each cipher key used gives different cipher video.



(a) Cipher Video frame (4 39;-39 4)

(b) Failed Decryption (4 39;-39 4)

(c) Failed Decryption (4 39;39 -4)

(d) Failed Decryption (-4 39;39 4)

(e) Failed Decryption (-4 -39;-39 4)

(f) Failed Decryption (-39 4; 4 39)

Figure 4.2.2 Cipher key sensitivity analysis

4.2.3 Key Space Analysis

The proposed method constructs cipher keys from two integer numbers (a, b) between 2 and $n - 1$ (frame height/width of video). Thus the domain of cipher key space increases as the size of video frame increases. For example, given a frame height of 256 (i.e. 8-bits number), a total of $2^8 \times 2^8 = 2^{16}$ different combinations of (a, b) can be made. Also given that cipher key A can be constructed in 8 different ways and $2^8 - 1$ different sides further increases the key space by a multiple of $2^{11} - 1$. This gives a possible key space of $2^{16} \times 2^{11} - 1 = 2^{27} - 1 \approx 134{,}217{,}727$. Obviously, this is fairly good enough to withstand Bruce-force attacks (given limited resources) especially for videos (e.g. live telecast of football match) whose financial value is only for a short time period.

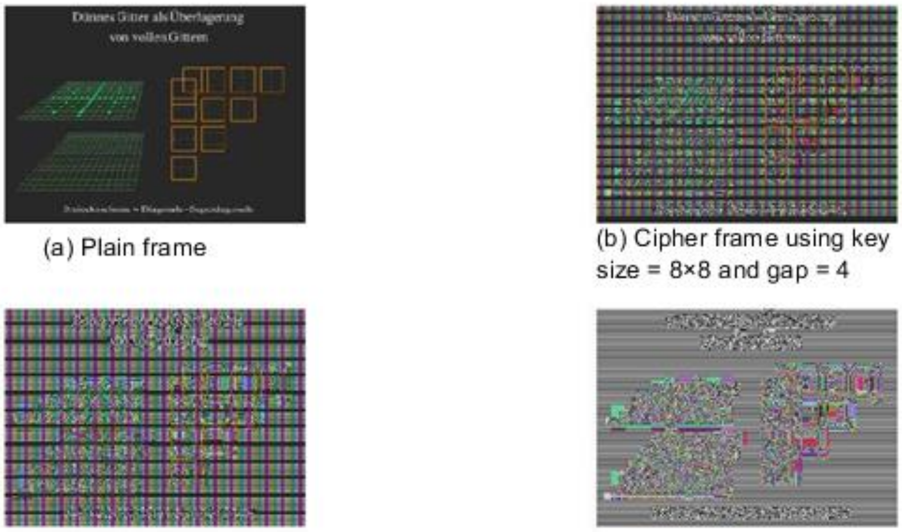
4.2.4 Visual Degradation

Simulation results for the analyses of visual degradation of a “carphone.mp4” video are presented in Sections C.4 – C.6 of Appendix C. A summary of these results is shown in Table 4.2.3. The higher values of MSE and lower values of PSNR mean difference exists between the cipher video and their plain form. Visual degradation increases as the dimension of the cipher key and block size increases.

Table 4.2.3 Average MSE and PSNR values for plain and cipher video of frame 6 of ‘carphone.mp4’ video

Size of cipher key	MSE	PSNR (dB)
8x8	113.6688	29.6696
16x16	286.4325	26.6425
64x64	311.6498	25.7414

It is also determined that the degradation in visual quality of cipher video is independent of the underlying make-up of the source video. The results obtained here is also opposed to and better than the work of Li et al. (2007). Figure 4.2.3 shows the performance of this proposed method on MPEG-1 video whose intra-frame DC coefficients are zeros.



(a) Plain frame

(b) Cipher frame using key size = 8×8 and gap = 4

(c) Cipher frame using key size = 16×16 and gap = 4

(d) Cipher frame with key size = 16×16 and gap = 0

Figure 4.2.3 The encryption results of the 150th frame in an MPEG-1 video

(Video Source: http://www5.in.tum.de/forschung/visualisierung/duenne_gitter/DG_4.mpg)

4.2.5 Chosen/Known-Plaintext Attacks

Figure 4.2.4 shows the histogram analysis of the effect of chosen/known-plaintext attacks on the proposed technique. Part c) is the histogram of XOR-ing the plain video frame **a)** and its corresponding cipher cipher frame **b)**. Part **e)** is obtained by XOR-ing **c)** and an unknown cipher frame **d)**. Since the histogram of part **a)** and part **e)** are completely different, the proposed technique is resistant against chosen/known-plaintext attacks.

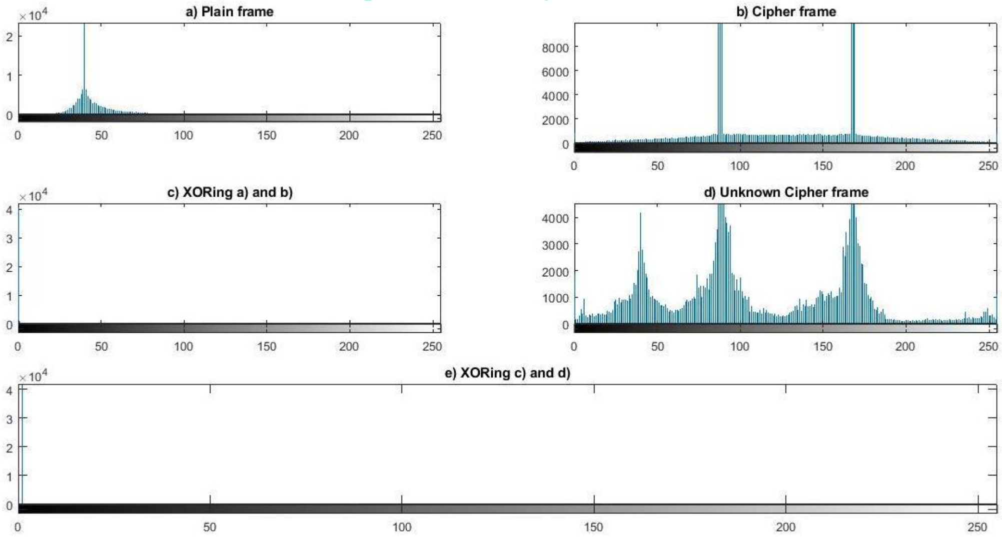


Figure 4.2.4 Histograms of plain and cipher video frames

4.2.6 Decryption Efficiency

In order to the efficiency of the proposed decryption algorithm, different cipher key sizes were used to encrypt and decrypt 'carphone.mp4' video. After each encryption/decryption process, the original video frame and the corresponding recovered cipher frames were extracted and the sum of their difference computed. It was found that the sum of the difference between the pixels of the original and the recovered frames are zero (0) for all sizes of the cipher keys used. These results indicate that the decryption process recovers plain video with an efficiency of 100%. Thus, viewers who purchase and obtain the right cipher key can view the high quality versions of the original video. Table 4.2.4 shows the sum of the pixel difference between original and recovered video frames.

Table 4.2.4 Sum of pixel difference between original and recovered video frames

Size of cipher key	**2x2**	**3x3**	**4x4**	**5x5**	**6x6**	**7x7**	**8x8**	**16x16**	**32x32**	**64x64**
Sum of difference	0	0	0	0	0	0	0	0	0	0

4.2.7 Conclusion

In this section, a new perceptual video encryption algorithm that uses specially formulated orthogonal matrices has been presented. The scheme relies on some properties of the orthogonal matrices to encrypt video through encoding. Various simulations and data analyses on the results obtained have been done using MATLAB.

In the first place, simulation results show that the scheme approximately encrypts only 1.3703% of a video length when a 2×2- block of pixels is encrypted and approximately 0.0242% of video length is encrypted when a 32×32- block of pixels are used. Both processing times are better than that of method one and the work of Kulkarni et al. (2013). Figure 4.2.1 further shows that, both encryption and decryption times are approximately equal and they both decrease as the block size increases. These low processing times imply that the scheme is ideal for on-the-fly encryption.

It is also shown that the proposed scheme is sensitive to any slight change in the cipher key values. This is clearly depicted in Figure 4.2.2, where a 2×2- block of pixels is adopted and the slight changes in the values led to unsuccessful recovery of the original high quality video.

The number of different key combinations required by any Bruce-force attack to successfully exhaust the key space is ≈ 134,217,72. This is approximately half of the required key combinations for the first proposed scheme. Even though this scheme has a good key space and could survive some Bruce-force attacks, it is outperformed by proposed scheme one on key space analysis.

The proposed scheme is also independent of the underlying mark-up of the source video since complete encryption can be achieved for video whose intra-frame DC coefficients are all zero (see Figure 4.2.3). Even though this scheme outperforms the work of Li et al. (2007), it is however been outperformed by proposed scheme one when Table 4.1.2 and Table 4.2.3 are compared.

On the issue of chosen/known-plaintext attacks, the proposed scheme is found to resist these attacks. This is clearly shown in the failed recovery of the histogram of the original high quality video in Figure 4.2.4.

Another analysis worth mentioning is the decryption efficiency of this proposed scheme. Legitimate viewers/consumers with the correct cipher key should be able to view the original high quality video. Barring any error in the communication channel, the proposed scheme has a zero (0) error recovery rate which implies that 100% of original high quality video are obtained by the decryption algorithms.

In a nut shell, this proposes scheme;

- ✓ Is faster in encryption and decryption of video data.
- ✓ Recover video data free of error.
- ✓ Can resist chosen/known-plaintext attacks.

4.3 Analyses of the Proposed Audio Encryption Algorithm Using K-Shuffle Technique

Both proposed audio encryption and decryption algorithms were coded in MATLAB Simulink in order to measure their performances against existing algorithms (see Section A.6 of Appendix A). Data analyses conducted include processing speed, key space, chosen/known-plaintext attacks and error rate.

