جـــامـعــة الـــشرق الأوسـط
MIDDLE EAST UNIVERSITY
Amman - Jordan

Agile Encryption Scheme for Multimedia Files Using Random Data

مخطط تشفير لملفات الوسائط المتعددة باستخدام بيانات عشوائية

Prepared by

Mohanad Ali Hussein Al-Halboosi

Supervised by

Dr. Mudhafar Al-Jarrah

Thesis Submitted in Partial Fullfillment of the Requirements for the Degree of Master in Computer Science

Department of Computer Science

Faculty of Information Technology

Middle East University

January, 2021

بسم الله الرحمن الرحيم

﴿ .... يَرۡفَعِ ٱللَّهُ ٱلَّذِينَ ءَامَنُواْ مِنكُمۡ وَٱلَّذِينَ أُوتُواْ ٱلۡعِلۡمَ دَرَجَٰتٖۚ وَٱللَّهُ بِمَا تَعۡمَلُونَ خَبِيرٞ ١١ ﴾ [سورة المجادلة، ١١]

صدق الله العظيم

Authorization

I **Mohanad Ali Hussein AL-Halboosi**, authorize Middle East University to provide and electronic copies of my thesis to the libraries, organization, or bodies and institutions concerned in research and scientific studies upon request.

Name: Mohanad Ali Hussein AL-Halboosi.

Date: 28/4/2021

Signature: [illegible]

Thesis Committee Decision

This thesis titled "**Agile Encryption Scheme for Multimedia Files Using Random Data**" was successfully defended and approved on 2021/1/26 .

Thesis Committee Members	**Signature**
(Supervisor)	
Dr. Mudhafar Munir Al-Jarrah	[illegible]
(Head of the Committee and Internal Examiner)	
Dr. Ahmad Al-Hmouz	[illegible]
Middle East University	
(Internal Examiner)	
Dr. Bassam Al-Shargabi	[illegible]
Middle East University	
(External Examiner)	
Dr. Adnan Hadi Mahdi	[illegible]
Al-Israa University	

Acknowledgment

First, I give thanks, and praise to Allah for his mercy, and reconcile and for granting me knowledge, confidence, patience to pass this Master thesis successfully.

Also, I would like to express my gratitude to my thesis supervisor, **Dr. Mudafar Al-Jarrah** for the complete guidance throughout the thesis stages, and for the critical assistance in designing and preceding the methodology of my research.

Finally, I thank all those, who have helped me directly or indirectly in the successful completion of my research work.

Mohanad Ali AL-Halboosi

The Researcher

Dedication

To the one who always kept me in her prayers, and did not save any effort to assist me throughout my life, **my Beloved Mother.**

To **My Father**, who has been always struggling to assure us a decent life, who raised me on the acts of mannerism, who kept admonishing me by the trust and honesty.

To my gorgeous **Brothers** who kept pushing me off the boundaries and assisted me in all the possible means.

To the woman who struggled with me through this journey and never stopped supporting and dedicating every possible means to support me through this journey, **My Wife**

To the light of my eyes, to my heart, **My Daughter**.

I dedicate my effort

Mohanad Ali AL-Halboosi

Table of Contents

List of Tables

List of Figures

List of Abbreviations

Abbreviations	Meaning
DES	Data Encryption Standard
AES	Advanced Encryption Standard
OPC	Open Platform Communications
RSA	Rivest-Shamir-Adleman
Enc	Encryption
PSNR	Peak Signal-to-Noise Ratio
GSM	Global System for Mobile
SPN	substitution–permutation network
ECB	Electronic Codebook Mode
CBC	Cipher-Block Chaining Mode
OCB	Offset Codebook Mode
OFB	Output Feedback Mode
CTR	Counter Mode
CFB	Cipher Feedback Mode
NIST	National Institute of Standards and Technology

Agile Encryption Scheme for Multimedia Files Using Random Data

Prepared by: Mohanad Ali Hussein Al-Halboosi

Supervised by: Dr. Mudhafar Munir Al-Jarrah

Abstract

In recent years there has been a big increase in the transmission of multimedia data for personal, business, and entertainment applications, over private and public networks. The exchange of multimedia files has been targeted by adversaries and hackers for various illegal purposes, which made it necessary to protect the security of these files by encrypting the files in a secure way to prevent access to their contents. However, multimedia files, especially videos, tend to be large and so require lightweight encryption methods, in particular when used on low performance computing platforms.

This thesis presents an agile (lightweight) symmetric encryption model to provide protection for the security of multimedia files using random data. The proposed model encrypts every byte of the plaintext file, including the header data, using stream cipher method, in order to provide higher level of security compared with selective encryption. The encryption algorithm utilizes a generated set of random numbers using a secret key as a seed, to be used in calculating the encrypted bytes. The decryption algorithm uses the same secret key and follows an inverse of the encryption steps. The secret key consists of a set of randomly generated decimal digits which provide a large key space.

The proposed model is implemented in MATLAB as a working system to encrypt multimedia files of various types, such as video, audio, or image, compressed or uncompressed. The experimental work was focused on video files as they tend to be large and are widely used in various applications such as streaming video. A dataset of public video files of various sizes of the MP4 type, which is the most widely used video type, is utilized in evaluating the performance efficiency of the system. The metric of throughput as KB per second was used as a measure of the performance efficiency. The experimental results showed an average of encryption throughput of 1297 KB / second, and a similar value for the decryption throughput. Evaluating the integrity of the encryption / decryption processes was carried out by comparing the plaintext and the decrypted files, which showed complete compatibility. The thesis ends with conclusions and suggestions for future work.

Keywords: Cryptography, Symmetric Encryption, Decryption, Multimedia Encryption, Lightweight Encryption, Encryption Throughput, Stream Cipher.

مخطط تشفير لملفات الوسائط المتعددة باستخدام بيانات عشوائية

إعداد: مهند علي حسين الحلبوسي

إشراف: د. مظفر منير الجراح

الملخص

شهدت السنوات الاخيرة إزديادا كبيرا في تراسل البيانات للملفات المتعددة الوسائط، للاستخدامات الشخصية وللاعمال ولتطبيقات التسلية، من خلال الشبكات العامة والخاصة. تعرض تبادل ملفات الوسائط المتعددة للاستهداف من قبل الخصوم والمتطفلين وذلك لاغراض غير شرعية متنوعة، والذي جعل من الضرورة حماية أمن هذه الملفات من خلال تشفيرها بإسلوب آمن لمنع الوصول الى محتوياتها. إلا أن ملفات الوسائط المتعددة وبالاخص ملفات الفديو تميل الى كونها كبيرة الحجم وبالتالي يتطلب تشفيرها الى طرق تشفير خفيفة الوزن، وبالذات عندما تستخدم على منصات حاسوبية منخفضة القدرة تعرض هذه الرسالة نموذج رشيق (خفيف الوزن) للتشفير المتماثل وذلك لتوفير حماية لامن ملفات الوسائط المتعددة باستخدام البيانات العشوائية. يشفر النموذج المقترح كل بايت من بايتات ملف النص العادي، ويشمل ذلك بيانات ترويسة الملف، باستخدام طريقة التشفير الدفقي، وذلك لتحقيق مستوى أعلى من الامن مقارنة مع التشفير الانتقائي. تستخدم خوارزمية التشفير مجموعة من الاعداد العشوائية المولدة بالاعتماد على مفتاح سري يكون بذرة بدء سلسلة الاعداد العشوائية، وتستخدم الاعداد العشوائية في إحتساب البايتات المشفرة. تستخدم خوارزمية فك التشفير نفس المفتاح السري المستخدم في التشفير وتتبع خطوات معكوسة لخطوات التشفير. يتكون المفتاح السري من مجموعة من الارقام العشرية المولدة عشوائيا، والذي يوفرمدى واسع للمفتاح.

تم تنفيذ النموذج المقترح كنظام عامل باستخدام لغة ماتلاب، لتشفير ملفات وسائط متعددة متعددة الانواع، مثل الفديو، الصوت، الصور، سواء كانت مضغوطة أو غير مضغوطة. تركز العمل التجريبي على تشفير الملفات الفديوية لميلها لان تكون كبيرة الحجم وواسعة الاستخدام في تطبيقات متنوعة مثل التدفق الفديوي. تم إستخدام مجموعة بيانات عامة لملفات فديوية باحجام مختلفة من نوع MP4 وهو النوع الاكثر استخداما ضمن الانواع الفديوية، وذلك لتقييم كفائة اداء النظام. أظهرت النتائج التجريبية معدل إنتاجية تشفيربلغ 1297 كيلو بايت بالثانية، وبقيمة مماثلة لعملية فك التشفير. تم تقييم سلامة عمليتي التشفير وفك التشفير من خلال مقارنة ملفات النص العادي مع ملفات النص الناتج عن فك التشفير، وبينت المقارنة تطابق تام. تنتهي الرسالة بإستنتاجات ومقترحات لاعمال مستقبلية.

الكلمات المفتاحية: علم التشفير، التشفير المتماثل، فك التشفير، تشفيرالوسائط المتعددة، التشفير خفيف الوزن، إنتاجية التشفير، التشفير الدفقي.

CHAPTER ONE
Introduction

CHAPTER ONE
Introduction

1.1 Research Context

The research in this proposal focuses on protecting the privacy and security of multimedia files through encryption. Multimedia files tend to be large, for which the classical encryption methods can be less useful in a practical situation due to algorithm’s complexity. The other issue in dealing with multimedia files is that they come in variety of formats and compression methods which needs to be addressed in a general scheme that is format independent.

1.2 Background

A problem of great importance in recent years has been the exchange of confidential multimedia data of photos, sound clips and video clips, which require reliable techniques to protect the confidential data from potential observers or intruders who might attempt to illegally access the data without authorization.

A common technique for protecting information in digital media such as video, audio, and images is information encryption (Yassein, M. B., 2017) , and this technology can be used by individuals and companies to send their private and confidential multimedia information in a safe way by using encryption tools that prevent unauthorized access to such data. However, the problem of encrypting information is that it can be used for unlawful purposes, such as exchange of messages of various media types by criminals and insiders within businesses who attempt to send private business documents to competitors.

Information encryption technology (Eskicioglu, A. M., 2004) is a security measure that aims to protect plain data in media files by scrambling the data in such a way that an intruder or hacker will not be able to make sense out of the scrambled data. Information encryption technology has mainly been used in encrypting textual data, such as e-mails, social media messages and sensitive documents such as contracts, legal agreements, treaties between nations and technical specifications of documents. However, more recently a lot of the sensitive and confidential documents come in multimedia files which require protection, such as entertainment video films, audio messages for business and marketing campaigns, as well as private video and audio clips whose privacy needs to be protected. Extending traditional encryption techniques (Kolhe, V., 2014) that have dealt mainly with textual documents, to deal with encryption of large multimedia, can require significant processing time, therefore new encryption models are needed that take into account the complexity and nature of the multimedia data and the need to have light-weight encryption for such media.

1.3 Problem Statement

The problem addressed in this proposal is the protection of privacy and security of multimedia files such as video, audio, images and text of various formats, through encryption. Well-established encryption algorithms such as DES and AES (Kaur , M., & Kaur, G., 2014) are not suited for encrypting large multimedia files due to the complexity of the algorithms that can take a long time to encrypt and decrypt such files. Furthermore, multimedia files come in a variety of formats and compression methods which needs to be tackled in an encryption scheme.

1.4 Goal and Objectives

The goal of this research is to enhance the privacy and security of multimedia files that are exchanged over the internet through lightweight encryptions.

The following objectives are considered:

1. Select multimedia files formats to be encrypted.
2. Design an encryption / decryption symmetric model that is based on using random data.
3. Implement the encryption / decryption model as a working system.
4. Test the implemented system on a set of multimedia files of various sizes.
5. Measure the encryption / decryption performance efficiency in terms of throughput per time unit.
6. Check the integrity of the encryption / decryption system.

1.5 Motivation

This research is motivated by the need to protect the privacy and security of multimedia files that are sent over the internet, which have seen an exponential increase in recent years due to the wide-spread use of social networking, teleworking, and distance learning.

1.6 Significance of Research

This research is expected to enhance and reduce the complexity of encrypting multimedia files of various formats and sizes which would lead to realizing effective tools for protecting private data from intruders and social networks espionage monitors.

1.7 Research Questions

In this research work we will attempt to answer the following questions:

1. What are the multimedia file types and formats that will be the target of encryption.

2. Will the proposed model deal with compressed as well as un-compressed data.
3. What are the structure and steps of the proposed encryption / decryption algorithms.
4. What will be the expected encryption time for multi-megabyte files.
5. How will the integrity of the output of the encryption / decryption processes be evaluated.
6. How does the proposed model compare with previous models in terms of encryption time.

1.8 Scope of Research

The scope of this research will cover the following points:

- Designing an encryption scheme that deals with multimedia files of various formats and types.
- Implementing the encryption / decryption scheme using a multimedia-supporting development environment.
- Experimenting with large multimedia files containing video and audio data to evaluate the time efficiency of the encryption model.

1.9 Limitations of the Proposed Research

This research is limited to symmetric encryption using a numeric secret key consisting of random decimal digits. Further work will be needed to adapt the proposed model for public key cryptography with asymmetric keys.

CHAPTER TWO
Literature Review and Related Work

CHAPTER TWO
Literature Review and Related Work

2.1 Introduction

The preservation of multimedia data and the rise of digital communication on the internet are becoming increasingly relevant. The usage of different kinds of apps of a wide variety of photographs and videos nowadays puts great emphasis on protection and privacy issues. Multimedia data encryption helps avoid undesirable and unwanted revelation through transit or storing of sensitive details.

There are three principle needs for encryption regarding mixed media insurance, which are namelessness, information trustworthiness, and validation. While AES (Kasat, 2015) and Rijndael (Jimeno, 2008) encryption calculations are utilized conversely, there is just a single differentiation, with help ranges and code key lengths. Rijndael has variable square and key: the square length and key length are discrete , As long as it is different 32 pieces and somewhere in the range of 128 and 256 pieces. Albeit a square or bigger key length can be determined in the Rijndael form, it may not be appropriate as of now. While AES indicates a square length of 128 pieces and the solitary worthy key lengths are 128 and 192 And 256. While gathering AES, the extra mass and key length in Rijnael don't matter to the new FIPS spec too.

With the headway of both PC innovation and the Internet, the utilization of interactive media information is expanding quickly. Hence, there is an incredible need to ensure delicate information before it is sent or appropriated. Accordingly, in this proposal the Rijndael calculation is considered, which assists with ensuring interactive media content by methods for coding strategy and a hypothetical report is composed dependent on the material contemplated.

2.2 Security of Multimedia

The proposed resource acknowledgment standard for a cell relationship could be joined with this reason for traffic load tuning.

