REPUBLIC OF TURKEY

AKSARAY UNIVERSITY

GRADUATE SCHOOL OF NATURAL AND APPLIED SCIENCE

DEPARTMENT OF ELECTRICAL ELECTRONIC AND

COMPUTER ENGINERING

"SECURE VOICE BY USING AES ALGORITHM"

MASTER OF SCIENCE THESIS

SABAH SALIH HUSSEIN

SUPERVISOR

Prof. Dr. Mehmet R. TOLUN

AKSARAY, 2017

AKSARAY UNIVERSITY GRADUATE SCHOOL OF NATURAL AND APPLIED SCIENCES

APPROVAL

SABAH SALIH HUSSEIN, M.Sc. with thesis student No. 152335801, successfully presented the M.Sc. thesis titled "SECURE VOICE BY USING AES ALGORITHM*",* prepared after satisfying all the requirements identified in the concerned bylaws, before the jury whose signatures are affixed below.

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

Aksaray University

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

Aksaray University

Jury Member: Asst. Prof. Dr. Kasim OZTOPRAK **.....**

Çankaya University

Date of Submission: 23 October 2017

Date of Defense: 20 November 2017

DECLARATION

The idea for this present study was taken from the literature regarding the topics. It was announced here that whole work presented in this project has been composed and originated by myself unless otherwise specified. The study was conducted completely under the supervision of Prof. Dr. Mehmet Reşit Tolun.

SABAH SALIH HUSSIN

PREFACE

This thesis was made as a completion for master degree of sciences; this study focused on encryption\decryption text and encryption \decryption voice by using AES Algorithm. The purpose of this project is how it can encrypt -decrypt text and audio then comparison between them which is more secure and faster, Also which is more difficult to hack by the hackers.

ACKNOWLEDGEMENT

At the beginning, I want to express an own word of thanks of the perfect thesis supervisor Professor. Mehemet Resit Tolun. without his advice the thesis would not have been finished , also very thankful for his faith in this study, particularly in the sometimes-hard state of affairs that he was written; he was every time generous through each phases to the research.

I owe the University of Aksaray, College of Engineering, Electric Electronic Computer Engineering .that give me a nice chance to take a master's degree I am thankful to Professor. Atilla Elic, Head of Electric Electronic Computer Engineering Department for his forbearance and help special thanks to my family for their helped and always was supporting me.

TABLE OF CONTENTS

Page

Page

ABSTRACT

SECURE VOICE BY USING AES ALGORITHM

This thesis study encryption \decryption text and voice by using AES algorithm, then comparing between them to know which one is better by using three standards. Speed, time duration and security. In this thesis we used five operation modes first ECB - Electronic Code Book. However the second one CBC - Cipher Block Chaining. The third one OFB - Output Feedback. The forth operation mode CFB - Cipher Feedback. The last operation mode is, CTR - Counter Mode. We cannot say which mode operation is the best. Because each of them have advantages and disadvantages but for encryption \decryption text counter mode is better than other mode operation. Because it is faster and simple for using more than other mode operations. After encrypted and decrypted audio by using the AES algorithm. Obtained different results however needed more time than text for encryption and decryption .in order to audio have more data than text Therefore, need more time than text for encryption and decryption (AbdelrahmanAltigani, Bazara Barry, 2013). According to speed for encrypt and decrypt text is faster than audio in order to for encryption and decryption audio have many data (AbdelrahmanAltigani, Bazara Barry, 2013).But text has less data if compared to voice. Then less data need less time. According to security audio is more secure than text and also more difficult to hack by hackers. Utilized Matlab code in this thesis in order to it has the capacity to plot graphical inclusive. use MATLAB is more appropriate than other packages like java and C#,in order to it has a large a collection of tools and libraries; moreover, for programming, it have not necessary for beginning from zero in order to MATLAB previously has functions for performing limited missions whilst some programs like C# should needs for beginning from zero.

Keywords: ECB - Electronic Code Book, CBC - Cipher Block Chaining, OFB - Output Feedback, CFB - Cipher Feedback, CTR - Counter Mode.

ÖZET

AES TARAFINDAN ŞIFRELEME VE SES KORUMASI ALGORITMA

Bu tez, şifreleme \ şifre çözme metin ve sesini AES algoritması kullanarak inceledikten sonra aralarında karşılaştırarak hangisinin daha iyi olduğunu üç standart kullanarak bildireceğiz. Hız, süre ve güvenlik. Bu tezde ilk olarak beş çalışma modu kullandık; ECB - Elektronik Kod Defteri. Ancak ikincisi CBC - Şifreli Blok Zinciri. Üçüncü OFB - Çıktı Geribildirimi. Dördüncü çalışma modu CFB - Cipher Feedback. Son çalışma modu, TO - Sayıcı Modudur. Hangi modun çalışmasının en iyi olduğunu söyleyemeyiz. Her birinin avantajları ve dezavantajları olduğu için şifreleme \ şifre çözme için metin sayacı modu diğer mod operasyonlarından daha iyidir. Çünkü diğer mod operasyonlarından daha fazla kullanmak daha hızlı ve kolaydır. AES algoritması kullanılarak şifrelenmiş ve şifreli sesi aldıktan sonra. Elde edilen farklı sonuçlar, ancak şifreleme ve şifre çözme için metinden daha fazla zamana ihtiyaç duyuyordu. Sesin metinden daha fazla veriye sahip olması için Bu nedenle, şifreleme ve şifre çözme için metinden daha fazla zamana ihtiyacı var (AbdelrahmanAltigani, Bazara Barry, 2013). Şifreleme ve şifre çözme hızına göre, şifreleme ve şifre çözme için birçok veri var (AbdelrahmanAltigani, Bazara Barry, 2013). Bununla birlikte, ses, sesle karşılaştırıldığında daha az veri barındırıyor. Daha az veri daha az zamana ihtiyaç duyar. Güvenlik gereğince ses, metinten daha güvenlidir ve bilgisayar korsanları tarafından kesilmesi daha da zorlaşmaktadır. Bu tezde Matlab kodundan yararlanılarak grafik dahil edilme kapasitesine sahip olur. Kullanımı MATLAB, java ve C # gibi diğer paketlere göre daha uygundur, çünkü bu araç ve kitaplıklardan oluşan geniş bir koleksiyona sahiptir; Dahası, programlama için, sıfırdan başlamak için gerekli değildir, çünkü MATLAB, C # gibi bazı programların sıfırdan başlamak için ihtiyacı olmakla birlikte, sınırlı görevleri yerine getirmek için önceden işlevleri vardır.

Anahtar Kelimeler: ECB - Elektronik Kod Defteri, CBC - Şifreleme Blok Zinciri, OFB - Çıktı Geri Beslemesi, CFB - Şifre Geribildirimi, CTR - Sayıcı Modu.

LIST OF FIGURES

Page

Page

LIST OF TABLE

Page

LIST OF ABBREVIATIONS

AES	Advanced Encryption Standard
ASCII	American Standard Code for Information Interchange
C	Cipher
CBC	Cipher Block Chaining Mode
CERT	Computer Emergency Response Team
CFB	Cipher Feed Back Mode
CTR	Counter Mode
D	Decipher
DES	Data Encryption Standard
E	Encryption
E-Mail	Electronic Mail
E –shopping	Electronic Shopping
ECB	Electronic Codebook Mode
ECC	Elliptic Curve Cryptosystem
GF	Galois Field
HW	Hardware
IV	Initialization Vector
NIST	National Institute for Standard and Technology
OFB	Output Feed Back Mode
RC4	Rivest Cipher 4
RSA	Rivest –Shamir –Adelman
S-Box	Substitution Box
SEM-AES	Semi Symmetric AES Encryption Algorithm

1. PREFACE RESEARCH

1.1 Introduction

This thesis discusses a Matlab application to the Advanced Encryption Standard (AES). AES is depending on the block cipher Rijndael, it became the appointed successor to the Data Encryption Standard (DES). That has been executed in a massive number of cryptographic modules all countries since 1978. Matlab is a programming language used as a matrix-oriented, completely appropriate to the matrix-based data structure of AES. Information security is a process of keeping information against unauthorized arrival for using or modification of data, whether in the stockpiling, processing of transfer stage. It is public term which could be utilized regard less to the form the data may take .Information security is not only one technology; rather, but it is strategy that consists of the operations, too less and policies corollary for detecting or stop threats and keep the data. Some types of ticklish information should be kept private Away from a massive domain of threats and do not change, changed or transmit without license. For example, number of credit card, special messages, etc.