4.3.1 Error Rate

The difference between the recovered audio signals and the original input signals were measured. Figure 4.3.1 show the MATLAB Simulink system for measuring the error rate. As an example, the cipher key (n, m, i) for the simulation is $(32, 32, 1)$ and the error rate computed between the original audio signal and the recovered audio signal is zero (0). This implies that the decryption algorithm insures that viewers who purchase and obtain the right cipher key will hear the high quality audio that accompanies source video.

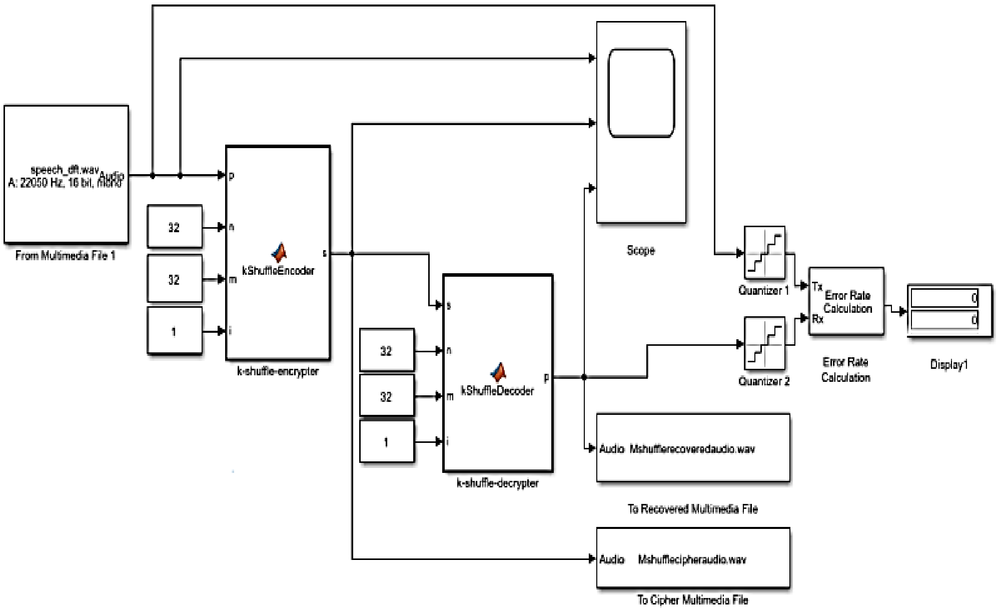


Figure 4.3.1 MATLAB Simulink to compute error rate of original and recovered audio signals

Table 4.3.1 summarizes the error rates of the proposed method, RSA method and LOG method proposed by Khalil (2016). As seen from the table, the proposed method is better than the RSA and LOG methods in terms of recovery error rate.

Table 4.3.1 Comparison of error rate of three encryption methods

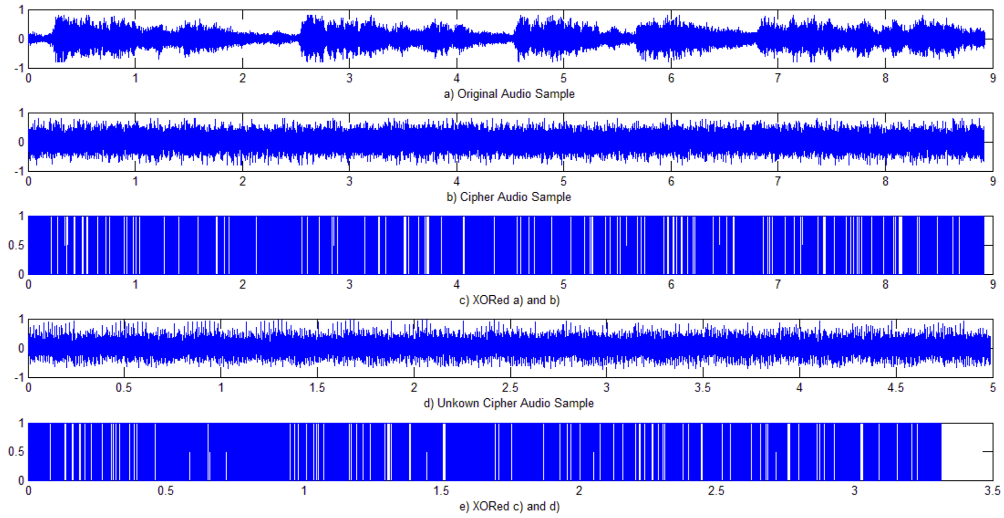
Method	**Cipher key**	**Error Rate**
RSA Method	$(e, n) = (5, 35)$	0.97370
LOG Method	$(a, b) = (0.0003, 300000)$	0.00017
Proposed Method	$(n, m, i) = (32, 32, 1)$	0.00000

4.3.2 Key Space

The proposed method uses three (3) parameters (n, m, i) for the cipher key. n and m increases as audio signal length increases and i does not exceed l. Thus each parameter has $n-$bits for $n-$bits audio sample. For a 16-bits audio sample, the number of combinations for n and m is $2^{16} \times 2^{16}$. Since the parameter i does not exceed l, the number of repetitions for any possible combination of n and m is $l-1$. Obviously, the key space is good enough for audio signal emanating from video data whose financial interest is only for a short time.

4.3.3 Chosen and Known-Plaintext Attacks

Chosen and known-plain message attacks are attacks in which the adversary besides knowing the encryption algorithm, can access or choose a set of plain messages, modify them and observe the corresponding cipher message. These attacks are considered as the most prevalent and most threatened in the point of view of data protectors. One of the requirements of a good cryptosystem is its security against these attacks. Part **c)** of Figure 4.3.2 is obtained by XOR-ing original audio sample **a)** and its corresponding cipher audio sample **b)**. Part **e)** is a XOR-ing of the audio sample in **c)** and an unknown cipher audio sample **d)**. Since part **e)** failed in recovering any meaningful (plain) audio sample, the proposed method is safe against chosen/known-plaintext attack.



a) Original Audio Sample

b) Cipher Audio Sample

c) XORed a) and b)

d) Unkown Cipher Audio Sample

e) XORed c) and d)

Figure 4.3.2 Failed chosen/known-plaintext attack on proposed method

4.3.4 Processing Time

The processing times of the encryption and decryption algorithms were measured to ascertain their feasibility in real-time operations (see Sections B.6 – B.11 of Appendix B). Table 4.3.2 summaries the average processing time for the simulation of ‘speech_dft.wav’ (4 seconds) audio file using the proposed methods.

Table 4.3.2 Average processing time of 'speech_dft.wav' audio file

Cipher key (n, m, i)	**Encryption time (sec.)**	**Decryption time (sec.)**
(2200, 50, 1)	3.07333	2.74167
(220, 500, 1)	0.32366	0.33366
(107, 1024, 1)	0.17133	0.34333
(22, 5000, 1)	0.03666	1.17433
(11, 10000, 1)	0.02400	4.21500
(220, 500, 2)	1.97566	0.61333

Table 4.3.2 shows that the processing time for the encryption algorithm decreases as the number of piles (k-shuffles) increases for one-iteration as against the increase in processing time for the decryption algorithm. It is however observed that the processing time is directly proportional to the number of repetition. That is, the more the repetitions, the more the processing time and key parameters can be chosen to achieve better processing time.

4.3.5 Conclusion

In this section, the k-shuffle technique for encryption and decryption of real-time audio data is presented. The algorithm is implemented using MATLAB Simulink simulation tools and the data obtained is analysed in various forms; error recovery rate, key space, chosen/known-plaintext attacks and processing time.

On error recovery rate the proposed audio encryption scheme has a zero (0) error rate and has outperformed the method proposed by Khalil (2016) (error rate = 0.00017). This shows that the decryption algorithm of the proposed scheme has a 100% recovery of original audio sample.

Key space analysis also shows that the scheme has ≈ 4,294,967,296 different key combinations for any adversary using Bruce-force attacks to exhaustively search the key space. This is considered good enough to ward off any Bruce-force attack for a period of time given limited computing resources. Also the key space is good since adversaries have to deal with two layers of security (video and audio). Thus, find the key for video encryption do not guarantee find the key for audio encryption in the video.

The issue of chosen/known-plaintext attacks that adversaries can adopt in order to recover the original audio samples is dealt with by the scheme. As shown in Figure 4.3.2, applying XOR-ing operations on known plain and cipher audio samples did not lead to a recovery of the original quality audio.

The processing time is also found to be better and ideal for implementation in a real-time audio and video processing. The results organized in Table 4.3.2 show that encryption time can be made better when the number of piles increases. This however is at the expense of the decryption process where the processing time increase as the number of piles increases. Nevertheless, a good encryption and decryption time can be obtained when the value of the number of piles is closer to the value of the number of data points.

In summary, given an error free communication channel, the results obtained show that the proposed method;

- ✓ have a zero (0) error rate for recovered audio signals.
- ✓ can survive chosen/known-plaintext attack.

- ✓ have a large enough key space to withstand Bruce-force attacks for audio data whose financial interest is only for a short time span (e.g. live telecast of soccer game).
- ✓ cipher key can be varied to achieve good processing time.

4.4 Analyses of the Proposed RNS Decoder for Cipher Video using the Moduli Set $\{\mathbf{2^n-1, 2^n, 2^n+1}\}$

In Section A.5 of Appendix A, the details of the MATLAB Simulink codes for simulating the proposed RNS encoder and decoder are presented. Results analyses were carried out on processing time and encoding efficiency.

4.4.1 Procession Time

The average processing time (in seconds) for ten (10) simulations of the encryption and decryption processes (see Section B.12 – B.14 of Appendix B) are summaries in Table 4.4.1. The results indicate that the 'RNSEncoder' consumes about 9% of the average encryption time while 25% of the average decryption time is consumed by the 'RNSDecoder'. Thus, RNS constitutes about 34% of the total processing time of any given encryption and decryption operation. Also, the increase in the decryption time enhances security especially against Bruce-force. Any adversary receiving the residual video will need extra efforts to decode them to the cipher state before efficient decryption can occur.