These cell interface methodologies can be combined with the fundamental force control plans dependent on the ideal objectives. The primary issue with this technique is to pick a fitting mix of cell connection and energy control innovation to accomplish a particular objective. For instance, the lower co-cell relationship dependent on impediment and OPC can't address the objective of intensifying uplink execution (P3), considering the way that all customers in this circumstance endeavor to connect with a practical impedance to a base BS created in Ultimately high sending capacity to all customers. Despite the fact that outline execution improves when customers with great channel conditions increment their transmission power, it is harmed when customers with support channel conditions increment their transmission strength.

Two big protection innovations are being developed to overcome the technological challenges:

1. End-to-end encryption multimedia crypting technologies as digital content is spread through a wide variety of delivery networks.
2. Watermarking is used to discourage copying, copyright ownership and authentication.

Classification implies shielding individual data from unapproved access. An undesirable gathering known for a derivation should not have the option to get to the materials. Information trustworthiness guarantees that data isn't altered in any unfortunate manner. Accordingly, approval techniques are concentrated in two gatherings:

2.3 Cryptography

Over the past decades, information technology has infiltrated more and more areas of oursociety. The development of digital information and telecommunication systems opened a wide range of new possibilities which were seized to improve the efficiency of different sorts of processes. Today, an everincreasing number of interactions between end users, organizations such as banks, and governments is carried out electronically. Two of the most remark- able breakthroughs were the world-wide expansion of the Internet and the spectacular growth of digital mobile networks (e.g., GSM). The number of users of both systems, which were nonexistent or confined to research com- munities until the early 1990s, is now in the order of a billion.

The success of these new technologies can be attributed to a number of in- trinsic advantages of digital systems: digital information is nearly insensitive to noise, it can be sent over long distances, copied or modified without any loss of quality. Moreover, the link between the information and its carrier has disappeared. The exact same piece of information can be transmitted over a wireless link, sent over an optical fiber, stored on a hard disk, and printed on a barcode. This allows a large number of very different devices to interact seamlessly.

However, the same properties which make digital information systems so attractive render them particularly vulnerable to a broad range of abuses. In a traditional mail system, the receiver of a letter can perform some simple tests to assure himself that the message was not compromised. He can check that the (sealed) envelope was not opened, study whether the handwriting or signature matches, and search for anomalies which might indicate that parts have been rewritten. These tests are all built on the assumption that any manipulation of the message will necessarily leave some traces on its physical carrier. Unfortunately, this is exactly what digital systems have tried to avoid. As was soon understood, the only way to secure digital systems without sac- rificing their advantages, is to transform the information in such a way that it protects itself, independently of how it is transferred or stored. The science which studies this problem is called cryptology, and an excellent (but maybe slightly outdated)

It should be noted that many of the security issues raised by modern in- formation technology are not new. Still, owing to the large scale of current information systems and the unprecedented impact they have on our daily live, cryptology has never been more important than today. Rapidly expanding networks are interconnecting more and more devices all over the globe, increasing both the number of interesting targets and the number of potential attackers. Moreover, eavesdropping on these networks has become much easier with the proliferation of wireless access points. Finally, the growing complexity of communication and information systems makes their security much harder to control, giving rise to some rather unexpected new problems such as computer viruses and worms.

2.3.1 Symmetric Encryption

The protection of digital information typically involves at least two distinct problems: secrecy protection (preventing information from being disclosed to unintended recipients) and authentication (ensuring that received messages originate from the intended sender, and were not modified on their way). This thesis is entirely devoted to the first problem, with the exception of the appendix, which discusses some specific aspects of the second one.

In cryptology, intended senders and recipients are distinguished from un- intended ones by assuming that they know some secret pieces of information, called keys. These keys can be shared between the sender and the receiver, or they can be different, in which case the sender and receiver are also prevented from impersonating each other. In this thesis, we will concentrate on the first case, called symmetric cryptography.

Symmetric cryptography addresses the problem of secrecy protection by using the shared secret key to transform the message in such a way that it cannot be recovered anymore without this key. This process is called symmetric encryption. Algorithms which perform symmetric encryption are known as ciphers. Based on the paradigm used to process the message, these ciphers are typically categorized into one of two classes: block ciphers and stream ciphers.

The security of symmetric encryption algorithms can in general not be proved (the notable exception being the one-time pad). Instead, the trust in a cipher is merely based on the fact that no weaknesses have been found af- ter a long and thorough evaluation phase. This explains the importance of a strong interaction between cryptography, the field which studies techniques to protect information, and cryptanalysis, which focuses on methods to defeat this protection. In this thesis, we will promote simplicity as an effective cata- lyst to enhance this interaction. While simple designs may be more likely to be vulnerable to simple, and possibly devastating, attacks, they certainly in- spire more confidence than complex schemes, if they survive a long period of public scrutiny despite their simplicity.

2.3.2 Encryption Algorithms

The purpose of an encryption algorithm is to protect the secrecy of messages which are sent over an insecure channel. A general encryption algorithm consists of two mathematical transformations : an encryption function E, and a decryption function $D = E^{-1}$. In order to communicate in a secure way, the sender (traditionally called

Alice) will apply the encryption function to C = E(P) over the insecure channel. Once C is received by the intended recipient (called Bob), the plaintext is recovered by computing D(C) = P . the original message P (the plaintext), and transmit the resulting ciphertext In order for this scheme to meet its purpose, i.e., to ensure that Alice's message will only be read by Bob, a number of conditions need to be fulfilled. First, the decryption function D must be known to Bob but kept secret from anybody else, with the possible exception of Alice. Secondly, the transformation E must be designed in such a way that an eavesdropper intercepting the ciphertext (often called Eve) cannot, at least in practice, extract any information about the plaintext, except maybe its length. Finally, the implementations of the transformations E and D should not require a prohibitive amount of computational resources.

2.3.3 Symmetric vs. Asymmetric Encryption

Until the 1970s, it was intuitively assumed that the previous conditions immediately implied that the encryption function E had to be secret as well. The reasoning was that if Eve were given E, it would suffice for her to reverse

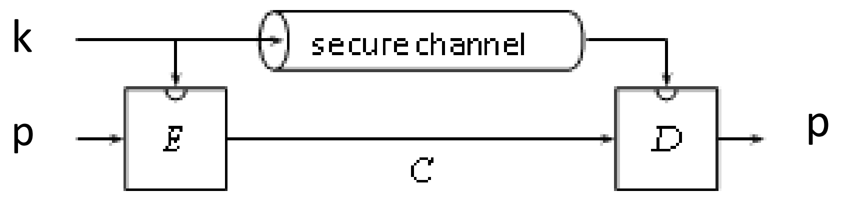


Figure 2.1 Model for symmetric encryption

This transformation to recover D. In the mid-1970s Diffie and Hellman realized that the secrecy of the encryption function was in fact not required, at least in theory, provided that one could construct so-called trapdoor one- way functions. These are functions which are easy to evaluate, but cannot be inverted efficiently, unless some extra information (the trapdoor) is given. Examples of trapdoor one-way functions were soon found, and allowed the development of practical public key encryption algorithms such as RSA .

While public key cryptography has the major advantage that Bob does not need to exchange any secret information with Alice before she can start encrypting, schemes which do rely on the secrecy of their encryption function still play a vital role in practical systems. The reason is that implementations of secret key or symmetric encryption algorithms, as they are called nowadays, are orders of magnitude more efficient than their public key (or asymmetric) counterparts. As its title suggests, this thesis will exclusively deal with symmetric encryption algorithms .

2.3.4 Kerckhoffs' Principle

In most situations, it is fairly hard to keep an encryption or decryption algorithm completely secret: either Alice and Bob have to design and implement their own algorithm, or they have to trust a designer not to disclose the algorithm to others. Moreover, for each correspondent Alice wants to communicate with, she will need a different algorithm. The solution to this problem is to introduce a secret parameter K as in Fig. 2.1, and to construct parametrized encryption and decryption functions, in such a way that $D_{K'}(E_K(P))$ does not reveal anything about P as long as $K' \neq K$. Instead of repeatedly having to design new secret algorithms, it now suffices to agree on a secret value for K, called the key. Typically, this key is a short binary string of 80 to a few hundred bits. Since the security of the resulting system only relies on the secrecy of the key, the functions E and D can now be shared and even made public. The principle that full disclosure of the underlying algorithms should never affect the security of a good encryption scheme, as long as the key is kept secret, is known as Kerckhoffs' principle.

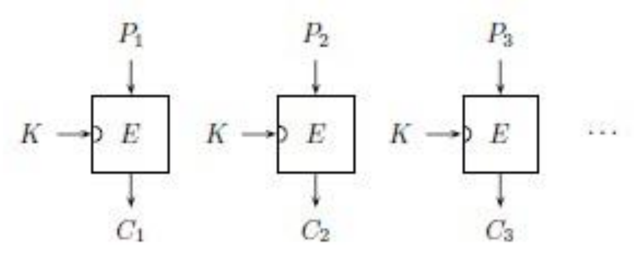


Figure 2.2 : A block cipher in so-called ECD mode

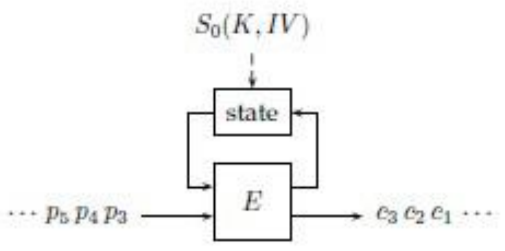


Figure 2.3 : A stream cipher

2.3.5 Stream and Block Encryption

Let us now take a closer look at the boxes E and D in Fig. 2.1. In practice, these boxes will not take a complete text as input and then transform it at once, but rather operate in a sequential fashion. With this respect, symmetric encryption algorithms are traditionally divided into two categories: stream ciphers and block ciphers. A block cipher divides the plaintext into separate blocks of fixed size (e.g., 64 or 128 bits), and encrypts each of them independently using the same keydependent transformation. A stream cipher, on the other hand, takes as input a continuous stream of plaintext symbols, typically bits, and encrypts them according to an internal state which evolves during the process. The initialization of this state is controlled by the secret key K and a public initial value IV . The differences between both systems are illustrated in Fig. 2.2 and Fig. 2.3.While the definitions above would at first sight allow to draw a clear the oretical distinction between stream ciphers and block ciphers based on the presence of a state, the situation is a bit more blurred in practice. The fact is that block ciphers are rarely used in the way shown in Fig. 2.2. Instead, the output of the key-dependent transformation is typically kept in memory and used as a parameter when encrypting the next block. While this approach, known as cipher block chaining , would fall into the category of stream encryption according to the previous definition, it is still commonly called block encryption. In order to resolve this apparent inconsistency, we will follow Daemen’s .

suggestion, and make a distinction between a block cipher, which is just an invertible key dependent transformation acting on blocks of fixed length, and a block encryption scheme, which encrypts plaintexts of arbitrary length and will typically use a block cipher as a component. While a block cipher is stateless by definition, this is rarely the case for a block encryption scheme . The need for a state in a block encryption scheme arises from a fundamental difference in approach between block ciphers and stream ciphers. A stream cipher tries to defeat the adversary by making the encryption of a plaintext symbol depend in an unpredictable way on the position in the stream. A block cipher, on the other hand, aims to make its output depend in an unpredictable way on the value of the plaintext block. A consequence of the latter approach is that repeated blocks in a plaintext message are easily detected at the output of a block cipher, thus providing the adversary with possibly useful information. The purpose of the state in a block encryption scheme is precisely to make such repetitions unlikely, by first “randomizing” the plaintext blocks before they are fed into the block cipher.

The different roles played by the internal states of block and stream encryption schemes are reflected in the following informal definitions:

Definition 2.1 (block encryption). A block encryption scheme is an encryption scheme whose state, if it has one, can be kept fixed without significantly reducing its security, provided that the plaintext symbols are independent and uniformly distributed.

Definition 2.2 (stream encryption). A stream encryption scheme is an encryption scheme whose state cannot be kept fixed without severely reducing its security, even if the plaintext symbols are independent and uniformly distributed. It is interesting to note that the two branches in symmetric cryptology have evolved in rather different circumstances. Block ciphers owe much of their popularity to a few successful designs (such as DES and its successor, AES) which are standardized, freely available, and can be deployed in many different applications. The most widely used stream ciphers, on the contrary, are proprietary designs (e.g., RC4, A5/1), closely tied to a particular application (e.g., GSM). Many of these designs were kept secret, until they eventually leaked out, or were reverse-engineered. This explains why, in the 1990s, stream ciphers have tended to receive less attention from the open research community than block ciphers.

2.3.6 Anatomy of a Block Cipher

Whereas stream ciphers are based on a variety of principles, most block cipher designs follow the same general approach. They typically consist of a short sequence of simple operations, referred to as the round function, which is repeated r times (called rounds). The first round takes an n-bit plaintext block

as input, and the last round outputs the ciphertext. Additionally, each round depends on a sub key (or round key) which is derived from a k-bit secret key (this derivation process is called the key schedule). Since the receiver must be able to uniquely decrypt the ciphertext, the round function has to be bijective for any value of the secret key. This is usually achieved in one of the following ways:

Feistel Ciphers. The round function of a Feistel cipher (named after H. Feistel, one of the IBM researchers who designed LUCIFER and DES) splits the input block into two parts L_{i-1} and R_{i-1}. The right part R_{i-1} is left unchanged and forms the left part of the output L_i . The right part of the output is constructed by adding a modified copy of R_{i-1} to the left part of the input L_{i-1},

i.e.,

$$L_i = R_{i-1},$$

$$R_i = L_{i-1} + f\,(R_{i-1}, K_i)\ .$$

It is not hard to see that this operation can be inverted by subtracting $f(L_i,K_i)$ from R_i, no matter how the function *f* is constructed. Many block ciphers are based on this structure, including DES.

SP Networks. Another approach consists in building a round function by combining layers of simple invertible functions: substitutions (called S-boxes) and permutations .The substitution layers act on small units of data (rarely more than 8 consecutive bits), and their highly nonlinear properties introduce local confusion into the cipher. The permutation layers, on the other hand, are simple linear transformations, but they operate on the complete block, and thus diffuse the effect of the substitutions. The terms confusion and diffusion, which were introduced by Shannon , will be a recurrent theme in this thesis. The most prominent block cipher based on an SP network is the Advanced Encryption Standard (AES). Notice also that the f -functions of many Feistel ciphers consist of a small SP network .

2.3.7 Modes of Operation

A block cipher in itself is just a component which describes a set of invertible transformations on blocks of fixed length n. Before Alice can actually start encrypting data, she needs to turn this block cipher into an encryption scheme. The different ways in which this can be achieved are called modes of operation. The purpose of a mode of operation is to extend the cryptographic properties of a block cipher to larger messages. The property which this thesis mainly focuses on is confidentiality, but modes providing message integrity and authenticity, possibly in addition to confidentiality, exist as well.

Although security obviously remains the primary criterion, other (noncryptographic) considerations often play an equally important role in the selection of a mode of operation:

Data Expansion. Some constructions require the plaintext length to be an ex- act multiple of the block length n. This implies that the original message may have to be expanded with extra padding bits, which is usually un- desirable.