1.2 Voice Security

The first chapter gives a little over view of the public aims and fundamentals for this thesis. Take a look closer to the date of voice security as well as on to the fundamental techniques and then a true to the human sound itself. These rules govern each type of data at any time, especially those that are moved in some way. System Communication is often seen as potential security leaks. FORTRAN moves data however these systems use data security technologies.

1.3 Information Security Elements

The elements of information security, the ability to communicate securely between transmitter and recipient depends on: Confidentiality, Data integrity, Authentication and Non-repudiation.

1.3.1 Confidentiality

Confidentiality is almost tantamount to particularity. The measures taken to ensure confidentiality are intended to prevent the access of critical information to the wrong persons, whilst making sure which the right person could actually get it. Arrival should be restricted to those authorized for viewing the data in question. It is common, as well, to classify the data according to the amount and kind of damage which can be done if they fall into the wrong hands. More strict or less restrictive measures could after that be executed. Some ways are utilized for warranty confidentiality like the account number of the bank's electronic user IDs are combined with the password (authentication). Biometric investigation, security codes, etc. However data cryptography is a widespread way of guarantee confidentiality. Look the security risk in Figure 1.1 which the sender and recipient can encounter in an unprotected environment.

Figure 1.1: Unprotected Communications, loss of privacy.

1.3.2 Integrity

Integrity includes preserve harmony and precision, and data reliability through their entire life cycle. Data should not be changed during transport, Steps should be taken for ensuring which data is not changed by unauthorized persons (for instance in violation of confidentiality). These procedures involve file license and user arrival controls. Version control could be utilized for preventing wrong changes or accidental deletions by authorized users to become issue. Moreover, there should be some means of detecting all changes in data which may happen as a outcome of non-human events like electromagnetic pulse (EMP). Some data may involve test kits, until cryptographic labs, to verify integrity. Backups or recurrence should be available for restoring the influenced data to their true case.

Figure 1.2: Unprotected communications loss of integrity.

1.3.3 Authentication

Data authentication, the receiver must be able to verify the source of the message as well as the mud entity of the sender to verify the claims of the sender. Intruder and prevents claims of farce as anyone else. He said he should not be able to claim to be the sender of the message. Sometimes, the authentication data is divided into two parts: the fact that the data has not been modified (data integrity or entity authentication), and the fact that the recipient knows who the sender (the origin of data) authentication. Consider the security risks in the 1.3 format, which could face the sender and receiver in the non-protected environment, when the sender sends a message to the receiver, but intercept the intruder and sent to the receiver, pretend that the sender and send. A variation on this theme is: the intruder can send a message to the receiver and pretend that the sender sent. If an entity source of the message cannot be verified by the recipient, and then they lacks authentication.

Figure1.3: Unprotected communications, lack of authentication.

1.3.4 Non-repudiation

Non-repudiation with proof of origin protects against rejection by one of the entities of the other participants in a communication of having participated in all or part of the communication. Non-repudiation with proof prevents the sender. Any attempt by the sender to falsely deny sending the message. At a time when non-repudiation the receiver to prevent any attempt by the recipient refused to receive the wrong messages. As an example, Figure 1.4 shows the importance of non-repudiation of origin. Suppose that your company is the owner in order to e-mail, he decides to let its customers in the system via e-mail. For us, it's really important that it can be reduced to a third party really. Particular client ordered goods, which claimed the company. Otherwise it would be easy for customers to prohibit the purchase of goods. He said it is also important for customers to be able to demonstrate to a third party, that is a fact of goods. Otherwise, it would be easy for the company to deny work. Paper and pencil world, offering reliability by manual signature.

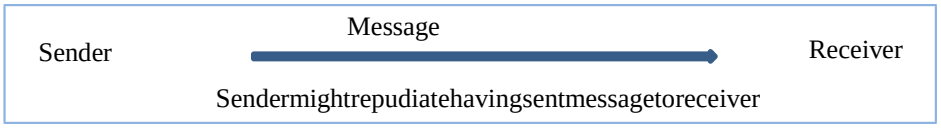


Figure 1.4: Unprotected communications, lack of non-repudiation.

1.4 Cryptanalysis

It is defined as the science and art of the ordinary to get encrypted, without knowing the encryption key. The goal is to recover the encryption key from the planner instead of recover a single message, If any kind of attack could entice the encryption key, the effect is very big and disastrous, All messages future and the past are encrypted using this encryption key risk. There are two ways of public attacks: Cryptanalysis attacks and Brute-force attacks.

1.4.1 Cryptanalysis attack

This kind of attack depends on the construction and nature of the encryption algorithm, and possibly some general knowledge of the characteristics of plaint excused, or even some sample of plaintext or cipher text pairs. This sort of attack exploits the encryption algorithm properties. It attempts to deduce specific plaintext or even to deduce the encryption key used. Cryptanalysis is divided into five types:

1.4.1.1-Cipher text only attack

The attacker knowing of the algorithm used, and having several cipher texts encrypted with the same cipher key. This sort of attack depends on the statistics on the frequency of letters in the cipher text. It attempts to deduce plaintext, possible encryption key or a part of it.

1.4.1.2 Known plaintext attack

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

1.4.1.3 Chosen plaintext attack

The attacker feeds the chosen plaintext into the black box. This type includes a black box encryption algorithm using an encryption key; both are chosen by the attacker. The black box encrypts the selected plaintext into a certain cipher text. Then, depending on the analysis the information accumulated between chooses plaintext and created cipher text pairs. It attempts to deduce the cipher key or part of it.

1.4.1.4 Chosen cipher text attack

Is an attack model for cryptanalysis where the cryptanalyst can collect information through obtaining the Plaintext of selected ciphertext. In the attack, enemy has opportunity to enter one or more known cipher texts into the system and get the resulting plaintexts from these parts of information the enemy can try to recover the hidden secret key used for decryption.

1.4.1.5 Chosen text

This is a mix of chosen Cipher Text and Plaintext Attacks. Sometimes, the chosen plaintext is encrypted to a cipher text, however at other times; the chosen cipher text is decrypted to a plaintext. Then, the attackers depend on analyzing the data cumulative between the plaintext and cipher text pairs. It as well try to deduce the encryption key or part of it.

1.4.2 Brute force attacks

This kind of attacks depends on trying every potential key combination on the chosen ciphertext till clear translation to the cipher text into plaintext is completed. This can too supposed as a kind of cipher text attack. The well-designed encrypt systems must be mathematically impossible. For example today AES with128 bit key size.

1.6 Research Motivation

Information security systems were in a critical situation in the last thirty years because of the numerous accidents that have taken place and led to data leakage for example**,** Cloud Computing Server**,** a Department of Veterans Affairs It was cracked when an employee has violated the service data by accessing it from home. The

employee was able to access the data records containing sensitive personal health information from more than 26000000 military veterans. These qualities, including health problems, social security numbers and addresses, etc. These kinds of servers are partly, not completely, protected against intruders, offering the information to pernicious attacks and often they are focused on a variety of attacks (Muhammad Usman and Usman Akram, 2014). As health information ten times more convenient than a credit card number on the black market, attackers are targeting $ 3 trillionU.S.health care companies; most of these organizations are not safe enough. In August 2014, after one of the largest US hospital operator attacking personal and health information to 4.5 million patients have been stolen by Chinese hackers, health professionals have warned the government to protect against intruders (EBay Hacked: 145 Million Customers Must Change Password, 2015). The largest auction site eBay in the world could suffer a gap in the security system and requires constant information security in all operations on the Internet. A security and good information means that sensitive data are protected from theft and misuse. This ensures security elements.