Table 4.4.1 Average Processing time of Encryption and Decryption Algorithms

Video File(Dimension)	Encryption (RNSEncoder) (sec.)	Decryption (RNSDecoder) (sec.)
vipmen.avi (160×120)	0.058(0.008)	0.061(0.026)
carphone.avi (176×144)	0.092(0.014)	0.098(0.035)
xylophone.mpg (320×240)	0.468(0.036)	0.475(0.097)

4.4.2 Encoding Analysis

Video data is made up of several frames arranged over time. The elements of these frames are called pixels. The weight of a pixel for RGB format of video span between 0 and $2^8 - 1$. Thus, 8-bits are required to encode and transmit each pixel. However, the introduction of RNS reduces this to 4-bits when $n = 4$ is used for the 'RNSEncoder'. This safes half of the bits required to encode and transmit cipher videos. Consequently, transmission speed is enhanced by the introduction of RNS. The dark frames of parts (b) and (c) of Figure 3.4.5 in Section 3.4.5 confirm the smaller values achieved through the RNS encoder.

4.6 Chapter Summary

The various proposed cryptosystems have been simulated and the results analysed to measure their performances. Conclusions have also been drawn on the performances of the proposed cryptosystems against some existing works in literature. The next chapter will draw conclusions and make some recommendations on the thesis.

CHAPTER FIVE

CONCLUSIONS AND RECOMMENDATIONS

5.1 Conclusions

This thesis proposed techniques for securing video of financial interest for content providers. Two perceptual video encryption techniques have been proposed; perceptual video encryption via unit anti-diagonal matrix and perceptual video encryption using orthogonal matrices. While the former is based on chaos (scrambling) technique, the latter is grounded on encoding technique. The thesis also proposed an integrated audio encryption technique and an RNS based encoder/decoder to enhance the security of audio signals and transmission speed of cipher video across networks.

The proposed perceptual video encryption algorithm via unit anti-diagonal matrix integrated rotation matrix to identify random points in video frames for encryption. Simulation and analysis of results showed that cipher video produced by this approach;

- ✓ have varying perceptibility which is controlled by changing the block size and the number of iteration,
- ✓ is not limited to specific video format,
- ✓ can resist chosen/known-plaintext attacks,

On the proposed algorithm using orthogonal matrix, the pixel values of source video are directly changed from their originals through encoding. Variable blocks of video frames are encrypted to yield cipher video that resists chosen/known-plaintext attacks. Investigation on algorithm showed that the technique is faster than other existing techniques. Also, the technique is found to be efficient in data recovery, yielding error free recovered video.

In the quest to secure audio signals contained in the source video, the thesis proposed an audio encryption algorithm by adopting the K-shuffle technique. This method scrambles audio bitstream into cipher audio bitstream by re-arranging the bitsteam in m −piles of n data each. Simulation and analysis of the results revealed that the technique;

- has an error-free recovery rate
- has a large enough key-space to resist Bruce-force attack for the video under consideration
- can survive known/chosen plaintext attack

Finally, transmission efficiency across networks is guaranteed by the adoption of RNS to encode and decode cipher video. Since smaller values require lesser bit representation and are also known to transmit faster than bigger values, the introduction of an RNS encoder produces two residual video (with smaller pixel values than the originals) for transmission. Additionally, the RNS encoding/decoding process adds another layer of security to the proposed techniques; the value of n needs to be known before a successful decoding can take place.

5.2 Recommendations

Based on the conclusions made, it is recommended that;

- the proposed audio encryption technique can be adopted by other researchers into their proposed techniques to enhance the security of audio signals.
- the transmission speed of multimedia services can be enhance if the proposed RNS decoder is integrated into existing/new cryptosystems.
- commercial video content provides/producers should adopt the proposed techniques to advertise their products. This will lead to the expansion of the customer based and consequently higher returns in their investments.

REFERENCE

Alhassan, S., & Gbolagade, K. (2013). Enhancement of the Security of a Digital Image Using the Moduli Set {2^n-1; 2^n; 2^n+1}. *International Journal of Advanced Research in Computer Engineering & Technology (IJARCET), 2*(7), 2223-2229.

Amir, H., & Shahrokh, G. (2006). A Novel Video Watermarking Method Using Visual Cryptography. *2006 IEEE International Conference on Engineering of Intelligent Systems* (pp. 1-5). Islamabad, Pakistan: IEEE.

Au-Yeung, J. S.-K., Zhu, S., & Zeng, B. (2010a). Partial Video Encryption Based on Alternative Integer Transforms. *Proceedings of 2010 IEEE International Symposium on Circuits and Systems*, 933-936.

Au-Yeung, J. S.-K., Zhu, S., & Zeng, B. (2010b). Quality Assessment for a Perceptual Video Encryption System. *2010 IEEE International Conference on Wireless Communications, Networking and Information Security* (pp. 102-106). Beijing: IEEE.

Au-Yeung, S.-K., & Zeng, B. (2012a). Improved Perceptual Video Encryption using Multiple 8×8 Transforms in MPEG-4. *7th International Conference on Communications and Networking in China* (pp. 185-188). Kun Ming: IEEE.

Au-Yeung, S.-K., & Zeng, B. (2012b). A New Design of Multiple Transforms for Perceptual Video Encryption. *2012 19th IEEE International Conference on Image Processing* (pp. 2637-2640). Orlando: IEEE.

Au-Yeung, S.-K., Zhu, S., & Zeng, B. (2009). Partial Video Encryption Based on Alternating Transforms. *encryption*, 893-896.

Au-Yeung, S.-K., Zhu, S., & Zeng, B. (2010). Improved Perceptual Video Encryption using Alternative Unitary Transforms. *2010 5th International ICST Conference on Communications and Networking in China* (pp. 1-5). Beijing: IEEE.

Au-Yeung, S.-K., Zhu, S., & Zeng, B. (2011a). Perceptual Video Encryption using Multiple 8×8 Transforms in H.264 and MPEG-4. *2011 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)* (pp. 2436 - 2439). Prague: IEEE.

Au-Yeung, S.-K., Zhu, S., & Zeng, B. (2011b). Design of New Unitary Transforms for Perceptual Video Encryption. *IEEE Transactions on Circuits and Systems for Video Technology, 21*(9), 1341-1345.

Babu, A. M., & Singh, K. J. (2013). Performance Evaluation of Chaotic Encryption Technique. *American Journal of Applied Sciences, 10*(1), pp. 35-41.

Banhanfar, S., & Zarei, N. (2013). Reverse Converter for the Moduli Set {2^n-1, 2^n, 2^n+1} Base on Grouping Number. *IJCSI International Journal of Computer Science Issues, 10*(6), 97-102.

Baraniecka, A., & Jullien, G. (1978). On Decoding Techniques for Residue Number System Realizations of Digital Signal Processing Hardware. *IEEE Transactions on Circuits and Systems, 25*(11), 935-936.

Bello, H. K., & Gbolagade, K. A. (2017). An Efficient CRT Based Reverse Converter for {2^(2n+1)-1, 2^(n-1), 2^(2n)-1} Moduli Set. *Journal of Advances in Mathematics and Computer Science, 25*(6), 1-9.

Bergeron, C., & Lamy-Bergot, C. (2005). Compliant Selective encryption for H.264/AVC video streams. *2005 IEEE 7th Workshop on Multimedia Signal Processing* (pp. 1-4). Shanghai: IEEE.

Bernatin, T., Kuzhaloli, S., Godwin Premi, M. S., & Brathesia Queen, L. (2016). Perceptual Video Encryption in Multimedia Secure Communication. *2016 Online International Conference on Green Engineering and Technologies (IC-GET)* (pp. 1-4). Coimbatore: IEEE.

Bhardwaj, M., Premkumar, A. B., & Srikanthan, T. (1998). Breaking the 2n-bit Carry Propagation Barrier in Residue to Binary Conversion for the {2^n −1,2^n ,2^n +1} Module Set. *IEEE Transactions on Circuits and Systems I: Fundamental Theory and Applications, 45*(9), 998-1002.

Bhargava, B., Shi, C., & Wang, S. Y. (2004). MPEG Video Encryption Algorithms. *Kluwer Academic Publishers, 24*, pp. 57–79.

Borujeni, S. E. (2000). Speech Encryption Based on Fast Fourier Transform Permutation. *ICECS 2000. 7th IEEE International Conference on Electronics, Circuits and Systems (Cat. No.00EX445). 1*, pp. 290-293. Jounieh: IEEE.

Chandra, M. A., Purwar, R., & Rajpal, N. (2012). A Novel Approach of Digital Video Encryption. *International Journal of Computer Applications (0975 – 8887)*, 38-42.

Chervyakov, N. I., Lyakhov, P. A., Kalita, D. I., & Shulzhenko, K. S. (2015). Effect of RNS moduli set selection on digital filter performance for satellite communications. *2015 International Siberian Conference on Control and Communications (SIBCON)* (pp. 1 - 7). Omsk: IEEE.

Daabo, M. I., Gbolagade, K. A., & Agbdemnab, P. A. (2016). Fast Overflow Detection Scheme by Operands Examinations Method for Length Three Moduli Sets. *Computer Engineering and Intelligent Systems, 7*(9), 8-13.

Diaconis, P., Graham, R. L., & Kantor, W. M. (1983). The Mathematics of Perfect Shuffles. *Advances in Applied Mathematics*, 175–196.

Ding, X., Deng, Y., Yang, G., Song, Y., He, D., & Sun, X. (2017). Design of New Scan Orders for Perceptual Encryption of H.264/AVC Videos. *IET Information Security, 11*(2), 55-65.

Duato, J., Yalamanchili, S., & Ni, L. (2003). *Interconnection Networks, an Engineering Approach* (1st ed.). San Francisco: Morgan Kaufmann.

Dubois, L., Puech, W., & Blanc-Talon, J. (2014). Smart Selective Encryption of H.264/AVC Videos using Confidentiality Metrics. *Annales des Télécommunications*, 569-583.