Error Propagation. Single bit transmission errors may have different effects on the decrypted ciphertext. Either the error only affects a single bit or block of the recovered plaintext, or it might propagate to one, a few or all subsequent blocks.

Random Access. A number of modes allow ciphertext blocks to be decrypted (or even modified) at arbitrary positions without first having to process all preceding blocks. This is particularly useful for storage encryption.

Parallel Processing. Some modes allow different blocks to be processed simultaneously, which may be an interesting way to increase the through- put in certain applications.

We now shortly review some of the most common confidentiality modes. Except for the OCB mode, all of these modes have been standardized by NIST in the first part of Special Publication . The remaining three parts of SP 800-38, which describe authentication and authenticated encryption modes, are definitely a recommended read, but lie beyond the scope of this thesis.

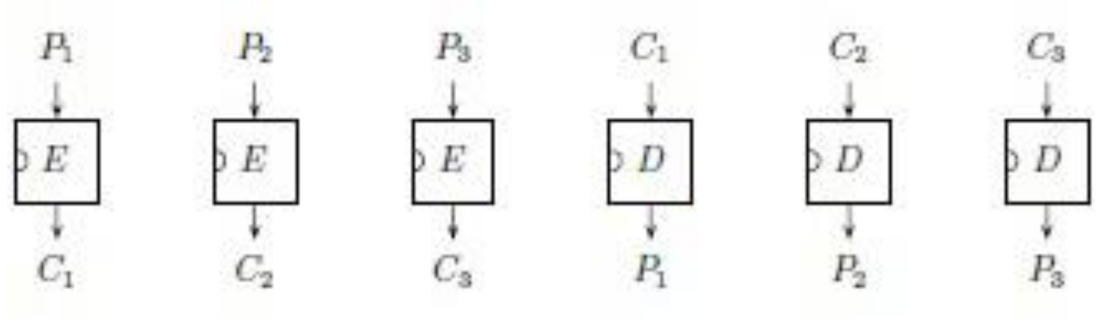


Figure 2.4 : A block cipher in ECB mode

Block Encryption Modes

The first three modes in this section turn a block cipher into a block encryption …

Electronic Codebook Mode (ECB). The ECB mode is without doubt the most straightforward way to encrypt messages whose length exceed the block length: the message is simply partitioned into n-bit blocks, each of which is encrypted independently. The scheme is depicted in Fig. 2.4 and can be described as follows:

Encryption: $C_i = E_K(P_i)$.

Decryption: $P_i = D_K(C_i)$.

The advantages of this mode are its simplicity and its suitability for parallel processing. Blocks at arbitrary positions can be encrypted or decrypted separately and errors do not propagate from one block to an- other, however, the major problem of this approach is that it does not hide all patterns in the plaintext: i.e., when- ever the plaintext contains identical blocks, so will the ciphertext. This limits the applications of the ECB mode to those (rare) cases where all blocks encrypted with a single key are guaranteed to be different.

Cipher-Block Chaining Mode (CBC). The CBC mode, which is presently the most widely used mode of operation, masks each plaintext block with the previous ciphertext block before applying the block cipher (see Fig. 2.5):

Encryption:	Decryption:
$C_0 = IV$,	$C_0 = IV$,
$C_i = E_K(C_{i-1} \oplus P_i)$.	$P_i = D_K(C_i) \oplus C_{i-1}$.

Since the output of a good block cipher is supposed to be completely unpredictable for anyone who does not know the key, all consecutive values of $C_{i-1} \oplus P_i$ will appear to be independent and uniformly distributed, and this regardless of the plaintext (assuming that the text itself does not depend on the key)

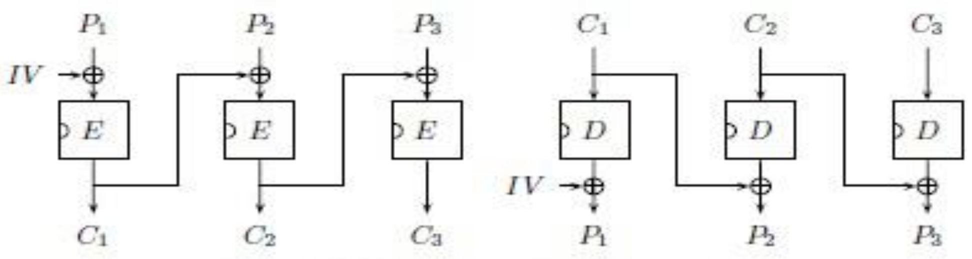


Figure 2.5 : A block cipher in CBC mode

Repetitons at the input of the block cipher are therefore unlikely to occur, which remedies the main short coming of the ECB mode. However, when the message length exceeds $2^{n/2}$ blocks, repeated values start to be unavoidable because of the birthday paradox. For this reason, the CBC mode (and in fact all modes in this section) should never be used to encrypt more than $2^{n/2}$ blocks with the same key . The cost of masking the plaintext in CBC is that the ciphertext feedback in the encryption part prevents the blocks from being processed in parallel. The decryption, on the other hand, depends only on two consecutive ciphertext blocks, and can still be performed independently for each block. This has the additional benefit that a bit error in the ciphertext can only affect the decryption of two blocks.

Offset Codebook Mode (OCB). The masking in CBC effectively destroys all dependencies in the plaintext. However, if the only purpose is to pre- vent repeated input blocks, then it suffices to require that the blocks are pairwise independent, or even weaker, just pairwise differentially uniform, i.e., that the difference between any two input blocks is uniformly distributed. The IAPM mode by Jutla and, derived from it, the OCB mode by Rogaway are block encryption modes based on this observation. Both modes are variants of the ECB mode in which a stream of words Z_i is added before and after the encryption (see Fig. 2.6):

Encryption: Decryption:

$$C_i = E_K(P_i \oplus Z_i) \oplus Z_i \,. \qquad P_i = D_K(C_i \oplus z_i) \oplus Z_i \,.$$

In the case of OCB, the words Zi are distinct multiples of a secret parameter L, which is derived from the key K by encrypting a nonce (i.e., a value used only once).

A first advantage of OCB compared to CBC is that it is completely parallelizable, both for encryption and decryption. But what makes the mode particularly attractive is the fact that it provides message authenticity at a very small additional cost: it suffices to compute a simple

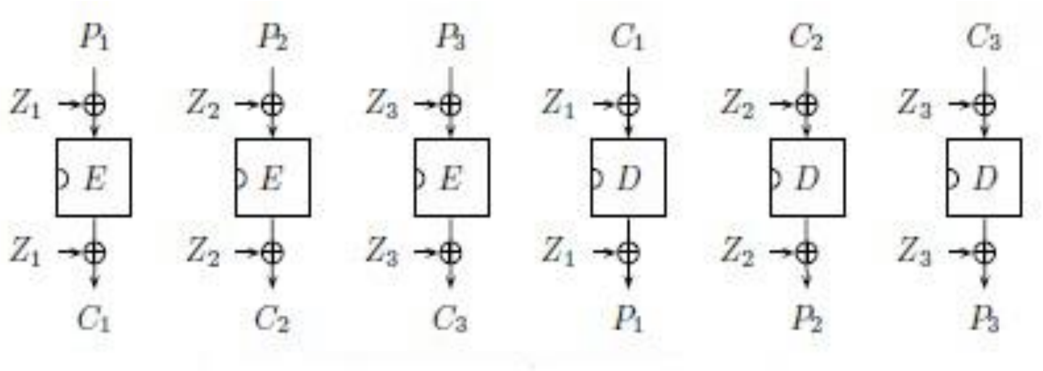


Figure 2.6 : A block cipher in OCB mode

(non-cryptographic) checksum of the plaintext, and to encrypt it with the same key K.

A drawback for the deployment of OCB is its intellectual property situation, which is somewhat unclear. This explains why the OCB mode was relegated to optional status in the IEEE 802.11i standard, in favor of the mandatory CCM mode, which is not encumbered by patents.

Stream Encryption Modes

Block ciphers can also be used to perform stream encryption, as illustrated by the three modes below. A noteworthy feature of these modes is that they only use the encryption function of the block cipher.

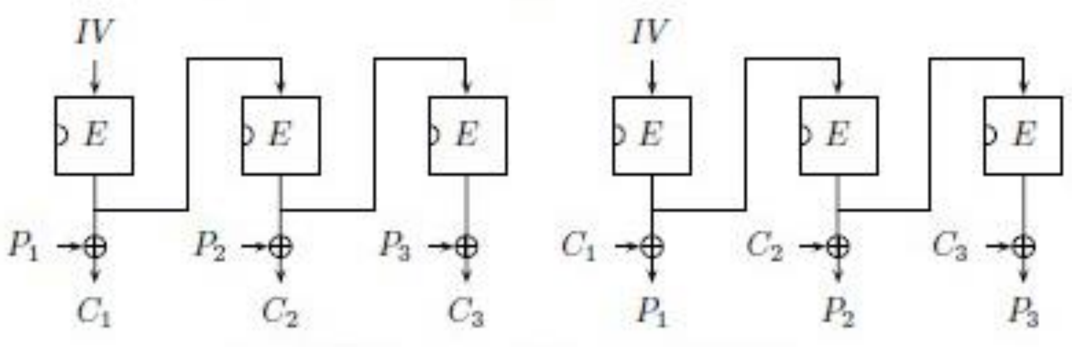


Figure 2.7 : A block cipher in OFB mode

Output Feedback Mode (OFB). The OFB mode, depicted in Fig. 2.7, encrypts plaintext blocks by combining them with a stream of blocks called key stream, which is generated by iterating the block cipher:

Encryption:	Decryption:
$Z_0 = IV$,	$Z_0 = IV$,
$Z_i = E_K(Z_{i-1})$,	$Z_i = E_K(Z_{i-1})$,
$C_i = P_i \oplus Z_i$.	$P_i = C_i \oplus Z_i$.

The generation of key stream blocks in OFB is independent of the plaintext. This means that the stream can be precomputed as soon as the IV

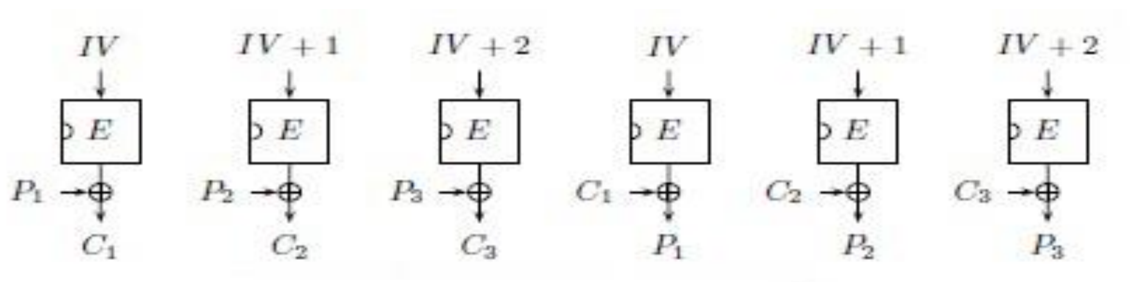


Figure 2.8 : A block cipher in CTR mode

is known, a feature which may be useful in real time applications. The mode is strictly sequential, though: the decryption of a single block at an arbitrary position in the ciphertext requires all preceding key stream blocks to be computed first.

Owing to the invertibility of E_K, all Z_i will necessarily be different, until one of them hits the value of Z_0 again, at which point the sequence will start repeating itself. A secure n-bit block cipher is not expected to cycle in much less than 2^{n-1} blocks, which implies that this periodicity has no practical consequences for a typical 128-bit block cipher. The mere fact that all Z_i within a cycle are different leaks some information as well, though. As a consequence, it is not recommended to encrypt much more than $2^{n/2}$ blocks with a single key.

Counter Mode (CTR). The CTR mode takes a similar approach as the OFB mode, but this time the key streamis generated by encrypting a counter (see Fig. 2.8) :

Encryption:	Decryption:
$Z_0 = IV$,	$Z_0 = IV$,
$C_i = P_i \oplus E_K(Z_i)$,	$P_i = C_i \oplus E_K(Z_i)$,
$Z_{i+1} = Z_i + 1$.	$Z_{i+1} = Z_i + 1$.

As opposed to the OFB mode, the CTR mode allows data blocks at arbitrary positions to be processed independently, both during encryption and decryption. This also allows pipelining in hardware, which can result in significant efficiency gains. Apart from this feature, the OFB and the CTR mode have very similar properties.

Cipher Feedback Mode (CFB). Both OFB and CTR (or OCB for that matter) require perfect synchronization during decryption, i.e., in order to decrypt a ciphertext block, the receiver needs to know the block's exact position in the stream. The CFB mode eliminates this requirement, and

is similar to CBC in this respect. The CFB mode is designed to process messages in r-bit segments, with

$1 \le r \le n$ (typically $r = 1$, $r = 8$, or, as in Fig. 2.10, $r = n$).

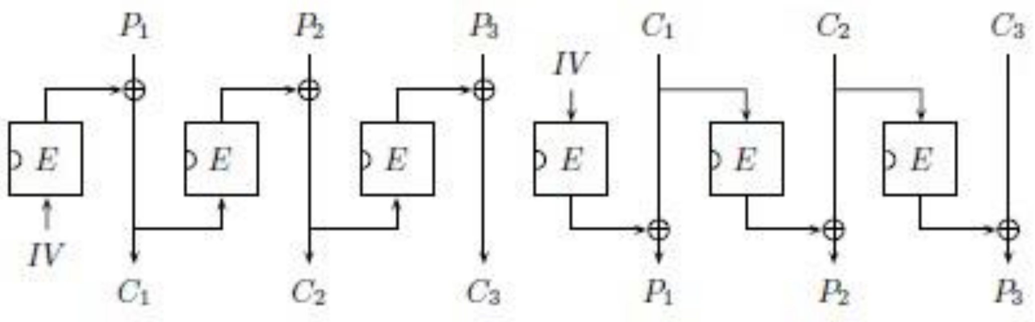


Figure 2. 9 : A block cipher in CFB mode

The encryption mode consists in shifting successive r-bit ciphertext segments back into an internal state block S_i, and combining the leftmost bits of $E_K(S_i)$ with the plaintext(see Fig. 2.9):

Encryption:

$$S_1 = IV\ ,$$
$$C_i = P_i \oplus E_K(S_i)[1 \cdot \cdot \cdot r]\ ,$$
$$S_{i+1} = (S_i \ll r) + C_i\ .$$

Decryption:

$$S_1 = IV\ ,$$
$$P_i = C_i \oplus E_K(S_i)[1 \cdot \cdot \cdot r]\ ,$$
$$S_{i+1} = (S_i \ll r) + C_i\ .$$

The feedback in CFB prevents the parallel encryption of plaintext blocks. Still, arbitrary ciphertext blocks can be decrypted independently, provided that the $\lceil n/r \rceil$ preceding blocks are available. As a direct consequence, single bit errors in the ciphertext cannot propagate over more

than $\lceil n/r \rceil$ successive blocks. Again, and for similar reasons as in CBC, a single key should not be used to encrypt more than $2^{n/2}$ blocks. For small values of r, additional precautions should be taken in order to avoid weak IV values. In particular, if the bits of the IV were to form a periodic sequence, then this would considerably increase the probability of repeated values at the input of the block cipher.