2. RELATED WORK

2.1 Introduction

With continued and fast widening of internet connecting through open networks, the security issue has become very necessary goal across networks. the formal statistics of Computer Emergency Response Team (CERT) claiming which in the latest forty years a lot of happenings occurred through internet communications in order to the networks are not very secured and the security become a decisive (Chen et al., 2007). The necessary or ticklish data transported through the network channel request quick insurance digital communication canal for achieving several targets of communication and data sending securely (Abaas, S. A., and Shibeeb, A. 2015). The field of data security has developed and growth widely in latest years. It provide a lot of mechanisms and methods which have been evolved To meet security needs.. But the more powerful instrument that is vastly used for controlling against a lot of types of security menaces is encryption mechanism (Pai, T. P., and Ravishankar, K. C. 2014). Encryption plays a focal role in modern cryptosystems. After that mitigate a lot of types of security menaces, equipping encryption for millions of ticklish financial, government, and own transactions daily. A perfect cryptography instrument could help mitigate the security intimidation and backing the data security broadly (Abaas, S. A., and Shibeeb, A. 2015). Many types of algorithms, like AES, Data Encryption Standard (DES), Elliptic Curve Cryptosystem (ECC), Rivest-Shamir-Adelman (RSA), Rivest Cipher 4 (RC4) etc, exist to encrypt and keep ticklish data. The algorithm utilized for encryption/decryption, the software program or hardware module utilized and the cipher key length specified the complication of cipher operation to turn into a plaintext to a cipher text and vice. Kirchhoff suggested which the security to each encipher system rely on the security principle. And they said about those situations which the secrecy of the cipher key is very necessary for security to the system instead of the cipher algorithm (Pai, T. P., and Ravishankar, K. C. 2014). The bigger key size, a lot of time the attacker need, for compute each the likely keys and implementation them to algorithm. This leads to upper security level (Bahrami, S., and Naderi, M. 2014). The bigger key area, possible keys could be

builded. Today, that is better to utilize the key size of 128, 192 or 256 bits. Such as, the key size of 256 bits could supply a 2^256 key space. For simplicity and velocity, the Symmetric algorithms are in large use. And also more convenient to encrypting a big text with top quickness. A lot of symmetric sketch exist, such as RC4, DES and AES (Abaas, S. A., and Shibeeb, A. 2015). Which, the basic defect of new cryptographic algorithms is warranty the security of data movable cross such unsafe medium as the internet? Consider a situation where a sender transmits a message to the recipient, the message can be bugged or changed through transition by the intruder who intercepts Telecommunications channel prior it reaches the receiver. This broken into exposes the Sent data to an unauthorized person.

2.2 Literature Review

DES was the standard symmetric key encryption algorithm. But it was brute-forced, and expires in 1998 to the United States. The National Institute for Standards and Technology (NIST) ran a competitive for the new Advanced Encryption Standard (AES) to substitute the DES that was cracked (Bahrami, S., and Naderi, M. 2014). In this world, a lot of suggestions have dealt with this contest, however just fifteen of them satisfy the main criteria of NIST. Out of the fifteen official nominees, five chosen finally. In October 2000, NIST. Declared the Rijndael as the winner of the five Finals algorithms. This algorithm was prepared through two Belgian investigators and put out in 2002. The Rijndael algorithm has been selection as the AES by NIST in order to it meets a lot of criteria which have been specified by the NIST Technology. Furthermore, it is one of the most suitable algorithms for encryption of data. After a little time, in November 2001, the Federal Information Processing Standards (FIPS) accepted and declared the Rijndael algorithm as a standard (Tayde, S., and Siledar, S. 2015). The AES, any input byte is always replaced by a stationary value supplied by S-box, no matters where it appears in the input or how many times it happens, the output byte is the same (Yong-Bin Kim and Kyung Ki Kim, 2014). The crackers are worthily focused on whitening the S-box because compromise the encryption key and the own info. For example, the crackers could Verify for the initial part of the S-box (that include 0x.63 0x.7C 0x.77 0x.7B values) and after finding the values of that part, then the S-box could be completely whitened and the key deduced as Kerins and Kursawe in 2006 aforementioned. A lot of researchers found out which the constant S-box a weakness and attempt to

reinforce the content and also the structure. Rachh and Ananda in 2008 designed S-Box utilizing combinational logic circuits even without calculating the Galois Field (GF). They suggested two techniques, the first one utilized truth table, and the other one is calculated the Algebraic Normal from (ANF) expression to all columns. Nevertheless, in compares with some other techniques, it has too long implement time (Bahrami, S., and Naderi, M. 2014). other technique in 2009 by Nalini and others suggested for constructing S-Box in HW design by utilizing multiplicative inverse in GF 2^8. However this path has a characteristic of low area overhead, it has highly long implementation time. Furthermore, the HW intricacy is increased widely (Karthigaikumar, P., and Rasheed, S. 2011). The asymmetric cryptosystem has a lot of advantages which cannot found in symmetric cryptosystem, like the existence of two various keys to perform telecommunication securely and for any session they are various (changed periodically). Furthermore, the number of keys utilized in huge networks could be smaller compared with networks that utilize symmetric key operation; (Yong-Bin Kim and Kyung Ki Kim, 2014). In 2012, Murthy and others say that in a large network, the number of keys in asymmetric can be smaller compared with a network that uses symmetric. nevertheless, It is strongly depend on modular arithmetic and exponentiation involving huge prime numbers, in order to all user must receive a public key from sender like independently and calculate the own key. This demand more than one session to create an own and a general key. This Lead to wastage of a lot of execution time. Too, for more than two nodes in the network, everyone must calculate the own and general key independently with other node (Yong-Bin Kim and Kyung Ki Kim, 2014).

Nevertheless, asymmetric key encryption has a lot of troubles, like: (Jin Gong, Wenyi Liu and Huixin Zhang, 2012).

1-The preparation to the key distribution Requests a lot of time if we compared with symmetric encryption.

2-A lot of time is required for encryption data in order to the mathematical processes that are required for creating public and private key.

3-The key sizes are too huge than those required to symmetric scheme.

4-Asymmetric general and own keys generated loss a lot of time. Furthermore, it decreases the Power of agreeable keys in a range because of to primitive values, whilst symmetric keys are commonly random numbers for period of time.

5-Till now, no public-key schema has been certain to be very secure.

6-The encrypted message could only be send to a single person. It is not practical to send it to more than one person as a recipient's own key is utilized.

On Murthy and others asymmetric algorithms have their advantages and disadvantages. They concluded which on the base of the application; the convenient authentication protocol must be selection. In asymmetric techniques, besides the key distribution trouble and a lot of other troubles, one of the big and very important troubles is the slow speed while asymmetric approaches are almost thousand times slower than symmetric (Gupta, S., and Goyal, D. 2014). It is very important in cryptography besides the Secret history. AES is a modernist symmetric block cipher that is one of the most prevalent algorithms which encipher data in many various applications. Ranging from huge data servers for small low-power embedded systems like the wireless standards 802.15.4 and protocols containing IPSec and SSH (Abaas, S. A., and Shibeeb, A. 2015). The main goals of AES Transitions are to collect non-linear relationship between plaintext, key, cipher texts and to diffuse the plaintext's block on some level thing. Adoption of round keys makes non-linear cryptanalysis. AES deploy the bit pattern to all rounds key by the way replacement and turnover (Karthigaikumar, P., and Rasheed, S. 2011). The relation among the plaintext and cipher text is concealed through propagation at bit level. It provides power to AES against a lot of cryptanalysis attacks that seek info on cipher text and plaintext (S.Dasa, and R.Ghosh, 2013). While utilizing a voice encryption algorithm such as AES, the particularity of the transported data is guaranteed. Nevertheless, a man-in-the-middle who knows about the encrypted communications among sender and the receiver could still interfere in different methods including traffic analyses and the deprivation of service attacks. In 2012 Soni and others and in 2014 Muhammad and Usman analyzed and compared the AES and DES encryption algorithms for more information you can watch Table 3.1. The main problems for all encryption algorithms are Consume necessary calculating resources such as CPU time, simulation (calculating) time, memory usages and the encryption level. It is necessary for each cryptography algorithm to have a high Avalanche Influence (Avalanche Influence is computed by alteration one bit in plaintext saving the key fixed or by alteration one bit in key keeping the plaintext fixed). Avalanche Influence = Number of flipped bits in ciphered text / Number of bits in ciphered text). It means which an encryption algorithm must alteration a lot of bits of the cipher message

when one bit (or some number of bits) of the plaintext or one bit (or some number of bits) of the encryption key is changed (Muhammad Usman , Usman Akram, 2014).

Table 2.1: Comparison between AES and DES.