Ellis, J. A., Fan, H., & Shallit, J. (2002). The Cycles of the Multiway Perfect Shuffle Permutation. *Discrete Mathematics & Theoretical Computer Science, 5*, 169-180.

Ernastuti. (2014). Perfect Shuffle Algorithm for Cryptography. *ARPN Journal of Engineering and Applied Sciences, 9*(12), 2383-2386.

Farkash, S., Raz, S., & D, M. (1991). Analog Speech Scrambling via the Gabor Representation. *17th Convention of Electrical and Electronics Engineers in Israel* (pp. 365-368). Tel Aviv: IEEE.

Furht, B., & Kirovski, D. (2014). *Multimedia Security Handbook* (1st ed.). CRC press.

Furht, B., Muharemagic, E., & Socek, D. (2005). *Multimedia Encryption and Watermarking.* New York: Springer Science+Business Media, Inc.

Gallaher, D., Petry, F. E., & Srinivasan, P. (1997). The digital parallel method for fast RNS to weighted number system conversion for specific moduli {2^k −1,2^k ,2^k +1}. *IEEE Transactions on Circuits and Systems II: Analog and Digital Signal Processing*, *44*(1), 53-57.

Gnanajeyaraman, R., Prasadh, k., & Ramar, D. (2009). Audio Encryption using Higher Dimensional Chaotic Map. *International Journal of Recent Trends in Engineering*, *1*(2), 103-107.

Hiasat, A. (2018). New Residue Number System Scaler for the Three-Moduli Set {2^(n+1) - 1, 2^n, 2^n - 1}. *Computers*, *7*(3), 1-7.

https://en.wikipedia.org/wiki/Audio_file_format. (n.d.). Retrieved July 26, 2018, from Wikipedia: https://en.wikipedia.org

https://en.wikipedia.org/wiki/Digital_audio. (n.d.). Retrieved July 26, 2018, from Wikipedia: https://en.wikipedia.org

Ibrahim, K. M., & Saloum, S. N. (1988). An Efficient Residue to Binary Converter Design. *IEEE Transactions on Circuits and Systems, 35*(9), 1156-1158.

Jarvinen, T. S., Takala, J. H., Akopian, D. A., & Saarinen, J. P. (2001). Register-Based Multi-Port Perfect Shuffle Networks. *ISCAS 2001. The 2001 IEEE International Symposium on Circuits and Systems (Cat. No.01CH37196). 4*, pp. 306-309. Sydney: IEEE.

Jenkins, W. (1978). Techniques for Residue-to-Analog Conversion for Residue-Encoded. *IEEE Transactions on Circuits and Systems, 25*(7), 555-562.

Kaushik, S., & Srivastava, A. (2013). Design of RNS Converters for moduli sets with Dynamic Ranges up to 6n-bit. *IOSR Journal of VLSI and Signal Processing, 2*(6), 14-19.

Khalil, M. I. (2016). Real-Time Encryption/Decryption of Audio Signal. *International Journal of Computer Network and Information Security(IJCNIS), 8*(2), 25-31.

Khalil, M. I. (2017). Quaternion-based Encryption/Decryption of Audio Signal Using Digital Image as a Variable Key. *International Journal of Communication Networks and Information Security, 9*(2), 216-221.

Kirthanaa, A., Mathan, N., & Vino, T. (2015). Improved perceptual Video Encryption and Decryption using S-Transform. *2015 International Conference on Control, Instrumentation, Communication and*

Computational Technologies (ICCICCT) (pp. 145-148). Kumaracoil: IEEE.

Kosek, M. R., Taylor, F. J., Griffin, M., & Hardy, R. R. (1989). RNS-based GaAs signal processing system. *IEEE Military Communications Conference, 'Bridging the Gap. Interoperability, Survivability, Security'. 2,* pp. 615-619. Boston: IEEE.

Kulkarni, A., Kulkarni, S., Haridas, K., & More, A. (2013). Proposed Video Encryption Algorithm v/s Other Existing Algorithms: A Comparative Study. *International Journal of Computer Applications*.

Kwon, G. R., Wang, C., Lian, S., & Hwang, S. S. (2012). Advanced Partial Encryption using Watermarking and Scrambling in MP3. *Multimed Tools Appl, 59*(3), 885-895.

Li, S., Chen, G., Cheung, A., Bhargava, B., & 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.

Lima, J. B., & da Silva Neto, E. F. (2016). Audio Encryption Based on the Cosine Number Transform. *Multimed Tools Appl, 71*(14), 8403–8418.

Low, J. Y., & Chang, C. H. (2011). A new RNS scaler for {2^n− 1, 2^n, 2^n+ 1}. *2011 IEEE International Symposium of Circuits and Systems (ISCAS)* (pp. 1431 - 1434). Rio de Janeiro: IEEE.

Lu, M. (2004). *Arithmetic and Logic in Computer Systems.* New Jersey: John Wiley & Sons, Inc.

Madain, A., Abu Dalhoum, A. L., Hiaryil, H., Ortega, A., & Alfonseca, M. (2014). Audio Scrambling Technique Based on Cellular Automata. *Multimed Tools Appl, 71*(3), 1803-1822.

Magli, E., Grangetto, M., & Olmo, G. (2011). Transparent Encryption Techniques for H.264/AVC and H.264/SVC Compressed Video. *Signal Processing, 91*(5), 1103-1114.

Makwana, V., & Parmar, N. (2014). Encrypt an Audio File using Combine Approach of Transformation and Cryptography. *(IJCSIT) International Journal of Computer Science and Information Technologies, 5*(3), 4473-4476.

Manpreet, K., & Ms. Sukhpreet, K. (2014). Survey of Various Encryption Techniques for Audio Data. *International Journal of Advanced Research in Computer Science and Software Engineering, 4*(5), pp. 1314-1317.

Mathew, J., Radhakrishnan, D., & Srikanthan, T. (1999). Residue-to-Binary Arithmetic Converter For Moduli Set {2^n -1, 2^n, 2^n +1, 2^(n+1) - 1}. *Workshop on Nonlinear Signal and Image Processing (NSIP'99)* (pp. 20-23). IEEE-EURASIP.

Medvedoff, S., & Morrison, K. (1987). Groups of Perfect Shuffles. *Mathematics Magazine, 60*(1), 3-14.

Mehrin, R., Mehdi, H., Saeid, B., & Mohammad, T. (2011). Fast Overflow Detection in Moduli Set {2^n – 1, 2^n, 2^n + 1}. *International Journal of Computer Science Issues, 8*(3), 407-414.

Menezes, A., van Oorschot, P., & Vanstone, S. (1996). *Handbook of Applied Cryptography*. London: CRC Press.

Meyer, J., & Gadegast, F. (1995). *Security Mechanisms for Multimedia Data with the Example MPEG-1 video.* Retrieved October 6, 2015, from http://www.gadegast.de/: www.gadegast.de/frank/doc/secmeng.pdf

Meyer-Base, U., Mellott, J., & Taylor, F. (1997). Design of RNS frequency sampling filter banks. *1997 IEEE International Conference on Acoustics, Speech, and Signal Processing. 3*, pp. 2061-2064. Munich: IEEE.

Omondi, A., & Premkumar, B. (2007). *Residue Number Systems: Theory and Implementation.* London: Imperial College Press.

Packard, R. W., & Packard, E. S. (1994). The Order of a Perfect k-Shuffle. *The Official Journal of the Fibonacci Association*, pp. 136-44.

Pande, A., & Zambreno, J. (2013). Advances in Multimedia Encryption. In A. Pande, & J. Zambreno, *Embedded Multimedia Security Systems* (pp. 11-22). London: Springer.

Pande, A., Zambreno, J., & Mohapatra, P. (2010). Joint Video Compression and Encryption using Arithmetic Coding and Chaos. *IEEE 4th International Conference on Internet Multimedia Services Architecture and Application*. IEEE.

Persi, D., Graham, R. L., & Kantor, W. M. (1983). The Mathematics of Perfect Shuffles. *Advance in Applied Mathematics*, *4*, pp. 175- 196.

Persson, A., & Bengtsson, L. (2009). Forward and Reverse Converters and Moduli Set Selection in Signed-Digit Residue Number Systems. *Journal of Signal Processing Systems, 56*(1), 1-15.

Piestrak, S. J. (1994). Design of High-Speed Residue-to-Binary Number System Converter Based on Chinese Remainder Theorem. *Proceedings 1994 IEEE International Conference on Computer Design: VLSI in Computers and Processors* (pp. 508-511). Cambridge: IEEE.

Piestrak, S. J. (1995). A High-Speed Realization of a Residue to Binary Number System Converter. *IEEE Transactions on Circuits and Systems II: Analog and Digital Signal Processing, 42*(10), 661-663.

Pitale, P., Pateria, A., Singh, P., & Golchha, N. (2015). Audio based Secure Encryption and Decryption. *International Journal of Computer Applications (0975 – 8887),* 1-4.

Raghunandhan, K. R., Dodmane, R., Sudeepa, K. B., & Aithal, G. (2013). Efficient Audio Encryption Algorithm For Online Applications Using Transposition And Multiplicative Non-Binary System. *International Journal of Engineering Research & Technology(IJERT),* 472-477.

Ramirez, J., Garcia, A., Meyer Base, U., Taylor, F., Fernandez, P. G., & Lloris, A. (2001). Design of RNS-based distributed arithmetic DWT filterbanks. *2001 IEEE International Conference on Acoustics, Speech, and Signal Processing. Proceedings (Cat. No.01CH37221). 2,* pp. 1193-1196. Salt Lake City: IEEE.

Ramnath, S., & Scully, D. (1996). Moving Card i to Position j with Perfect Shuffles. *Mathematics Magazine, 69*(5), 361-365.

Raphael, A. J., & Sundaram, V. (2010). Cryptography and Steganography – A Survey. *International Journal of Computer and Technology Applications, 2*(3), pp. 626-630.