2.3.8 Dedicated Stream Ciphers

In its most general form, a stream cipher consists of a transformation which takes as input a plaintext symbol and the current state, and combines both to produce two outputs: a ciphertext symbol and the next state. This general construction was depicted in Fig. 2.3. In the vast majority of practical ciphers, however, the transformation E can be further decomposed into separate functions in either of the three ways shown in Fig. 2.10.

Synchronous Stream Ciphers

The first and by far most common approach is to update the state independently from the plaintext, in which case the stream cipher is said to be synchronous.

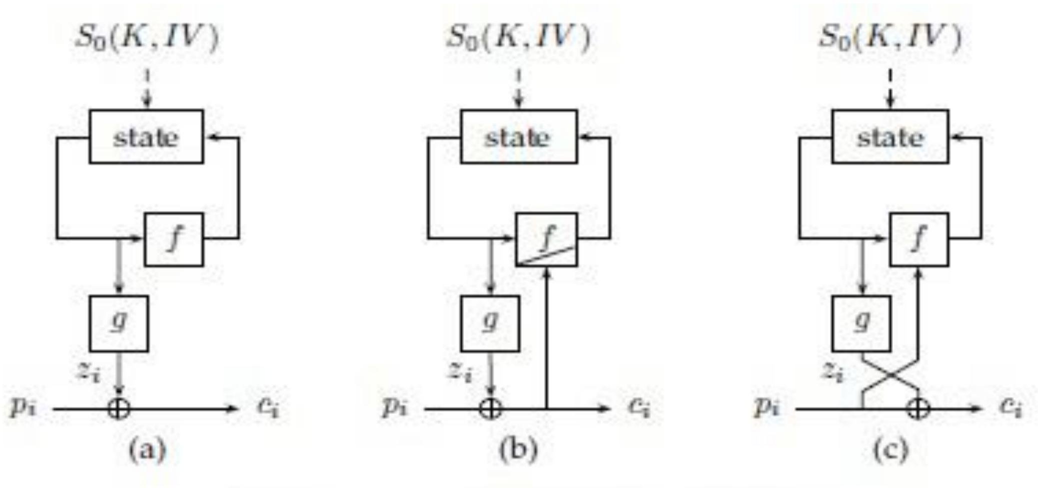


Figure 2.10 : Three type of stream ciphers

(a) binary additive ,(b) self-synchronizing , and (c) accumulating

The computations performed by a synchronous stream cipher can be described by three functions f, g, and h. The function f computes the next state, the function g produces key stream symbols zi, and the function h outputs ciphertext symbols by combining plaintext and key stream symbols:

Encryption:	Decryption:
$z_i = g(S_i)$,	$z_i = g(S_i)$,
$c_i = h(p_i, z_i)$,	$p_i = h^{-1}(c_i, z_i)$,
$S_{i+1} = f(S_i)$.	$S_{i+1} = f(S_i)$.

In addition to being synchronous, most modern stream ciphers are also binary additive, which means that their combining function *h* is simply defined as $c_i = p_i \oplus z_i$, as in Fig. 2.10 (a).

Note that we already saw two special examples of synchronous stream ciphers in the previous section, namely the OFB mode and the CTR mode. A notable common feature of these block cipher based constructions is the way in which they use their state: one part of the state (the key K) is kept secret, but stays constant during the whole encryption process; the other part is continuously updated, but is either known to the attacker, or directly used as key stream and thus easily derived from fragments of known plaintext. With respect to the functions *f* and *g*, OFB and CTR take diametrical approaches. The OFB mode uses an extremely simple *h*-function, and relies completely on the strength of *f* for its security. The CTR mode works the other way around. In order to keep deriving unpredictable key stream bits from *a* state whose bits are all either constant or known, the OFB and CTR constructions have to place heavy demands on either *f* or *g*, and this justifies the need for *a* relatively complex component such as a block cipher. Of course, nothing forces a synchronous stream cipher to use its state in this particular way. In fact, most dedicated synchronous stream ciphers will make sure that none of the state bits stay constant for more than a few iterations, and that only a fraction of these bits are revealed to the attacker at any time. This strategy, as opposed to the OFB and CTR constructions, allows the cipher to gradually accumulate unpredictable bits in its state. The advantage of this approach is that its security depends more on the interaction between f and g, than on the individual functions themselves. In principle, neither of these functions is required to be particularly strong on its own, which explains why dedicated synchronous stream ciphers have the potential to be significantly more efficient and compact than block cipher based schemes.

Examples of synchronous stream ciphers include modern stream ciphers, such as the widely used RC4 and A5/1, but also historical rotor-based machines such as Enigma (the latter is an illustration of a non-additive synchronous stream cipher). Note that even though all these ciphers share the same high level structure, the design of their components is often based on very different principles. Self-Synchronizing Stream

Ciphers The synchronous stream ciphers in the previous section derive their name from the fact that their state must be perfectly synchronized with the incoming cipher text streamin order to recover the plaintext. On unreliable channels, this requires an external synchronization mechanism, which can be impractical in certain applications. In these cases, self-synchronizing stream ciphers can come in handy. The underlying idea of self-synchronizing stream ciphers is to use the ciphertext stream itself to synchronize the state. The encryption and decryption operations are defined as follows:

Encryption:	Decryption:
$z_i = g(S_i)$,	$z_i = g(S_i)$,
$c_i = h(p_i, z_i)$,	$p_i = h^{-1}(c_i, z_i)$,
$S_{i+1} = f(S_i, c_i)$.	$S_{i+1} = f(S_i, c_i)$.

The function f is non-injective with respect to S_i, and defined in such a way that the state Si can always be computed as a function of the initial state and the last t ciphertext symbols, i.e.,

$$S_i = f(f (\cdot \cdot \cdot f(S_0, c_{i-t}) \cdot \cdot \cdot), c_{i-1}) , \forall i \geq t .$$

Fig.2.10(b) shows the encryption mode of a self synchronizing stream cipher where $h(p_i, z_i) = p_i \oplus z_i$. The narrowing shape of *f* symbolizes the non-injective nature of the function. Block ciphers in CFB mode are the most widely used self-synchronizing stream ciphers. The state in this case consists of a constant part which stores the secret key, and a variable part which contains the last $t = \lceil n/r \rceil$ ciphertext symbols. The state update function f is just a shift of this second part. As in CTR mode, the security of CFB completely relies on the strength of g. Dedicated (as opposed to block ciphers based) self-synchronizing stream ciphers are relatively rare. The reason for this is that the switch to a dedicated self-synchronizing stream cipher is not likely to result in the same efficiency gain as in the synchronous case. A first limitation is that self-synchronizing stream ciphers cannot accumulate unpredictability in their state to the same extent as synchronous stream ciphers, simply because (2.1) does not allow f to perform any computation that would affect the state for more than t iterations. The second complication is that the state update depends on ciphertext symbols, which means that an adversary who can influence the ciphertext (either by corrupting the communication channel, or by controlling parts of the plaintext), can also influence how the state is updated. In particular, the adversary can force the decrypting cipher to return to any previous state by replaying fragments of the ciphertext. This allows chosen-ciphertext (or - plaintext) attack scenarios which do not apply to synchronous stream ciphers. Because of these two structural properties, self-synchronizing designs still have to rely to a large extent on the individual strength of the combined function $g \circ f$, which limits the potential for large efficiency gains.

Accumulating Stream Ciphers

A third class of stream ciphers, which has only started to appear very recently, follows the structure of Fig. 2.10 (c), and can be described by the following equations:

$$\begin{array}{ll}
\text{Encryption:} & \text{Decryption:} \\
z_i = g(S_i)\ , & z_i = g(S_i)\ , \\
c_i = h(p_i, z_i)\ , & p_i = h^{-1}(c_i, z_i)\ , \\
S_{i+1} = f(S_i, p_i)\ . & S_{i+1} = f(S_i, p_i)\ .
\end{array}$$

The expressions above resemble somewhat the equations of a self-synchronizing stream cipher, but in fact, the construction serves the opposite purpose. Whereas a self-synchronizing stream cipher aims to limit the effect of communication errors (bit flips, insertions, or deletions), the goal of the current construction is to make sure that any modification in the ciphertext (be it accidental or malicious) would have a very high probability to permanently disturb the state. In order to achieve this, condition (2.1) is dropped, allowing the function f to be invertible with respect to both S_i and p_i. The class of ciphers based on this construction has not been given a special name in the literature so far, but in this thesis we will refer to them as accumulating stream ciphers. The main use of accumulating stream ciphers is to perform authenticated encryption. When used for this purpose, the state has to undergo some additional processing at the end of the encryption process, and the result is used as a message authentication code (MAC). Phelix and Shannon are two ciphers based on this paradigm. An issue that complicates the design of efficient and secure accumulating stream ciphers is the fact that the scheme provides the adversary with a means to influence the state, just as in self-synchronizing stream ciphers. This time, however, there is no obvious way for the adversary to reset the state. That is, unless she can force a reinitialization of the cipher with the same key and IV. If this possibility cannot be excluded, $g \circ f$ would need the same kind of strength as in a self-synchronizing stream cipher in order to resist attacks , and there would be little hope for the cipher to attain the same efficiency as a synchronous cipher.

2.4 History of Multimedia Encryption

Mixed media coding innovation was first presented in 1980 and turned into an interesting issue of examination during the 1990s, its advancement can be arranged into three stages; Raw information encryption, compacted information encryption and halfway encryption.

Before the 1990s, just some interactive media encoding strategies were normalized. The vast majority of the interactive media information (picture, video) has been moved or put away in crude structure. Mixed media encoding was essentially founded on pixel exchanging or encoding, for example the video/picture is adjusted with the goal that the subsequent information gets unimaginable. For instance, fill-in-clear bends are utilized to alter picture/video information, which confounds the connection between video pictures/neighboring pixels. European broadcasting companies utilize the Eurocrypt signal coding standard, which changes the field line by line (Saha Arunabh, 2015). These techniques are utilized because of their ease and computational multifaceted nature. Be that as it may, this kind of adjustment changes the connection between adjoining pixels, causing the pressure cycle to not work. In this manner, these encryption calculations are just helpful for an application that doesn't need pressure.

In the mid 1990s, with the headway of media innovation, some picture and sound/video encoding norms were grown, for example, JPEG, MPEG, and so forth This application.

After the headway of web innovation in the last part of the 1990s, assembling a mixed media application required more cooperation and activity continuously. By encoding just certain bits of information, the size of the last encoded record can be diminished by improving the productivity of the encryption. Because of the fast development of the organization, propels in data security become essential to ensure secrecy and protection. Cryptographic calculations assume a crucial part in data security. The various calculations utilized are AES and DES to scramble and decode information .The most utilized encryption calculations incorporate AES and DES. These calculations can be utilized by somebody who needs to secure a lot of data by making an information document that can be encoded or decoded with the AES or DES calculation. There was an audit of AES, DES, and Blowfish by Ashwin Kumar and K.S. Sandha with respect to security and force use and why AES has favored execution over some different records. More relationships are made among AES and DES, so it is ideal to give better synchronization of more secure substances (Thuraisingham Bhavani, 2007). It likewise shows that the two math measures take diverse time contingent upon the gadget. Encoding whole information records can be a down to earth approach to ensure a lot of information. Notwithstanding, clump record encryption can be insufficient and lumbering as it is absurd to expect to get to a particular segment of the encoded information in the document. Regardless of whether the application just requirements admittance to a specific bit of information, the whole record should be unscrambled. Without the capacity to interpret a segment of a record, it is hard to plan an information preparing framework that can give an alternate degree of information admittance to various applications.

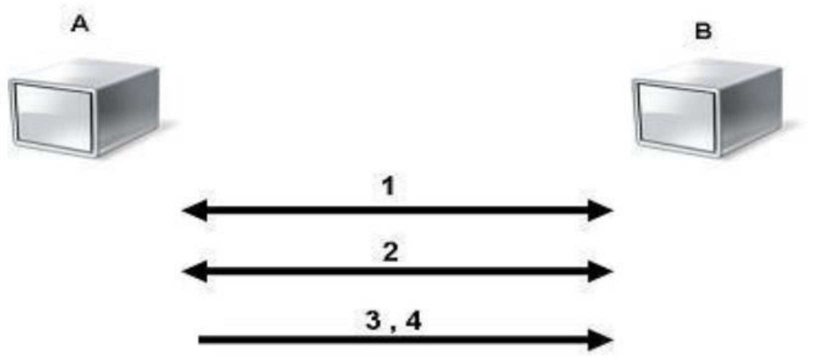
2.5 Symmetrical Key Cryptosystems

All exemplary cryptographic frameworks created preceding 1970 are instances of symmetric expert code frameworks. Likewise, the majority of the cryptographic frameworks created after 1970 are the equivalent (Andrei Toma, 2015). Incorporate some regular instances of current symmetric key:

1. Advanced Standard Encryption AES

2. Standard for Data Encryption

All symmetric keys have average interests, as they depend on an ordinary riddle between interconnected gatherings. The riddle can be utilized as an encryption and unscrambling key. The hindrance of a balanced trade is that you can't manage a huge trade association. Then again, symmetric key requires more unobtrusive size to have a level of security similar to that of open code systems, making the connection quicker and more unassuming (see Fig. 2.11):

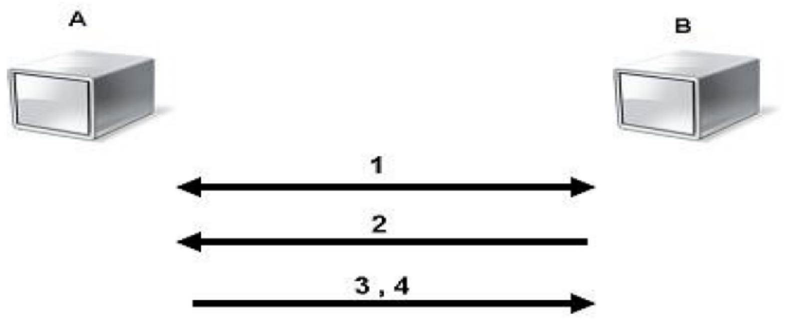


1. Cryptosystems A and B accept.
2. A and B plan to use the key.
3. A message with the shared key is encrypted.
4. B decrypts the code with the mutual key.

Figure 2.11 Symmetric encryption

2.5.1 Public Key Cryptosystems

In this sort of cryptography system, there are two separate keys: a public key, known to all, and a dark key, known distinctly to the proprietor. This sort of structure is known as "topsy turvy" in light of the fact that it utilizes a substitute key for unscrambling and encryption (public key and private key). The data is scrambled with a public key and should be encoded with the private key. (see Fig. 2.12):



1. Cryptosystems A and B accept.
2. B sends it to the general public.
3. A message is encrypted with the agreed cipher and public key B.
4. B decrypts the letter by using the private key and the negotiated chip..