FACTORS	DES	AES
Key size	56 bits	128,192 or 256 bits
Block Size	64 bits	128 bits
Encryption type	Symmetric block cipher	Symmetric block cipher
Developed year	1977	2000
Security Proven	Proven inadequate	Considered secure
Cryptanalysis Resistance	Vulnerable to differential and linear cryptanalysis; weak substitution tables	Strong against differential, truncated differential, linear, interpolation and square attacks
Possible Keys	2^56	2^128, 2^192 and 2^256
Possible ASCII Printable Character Key	957	95^16, 95^24 or 95^32
Speed (AES_256)	NA NA NA	Visual C++ 51.2 Mbits/S C 19.8 Mbits/S Java 709 Kbit/S
Avalanche effect (1 bit in the plaintext is changed)	43 bits out of 128 bits	83 bits out of 128 bits
Avalanche effect (1 bit in key is changed)	41 bits out of 128 bits	81 bits out of 128 bits
memory required for implementation	43.3 KB memory	10.2 KB memory

From the experimental results in Tables 3.1, Authors (Abomhara, M. A. S. 2010). Have completed which AES is more secure than DES and AES is also twice as fast as DES. But one of the power factors is the key sizes of AES that is too huge than

that of DES's. DES utilizes 56 bits of key size that is not impedance to cryptanalysis, particularly brutal force offensive, while AES utilizes 128, 192 and 256. Bits of key size for cryptography algorithm that is impedance to cryptanalysis, particularly brutal force offensive. AES is impedance against a lot of kinds of attacks like differential, linear and statistical analysis (S.Dasa, et al., 2013). And therefore AES could be selection as first choice as symmetric block cipher. The paper was not mention that mode of processes is utilized to conduct the avalanche influence computation. For example, a count mode of process encrypts all block independently, and any bit change in plaintext or cipher key influence the equal bits in the cipher text only till if it encrypts only one block. Comparing the Avalanche Influence in (Muhammad Usman and Usman Akram, 2014). It could be finished which the new way in does not have Influences of performance of security. Completely reverse, the suggested ways can reduce the performance of security. Furthermore, a lot of time is needed to map the plaintext and key to binary codes. In 2012 Oyshee and others in had suggested a technique to discovery a new affine transference and tried to minimize the implementation time. As a result a new S-Box with various values is created. While the algorithm with the suggested S-Box and inverse S-Box has been executed, the experimental result indicates which the wanted time is optimized by 10 microseconds almost with various values (Gupta, S., Kishor, L., and Goyal, D. 2014). In spite of the suggested way it lowering time consumption, thither is a small time compared with the standard S-Box. For this cause it is very difficult to implement the new way till in HW or SW. too any changing in the algorithm structure must inform the receiver. The S-Box in AES It is the most complex part till in HW or SW in order to it is the only non-linear function whilst the rest are liners' (Abaas, S. A., and Shibeeb, A. 2015).. Therefore, a lot of researchers try to enhance this function till in HW or in SW design. The five unpretentious style lookup tables in 2012 by Gong and others. Have been executed utilizing S-box have the capability to combine various steps of the round functions that leads to curtailment the magnitude of code. The execution capacity has too been improved according to the authors. This could be simply executed in VB, C, VHDL or other languages (Gupta, S., Kishor, L., and Goyal, D. 2014). Nevertheless, the structure of utilizing S-Box do not changed when the authors exchange the S-box table with five simple style lookup tables. Using more than one table, it makes it difficult the calculations and replacement operation. it leads to need of a lot of time (this way is already utilized

in DES algorithm where it uses 8 S-Boxes tables and replaced by single S-Box in AES. it makes the replacement operation efficient in terms of time difficulty. For the suggested AES, the structure of AES is completely changed when it combines various steps, these changes demand the new execution till in HW or SW and the receiver must know about each changes in standard design. Based on of the experimental results, they have discovered which this way is four times faster than the sequential one (Singh, Sartaj, 2012). But the problem of security of the algorithm. Too depend on the method of process. In practice, five methods of processes can be used, while only two of them could be used in this suggested way (Yadav et al., 2016).to know solution of issue, the key of the AES-CTR and AES-EBC mode should be updated periodically to avert the pattern at block level. The S-Box in AES is generated, using a determine irreducible polynomial (11B) in the modulus GF (28) with determined additive constant {63} in GF (28) (Ramesh K, Ramesh S, 2009). In 2013, Dasa and others in (Ramesh K, Ramesh S, 2009). Have proved which, by save the criteria defined by Rijndael algorithm (AES), the new S-Boxes had been generated utilizing various irreducible polynomial and addition steady. This reinforcement has the same or till better influence in comparison with the standard S-Box. With this new manner, the authors think which the security in AES is strengthen (they have found them as additional keys for AES) that are indeed the unspecified module and additive steady polynomials used by the sender. According to the authors, the sender could select and generate each S-Box according to his / her particular choice (i.e., unknown S-Boxes) from a huge set of options to block linear and differential cryptanalysis. In situation of suspicion of any intruder, an S-Box could be substituted by another one by the sender (Singh and Sartaj, 2012). This new way is characterized by NIST and it increases the prevention of linear and differential cryptanalysis. Nevertheless, all time the user selected or generates each S-Box to replace the present S-Box, the receiver must inform about new S-Box and how he could get this. Furthermore, the receiver should be capable to generate the new S-Box to substitute the existent one with the new one. Furthermore, it demand extra time and power consuming (Yadav et al., 2016).Ultimately, suggest that the AES is a hardware module, how could the receiver change the S-Box. In order to S-Box in AES every time produces the same output value for the same input value independent of the gross numbers of the value or while it show in the message, in 2013, the authors of paper see it as a feebleness which provided by the replacement

step to the algorithm, attacks are efficiently centered on whitening the S-box because compromise the encryption key and the own info. Furthermore, when adding the encryption key to the mass data entry using the AddRoundKey function works independently and produce the same output with the same input values, It is not important how many times a value shows in the data block in order to the encryption key is constant. The assailant that uses the statistical properties to the info to attack the cipher text could make utilize of this fragility. The statistical properties could be found during searching for some values of replacement box in a lot of files (Ning et al., 2014). For avert this kind of attack in the AES cryptography algorithm, there are a lot of methods to adjust the S-box values; one of these methods is utilizing the 8 by 8 binary diversion matrix and the transposed 8-bit vectors to calculate the replacement values. Another method to calculate the replacement value of a byte to the S-Box is based on polynomials. But it is necessary to calculate the replacement value for the S-Box in runtime and if could be, to generate replacement value dynamically. The authors of paper mentioned in 2013. The standard S-Box is created, utilizing determine irreducible polynomial (11B) in the modulus GF (2^8) with a determine additive fixed (63) in GF (2^8) and the S-box could be calculated and stored in advance. The authors (Bahrami, S., and Naderi, M. 2014). Suggest generating dynamic S-Box with random irreducible polynomials by using some subsequences of the encryption key. This dynamic S-Box enhances the security of the S-Box and makes the AES more powerful, according to them. The encryption key of the suggested way contains three parts (k1, K2, k3). The Transactions k1 and K2 of irreducible polynomials alteration randomly in all rounds and for all input byte. The first key (k1) could be chosen randomly from the array to the 112 non-symmetric invertible elements. The second key (k2) could be selected as integer value from 1 to 255. The third key (k3) is the key utilized on the Add Round Key function. The key length with 16 bits increases for this suggested way. The key length for AES_128 is from 128 bits to 144 bits, AES_192 is from 192 bits to 208 bits and AES_256 is from 256 bits to 272 bits. According to the authors, this way did not give any opportunity for the enemy to whitening the S-Box values of the AES in order to the S-Box, values are obscure. It increases the level of security. Furthermore, the intricacy of S-Box is even reduction. Nevertheless, the authors had not mentioned anything about the time required to perform these processes .It did not prove to them in practice. All time S-Box is created in the run-time an extra time is needed (more

security but less speed). The authors have tried to save the S-Box values secrets for the public, however have made known the power of algorithm is in the encryption key and the algorithm consists of the S-box if known for public (Yi et al., 2016). At recent days,, this technology It has enabled biometric technology setting up the encryption key for the cryptography system. In 2013, Tajuddin and Nandini in paper (Yi et al., 2016). Execution a method setting up the encryption key utilizing retina biometric technology in order to it offer a unique identity which cannot be repeated. They have executed the cryptography algorithm which creates the unique encryption key from the person's physiological or behavioral advantages like retinal blood vessels, by extracting retina advantages of blood vessels and calculating the number of bifurcation points, degree of bifurcation point or the number of eye grounds retrieval in the retina images. In spite of this new technique is very complicated for the enemy in order to crack or guess biometric key And minimize the cost in comparison with missing key (Singh and Sartaj, 2012). Generally, a biometric system works either in investigation mode or in identity mode, count on the application context, like enter in the system or select a system. Whilst cryptography system is generally utilized between two parties that try to connect securely through unsecure network, it is difficult for the pool these two techniques. Furthermore, for the extraction a characteristic set from the data that obtained (retina images) several own retina reader and own software there is a need for a program to feature extraction retina. These requirements and calculations increase the cost and the time delay that are the necessary characteristics in the communication system. (Singh and It is hard to the receiver and the sender to remember the key that would Sartaj, 2012) be meaningless (it is difficult to share between two parties). In the asymmetric situation, keys are numbers with own mathematical characteristics which demand a key generation schema and not any keys are agreeable. The AES is computationally secure against wanton force offensives, the combination between AES, that is responsible of just scrambling the confidential data and the steganography protocol Word Shift Coding Protocol (WSCP), that is responsible for hiding the confidential data, strengthens the security level of transmitted messages widely. The AES way is utilized to encrypt the message, then, the encrypted message is transformed to a binary form, and hidden in a carrier. As said previously, the result has better confidentiality for the message but the extra layer of security case is delayed in simulation time so at a time borne (Supportz, 2015). The steganography protocols