Sadkhan, S. B., & Mohammed, R. S. (2015). Proposed Random Unified Chaotic Map as PRBG for Voice Encryption in Wireless Communication. *International Conference on Communication, Management and Information Technology (ICCMIT 2015)* (pp. 314 – 323). Prague: Procedia Computer Science.

Schneier, B. (1996). *Applied Cryptography, Second Edition: Protocols, Algorithms, and Source Code in C.* Wiley Computer Publishing, John Wiley & Sons, Inc.

Shahid, Z., Chaumont, M., & Puech, W. (2010). Selective encryption of C2DVLC of AVS video coding standard for I & P frames. *2010 IEEE International Conference on Multimedia and Expo* (pp. 1655-1660). Suntec City, Singapore: IEEE.

Sharma, S., Kumar, L., & Sharma, H. (2013). Encryption of an Audio File on Lower Frequency Band for Secure Communication. *International Journal of Advanced Research in Computer Science and Software Engineering, 3*(7), 79-84.

Shetty, A., Kiran, R., Shetty, S., Naik, S., Nayak, S., & D'Souza, D. J. (2017). Image Cryptography using RNS Algorithm. *2017 IEEE International*

Conference on Power, Control, Signals and Instrumentation Engineering (ICPCSI) (pp. 1936-1939). Chennai: IEEE.

Singh, T. (2014). Residue number system for fault detection in communication networks. *2014 International Conference on Medical Imaging, m-Health and Emerging Communication Systems (MedCom)* (pp. 157-161). IEEE: Greater Noida.

Socek, D., Kalva, H., Magliveras, S. S., Marques, O., Culibrk, D., & Furht, B. (2007). New approaches to Encryption and Eteganography for Digital Videos. *Multimedia Systems, 13*(3), 191-204.

Spinsante, S., Chiaraluce, F., & Gambi, E. (2005). Masking Video Information by Partial Encryption of H.264/AVC Coding Parameters. (pp. 1-4). Antalya: 2005 13th European Signal Processing Conference.

Sridharan, S., Dawson, E., & Goldburg, B. (1991). Fast Fourier Transform Based Speech Encryption System. *IEE Proceedings I - Communications, Speech and Vision. 138*, pp. 215-223. IET.

Stallings, W. (2006). *Cryptography and Network Security Principles and Practice* (5th ed.). New York: Prentice Hall.

Steve, B., Persi, D., & Ron, G. (2014, December 16). *shuffling.pdf.* Retrieved September 30, 2018, from Stanford University: https://statweb.stanford.edu/~cgates/PERSI/papers/shuffling.pdf

Stone, H. S. (1971). Parallel Processing with the Perfect Shuffle. *IEEE Transactions on Computers, C-20*(2), 153-161.

Sultana, S. F., & Shubhangi, D. C. (2017). Video Encryption Algorithm and Key Management using Perfect Shuffle. *International Journal of Engineering Research and Application, 7*(7), 1-5.

Tamimi, A. A., & Abdalla, A. M. (2014). An Audio Shuffle-Encryption Algorithm. *Proceedings of the World Congress on Engineering and Computer Science 2014. 1*, pp. 409-412. San Francisco: Newswood Limited.

Taylor, F. J. (1990). An RNS Discrete Fourier Transform Implementation. *IEEE Transactions on Acoustics, Speech, and Signal Processing, 38*(8), 1386-1394.

Uhl, A., & Pommer, A. (2005). *Image and Video Encryption From Digital Rights Management to Secured Personal Communication.* Boston: Springer Science + Business Media, Inc.

Wang, L.-F., Wang, W.-D., MA, J., Xiao, C., & Wang, K.-Q. (2008). Perceptual video encryption scheme for mobile application based on H.264. *The Journal of China Universities of Posts and Telecommunications*, 73-78.

Wang, W., Swamy, M. N., & Ahmad, M. O. (2014). RNS Application for Digital Image Processing. *4th IEEE International Workshop on System-on-Chip for Real-Time Applications* (pp. 77-80). Banff: IEEE.

Wang, Y. (2000). Residue-to-Binary Converters Based on New Chinese Remainder Theorems. *IEEE Transactions on Circuits and Systems II: Analog and Digital Signal Processing, 47*(3), 197-205.

Wang, Y., O'Neill, M., & Kurugollu, F. (2013). Partial Encryption by Randomized Zig-Zag Scanning for Video Encoding. *2013 IEEE International Symposium on Circuits and Systems (ISCAS2013)* (pp. 229-232). Beijing: IEEE.

Wang, Y., Song, X., Aboulhamid, M., & Shen, H. (2002). Adder Based Residue to Binary Number Converters for {2^n-1, 2^n, 2^n + 1}. *IEEE Transactions on Signal Processing, 50*(7), 1772-1779.

Watanabe, O., Fukuhara, T., & Kiya, H. (2015). A Perceptual Encryption Scheme for Motion JPEG 2000 Standard. *2015 15th International Symposium on Communications and Information Technologies (ISCIT)* (pp. 125-128). Nara: IEEE.

Wong, A., & Bishop, W. (2006). An Efficient, Parallel Multi-Key Encryption of Compressed. *IASTED International Conference on Signal and Image Processing*.

Yahya, A. A., & Abdalla, A. M. (2008). A Shuffle Image-Encryption Algorithm. *Journal of Computer Science, 4*(12), 999-1002.

Yang, W., Zhao, M., Chen, X., Huang, L., & Wang, J. (2011). Application of residue number systems to Bent-pipe satellite communication systems. *2011 6th International ICST Conference on Communications and Networking in China (CHINACOM)* (pp. 1083-1087). Harbin: IEEE.

Yao, Y., Zhou, J., Yan, B., & Li, Y. (2018). RNS-Based Embedding Scheme for Data Hiding in Digital Images. *2018 17th IEEE International Conference On Trust, Security And Privacy In Computing And*

Communications/ 12th IEEE International Conference On Big Data Science And Engineering (TrustCom/BigDataSE) (pp. 1480-1483). New York: IEEE.

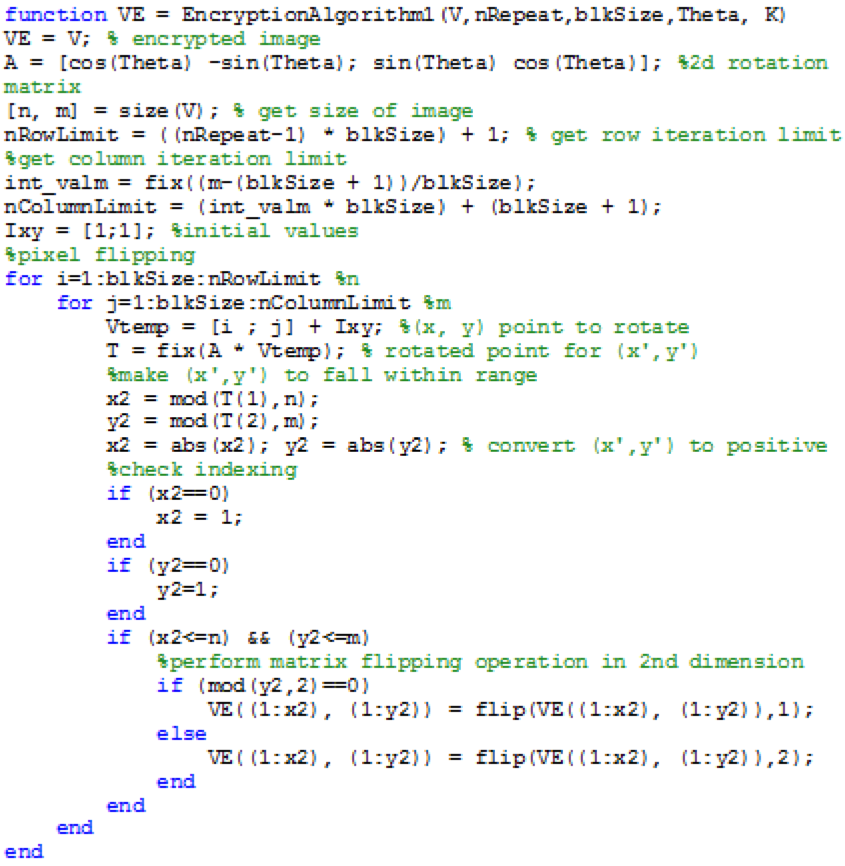
Zarei, B., & Askarzadeh, M. (2010). A High-speed Residue Number Comparator for the 3-Moduli Set {2^n-1, 2^n, 2^n+1}. *International Journal of Advanced Research in Computer Science, 3*(1), 270-272.

Zeng, B., Au-Yeung, S.-K., Zhu, S., & Gabbouj, M. (2014). Perceptual Encryption of H.264 Videos: Embedding Sign-Flips into the Integer-Based Transforms. *IEEE Transactions on Information Forensics and Security*, *9*(2), 309-320.