Figure 2.12. Asymmetric encryption

2.5.2 Rijndael Algorithm

The Rijndael account has been chosen by the US National Institute of Standards and Technology (NIST) as a proposed AES account. This figuring was arranged by two Belgian cryptologists, Dr. Vincent Regmen and Dr. Joan Damon. Rijndael is a square token and is the most mainstream sort of private key calculation (Rijmen, 2002). The essential clarifications behind NIST for picking this record are:

1. The parallel and symmetrical structure

2. There is a lot of versatility for implementers

3. It has not allowed successful attacks

4. It is suited to specialized processors.

5. Pentium

6. RISC and processors in parallel

7. They are perfect for intelligent cards

8. It is for special hardware versatile

Regardless of these reasons, the count can be performed effectively on countless processors and gadgets. In contrast to other people, it has a short encoding and de-trap time and gives the best presentation to both the gear and programming utilized.

2.6 Specification of Algorithms

Rijndael is a rehashing ace box image that contains block length and key length, for instance, both key and square size should be 128, 150, 192, 224, or 256 pieces. The differentiation among Rijndael and AES is the scope of characteristics called for to encode and obstruct length. AES limits block length to 128 and underpins key length 128, 192 or 256 pieces. The additional square and Rijndael key length are not safeguarded in the current FIPS standard as they were not assessed during the AES political decision cycle.

2.6.1 Byte and State of Rijndael

AES (Rijndael) info and yield is a line of 128 pieces, which is the size of the code block and the key length can be looked over 128, 192 or 256 pieces. A Rijndael byte can be characterized as: a bunch of 8-bit successions speaking to a limited area component. Utilizing polynomial documentation, byte "b" can be spoken to as follows:

$$b_7x^7+b_6x^6+b_4x^4+b_3x^3+b_2x^2+b_1x^1+b_0x^0$$

Equation 2.1. Byte and State of Rijndael

Where; b7, b6, b5, b4, b3, b2, b1, b0 are either 0 or 1 given that Rijndael uses a finite field.

2.6.2 The Rounds

Rijndael consists of a variable number of rounds depending on the block of cipher and the length of the key, as follows:

i. 10 is 12b bit long for both block and key length.

ii. 12 if any one of the block or keys is 192 bits, but none of them is larger than 192 bits.

iii. 14 If there is a 256-bit block or key.

Rijndael is a calculation that comprises of adding an underlying key, an Add Round Keys step, which is trailed by routine rounds, however the last round skirts the Merge Columns step. Each round is a four-byte change succession known as steps. So when utilizing Rijndael encryption calculation the means included are:

A) Substitution by Byte

Every 8-cycle byte is in reverse planned to another byte. Every byte in the square is supplanted by its substitute in the S-box (a square that maps the "n" digit to the "m" bit). As it is the lone piece of the nonlinear coding, it is viewed as the main piece of the calculation. As in the figure (2.13)

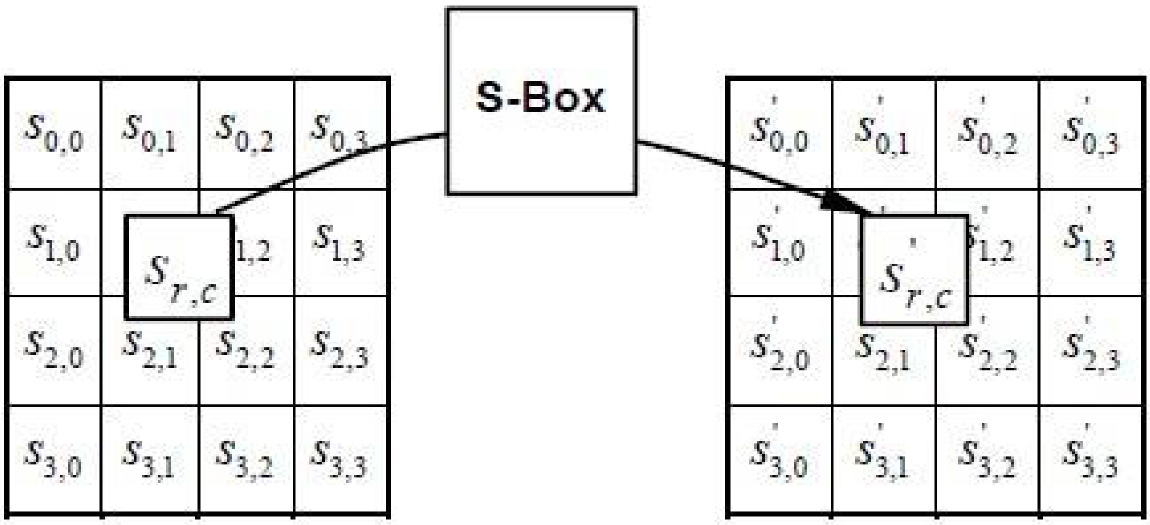

S-Box
$S_{0,0}$ $S_{0,1}$ $S_{0,2}$ $S_{0,3}$
$S_{1,0}$ $S_{r,c}$ $S_{1,2}$ $S_{1,3}$
$S_{2,0}$ $S_{2,1}$ $S_{2,2}$ $S_{2,3}$
$S_{3,0}$ $S_{3,1}$ $S_{3,2}$ $S_{3,3}$
$S'_{0,0}$ $S'_{0,1}$ $S'_{0,2}$ $S'_{0,3}$
$S'_{1,0}$ $S'_{r,c}$ $S'_{1,2}$ $S'_{1,3}$
$S'_{2,0}$ $S'_{2,1}$ $S'_{2,2}$ $S'_{2,3}$
$S'_{3,0}$ $S'_{3,1}$ $S'_{3,2}$ $S'_{3,3}$

Figure 2.13. Substitution by Byte

B) Shift Rings

In this progression, the lines are looked in four changed manners. Evaluation 0 not contacted; Row 1, Row 2 and Row 3 are turned to one side separately. In Shift Rows Transform, the byte in the last three columns is occasionally moved by an alternate number of bytes from the primary line, r = 0 without contacting. The chart in Fig. 2.14 Shows the Shift rows pass the last three rows in the state cyclically .

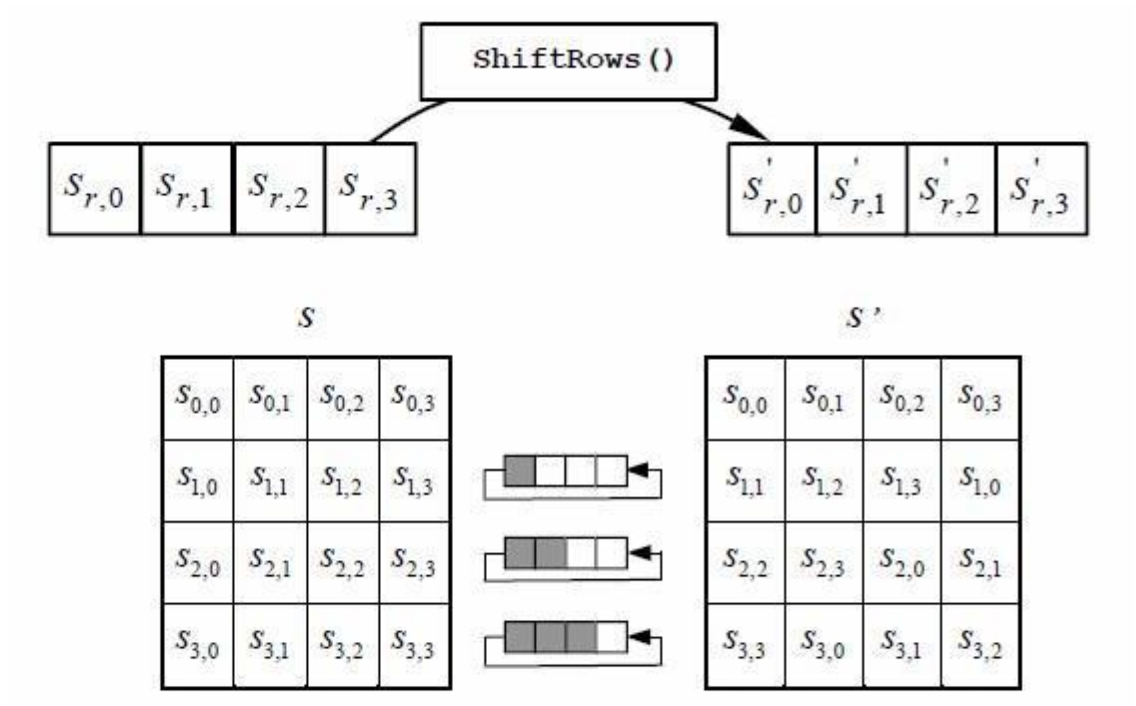

ShiftRows()
$s_{r,0}$ $s_{r,1}$ $s_{r,2}$ $s_{r,3}$
$s'_{r,0}$ $s'_{r,1}$ $s'_{r,2}$ $s'_{r,3}$
s
s'
$s_{0,0}$ $s_{0,1}$ $s_{0,2}$ $s_{0,3}$
$s_{1,0}$ $s_{1,1}$ $s_{1,2}$ $s_{1,3}$
$s_{2,0}$ $s_{2,1}$ $s_{2,2}$ $s_{2,3}$
$s_{3,0}$ $s_{3,1}$ $s_{3,2}$ $s_{3,3}$
$s_{0,0}$ $s_{0,1}$ $s_{0,2}$ $s_{0,3}$
$s_{1,1}$ $s_{1,2}$ $s_{1,3}$ $s_{1,0}$
$s_{2,2}$ $s_{2,3}$ $s_{2,0}$ $s_{2,1}$
$s_{3,3}$ $s_{3,0}$ $s_{3,1}$ $s_{3,2}$

Figure 2.14. Shift rows pass the last three rows in the state cyclically

C) Columns Mix

In that method, bytes are combined linearly in columns after the matrix is multiplied. The chart in Fig. 2.15 Shows the Mix column matrix transformation .

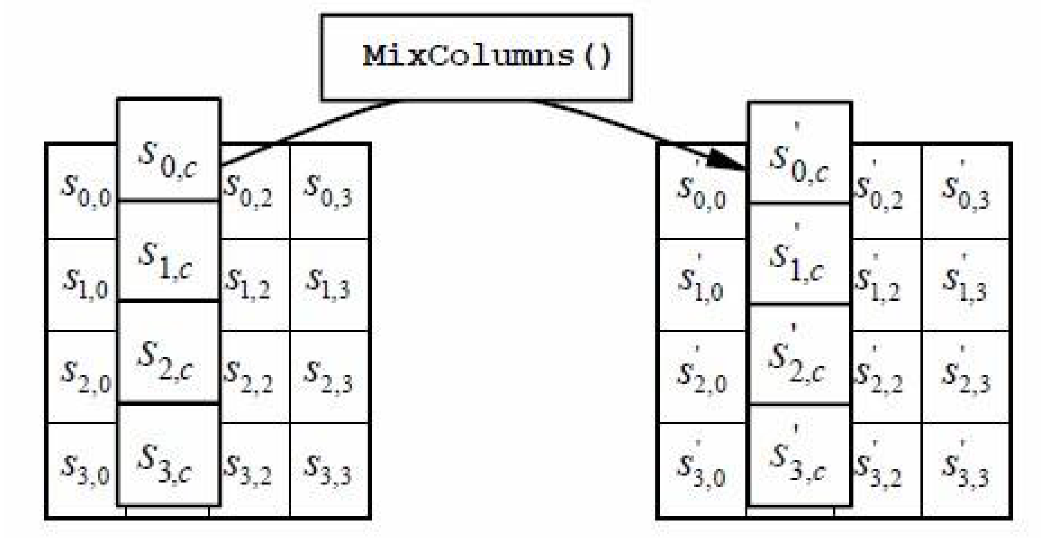

MixColumns()
$s_{0,0}$ $s_{0,c}$ $s_{0,2}$ $s_{0,3}$
$s_{1,0}$ $s_{1,c}$ $s_{1,2}$ $s_{1,3}$
$s_{2,0}$ $s_{2,c}$ $s_{2,2}$ $s_{2,3}$
$s_{3,0}$ $s_{3,c}$ $s_{3,2}$ $s_{3,3}$
$s'_{0,0}$ $s'_{0,c}$ $s'_{0,2}$ $s'_{0,3}$
$s'_{1,0}$ $s'_{1,c}$ $s'_{1,2}$ $s'_{1,3}$
$s'_{2,0}$ $s'_{2,c}$ $s'_{2,2}$ $s'_{2,3}$
$s'_{3,0}$ $s'_{3,c}$ $s'_{3,2}$ $s'_{3,3}$

Figure 2.15. Mix column matrix transformation

D) Add to Round Keys

In this last stage, the subkey is added by consolidating every byte with the byte relating to the subkey with XOR (a coherent activity gives a genuine yield when the two sources of info are unique). (See Figure 2.16)

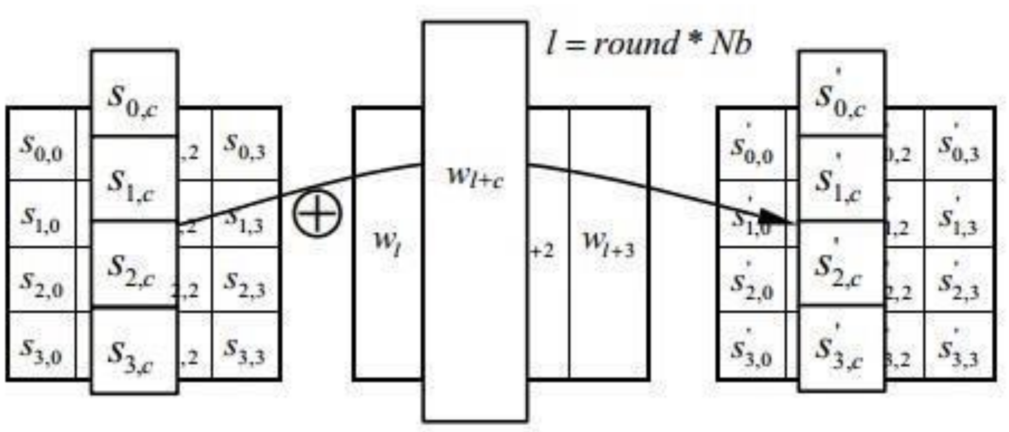


Figure 2.16 XORs each State column with a term from the main schedule Add Round Sey ().

2.7 Schedule of Keys

Round keys with key writing computer programs are gotten from the code key. Next, the AES calculation grows the key with the encryption key, takes the encryption key, and extends it to create the complete Nb (Nr + 1) 32-bit words (where, Nb is the all out section number, for AES the worth is 4 and Nr is the absolute number For adjusts, for example 10, 12 or 14). Each Rijndael round that contains Add Round Keys requires a bunch of four 32-digit catchphrase information. The key development calculation expands the information encryption key by changing the primary Nk words with the info encryption key (where, Nk is the quantity of 32-cycle capacities in the AES figure key, the qualities are 4, 6 or 8). At that point each next word w [i] is equivalent to OR restrictive to the past word w [i −1], and Nk is w [i− Nk]. For words in places that are products of Nk, first the past word w [i - Nk] is turned one byte to one side, at that point its bytes are changed over utilizing Sbox for the byte replacement step and afterward a selective OR-ed with a consistent dependent on adjusting Before utilizing OR with w [i - Nk].