use a carrier that is a medium to carry the confidential message; these medium could be a text, an audio, a video, an image …etc. And these protocols different according to the medium in use and other standard. Utilizing a steganography protocol alone for hiding the message could not be enough, in order to in case the enemy notices which the steganography is utilized, the message could be guessed or cracked while the AES could be attacked by a man-in-the-middle. The experimental analysis and results of the suggested way has proved which has a long delay time to the encryption for comparing messages with the standard AES.There are a lot of causes for delay time like required time for hiding each encrypted characters in the carrier or to retrieve the cover characters and checking the rest of the carrier till after each cover characters are retrieved (Bahrami, S., and Naderi, M. 2014). First, the size of the message must not be huge; it is relatively difficult for hiding huge data that require huge covers. Second, these systems involve dealing with the intricacy, third these ways increase the simulation time required to encrypt messages, and lately, a lot of time needed to transmit the message in order to very large data must be transferred (data with the carrier) (Supportz,2015). In order to the Sub Byte diversion function (the replacement table) is the only non-linear diversion function whilst the rest of diversion functions are linear, it is too much costly and intricate portion in the AES encryption algorithm when it is executed in HW design (Nalini et al., 2009). A lot of techniques by a lot of researchers have been utilized, like a simple look-up table (16-by-16 LUT) which includes all possible values of S-Box in ROM chip. The merit of this kind of the S-Box is which it offers shorter the delay time. Nevertheless, it has troubles with high speed pipelined design and it could not be utilized in high speed applications. It too requires a wide area for SubByte and InvSubbyte for encryption, respectively; paper mentions it in (Cyberwarzone.2014). Another method to execute SubByte transformation in HW design is by utilizing combinational logics circuit that is computed from the arithmetic characteristics. This S-Box approach has a merit without delay. Time of S-Box transformation and good performance for comparing with the 16x16 LUT S-Box, But it demand a very large area for logic circuits, paper mentioned it in 2014. In 2014, Ho Lee, Yong Kim and Kyung Kim have suggested a modern S-Box architecture utilizing Programmable Logic Array (PLA) and power-gated technique. By the authors, from the experimental results analysis, and has a better performance comparing with the standard S-Box architectures. The merit of the modern approach is which it minimizes the intricacies

of hardware, and increases the implement time with high speed and consume small power (Cyberwarzone.2014). Each of these methods are designed in hardware It is well known which execution anything in HW needs a huge cost comparison with the same things execution with software. With the High-Level languages like C/C++ the quickness of conventional hardware design could be obtained and any modification in the program could be simply maintained. Furthermore, wherefore the drawback of rapidity and security is not completely solved. In 2014, Feng-Hsiag Hsiao and Guang-Wei Liou have too suggested a hybrid approach which combining the AES with chaotic synchronization cryptosystem. Then key expansion, the first round key and the message are utilized as input to the AES algorithm to make an intermediate state called encrypted message, after that the output is directed To the anarchist system called master system which transform the encrypted message to the cipher text message utilizing anarchist masking. The cipher text and the n-the round key (It was created by making a key expansion) are send aloof to the receiver via an channel unsafe. While the Chaos systems are synchronized in the sender and receiver, the encrypted message is received. At the receiver side, the Chaos system called slave system transform the decrypted cipher text to the encrypted message. After that, the AES algorithm utilizes the n-the round key to decrypt the encrypted message to a plaintext message (Gupta, S., and Goyal, D. 2014). A high level of is supplied by using hybrid (RSA & AES) algorithm, by the paper. However the authors are not demonstrating the time needed for this method that generally takes a lot of time and causes transfer overhead. The AES algorithm is utilized by a lot of implementations and a lot of these implementations utilize AES algorithm in combination with another security algorithms or protocols. The big trouble is how the confidentiality of the transmitted message. In 2015, the authors have utilized a compiler which automatically adds a part of program in assembly language for the encryption algorithm against side-channel attacks for protection the message. The compiler chooses some instructions for this mission. The compiler is valuation based on two block ciphers, AES and Clefia. The authors does not remind anything about the area required for these symbol , who the recipient to the message knows the symbol used and if they are chosen automatically, how could we warranty which at the receiver's side the same symbol part is chosen.

2.3 Conclusion

In this Chapter, a state-of-the-art to the related business for Advanced Encryption Standard (AES) schema was offered. The two basic goals of encryptions are security and implementation time (speed). There are a lot of studies and theses which attempted to strengthen these goals to a better merit. Nevertheless, when these studies and theses are deeply studied, one could deduce which the two basic goals of security and implementation time to the data have not been achieve efficiently. A lot of investigations in the lot of sides of security and implementation time have been conducted because of the strengthen security and implementation time of algorithms. For example, S-Box preserves several times which it promotes the security however it increases the delay or requires more space or vice versa. Moreover, in the situation of the hybrid approach, it increases the security; however it spends a lot of time. Despite of the three are a lot of studies which have attempted to manipulate the S-Box or derived a lot of look up charts from the standard S-Box for standing in the same or till in the better merit list comparing to the standard S-Box, just a little security strengthened is achieved at the expense of very big area requirement and too much delay. Extra methods of strengthen contain combining the AES with another approaches to increase the security; nevertheless, they increase the delay (spend more time). A lot of studies have attempted to minimize the processing time like the parallel AES schema which depend on hardware applications. For example, the AES-CTR and AES-EBC process modes which work parallel, have a trouble with processing a big data, there is the high opportunity of an enemy managing to use the pattern at block level. In this study, an algorithm which would be called Semi-symmetric AES algorithm (SEM-AES) in that AES with updated key would be suggested for using. The suggested algorithm indicating the superiority. By means of security on AES scheme while maintaining the same required processing time. It makes a different cipher text all time it runs encrypting/decrypting, although the same key and messages are fed into the system.

3. BACKGROUND THEORY FOR AES ALGORITHM

3.1 Introduction

In this chapter, the theory will be backlit for the Advanced Encryption Standard (AES) planner have an idea about the frame and algorithm performance that is, how it works, and the resulting. The goal of the message encrypted to limit the operation mode which uses AES for encryption data (selected on the basis of the proposed mode of operation). The AES system is iterative process, i.e., It contains a number of rounds utilized to state; the number of rounds depends on the key size. It is10, 12 and 14 repetition to the key size 128-bit, 192-bitand256-bit, respectively. Any round contains four transformations added to state values these are: Byte-Substitution, Shifting-Rows, Mixing Columns and Add Round Key. The latest round in each is exclusion in order to it does not have Mixing Columns. The encryption key of the length (16, 24 or 32 bytes) is expanded in to a key schedule that contains a linear array of 44, 52 or 60 words of 32 bits each, respectively. Any key of the four sub keys is utilized by AddRoundKey procedure to all rounds. Lastly, Cipher text appointed base64 encoded data before transmission (or for decoding base64, while receiving). This goal is to change non- universal ASCII characterset to characters that belong to the universal ASCII characterset. The AES roads described bytes authors (Supportz, 2015).

3.2 Message to Blocks Conversion

Algorithm of AES is an encrypt/decrypt stationary -sizeblockofdataof128-bit (16bytes) length as well. Typically, the size of messages arbitrary length, Message length from one to another could different (for example 1MB). To transform a message into blocks of plaintext, at the beginning should be divided into a number of constant-size blocks of 128-bit length. Then, Encryption of these blocks and decryption based on the mode of operation utilized. AES block cipher algorithm, which demand the input of messages to be an exact multiple block size of 128 bits (16 bytes). If a plaintext input that should be encrypted is not an exact multiple of the block size of 128-bit before the encryption operation, it must be supplemented by the

addition of padding characters to the end of the message. Figure3.1 explain transmutation message to a number of blocks. Typically, the characters utilized for the stuffing are (zero (0) value, distance or character x) and called phantom characters. On the receiving side, after decoding the encrypted encryption text message recipient knows that the recurring characters in the end of the letter are a fake character. These characters will be removed by the receiver.

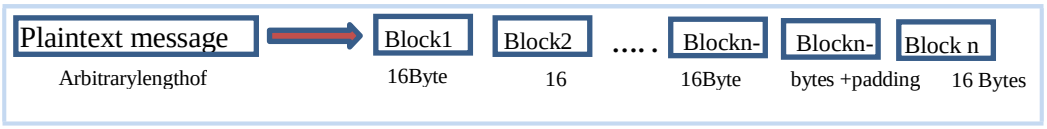


Figure 3.1: Message to blocks conversion.