APPENDIX A

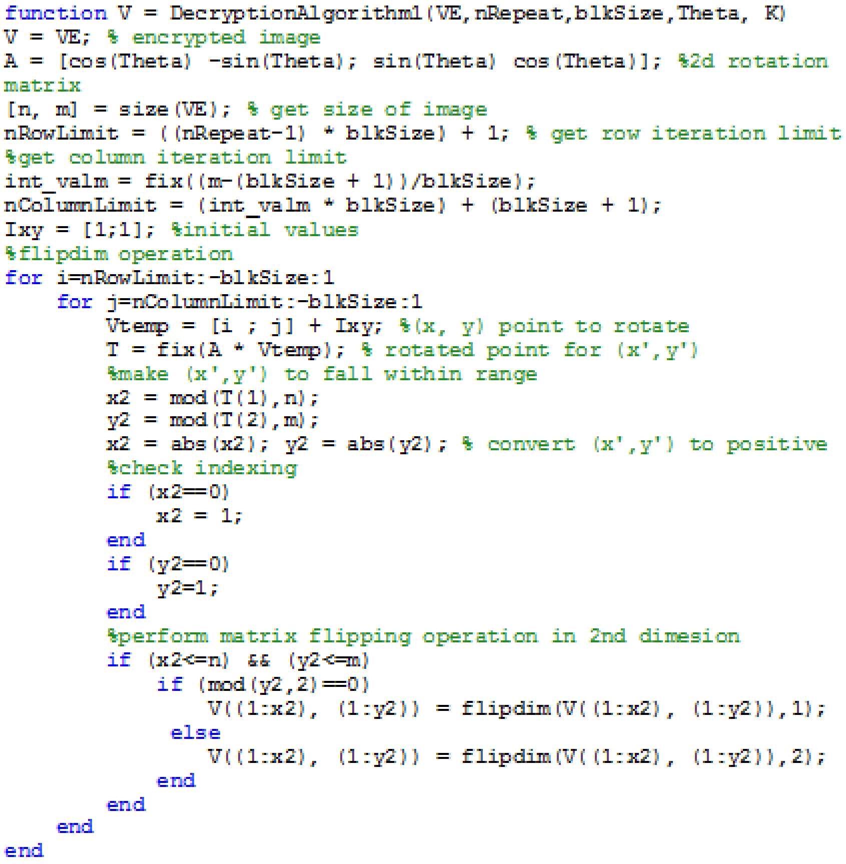
MATLAB SOURCE CODES

A.1 Source code to simulate the encryption algorithm of the Proposed Perceptual Video Encryption Algorithm via Unit Anti-diagonal Matrix.



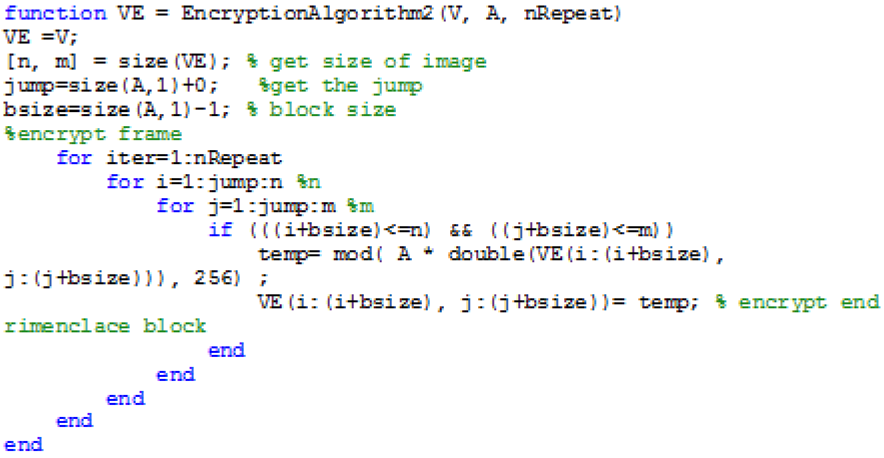
```
function VE = EncryptionAlgorithm1(V,nRepeat,blkSize,Theta, K)
VE = V; % encrypted image
A = [cos(Theta) -sin(Theta); sin(Theta) cos(Theta)]; %2d rotation
matrix
[n, m] = size(V); % get size of image
nRowLimit = ((nRepeat-1) * blkSize) + 1; % get row iteration limit
%get column iteration limit
int_valm = fix((m-(blkSize + 1))/blkSize);
nColumnLimit = (int_valm * blkSize) + (blkSize + 1);
Ixy = [1;1]; %initial values
%pixel flipping
for i=1:blkSize:nRowLimit %n
    for j=1:blkSize:nColumnLimit %m
        Vtemp = [i ; j] + Ixy; %(x, y) point to rotate
        T = fix(A * Vtemp); % rotated point for (x',y')
        %make (x',y') to fall within range
        x2 = mod(T(1),n);
        y2 = mod(T(2),m);
        x2 = abs(x2); y2 = abs(y2); % convert (x',y') to positive
        %check indexing
        if (x2==0)
            x2 = 1;
        end
        if (y2==0)
            y2=1;
        end
        if (x2<=n) && (y2<=m)
            %perform matrix flipping operation in 2nd dimension
            if (mod(y2,2)==0)
                VE((1:x2), (1:y2)) = flip(VE((1:x2), (1:y2)),1);
            else
                VE((1:x2), (1:y2)) = flip(VE((1:x2), (1:y2)),2);
            end
        end
    end
end
```

A.2 Source code to simulate the decryption algorithm of the Proposed Perceptual Video Encryption Algorithm via Unit Anti-diagonal Matrix.



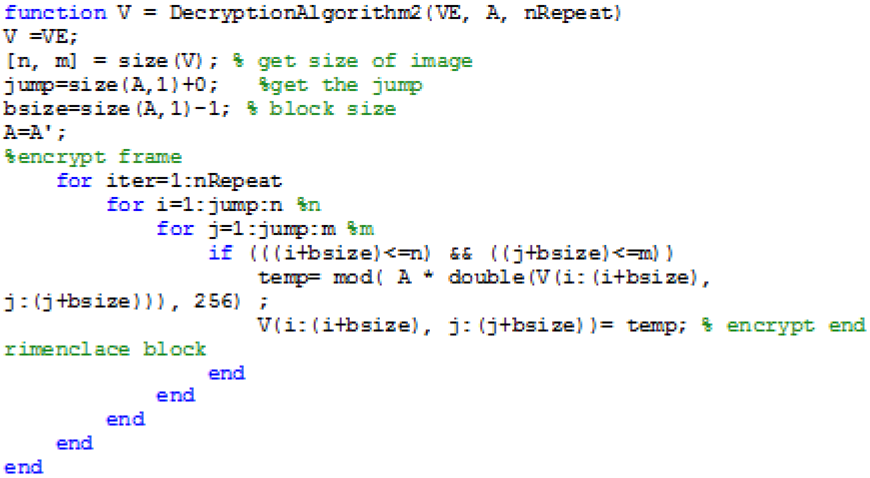
```
function V = DecryptionAlgorithm1(VE,nRepeat,blkSize,Theta, K)
V = VE; % encrypted image
A = [cos(Theta) -sin(Theta); sin(Theta) cos(Theta)]; %2d rotation
matrix
[n, m] = size(VE); % get size of image
nRowLimit = ((nRepeat-1) * blkSize) + 1; % get row iteration limit
%get column iteration limit
int_valm = fix((m-(blkSize + 1))/blkSize);
nColumnLimit = (int_valm * blkSize) + (blkSize + 1);
Ixy = [1;1]; %initial values
%flipdim operation
for i=nRowLimit:-blkSize:1
    for j=nColumnLimit:-blkSize:1
        Vtemp = [i ; j] + Ixy; %(x, y) point to rotate
        T = fix(A * Vtemp); % rotated point for (x',y')
        %make (x',y') to fall within range
        x2 = mod(T(1),n);
        y2 = mod(T(2),m);
        x2 = abs(x2); y2 = abs(y2); % convert (x',y') to positive
        %check indexing
        if (x2==0)
            x2 = 1;
        end
        if (y2==0)
            y2=1;
        end
        %perform matrix flipping operation in 2nd dimesion
        if (x2<=n) && (y2<=m)
            if (mod(y2,2)==0)
                V((1:x2), (1:y2)) = flipdim(V((1:x2), (1:y2)),1);
             else
                V((1:x2), (1:y2)) = flipdim(V((1:x2), (1:y2)),2);
            end
        end
    end
end
```

A.3 Source code to simulate the encryption algorithm of the Proposed Perceptual Video Encryption Algorithm using Orthogonal Matrix.



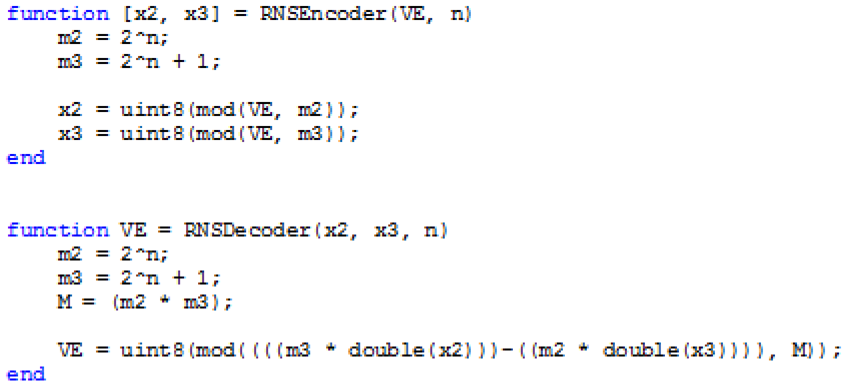
```
function VE = EncryptionAlgorithm2(V, A, nRepeat)
VE =V;
[n, m] = size(VE); % get size of image
jump=size(A,1)+0;    %get the jump
bsize=size(A,1)-1; % block size
%encrypt frame
    for iter=1:nRepeat
        for i=1:jump:n %n
            for j=1:jump:m %m
                if (((i+bsize)<=n) && ((j+bsize)<=m))
                    temp= mod( A * double(VE(i:(i+bsize),
j:(j+bsize))), 256) ;
                    VE(i:(i+bsize), j:(j+bsize))= temp; % encrypt end
rimenclace block
                end
            end
        end
    end
end
```

A.4 Source code to simulate the decryption algorithm of the Proposed Perceptual Video Encryption Algorithm using Orthogonal Matrix.



```
function V = DecryptionAlgorithm2(VE, A, nRepeat)
V =VE;
[n, m] = size(V); % get size of image
jump=size(A,1)+0;    %get the jump
bsize=size(A,1)-1; % block size
A=A';
%encrypt frame
    for iter=1:nRepeat
        for i=1:jump:n %n
            for j=1:jump:m %m
                if (((i+bsize)<=n) && ((j+bsize)<=m))
                    temp= mod( A * double(V(i:(i+bsize),
j:(j+bsize))), 256) ;
                    V(i:(i+bsize), j:(j+bsize))= temp; % encrypt end
rimenclace block
                end
            end
        end
    end
end
```

A.5 Source code to simulate RNS encoding and decoding of cipher and plain video.