2.8 Comparison of the AES and DES

AES and Triple DES (TDES or 3DES) are the two most normally utilized square codes. By plan, AES is quicker by cycles, which implies that exchanging between gadgets is less complex than exchanging between programs. Nonetheless, DES encodes information in 64 pieces and uses a 56-bit key, so it has an inexact likelihood of 72 billion (Joseph Zambrino, 2015). In spite of the fact that the number is tremendous, given the registering intensity of the present innovation, it isn't sufficient and could be assaulted. Hence, because of DES's failure to stay up with mechanical turns of events, it's anything but a sufficient security. Because of the broad utilization of DES, the speediest arrangement was to move up to 3DES, which is protected enough for the present innovation. The Rijndael calculation was picked to supplant the 3DES. The primary explanation behind picking Rijndael as the new age of AES is:

i. Safety: security

ii. Performance of Hardware and Applications.

iii. Suitable for space-restricted environments.

iv. power research opposition and other assaults on execution.

AES works immediately even on little electronic gadgets and gadgets (cell phones and brilliant cards). It can give greater security since it has a bigger and longer key and square size. It likewise utilizes a 128-cycle block size and works with 128,192 and 256-bit keys. Rijndael is adaptable and can work with any key and square size various of 32 pieces and ranges somewhere in the range of 128 and 256 pieces. AES is an option in contrast to 3DES, and both encryption keys will coincide until 2030 for a smooth, steady change. A point by point examination among AES and DES is appeared in the accompanying table:

Table 2.1 AES and DES contrasts

Factors	**AES**	**DES**
Key length	128, 192 or 256 bits	56 bits
Cipher type	Symmetric block cipher	Symmetric block cipher
Block size	128, 192, or 256 Bits	64 bits
Cryptanalysis resistance	Strong against differential, linear, interpolation and square attacks.	Vulnerable to differential and linear cryptanalysis; weak substitution tables
Security	Considered secure	Proven inadequate
Possible keys	2128,2192 or 2256	256
Times required checking all possible keys 50 billion Keys per second	For a 128-bit key 5 x 10^{21} years	For a 56-bit 400 days
Rounds	10,12 and 14 for 128, 192 And 256-bits respectively	16

2.9 Classification of Cryptography Methods

The high development of organization innovation is prompting a radically shared culture of information trade. Consequently, it is bound to be copied and reallocated by programmers. Hence, data should be secured while it is being communicated. There are a few encryption procedures that are utilized to forestall data robbery. It incorporates different encryption strategies used to secure delicate information:

2.9.1 Optical Encryption

Optical encoder utilizes optical devices to construct actual picture encoding frameworks, which depend principally on optics to change the recurrence segment of a picture. Shading pictures are first changed over to filed picture arranges prior to being encoded. During this encoding cycle, the picture is encoded in consistent repetitive sound covers of two stages: one at the information level and the other in the Fourier level. During the time spent unraveling, the shading picture can be reestablished by changing the listed picture back to RGB setup. This technique is more remarkable and incredible than some other strategy right now.

2.9.2 Selective Encryption

Basically, specific encryption expects to keep all advanced picture bits from being encoded while giving secure encryption simultaneously. The central issue is just to encode a little segment for quick encryption. Particular encryption incorporates five strategies:

i. Methods based on DCT

ii. Methods based on Fourier

iii. Methods of scanning

iv. Methods of Chaos

v. Methods based on quad-tree

These strategies to fulfill application necessities should be quick while keeping up the pressure proportion as it was initially. A few distinct strategies have been proposed as coding for DCT based techniques. The strategies known as crisscross were produced utilizing Tang calculation, another calculation from Qiao and Nahrstedits dependent on the recurrence circulation of two contiguous bytes in the MPEG bit stream. This strategy offers a few favorable circumstances: adaptability, lavishness, specific measurements, and adaptability of coordination.The chart in Fig. 2.17 Shows the Mechanism for selective encryption .

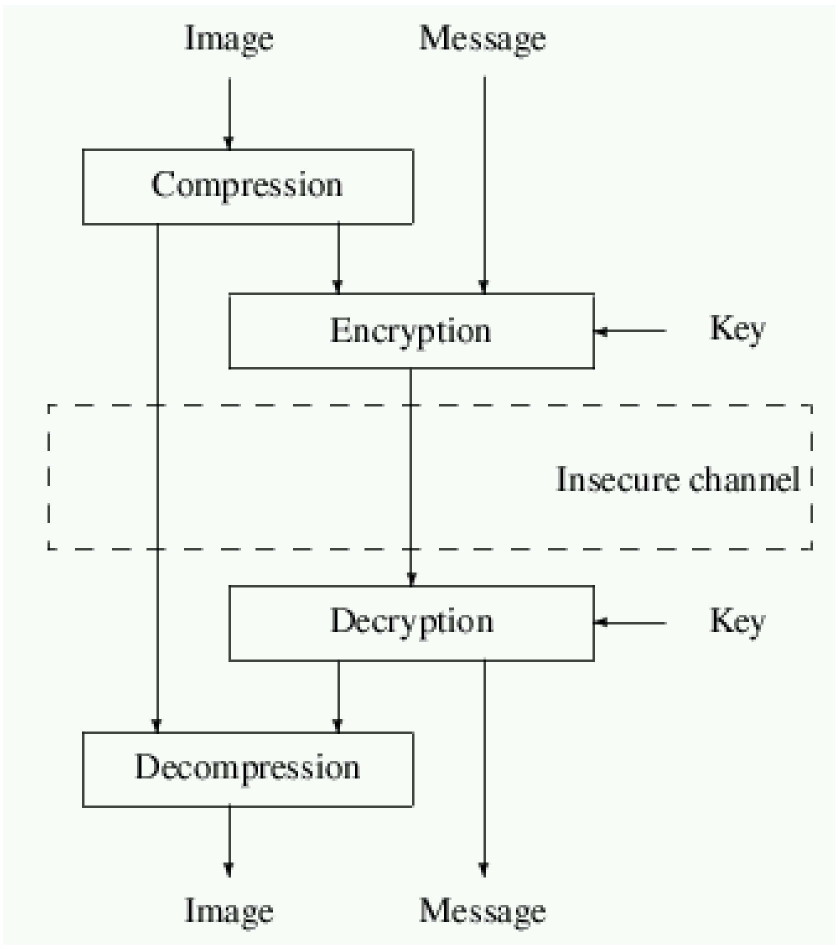


Figure 2.17. Mechanism for selective encryption

2.9.3 Chaotic Encryption

Chaotic codec is described by high affectability to its underlying qualities, just as to its boundaries, blending property, and strength. Ergodicity is a cycle where each progressive example is similarly illustrative of the entirety. This technique is one of the great contender for coding. The fundamental reason for this sort of encryption is picking the best muddled guide for some coding ventures (Joseph Zambrino, 2015). The disorderly guide utilized in the coding cycle ought to have the accompanying attributes:

<u>Property Mixing</u>

The mixing property is identified with the disperse property of the calculation. Simply envision that an ordinary content gathering has a prime locale in the stage space of the guide, the mixing property is the thing that implies the scattering of one plain content over a large number of the coded text.

<u>Rugged Chaos</u>

The favored calculation should have the option to disseminate the impact of a solitary primary number over many coded numbers. The vital alludes to the system of the cryptographic calculation. So we need to stress over transformation, including boundaries and variable.

<u>Set of Parameters</u>

The favored calculation should have the option to disseminate the impact of a solitary primary number over many coded numbers. The vital alludes to the system of the cryptographic calculation. So we need to stress over transformation, including boundaries and variable.

2.9.4 Nonchaotic Encryption

We have proposed encoding pictures utilizing Sudoku Matrix. Picture encoding as per this framework comprises of three phases. During the principal stage, a sudoku reference network is made and utilized in the coding cycle. At that point the pixel force of the picture is changed utilizing Sudoku reference framework esteems lastly the pixel esteem is modified utilizing a similar Sudoku grid as the planning cycle. So with the assistance of this grid, we can encode any advanced picture (parallel pictures, grayscale pictures, and RGB pictures) (Ching-Yung Lin, 2006).

The strategic guide is utilized to control the size of the Sudoku lattice. We have sent you an ideal Latin square picture encoder, which gives a 256-digit key length for creating a latin square and along these lines delivers a 256 x 256 square picture like a sudoku network. That is, without two numbers in a similar square, they can be adjusted in a similar line, segment or square. With LSIC, numerous alluring solid encryption properties can be accomplished, including:

i. Wide space in key
ii. High key sensibilities
iii. Cipher text spread uniformly
iv. Excellent confusion and dissemination
v. Semantically reliable

At that point, two new calculations dependent on the Fibonacci code were presented: one for the spatial space and the other for the recurrence area (otherwise called the JPEG space). The security key for picture encoding calculations is:

- Setting Parameter (p and i)
- Size of the original picture

There are a ton of opportunities for security keys, which makes it hard for unapproved clients to hack the encoded picture, along these lines making it safer.

2.10 Visual Cryptography

Visual encryption utilizes the qualities of an individual's capacity to translate a scrambled picture. It doesn't need information on cryptography or troublesome computations. Continue ensuring that programmers can't get any mystery photograph thoughts from a solitary cover photograph.

This framework was presented utilizing a Hilbert bend and two tails, by Sen Jin Lin, Ja Chen Lin, and Wen-Bin Fang. A Hilbert bend was utilized to decrease the issue of the 2D picture to a 2D arrangement while the white tail was utilized to total enough white pixels ('m' white pixels) for every one of these pixels' m 'whites can just utilize a grid rather than the network' m ''. Dark paste is utilized for a comparative reason likewise. Stacking shadows brings about an excellent picture. In any case, C. Chang, C. Tsai Wat. Chen recommended that the visual coding plan should uphold enormous picture organizations, for example, shading and grayscale. They additionally added that arbitrary looking activities seem untrustworthy, which opens them to assaults in the center, to maintain a strategic distance from such assaults, just essential activities should be performed. Parallel coding was utilized by Y.C. He, F Lin, C.Y. Change to speak to the predetermined subpixels of each square, at that point arbitrarily AND/OR to ascertain the parallel image of the stacked sub pixel for the whole square in the coat picture. The image can go from 0 to 255, however it can even be estimated relying upon the factor. Consequently, the mystery photograph can be 256 tones or genuine nature (Sonia Vatta, 2013). The chart in Fig. 2.18 Shows the Example of visual cryptography .

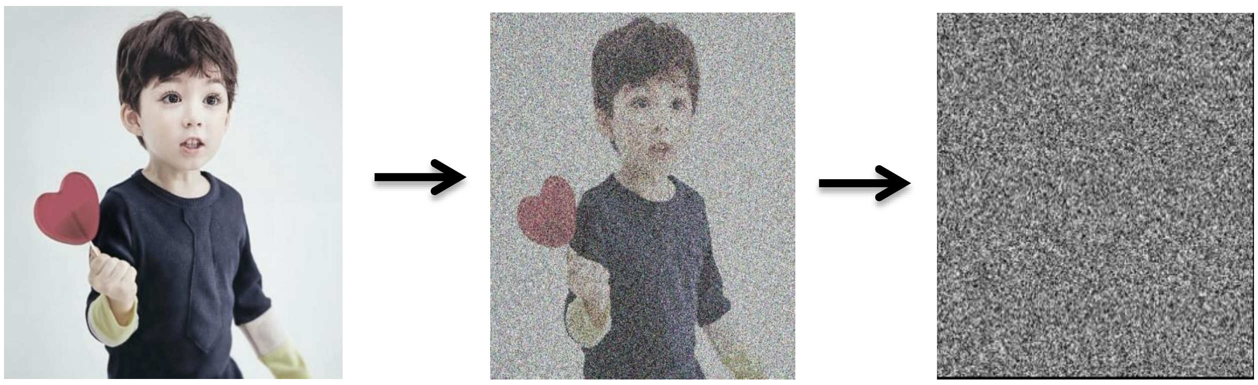


Figure 2.18. Example of visual cryptography

2.11 Analysis of Security

Security is a significant issue in cryptography. A decent picture encoding plan ought to withstand different assaults, for example, realized plain content assault, figure just content assault, factual examination assault, and savage power assaults. Here are some security breaks down:

2.11.1 Exhaustive Search of The Key

Solid encryption calculation should be unified with huge key space. The bigger the key zone, the less possibility of assaulting the encoded plan. Assume the calculation contains a K-piece key, at that point a thorough key hunt requires 2K endeavors to break the key.

2.11.2 Key Sensibility Analysis

The way toward encoding the ideal picture should be drawn nearer delicately with a mystery key, which implies that a solitary piece change in the mystery key delivers a totally extraordinary scrambled picture (Khaled F. Hussain, 2011). This sort of affectability can be managed regarding two angles:

i. Encryption Process

Two distinctive ciphertext pictures, C1 and C2, were produced against a similar plain book picture utilizing the K1 and K2 figure keys, with just the slightest bit contrast between the two keys.

ii. Decoding

Two diverse decoded pictures, D1 and D2, were produced against the equivalent ciphertext picture utilizing two code keys, K1 and K2, with just the slightest bit distinction between the two keys.

2.11.3 Histogram Analysis

This is one of the most unique approaches to represent the nature of the encoded picture, as the excellent picture encoding strategy encodes an irregular plain book picture, so it is important to show the dispersed diagram. Consistently to encoded text picture.

2.11.4 Information Entropy Tests

Despite the fact that histogram examination is the most sensible investigation, as it shows how pixels can be circulated equitably as ciphertext, you actually have an issue, which demonstrates how well or ineffectively dispersed the chart is. Data entropy can be characterized as the critical proportion of the irregularity of a sign source.

$$(X) = -\sum_{i=1}^{n} \Pr(x_i)\, log_2 \Pr(x_i)$$

$$\Pr(X = x_i) = \frac{1}{F}$$

Equation 2.2 Information entropy equation

The information entropy can be used to measure the picture randomness seen in the above equation.

Where;

X= test image Xi =possible value in X

Pr (x_i) = Random pixel chance in X with Xi value

H(x) = achieved when F = amount of intensity scale allowed for the image format X is equally distributed.

2.11.5 Peak Signal-to-Noise Ratio (PSNR)

In signal processing the term peak signal-to-noise ration (PSNR) is generally used to determine the ratio between the maximum possible power of a signal (referred as valuation valuations of a signal s) and the power of corrupting noise that affects the fidelity of the signal representation. With the PSNR as quality measure, the similarity between two signals, for example the unencrypted and the selectively encrypted smart meter data, can be determined. Usually the PSNR is expressed on a logarithmic decibel scale(Kotevski and Mitrevski, 2010).