3.3 Block to State Transformation and Vice Versa

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

3.3.1 Block to state transmutation

The input block is a series of 16 bytes can be changed to an intermediate state could be represented as a matrix of bytes, consisting of four columns and four rows, called state array. Figure 3.2 shows the way of how a block is changed to a state. The first four bytes(0,1,2and3) become the first column of the state array arranged from up to down such that $S_{0,0}$=b0,$S_{1,0}$=b1,$S_{2,0}$=b2and$S_{3,0}$=b3.The second four bytes(4,5,6and7) become the second column of the state array arranged from up to down such that $S_{0,1}$=b4,$S_{1,1}$=b5,$S_{2,1}$=b6and$S_{3,1}$=b7.Thebytes(8,9,10and11)become the third column of the state array arranged from up to down such that $S_{0,2}$=b8,$S_{1,2}$=b9,$S_{2,2}$=b10and$S_{3,2}$=b11.The bytes(12,13,14 and 15)become the fourth column of the state array arranged from up to down such that $S_{0,3}$=b12, $S_{1,3}$=b13, $S_{2,3}$=b14and$S_{3,3}$= b15.

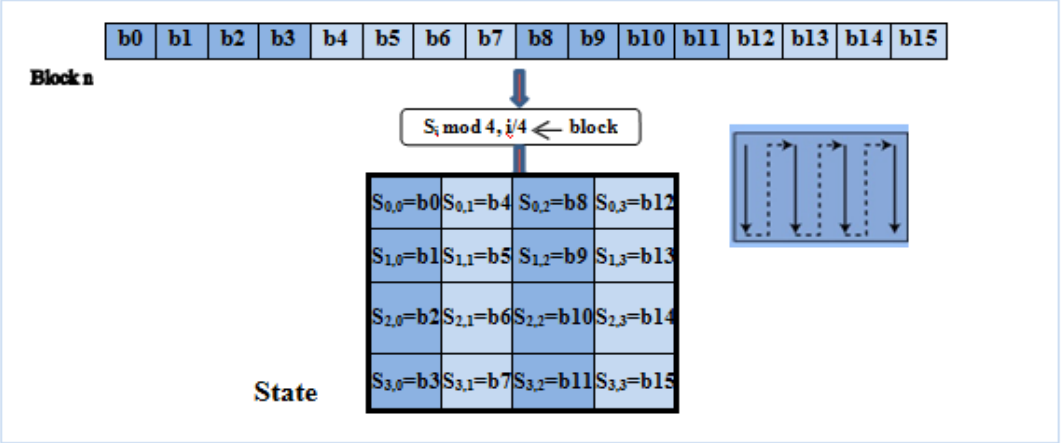


Figure 3.2: Block to state transmutation.

3.3.2 State to Block transmutation

After the encryption or decryption, then it should set an intermediate state for a series of 16 bytes can be represented as a set of a series of 16-byte returns called output block. Figure3.3 shows the mechanism of a state is transform to a block. The first column of the state becomes the bytes(0,1,2and3) of the output block for example b0=$S_{0,0}$,b1=$S_{1,0}$,b2=$S_{2,0}$andb3=$S_{3,0}$.The second column of the state becomes a byte (4,5,6and7)of the output block for example, b4=$S_{0,\ 1}$, b5=$S_{1,\ 1}$, b6=$S_{2,\ 1}$ and b7=S_3. bytes (8,9,10and11) of the output blocks that b8=$S_{0,2}$, b9=$S_{1,2}$, b10=$S_{2,2}$ and b11=$S_{3,2}$.The fourth column of the state becomes the bytes(12,13,14and15)of the output block such that b12=$S_{0,3}$, b13=$S_{1,3}$, b14=$S_{2,3}$ and b15=$S_{3,3}$.

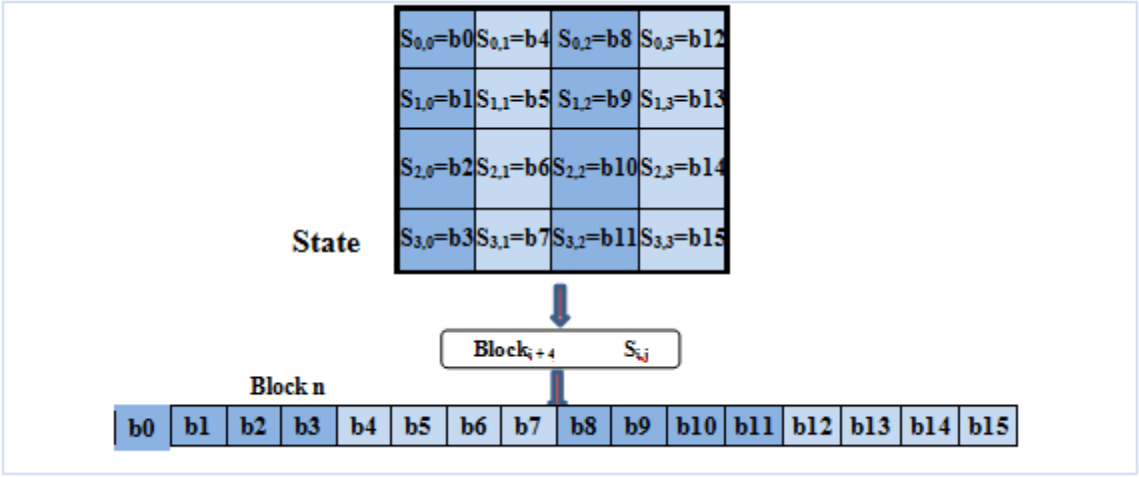


Figure 3.3: State to block transmutation.

3.5 Encryption/Decryption

In the AES algorithm, the input block length, the output block length and the medium value as it is called the State are finite to 16 bytes. The state is a matrix of 4 columns and 4 rows. This state is represented by Nb = 4, that reflects the number of 32-bit words in the state. whilst the cipher key length can be autonomous limited to 16 byte, 24 byte or 32 byte, the length of the key is represented by a matrix of 4 rows and 4, 6 or 8 columns that is represented by Nk = 4, 6, or 8, reflecting the number of 32-bit words in the Cipher Key. The AES algorithm is a repeated process. i.e., it contains a number of rounds applied to prevent the data encryption operation. The number of rounds to perform on the one block of data through the implement to the AES algorithm depends on the key size. For the key size with 128-bit length, the number of rounds (Nr) is 10 repetitions. It is called AES_128. While the key size with 192-bit length, the number of rounds (Nr) are 12 repetitions. It is called AES_192. As for the key size with 256-bit length, the number of rounds (Nr) is 14 repetitions. It is called AES_256. The connection between the cipher key length, data block size and number of rounds desired for all AES-type are given in Table 3.1.

Table 3.1: Relation among round, key and data.

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

All rounds for decipher and cipher implement four transformations on blocks of data. These functions (and inverse) are explained in parts 3.7 and 3.8 below.

3.6 Key Expansion

The procedure for a key expansion of the AES algorithm takes as input the length of the encryption key is Equal to key[4*Nk](16,24or32bytes)that could be seen as array of inputs Nk-wordof32 bits each where Nk=4,6,8 count on key. And as output, it generates a key schedule which includes a line array of 44/52/60 32-bit words each. Total number of words which the key expansion. Routine generates count on the key size that is Nb (Nr+1) words. Figure3.9 shows an example of 16 bytes key, How to extend the generated table of key encryption key. The expansion of the encryption key into the key schedule proceeds according to the fake code in the shape 3.10.

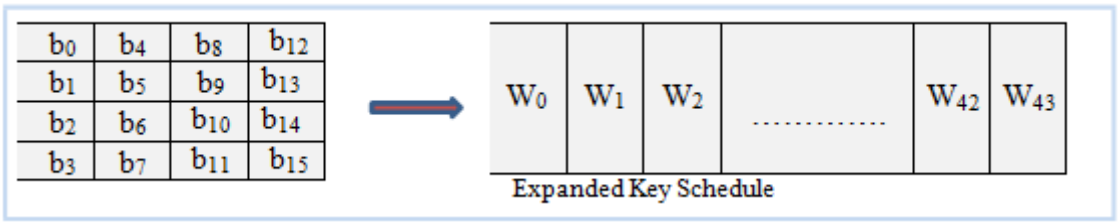


Figure 3.4: Key expansion schedule from cipher key.

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

Figure 3.5: Pseudo code for key expansion procedure.

For the initial Round and for all of the Nr rounds (where Nr=10, 12 or 14).