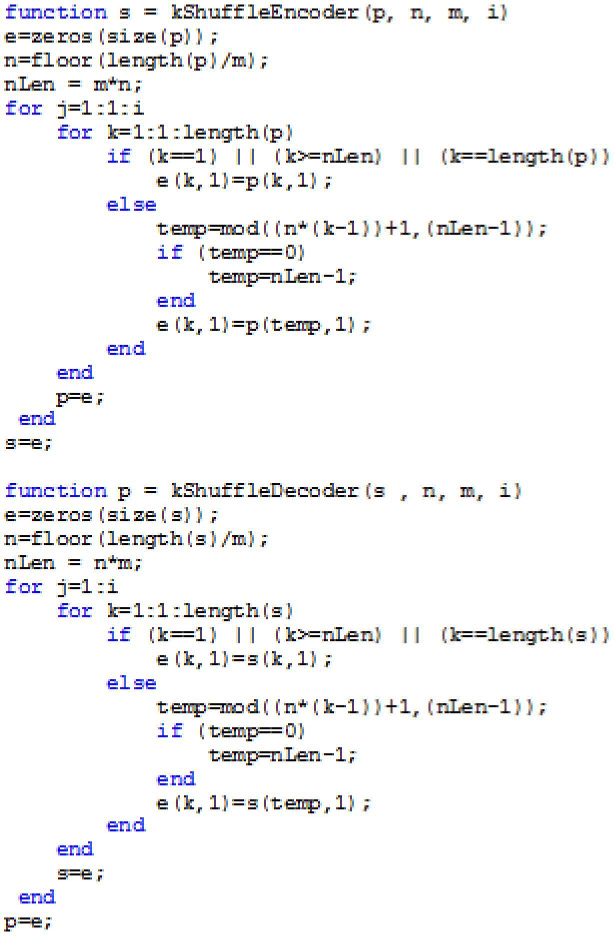
```
function [x2, x3] = RNSEncoder(VE, n)
    m2 = 2^n;
    m3 = 2^n + 1;

    x2 = uint8(mod(VE, m2));
    x3 = uint8(mod(VE, m3));
end


function VE = RNSDecoder(x2, x3, n)
    m2 = 2^n;
    m3 = 2^n + 1;
    M = (m2 * m3);

    VE = uint8(mod((((m3 * double(x2)))-((m2 * double(x3)))), M));
end
```

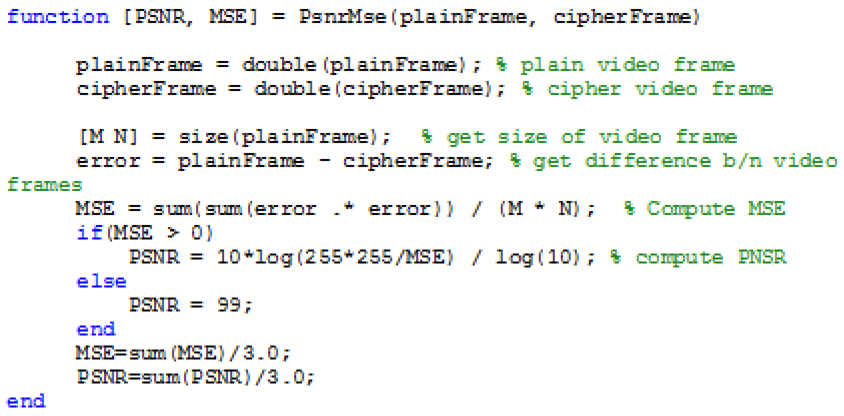
A.6 Source code to simulate the proposed Audio encryption and decryption algorithms of real-time audio signals.



```
function s = kShuffleEncoder(p, n, m, i)
e=zeros(size(p));
n=floor(length(p)/m);
nLen = m*n;
for j=1:1:i
    for k=1:1:length(p)
        if (k==1) || (k>=nLen) || (k==length(p))
            e(k,1)=p(k,1);
        else
            temp=mod((n*(k-1))+1,(nLen-1));
            if (temp==0)
                temp=nLen-1;
            end
            e(k,1)=p(temp,1);
        end
    end
    p=e;
 end
s=e;

function p = kShuffleDecoder(s , n, m, i)
e=zeros(size(s));
n=floor(length(s)/m);
nLen = n*m;
for j=1:i
    for k=1:1:length(s)
        if (k==1) || (k>=nLen) || (k==length(s))
            e(k,1)=s(k,1);
        else
            temp=mod((n*(k-1))+1,(nLen-1));
            if (temp==0)
                temp=nLen-1;
            end
            e(k,1)=s(temp,1);
        end
    end
    s=e;
 end
p=e;
```

A.7 Source code to compute MSE and PSNR



```
function [PSNR, MSE] = PsnrMse(plainFrame, cipherFrame)

      plainFrame = double(plainFrame); % plain video frame
      cipherFrame = double(cipherFrame); % cipher video frame

      [M N] = size(plainFrame);  % get size of video frame
      error = plainFrame - cipherFrame; % get difference b/n video
frames
      MSE = sum(sum(error .* error)) / (M * N);  % Compute MSE
      if(MSE > 0)
          PSNR = 10*log(255*255/MSE) / log(10); % compute PNSR
      else
          PSNR = 99;
      end
      MSE=sum(MSE)/3.0;
      PSNR=sum(PSNR)/3.0;
end
```

APPENDIX B

PROCESSING TIME

B.1 Processing time used by proposed encryption algorithm to encrypt a 12 seconds ‘carphone.mp4’ video. Proposed Perceptual Video Encryption Algorithm via Unit Anti-Diagonal Matrix.

Count	Processing Time (sec)
1	0.147882484
2	0.148132017
3	0.146800937
4	0.147775838
5	0.147122111
6	0.146804485
7	0.146435371
8	0.146434424
9	0.147058495
10	0.145420991

B.2 Processing time used by proposed encryption algorithm to encrypt a 4 seconds “Xylophone.mpg” video. Proposed Perceptual Video Encryption Algorithm via Unit Anti-Diagonal Matrix.

Count	Processing Time (sec)
1	0.090266858
2	0.089747127
3	0.090697972
4	0.090530120
5	0.088425544
6	0.091029314
7	0.090645205
8	0.089891823
9	0.090782681
10	0.090074519

B.3 Processing time used by proposed encryption algorithm to encrypt a 10 seconds “news.avi” video. Proposed Perceptual Video Encryption Algorithm via Unit Anti-Diagonal Matrix.

Count	**Processing Time (sec)**
1	0.249876782
2	0.250849358
3	0.251988517
4	0.250744081
5	0.249453768
6	0.250612018
7	0.250634332
8	0.250758214
9	0.250681545
10	0.249034255

B.4 Processing time (seconds) used by proposed encryption algorithm to encrypt a 12 seconds ‘carphone.mp4’ video using different cipher key sizes. Proposed Perceptual Video Encryption Algorithm using Orthogonal Matrix.

Count	2x2	3x3	4x4	5x5	6x6	7x7	8x8	16x16	32x32
1	0.322	0.200	0.133	0.094	0.054	0.089	0.070	0.040	0.023
2	0.352	0.219	0.134	0.104	0.024	0.119	0.095	0.041	0.025
3	0.352	0.193	0.143	0.113	0.027	0.114	0.101	0.041	0.023
4	0.346	0.197	0.148	0.092	0.026	0.115	0.103	0.040	0.026
5	0.347	0.204	0.135	0.100	0.024	0.114	0.098	0.040	0.026
6	0.385	0.237	0.130	0.095	0.025	0.087	0.093	0.041	0.024
7	0.347	0.203	0.144	0.109	0.028	0.118	0.097	0.039	0.023
8	0.378	0.190	0.122	0.093	0.025	0.114	0.096	0.040	0.025
9	0.379	0.218	0.136	0.094	0.025	0.123	0.098	0.039	0.023
10	0.355	0.191	0.145	0.122	0.026	0.113	0.087	0.039	0.024

B.5 Processing time used by proposed decryption algorithm to decrypt a 12 seconds cipher ‘carphone.mp4’ video using different cipher key sizes. Proposed Perceptual Video Encryption Algorithm using Orthogonal Matrix.

Count	2x2	3x3	4x4	5x5	6x6	7x7	8x8	16x16	32x32
1	0.394	0.193	0.131	0.108	0.025	0.116	0.098	0.045	0.035
2	0.387	0.210	0.156	0.096	0.026	0.12	0.105	0.042	0.025
3	0.372	0.204	0.135	0.093	0.025	0.117	0.089	0.042	0.025
4	0.344	0.226	0.151	0.099	0.025	0.104	0.073	0.043	0.024
5	0.383	0.192	0.193	0.096	0.026	0.117	0.089	0.040	0.024
6	0.362	0.210	0.131	0.096	0.025	0.115	0.100	0.040	0.023
7	0.339	0.225	0.137	0.117	0.026	0.121	0.069	0.039	0.023
8	0.377	0.212	0.150	0.097	0.026	0.116	0.085	0.041	0.023
9	0.344	0.253	0.131	0.122	0.025	0.084	0.079	0.040	0.022
10	0.368	0.226	0.136	0.096	0.025	0.119	0.081	0.039	0.022

B.6 Processing time for the simulation of a 4 seconds ‘speech_dft.wav’real-time audio file using the cipher key (n, m, i) as (2200, 50, 1). Proposed Audio Encryption Algorithm using K-Shuffle Technique

Count	**Encryption time (sec.)**	**Decryption time (sec.)**
1	2.991782607	2.701676
2	3.114035681	2.863199
3	3.116616329	2.653202
4	2.998203814	2.811165
5	3.060269110	2.779412
6	3.081874217	2.714622
7	3.028650186	2.712913
8	3.104154294	2.811416
9	3.039563766	2.702255
10	3.086250477	2.703475