For example, the PSNR between the unencrypted signal orig and the selective encrypted signal enc is calculated by (Mallat, 2009) and (Wang and Bovik, 2009):

$$\text{PSNR(orig,enc)} = 10 \, . \log_{10} \frac{\text{valuation}}{\text{MSE (orig , enc)}} \{\text{dB}\}$$

The mean-squared error (MSE), as part of equation 6.1, provides a quantitive score to describe the degree of similarity between two signals (Wang and Bovik, 2009). The MSE is defined by (Wang and Bovik, 2009):

$$\text{MSE(orig,enc)} = \frac{1}{N} \sum_{i=1}^{N} (\text{orig}_i - \text{enc}_i)^2$$

As discussed by Wang and Bovik (2009), the use of the MSE as signal fidelity measure has the following advantages:

- **Simplicity:** The computation of MSE is performed parameter free and only by one multiply and two additions per sample.

• **Constant and Direct inTerpretations of Similarity:** All norms of two signals are described by non-negative values. Furthermore, the difference between two signals equals zero if and only if all corresponding samples of the signals are equal.

• **Clear Physical Meaning:** Even after performing a transform to a signal, the energy of a signal distortion in the transform domain is still the same as in the signal domain.

• **Excellent Metric for Optimization:** With its properties of convexity, symmetry and differentiability, MSE represents an excellent metric in the context of optimization.

• **Trust in Convention:** Historically the MSE has been used for a wide variety of signal processing applications. E.g., in signal compression, restoration, denoting, reconstruction, etc.

According to Kotevski and Mitrevski (2010), theoretically the highest PSNR would be 100 dB, but in reality the estimated value for image processing lies between 30 dB and 40 dB.

2.11.6 Coefficient of Correlation Analysis

It is this factor that decides the quantity of similitudes between two factors to one another. This is normally used to quantify the nature of encryption for a wide range of encryption. The connection coefficient (rxy) can be determined as follows:

$$R_{xy} = \frac{COV\ (X,Y\ )}{\sigma x \sigma y}$$

$$\sigma_x = \sqrt{VAR\ (\ X\ )}$$

$$\sigma_y = \sqrt{VAR\ (\ y\ )}$$

$$\mathrm{VAR}\ (x) = \frac{1}{N} \sum_{i=1}^{N} (x_i - E(x))^2$$

$$\mathrm{VAR}\ (y) = \frac{1}{N} \sum_{i=1}^{N} (y_i - E(y))^2$$

$$\mathrm{COV}(x\ ,\ y)\ = \frac{1}{N} \sum_{i=1}^{N} (x_i - E(x))\ (y_i - E\ (y))$$

Equation 2.3 Coefficient of correlation equation

Where; x and y = the two pixel esteems at similar area in the first and encoded pictures Cov (x, y) = the fluctuation in these pixels VAR (x) and VAR (y) = the difference estimations of the x and y pixel esteems in both the first and encoded picture; Xσ and yσ = standard deviations of pixel x and y esteem;

E = Waiting for an operator ; N = Dimension of the image.

CHAPTER THREE
Methodology and the Proposed Model

CHAPTER THREE
Methodology and the Proposed Model

3.1 Methodology Approach

The methodology approach adopted in the thesis work is based on problem solving and experimental evaluation. The proposed model is designed and implemented to provide the security protection for multimedia files of various types through light weight encryption to make it possible to encrypt multi-megabyte files without the complexity of traditional encryption algorithms while at the same time providing full encryption of all elements of the encrypted file. The experimental work will test the implemented model using compressed multimedia files, in particular compressed video of realistic sizes that are used exchanged over the internet.

3.2 Outline of the Proposed Model

The proposed model is designed and implemented as a symmetric encryption system using a common secret key, to provide full-encryption of the plaintext multimedia file where each byte of the file, including the header blocks, are encrypted, in order to have a comprehensive protection of the entire file.

Due to the large size of multimedia files, in particular video files, the designed encryption model will avoid complex computation that can take an extensive time, by employing Byte replacement steps using pseudo random data combined with the original bytes. To compensate for the simple computational steps, the proposed model is designed to have a large numeric secret key which will provide strong security against brute force attacks. Furthermore, the model will deal with compressed and uncompressed multimedia files.

3.3 Design Guidelines of the Proposed Model

The proposed encryption model is designed to meet three essential criteria:

- Reduced computing steps to achieve light weight encryption.
- Encryption of every byte of the plaintext file, including header data, to achieve high security.
- Symmetric secret key with a large key space.

The following design guidelines are taken into consideration:

1- The plaintext multimedia file including the header data are processed as a stream of bytes regardless of the type of file.

2- Both compressed and un-compressed multimedia files are processed the same way.

3- Encrypt every byte of the plaintext multimedia file.

4- The byte level encryption algorithm should utilize random numbers generated using a pseudo random number generator.

5- A numeric secret key is used as a seed in the random generation process.

6- The encrypted ciphertext file should be of the same size as the plaintext file.

7- The decryption process follows an inverse of the encryption process and should result in a multimedia file of the same type and size as of the plaintext multimedia file.

3.4 Implementation Guidelines

The proposed model should be implemented using multimedia-oriented programming language such as Python or MATLAB, which provide file handling features to deal with a wide range of multimedia file types. Implementation of the proposed model is to be in Microsoft Windows environment, with the possibility of future an alternative version on a mobile device environment such as Android.

The numeric secret key should be of the integer type and should have a maximum value that is limited by the maximum value of the integer type in the implementation language, such as 16 decimal digits.

3.5 Testing and Evaluation Issues

The implemented system should restore the plaintext multimedia file from the encrypted ciphertext file without any changes to the contents or format as compared with the plaintext file. To test this criterion, the original plaintext file and the decrypted plaintext file should be compared at byte-level and should have no differences. Also, the decrypted plaintext file should have the same viewing features, a video file should be played and take the same viewing time as the original file.

Performance efficiency evaluation of the implemented system in terms of encryption / decryption time should be measured per file size unit such as KiloByte (KB).

The testing and evaluation should be done using an arbitrary secret key within the range of possible key values.

3.6 The Multimedia Encryption / Decryption Algorithms

3.6.1 The Multimedia Encryption Algorithm

The process of encrypting a multimedia file follows the following algorithmic steps:

1. Read the plaintext multimedia file (header and data blocks) and convert it to an array of bytes (Plain-Byte)

2. Calculate the size of the Plain-Byte array and store it in Byte-Count

3. Create an array to store the cipher bytes (Cipher-Byte) with size = Byte-Count

4. Read the secret key that was generated in the key generation process and store it in SK

```
5. Set the seed of the random number generator to SK

6. Set the loop counter (i) to 1

7. While i < Byte-Count do

Generate a random number with value range 0..255 and store it in Rand-Byte
      If Rand-Byte >= Plain-Byte(i)
            Cipher-Byte(i) = Rand-Byte – Plain-Byte(i)
      else
            Cipher-Byte(i) = 256 – Plain-Byte(i) + Rand-Byte
      End If
      i = i + 1
  End While

8. Save the Cipher-Byte array in a ciphertext file with a "bin" format type.
```

3.6.2 The Multimedia Decryption Algorithm

The process of decrypting an encrypted multimedia file follows the following algorithmic steps:

```
1. Read the cipher text multimedia file (header and data blocks) and convert it to an
array of bytes (Cipher-Byte)

2. Calculate the size of the Cipher-Byte array and store it in Byte-Count

3. Create an array to store the decrypted plaintext bytes (Decrypt-Byte) with size =
Byte-Count

4. Read the secret key which was generated in the key generation process and store it in
SK

5. Set the seed of the random number generator to SK

6. Set the loop counter (i) to 1

7. While i < Byte-Count do

    Generate a random number with a value range (0..255) and store it in Rand-Byte
```

```
        If Cipher-Byte(i) > Rand-Byte
                Decrypt-Byte(i) = 256 - Cipher-Byte(i) + Rand-Byte
        else
                Decrypt-Byte(i) = Rand-Byte – Cipher-Byte(i)
        End if
    i = i + 1
    End While
```

7. Save the Decrypt-Byte array in a file with the same format as the plaintext file.

3.6.3 The Key Generation Algorithm

In symmetric encryption, there is only one key which is the secret key. In the proposed model, the secret key will be generated automatically rather than being selected manually by the user, which is not practical for a large key. The secret key is designed to consist of a fixed number of decimal digits, and it will be randomly generated using a pseudo random number generator. The secret key will be saved in a file (Key-File), and the same file will be read by the encryption and decryption algorithms.

CHAPTER FOUR
Implementation and Experimental Results

CHAPTER FOUR
Implementation and Experimental Results

4.1 Introduction

This chapter presents implementation of the proposed multimedia encryption model and the results of the experimental work a using public dataset of multimedia files. The chapter is organized as follows: section 4.2 presents an overview of the implementation; section 4.3 provides a description of the dataset that was used in the experimental work; section 4.4 presents a description of implemented multimedia encryption / description system; and section 4.5 presents the experimental work and discussion of results.

4.2 Implementation Overview

The aim of the implementation phase is to construct a working system for the encryption and decryption of multimedia files based on the proposed algorithms presented in chapter three, and to conduct experimental work using a public dataset.

The proposed encryption and decryption algorithms were implemented in the MATLAB language version 2016a, to take advantage of the powerful programming facilities that deals with various multimedia files. The two algorithms are implemented as two separate programs; the encryption program which gets as input a numeric secret key, encrypts every byte of the multimedia file including the header blocks, using the generated random numbers, and outputs the encrypted file which has the same size as the original plaintext file. The decryption program reads the encrypted file and reverses the encryption process, using the same secret key which is used for encryption, and the same sequence of random numbers, and outputs the decrypted file. The two programs invoke the pseudo random number generator function for encryption and decryption, using the secret key as a seed.

4.3 The Dataset

This section describes the dataset of sample files that were used in testing the implemented system. We have chosen to use video files for testing the system for several reasons: they are larger and more versatile than static images, which provide richer testing options; private video files are widely exchanged in personal and business communications; and most importantly encryption of streaming video is highly needed in the entertainment and internet industries.

For the purpose of this research, video files from a public source (Big list of sample videos for testers, 2020) which contains a wide range of file types and sizes. The MP4 video file type was chosen for testing as it is compressed, and it is highly used in popular movies' libraries such as YOUTUBE, multimedia applications and in streaming videos. It is worth noting that the implemented system can process any type of multimedia file that is supported by the MATLAB language regardless whether it is compressed or uncompressed.

Table 4.1 shows the selected sample of video files that were used in the experimental work.

Table 4.1: Dataset of video test files

#	Format	Title	Dimensions	File Size (MB)
1	MP4	DLP_PART_2_768k	400 × 300	5.26
2	MP4	Small	560 × 320	0.17
3	MP4	jellyfish-25-mbps-hd-hevc	1920 × 1080	31.30
4	MP4	lion-sample	384 × 288	10.27
5	MP4	page18-movie-4	480 × 270	27.80
6	MP4	metaxas-keller-Bell	176x98	20.31
7	MP4	Panasonic_HDC_TM_700_P_50i_2	1920 × 1080	25.19
8	MP4	star_trails	1280 × 720	37.83
9	MP4	TRA3106	720x496	6.77
10	MP4	Dolbycanyon	720x480	3.01

4.4 The Multimedia Encryption / Decryption System

The implemented multimedia encryption system (MEDS) consists of two main programs as follows:

a. The Encryption Program:

This program implements the encryption algorithm presented in chapter three; it works as follows:

- Read the plaintext multimedia file to be encrypted, including header data, and store it in a one-dimensional array of bytes.

- Read the numeric secret key from the Secret Key file.

- Seed the random number generator using the secret key.

- Create an empty one-dimensional array of bytes (the encrypted array) with the same size in bytes as the input file array, to store the encrypted bytes.

- Encrypt every byte of the plaintext file using the next random number which is between 0 and 255 and store the encrypted bytes in the encrypted array.

- Create an output ciphertext file with the “.bin” format type and save the encrypted array into this file (the ciphertext file should have the same file size as the plaintext file).

b. The Decryption Program:

This program implements the decryption algorithm presented in chapter three; it works as follows:

- Read the ciphertext multimedia file to be decrypted, including header data, and store it in a one-dimensional array of encrypted bytes.

- Read the numeric secret key from the Secret Key file that was used in the encryption phase.

- Seed the random number generator using the secret key.

- Create an empty one-dimensional array of bytes (the decrypted array) with the same size in bytes as the encrypted array, to store the decrypted bytes.

- Decrypt every byte of the encrypted array using the next random number which is between 0 and 255 and store the decrypted bytes in the decrypted array.

- Create an output file with the same format type as the plaintext file and save the decrypted array into this file (the decrypted file should have the same size as the ciphertext file).

4.5 Experimental Work and Discussion of Results

The implemented multimedia encryption system (MEDS) was executed on a Laptop computer with intel i5 CPU and 8GB of memory. The experimental work involved evaluating performance efficiency, integrity evaluation of the encryption / decryption processes, visual cryptography results, and statistical distortion results. The MEDS system was used in encrypting / decrypting the dataset files presented in section 4.1. For each file, the elapsed processing time in seconds was calculated during the encryption and decryption processes.

Table 4.2 presents the experimental results, showing the processing time in seconds of the encryption and decryption processes for each of the selected files, as well as showing the throughput in KiloBytes (KB) per second.

Table 4.2 Encryption throughput experimental results

No	File Name	Size MB	Encryption Time (Second)	Encryption Throughput (KB / Second)	Decryption Time (Second)	Decryption Throughput (KB / Second)
1	DLP_PART_2_768k	5.26	4.10	1316.04	4.10	1313.45
2	Small	0.17	0.16	1119.76	0.16	1108.83
3	jellyfish-25-mbps-hd-hevc	31.3	24.18	1325.53	24.58	1304.09
4	lion-sample	10.27	8.02	1311.42	8.03	1309.89
5	page18-movie-4	27.8	21.58	1319.55	21.66	1314.34
6	metaxas-keller-Bell	20.31	19.29	1337.32	19.63	1314.12
7	Panasonic_HDC_TM_700_P_50i	25.19	29.06	1332.74	29.47	1314.24
8	star_trails	37.83	15.76	1319.13	15.79	1316.58
9	P6090053	6.77	1.91	1292.26	1.94	1266.56
10	Dolbycanyon	3.01	2.36	1304.11	2.36	1305.56
	AVERAGE	-	12.64	1297.78	12.77	1286.77

4.5.1 Performance Efficiency Results

Performance efficiency of the encryption and decryption algorithms were evaluated using the Throughput per Second metric, which is often used in evaluating the efficiency of encryption algorithms, as in (Abd Elminaam, et al, 2010).