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

Figure 3.6: Round keys.

The key expansion scheme contain building the initial round key (4 words of 32 bits) immediately from the key by dividing it to four words, that shown in Fig 3.12, building the 1 to Nr round keys (each round is 32 bits about 4 words), while all word (Wi) in the round key is formed on the basis of the prior word (Wi-1) and on the word (Wi-Nk). Except for the words in positions that are a multiple of Nk, while a private transmutation is implemented to the word (Wi-1) by rotating Wi-1, its value is exchanged and XOR-ed with a fixed value. The outcome is called (ti) that is eventually XOR-ed with the word (Wi-Nk, Show shapes 3.12 and3.13forkey expansion. In the 256-bit key (Nr= 14), if Nk= 8 and i-4 is a multiple of Nk, then Sub Word is applied to w[i-1] prior to the XOR.

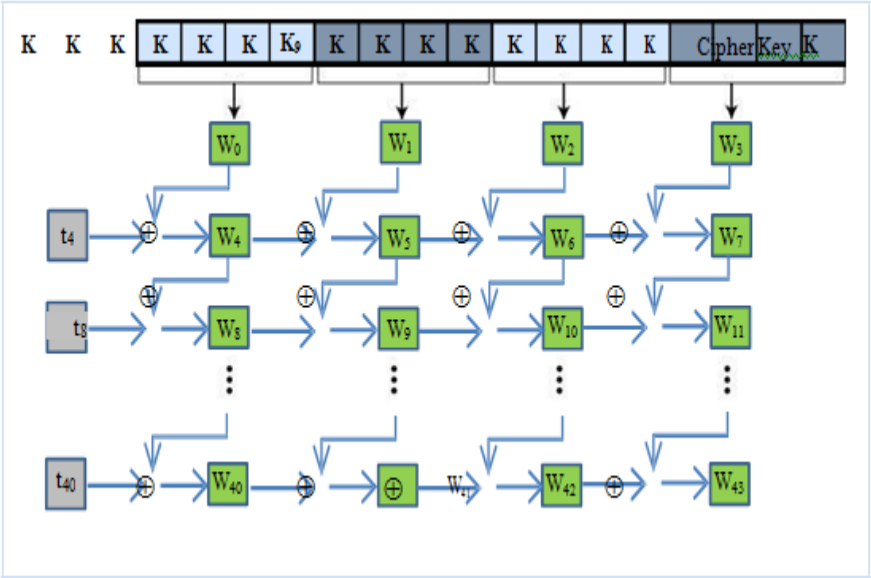


Figure 3.7: Key expansion constructions.

RotWordprocedure is a function that accepts a word as input, performs cyclically left shifting on a word as output word.

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

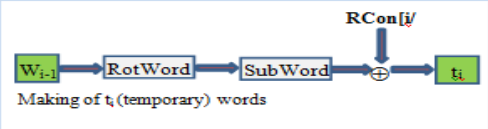
Sub Word procedure is a function that perform subtype substitution on each of the four bytes of input word using the S-box table.

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

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

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

Figure3.13showsan example of constructing temporary wordstj,when i =4Nr.tj =SubWord(RotWord(temp))□□RCon[j]– the round constant, Figure3.14.



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

Figure 3.8: Temp to construction.

Figure 3.9: Round constant (RCon).

3.7 Cipher

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

1-SubBytes (substitution).

2-ShiftRows (permutation).

3-Mix Columns (mixing).

4-AddRoundKey (round key addition).

The Cipher is described in the cipher structure in Figure 3.15 and in the pseudo code in Figure 3.16.

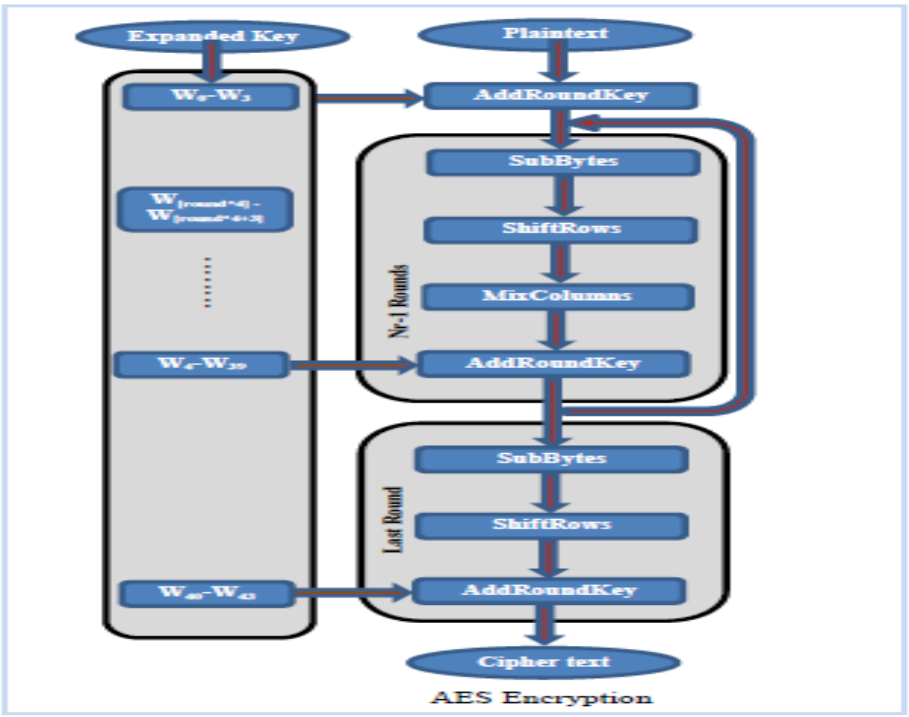


Figure 3.10: Cipher structure.

```
AES_Cipher(byte input[4*Nb], byte output[4*Nb], word  W[Nb*(Nr+1)])

begin
   byte state_Matrix[4,Nb]
   state_Matrix = input
   AddRoundKey(state_Matrix, W[0, Nb-1])                    // See Sec 2.7.4.

   for round = 1 step 1 to Nr–1     // round 1 to 9,11 or 13
         SubBytes(state_Matrix)                             // See Sec.2.7.1
         ShiftRows(state_Matrix)                            // See Sec.2.7.2
         MixColumns(state_Matrix)                           // See Sec.2.7.3
        AddRoundKey(state_Matrix,W[round*Nb, (round+1)*Nb-1])
   end for

   SubBytes(state_Matrix)    // last round ,Nr=10 , 12 or 14
   ShiftRows(state_Matrix)
   AddRoundKey(state_Matrix, W[Nr*Nb, (Nr+1)*Nb-1])
   Output = state_Matrix
end
```

Figure 3.11: Pseudo code for cipher.

3.7.1 Sub bytes transformation function

The Sub Bytes or substitution transformation is only non-linear byte substitution which treats all byte to the state independently; by utilize S-Box table that contains 16 columns indexed from 0 to F in hexadecimal values and 16 rows indexed from 0

to F in hexadecimal, this table consist of values ranging from 0 to FF in hexadecimal values. The substitution process takes input from the 8-bit,the last nibble is used as index to column, while, the first nibble of byte uses as index to row, Then substitutes the value of the intersection between the row and column, and as a result, yield of 8 different bit data. The goal of this mission is to create a state of confusion. Table S-Box is the most expensive component in a complex and important AES. It is regarded as the heart of the AES algorithm. It is also the most time-consuming element of force. S-box is invertible table and constructed by composing two transformations.

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

The substitution mission utilized the S-box; this box is presented in hexadecimal format as clarify in Figure 3.17. It works as follows: -

for instance, if input value S1,2 = {65}, after that the new value could be specified by the crossroads of the row with index '6' and the column with index '5' (see Figure 3.17). This could produce the result value S'1, 2 = {4d}.

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

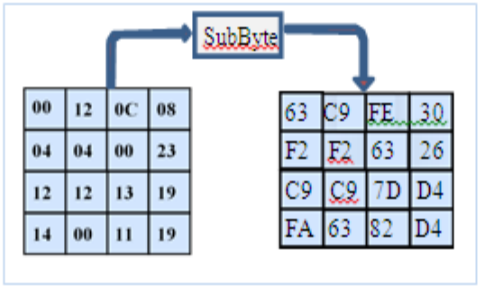


Figure 3.12: S-Box: substitution values. **Figure 3.13:** An example of sub byte.

3.7.2 Shift rows transformation function

The Shift Rows function supplies a simple replacement to the data, while the other transmutation contains replacements. The goal of this function is for providing Deploy values between columns to the state.