B.7 Processing time for the simulation of a 4 seconds ‘speech_dft.wav’real-time audio file using the cipher key (n, m, i) as (220, 500, 1). Proposed Audio Encryption Algorithm using K-Shuffle Technique

Count	**Encryption time (sec.)**	**Decryption time (sec.)**
1	0.300672248	0.37861
2	0.350082316	0.321191
3	0.264789389	0.340204
4	0.318373502	0.282594
5	0.319525707	0.359173
6	0.324725784	0.345871
7	0.251004364	0.334478
8	0.247179367	0.407748
9	0.312034405	0.399778
10	0.357255668	0.327567

B.8 Processing time for the simulation of a 4 seconds ‘speech_dft.wav’real-time audio file using the cipher key (n, m, i) as (107, 1024, 1). Proposed Audio Encryption Algorithm using K-Shuffle Technique

Count	**Encryption time (sec.)**	**Decryption time (sec.)**
1	0.217417509	0.366421
2	0.086068453	0.411095
3	0.257476350	0.365261
4	0.214790410	0.334598
5	0.095802954	0.352333
6	0.109913014	0.364416
7	0.153648393	0.426680
8	0.170322141	0.364264
9	0.219208965	0.408570
10	0.110625125	0.391627

B.9 Processing time for the simulation of a 4 seconds 'speech_dft.wav'real-time audio file using the cipher key (n, m, i) as (22, 5000, 1). Proposed Audio Encryption Algorithm Using K-Shuffle Technique

Count	Encryption time (sec.)	Decryption time (sec.)
1	0.032927513	1.174573
2	0.032584985	1.181125
3	0.034481225	1.181489
4	0.037823755	1.161516
5	0.034850132	1.170565
6	0.032674031	1.174449
7	0.038252853	1.175735
8	0.036051086	1.179785
9	0.039470783	1.180672
10	0.031394559	1.176867

B.10 Processing time for the simulation of a 4 seconds ‘speech_dft.wav’real-time audio file using the cipher key (n, m, i) as (11, 10000, 1). Proposed Audio Encryption Algorithm Using K-Shuffle Technique

Count	Encryption time (sec.)	Decryption time (sec.)
1	0.033493346	4.024848
2	0.023717265	4.777814
3	0.028177893	3.669613
4	0.026987248	3.588494
5	0.020505872	5.093799
6	0.024989628	4.902156
7	0.017466453	4.328754
8	0.027097091	3.969181
9	0.020986304	3.876807
10	0.026970715	4.659365

B.11 Processing time for the simulation of a 4 seconds ‘speech_dft.wav’real-time audio file using the cipher key (n, m, i) as (220, 500, 1). Proposed Audio Encryption Algorithm Using K-Shuffle Technique.

Count	Encryption time (sec.)	Decryption time (sec.)
1	1.966403338	0.563969
2	1.889278521	0.604791
3	2.060721260	0.591060
4	2.007818352	0.611690
5	1.962227615	0.667406
6	1.963578014	0.500037
7	2.045058232	0.587514
8	1.939991447	0.573837
9	1.958520744	0.613301
10	1.847432089	0.628890

B.12 Processing time for the simulation of a 160 × 120 'vipmen.avi' real-time video file. Proposed RNS Decoder for Cipher Video using the Moduli Set $\{\mathbf{2^n - 1, 2^n, 2^n + 1}\}$.

Count	Encryption (sec)	RNSEncoder (sec)	Decryption (sec)	RNSDecoder (sec)
1	0.057859	0.007872	0.061001	0.025967
2	0.058089	0.007979	0.060922	0.026010
3	0.058270	0.008065	0.060989	0.025912
4	0.058032	0.007951	0.060925	0.025790
5	0.058211	0.008078	0.060788	0.026105
6	0.057982	0.008132	0.060969	0.026030
7	0.058036	0.008044	0.060855	0.025849
8	0.057851	0.008082	0.061039	0.025968
9	0.057945	0.008022	0.061103	0.025983
10	0.057897	0.007841	0.061051	0.025943

B.13 Processing time for the simulation of a 176 × 144 'carphone.avi' real-time video file. Proposed RNS Decoder for Cipher Video using the Moduli Set $\{\mathbf{2^n-1,2^n,2^n+1}\}$.

Count	Encryption (sec)	RNSEncoder (sec)	Decryption (sec)	RNSDecoder (sec)
1	0.092064	0.013953	0.097920	0.035022
2	0.091880	0.014125	0.097996	0.035072
3	0.091969	0.014142	0.098005	0.034828
4	0.092025	0.014059	0.097877	0.034899
5	0.092043	0.014232	0.097943	0.034967
6	0.091953	0.013973	0.097945	0.034963
7	0.091967	0.013928	0.098059	0.034995
8	0.092066	0.014038	0.097847	0.034941
9	0.092064	0.013964	0.098100	0.035018
10	0.092050	0.013902	0.098084	0.035029

B.14 Processing time for the simulation of a 320 × 240 ‘xylophone.mpg’ real-time video file. Proposed RNS Decoder for Cipher Video using the Moduli Set $\{\mathbf{2^n-1,2^n,2^n+1}\}$.

Count	Encryption (sec)	RNSEncoder (sec)	Decryption (sec)	RNSDecoder (sec)
1	0.467940	0.036054	0.475194	0.097063
2	0.468119	0.036021	0.474921	0.097132
3	0.468051	0.036139	0.475072	0.096988
4	0.467820	0.035995	0.474989	0.096917
5	0.468003	0.035926	0.474951	0.096943
6	0.468158	0.035979	0.475050	0.096802
7	0.468119	0.035984	0.474887	0.096864
8	0.467964	0.035927	0.474961	0.096902
9	0.468006	0.035820	0.475017	0.097011
10	0.468038	0.035892	0.474996	0.097035

APPENDIX C

VISUAL DEGRADATION

C.1 MSE and PSNR values of selected plain and cipher video frames of "carphone.mp4" video with cipher key (1, 32, 60, [10, 1]). Proposed Perceptual Video Encryption Algorithm via Unit Anti-Diagonal Matrix

Video Frames	MSE	PSNR (dB)
1	1325.964283	17.26268
3	1322.960546	15.54992
4	1325.715779	17.01677
6	1325.931867	17.93214
8	1325.736093	15.36716
11	1325.29921	15.99893
66	1324.119737	19.14792
72	1325.143452	16.90455
82	1324.933104	19.83928
90	1323.870579	17.03773

C.2 MSE and PSNR values of selected plain and cipher video frames of "xylophone.mpg" with cipher key (1, 62, 113, [5, 5]). Proposed Perceptual Video Encryption Algorithm via Unit Anti-Diagonal Matrix

Video Frames	**MSE**	**PSNR (dB)**
3	647.2230937	19.99174
6	646.2120008	20.61262
9	645.1998191	19.62453
12	645.9982691	19.9192
47	647.2089151	19.49228
66	647.5414229	20.15602
80	646.3558046	20.58952
85	645.8140056	21.36871
90	647.2349899	18.62621
120	645.2232225	20.72978

C.3 MSE and PSNR values of selected plain and cipher video frames of “News.avi” with cipher key (1,6, -15,[1,1]). Proposed Perceptual Video Encryption Algorithm via Unit Anti-Diagonal Matrix

Video Frames	**MSE**	**PSNR (dB)**
4	1457.136779	16.81
7	1457.220518	16.1857
23	1457.070831	16.56158
55	1458.298088	18.57598
67	1456.885845	15.96046
75	1456.141155	17.47789
82	1456.500662	15.8373
91	1457.251462	15.85737
103	1457.126692	16.99987
130	1456.077771	16.45362

C.4 MSE and PSNR values of frame 6 of plain and cipher “carphone.mp4” video with cipher key size 8×8. Proposed Perceptual Video Encryption Algorithm using Orthogonal Matrix.

Video Frames	MSE	PSNR (dB)
1	114.0508424	30.90134
3	113.49993	31.42981
4	113.6372717	29.33723
6	112.8944093	30.51935
8	112.9281307	30.32323
11	113.6997945	27.90199
66	112.3996578	27.79928
72	114.9251264	30.40666
82	111.8525017	31.6287
90	114.4205549	30.70992

C.5 MSE and PSNR values of frame 6 of plain and cipher “carphone.mp4” video with cipher key size 16×16. Proposed Perceptual Video Encryption Algorithm using Orthogonal Matrix.

Video Frames	**MSE**	**PSNR (dB)**
1	285.4137328	25.99411
3	287.4263238	25.84614
4	286.5063399	27.29025
6	284.4865123	26.08338
8	285.7369535	26.42255
11	285.2448346	27.23504
66	285.5906911	25.92997
72	285.5579738	26.21973
82	285.8267913	27.60389
90	286.4669528	26.97408

C.6 MSE and PSNR values of frame 6 of plain and cipher "carphone.mp4" video with cipher key size 64×64. Proposed Perceptual Video Encryption Algorithm using Orthogonal Matrix.

Video Frames	**MSE**	**PSNR (dB)**
1	311.7220451	24.61905
3	312.2771455	23.79666
4	311.4364477	25.58893
6	312.2115124	25.13068
8	310.6646048	27.84362
11	311.1490233	25.62854
66	311.7078789	26.21865
72	311.5063946	27.47991
82	311.5285881	28.21993
90	310.4622174	24.35861

APPENDIX D

PUBLICATIONS

1. Alhassan, S., Iddrisu, M. M. Daabo, M. I. (2019). Perceptual Video Encryption using Orthogonal matrix, *International Journal of Computer Mathematics: Computer Systems Theory*.

2. Alhassan, S., Iddrisu, M. M. Daabo, M. I. (2019). Improved Perceptual Video Encryption Technique using Residue Number System. Earthline J. Math. Sci. 2(1), pp. 111-125.

3. Alhassan, S., Iddrisu, M. M. Daabo, M. I. (2018). Perceptual Video Encryption via Unit Anti-diagonal Matrix, *Appl. Math. Inf. Sci.* Vol. 12(5), pp. 923-929.