The throughput is the amount of work carried out during the selected time unit. In our work the throughput is measured as the number of KiloBytes encrypted or decrypted per second. The results in Table 4.2 show that the encryption and decryption throughput per second for each processed file is very close; this is due to the fact that the decryption algorithm steps are inverse of the encryption steps. Moreover, despite differences in file size, the throughput per second for the different files are similar, with the exception of file no. 2 which is much smaller than the other files and has shown lower throughput per second; this is due to the effect of the file input-output overhead on the overall processing time.

The average throughput per second for the two algorithms, 1297.78 KB/Second and 1286.77 KB/Second, which indicates that the encryption of 100 MB MP4 video file takes about 1.3 minutes.

4.5.2 Integrity Evaluation

To evaluate integrity of the encryption and decryption processes, a byte-level comparison between the input plaintext video file and the decrypted video file was carried out. The byte-level comparison was executed using the File-Compare (FC) command in Windows, which detects any differences between the compared files. In the experimental work, all the comparisons resulted in zero differences between the plaintext file and the decrypted file, hence this outcome confirms that integrity of the plaintext file was maintained. Furthermore, the decrypted video files executed correctly and showed exactly the same video and audio output as what the plaintext video files showed.

4.5.3 Visual Cryptography Results

Visual comparison between plaintext and ciphertext files provides a way of assessing as to how well the plaintext file was scrambled into an unrecognizable ciphertext file. In the present experimental work, uncompressed images were chosen for two reasons: images are more suitable to present in a report than video; and encryption of compressed images can result in non-displayable ciphertext images.

Four standard test images from the Gonzales image database (Gonzales & Wood, 2018) were chosen, these are Lena, Peppers, Livingroom and Pirate, with image size of 768 KB and 512x512 dimensions. The original images were converted from TIF format to BMP, which is the format often used in similar work.

Fig. 4.1 shows the selected plaintext images and the corresponding plaintext images. The ciphertext images do not show any resemblance with of the plaintext images, and the four plaintext images appear visually similar, which indicates that the encryption process has resulted in a strong distortion of the plaintext image, leading to a ciphertext image that is totally un-similar to the plaintext image.

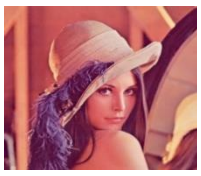
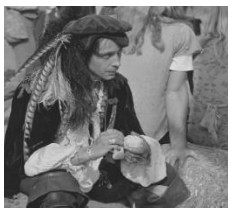
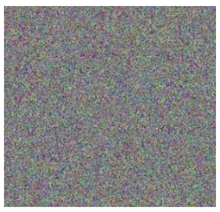
 Lena Plaintext	Lena Ciphertext
Peppers Plaintext	Pepper Ciphertext
LivingRoom Plaintext	LivingRoom Ciphertext
 Pirate Plaintext	 Pirate Ciphertext

Figure 4.1: Image distortion due to encryption

4.5.4 Statistical Distortion Evaluation

The Peak-Signal-to-Noise-Ratio (PSNR) metric is used in the comparison between two images in various image processing applications. High PSNR value indicates lower differences or stronger similarity between the images; while in encryption, lower PSNR values indicate higher dissimilarity or anomaly between the plaintext and ciphertext images, hence stronger encryption.

Table 4.3: Encryption PSNR results of the standard images

Image Name	PSNR (dB)
Lena.bmp	8.60
Peppers.bmp	8.07
Mandril.bmp	8.70
Lake.bmp	8.24
Average	8.40

Table 4.3 shows the PSNR results of the comparison of the four standard test images against the corresponding ciphertext images. First, it appears that there is little variation in the PSNR values despite the different color patterns of the four images. Second, the average PSNR is close to the results obtained in (Al-Hussainy, et. al., 2018) for the DES, AES and LWC algorithms, but it is slightly higher which can be related to the effect of using larger images of higher photographic quality.

4.5.5 Comparison with Previous Models

Several research works have compared new encryption models with the standard models AES and DES, the most recent of which is the paper by Al-Husainy and Sherjabi (2020), referred to herewith as H&S model, which presented a lightweight encryption model for IoT applications.

The proposed MEDS model is compared with the H&S and AES models in terms of encryption time and encryption distortion. Three uncompressed BMP images were used in the comparison as shown in Fig. 4.2.

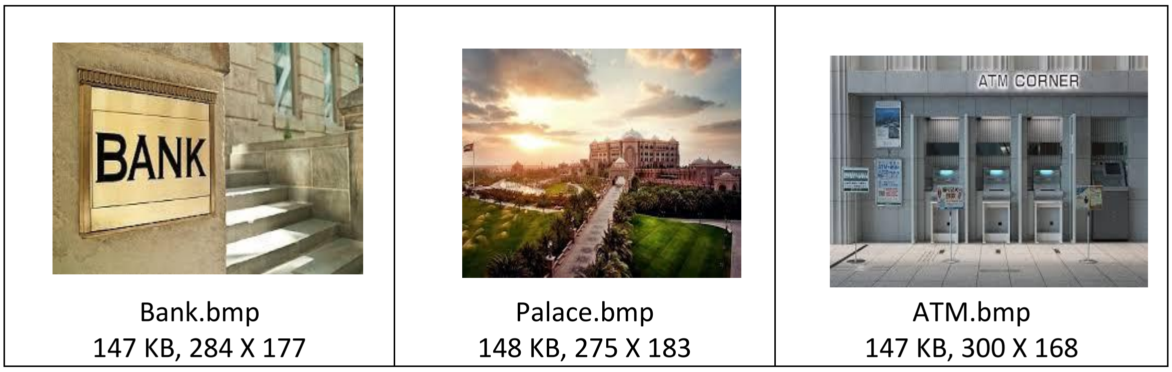


Figure 4.2: Three images for comparison

Table 4.4 presents results of the encryption time in millisecond of the proposed MEDS model and the two reference models. The encryption times for the reference models are as in the published paper, they were measured on a computing platform similar to the one used in the MEDS experiment. The MEDS encryption time is shown to be lower than that of the two reference models.

Table 4.4: Encryption time comparison with other models

Image	Encryption Time (msec)		
	MEDS	**H&S**	**AES**
Bank	166	210	405
Palace	170	134	271
ATM	195	168	266

Table 4.5 presents the encryption distortion results expressed as PSNR for the MEDS model and the two reference models, using the three comparison images. The results indicate a slightly higher PSNR value of the proposed model.

Table 4.5: Encryption PSNR comparison with other models

Image	**PSNR (dB)**		
	MEDS	**H&S**	**AES**
Bank	8.85	8.00	8.05
Palace	7.51	7.13	7.15
ATM	9.41	8.11	8.09

4.5.6 Security Strength Evaluation

Security strength of symmetric encryption algorithms are usually evaluated in terms of key size (aka key length or key space) which is equivalent to the complexity of a brute-force attack (Bernstein, 2005). The National Institute of Standards and Technology (NIST) defines the relative strength of different types of encryption algorithms in “bits of security”. NIST recommends (Barker, 2020) key size with a minimum strength of 112 bits to protect data.

In the proposed model, the key size was chosen to consists of 40 decimal digits, which gives 1040 possible combinations for brute force attacks. The key size of 40 decimal digits is equivalent to key size of 133 bits, as 1040 is equal to about 2133, hence, the security strength of the proposed model is evaluated as being at 133 bits level, which is above the minimum 112 bits as recommended by NIST, and higher than the AES-128 key size.

Based on the experimental results which showed that encrypting one MB of data takes less than one second, so a brute force attack on such data file would take on average about 1040 seconds, a very long time, and even when using faster computers, the proposed model still maintains a high level of data protection.

CHAPTER FIVE
Conclusion and Future Work

CHAPTER FIVE
Conclusion and Future Work

5.1 Conclusion

The research in this thesis presented the design and implementation of a new symmetric encryption model, with the feature of being stream cipher, non-selective, full (all-byte) encryption approach to provide high security, and to be computationally lightweight so as to be practical in encrypting / decrypting large multimedia files. The model uses a randomly generated numeric secret key, consisting of 40 decimal digits, as a seed to generate random numbers of values between 0-255, where each random number is matched with a byte of the plaintext file to generate a byte of the ciphertext file.

The implemented system does not compress or decompress a multimedia file, it is processed as a stream of bytes regardless of its structure or file type.

The experimental work involved encrypting / decrypting a set of compressed public MP4 video files of various sizes. The results showed that despite encrypting / decrypting each byte of the plaintext file, the average throughput per second was about 1297 KB per second, hence a relatively large video file of 100 MB would take about 78 seconds to encrypt or decrypt. This throughput per second makes the system practical to use for protecting multimedia files of realistic sizes on ordinary computing platforms.

Furthermore, although the encryption model is considered lightweight in terms of the involved operations, but due to using a large secret key and random data, as well as encrypting each byte of the plaintext file, the model provides a secure approach for protecting multimedia files.

5.2 Future Work

The research outcome of this thesis provides opportunities for further work on the lightweight cryptography of multimedia data, especially video. The following ideas are suggested for future work:

- Investigating the possibility of developing a lightweight public key cryptography model, using asymmetric keys, based on adaptation of the present work.
- Extending the proposed model to deal with in-transit (streaming) video.
- Enhancing security of the proposed model through techniques to use larger key space.
- Investigating alternative units of encryption such as 32-bit word instead of 8-bit bytes to reduce encryption time of the proposed model.

References

Abd Elminaam, D. S., Abdual-Kader, H. M., & Hadhoud, M. M. (2010). Evaluating the performance of symmetric encryption algorithms. IJ Network Security, 10(3), 216-222.

Abdulrahman, H., Chaumont, M., Montesinos, P., & Magnier , B. (2016). Color images Encryption using RGB channel geometric transformation measures. In Wiley Online Library.

Al-Hussainy, M. F., & Al-Shargabi, B. (2020). Secure and Lightweight Encryption Model for IoT Surveillance Camera, International Journal of Advanced Trends in Computer Science and Engineering, Vol. 9, No. 2.

Al-Hussainy, M. F., Al-Sewadi, H. A., & Masadeh, R. M. (2018). Lightweight Image Cryptosystem Using Auto Generated Key, Journal of Engineering and Applied Sciences, 13(17), 7418-7425.

Aljawarneh, S., & Yassein, M. B. (2017). A resource-efficient encryption algorithm for multimedia big data. *Multimedia Tools and Applications*, *76*(21), 22703-22724.

Aljawarneh, S., & Yassein, M. B. (2018). A multithreaded programming approach for multimedia big data: encryption system. *Multimedia Tools and Applications*, *77*(9), 10997-11016.

Arunabh, S. (2015). Overview of Multimedia Security, Avaiable at: http://www.academia.edu/8199308/Overview_of_Multimedia_Security [Accessed 25-10-2020].

Badr, S., Salama , G., Selim , G., & Khalil ,A. (2014). *A Review on Encryption Techniques: From Image Format Point of View*. International Journal of Computer Applications .

Barker, Elaine (2020). Recommendation for Key Management, NIST Special Publication 800-57 Part 1, Revision 5.

Bernstein, Daniel J. (2005). Understanding Brute Force. ECRYPT STVL Workshop on Symmetric Key Encryption.

Bhavani, T., (2007). Security and privacy for multimedia database management systems, pp. 14-29. Available at: https://www.utdallas.edu/~bxt043000/Publications/Journal-Papers/DAS/J33_Security_and_privacy_for_multimedia_database_management_systems.pdf [Accessed 1.11.2020]

Bhetwal, K. (2016). Multimedia security using encryption and decryption.

Big List of Sample Videos for Testers. (2020). Retrieved 15 June, 2020, from https://standaloneinstaller.com/blog/big-list-of-sample-videos-for-testers-124.html

Daemen, J., & Rijmen, V. (2002). *The design of Rijndael* . New York: Springer-verlag..

Deshmukh, P., & Kolhe, V. (2014, February). Modified AES based algorithm for MPEG video encryption. In International Conference on Information Communication and Embedded Systems (ICICES2014) (pp. 1-5). IEEE.

El-said, S. A., Hussein, K. F. A. & Fouad, M. M., (2011). International Journal of Signal Processing, Image Processing and Pattern Recognition. Securing Multimedia Transmission Using Optimized Multiple Huffman Tables Technique, 4(1), pp. 49-64, Available at: http://www.sersc.org/journals/IJSIP/vol4_no1/4.pdf [Accessed 20-10-2020].

Emanuil Rednic; Andrei Toma, n.d. Software Analysis. SECURITY MANAGEMENT IN A MULTIMEDIA SYSTEM, 4(2), pp. 237-247.

Furht, B., Socek, D., & Eskicioglu, A. M. (2004). Fundamentals of multimedia encryption techniques. *Multimedia Security Handbook*, *4*.

Gaur, R., & Chouhan,V.S.(2017). *Classifiers in Image processing*, International Journal on Future Revolution in Computer Science & Communication Engineering.

Goljan, M., Fridrich , J. , & Cogranne , R. (2014). *Rich Model for Encryption of Color Images*. In IEEE international workshop on information forensics and security (WIFS), Atlanta, GA, USA.

Gong, R. & Wang, H. (2012) *Encryption for GIF images based on colors-gradient cooccurrence matrix*. Optics Communications, (vol. 285), no. 24, pp. 4961-4965.

Gonzales, R. C. & Wood, E. R. (2018). Digital Image Processing, 4th Edition, Pearson.

Hall-Beyer, M. (2017). *GLCM texture: tutorial.* Arts research and publications.

Kaur ,M., & Kaur, G.(2014). ”*Review of Various Encryption Techniques*”, International Journal of Computer Science and Information Technologies .

Kekre, H.B., Athawale , A.A., & Patki, S.A. (2011). *“Encryption of LSB Embedded Images Using Gray Level Co-Occurrence Matrix”.* International Journal of Image Processing (IJIP), Volume (5), Issue (1)

Li, Chaoyun (2020). New Methods for Symmetric Cryptography, Ph.D. dissertation, KU Leuven University, Belgium.

Lindström, B. (2016). Evaluation of the impact of encryption on video transmissions (in e-health applications).

Pande, A. & Zambreno,J. (2013). Advances in Multimedia Encryption. In: Embedded Multimedia Security Systems. London: Springer-Verlag, pp. 11-22. Available at: http://www.springer.com/cda/content/document/cda_downloaddocument/97814471 44588- c2.pdf?SGWID=0-0-45-1345406-p174549534 [Accessed 20-10-2020].

Pande, A., Mohapatra, P., & Zambreno, J. (2015). Securing Multimedia Content Using Joint Compression and Encryption, Available at: http://spirit.cs.ucdavis.edu/pubs/journal/amit-multi.pdf [Accessed 20-10-2020].

Pujol, F. A., Mora, H., Sánchez, J. L., & Jimeno, A. (2008). A client/server implementation of an encryption system for fingerprint user authentication. *Kybernetes*.

Rasool , Z. (2017). *The Detection of Data Hiding in RGB Images Using Statistical Steganalysis* (Master Thesis, Middle East University).

Varhade, S. A., & Kasat, N. N. (2015). Implementation of AES Algorithm Using FPGA & Its Performance Analysis. *International Journal of Science and Research (IJSR)*, *4*(5).