These transmutation operations the State by cyclically shifting each row of 0, 1, 2 and 3 of the state with diverse numbers of bytes like:-

- 1st row is unchanged
- 2nd rowcircularshifted1byte to left
- 3rd row circularshifted2bytestoleft
- 4th row circularshifted3bytestoleft

The shift rows process exchanges an individual byte from one column to other that is a linear road of a multiple of 4 bytes. This spread guaranteed which the 4 bytes which belong to one column diffusion out over each columns in the state. Watch 3.19 and 3.20.

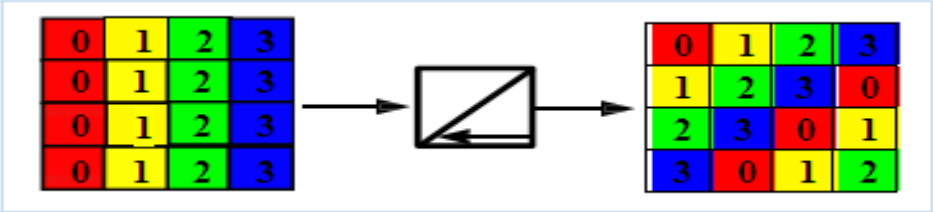


Figure 3.14: Shift rows.

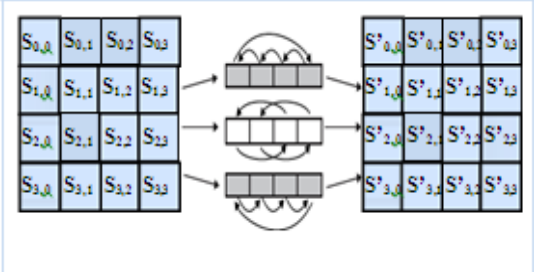


Figure 3.15: Shift rows transformation.

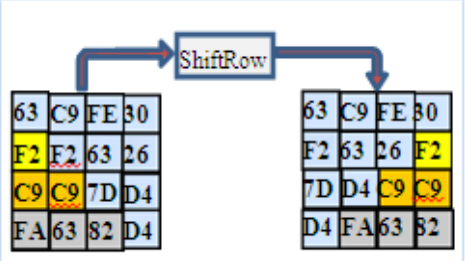
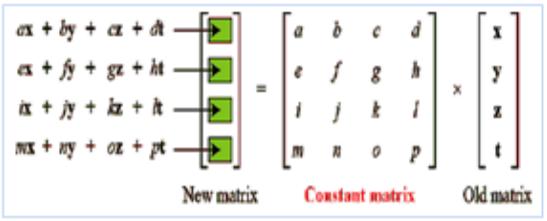


Figure 3.16: An example of shift row.

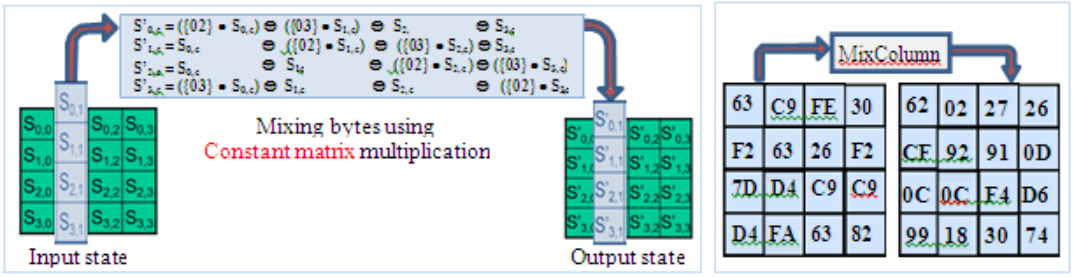
The Mix Columns transmutation function supplies a simple "replacement" to the data. The purpose to this function is for providing Spread of values between columns of the state. These lead the Spread of the cipher values**.** All byte of a column in the state is processed separately and set them in a new value which depends on each four bytes in which column in mixture with fixed matrix, for more info watch figure 3.22. It is designed as a matrix multiplication while all byte is deal with as a polynomial which makes use of arithmetic on GF (28) utilizing main polynomial m(x) =x8+x4+x3+x+1, the result is a fixed matrix, watching Figure 3.23. The fixed matrix utilized is depending on a linear code with maximal distance between code words – this giving good mixture to the bytes within all columns.



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

Figure 3.17: Matrix multiplication. **Figure 3.18:** Constant matrixes as value

The Mix Columns transmutation function adds an inter-byte transmutation which modifications the bits inside a byte, depend on the bits inside the neighboring bytes. These provide Spread at the bit level watching Figure 3.24. While dot (.) operator represent multiplication on Finite field GF (28). Figure 3.25 shows an example of how state values are replaced of the new values utilizing the Mix Columns technique.



63	C9	FE	30
F2	63	26	F2
7D	D4	C9	C9
D4	FA	63	82

62	02	27	26
CF	92	91	0D
0C	0C	F4	D6
99	18	30	74

Figure 3.19: A pply mix column. **Figure 3.20:** Example of mixcolumn.

3.7.4 Add round key transformation function

The Add Round Key transmutation function supplies a simple "substitution" to the data. This transmutation is a simple bitwise XOR process which added a Round Key to the current block to the state, like that.

$[S_{0,c}, S1,c,S2,c,S3,c]$XOR $[W_{round*4+c}]=[S'_{0,c}, S'1,c,S'2,c,S'3,c]$

Where 0<= c<4,0<=round<= Nrand[Wi]= keyword

All Round Key contain four words of 32-bit all provided by the key schedule. All of these four words to the round key XOR-ed with the columns of the State, AddRoundKey Working one column of the state at a time autonomous, to be clearer see Figure 3.26. The process is matrix addendum.

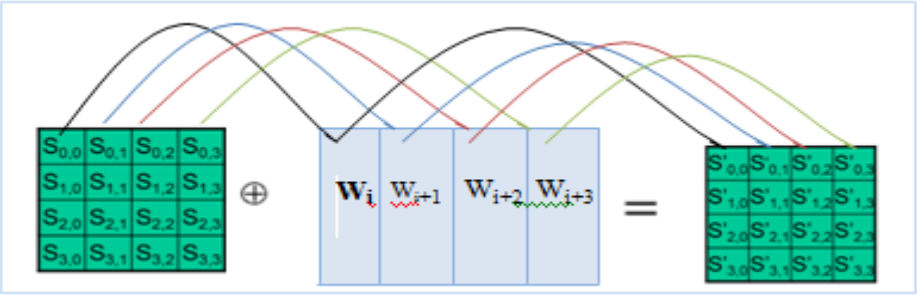


Figure 3.21: Add round key function.

This is only transmutation task which utilize the round key and obscures the output state. And it is a conformable task that means its inverse for decryption since bitwise XOR processes its own inverse, with reversed keys.

3.8 Decipher

AES algorithm in the decoding step is not conformable to the encryption step. That means not the same transmutation proceedings and structures are utilized for decipher and cipher. Nevertheless each transmutation procedures for ciphering were readily reversible and their inverse format is used in deciphering. But in the case of the cipher key, the expanded key used in reverse order in decipher algorithm. Prior decoding the encrypted block codes, and convert them to the state array as described previously. After adding the Round key to the state array in the first round, a number of rounds applied to the state array, number of rounds which applied for the state depends on the key size, then could be 10, 12, or 14 repetition to the key 128,192 and 256 bits respectively just such as cipher procedure. Then the eventual round added to the state array that is different from the Nr-1. Rounds in order that don't have Inverse Mix columns function. In the end the state transform to the output array that is the plaintext.

1-InvShiftRows (permutation).

2-InvSubBytes (substitution).

3-Add Round Key (round key addition).

4-InvMixColumns (mixing).

It described decoding in pseudo code in Figure 3.22 and described the structure of the decoder in Figure 3.23.

```
AES_Decipher (byte input [4*Nb], byte output[4*Nb], word W[Nb*(Nr+1)])

begin

    byte state_Matrix[4,Nb]
    state_Matrix = input
    AddRoundKey (state_Matrix, W[Nr*Nb, (Nr+1)*Nb-1])  // See Sec2.8.3

    for round = Nr-1 step -1 down to 1      // round 9,11 or 13 down to 1
        InvShiftRows(state_Matrix)          // See Sec 2.8.1
        InvSubBytes(state_Matrix)           // See Sec 2.8.2
        AddRoundKey(state_Matrix, W[round*Nb, (round+1)*Nb-1])
        InvMixColumns (state_Matrix)        // See Sec 2.8.4
    end for

     InvShiftRows (state_Matrix)    // last round 10 , 12 or 14
     InvSubBytes (state_Matrix)
    AddRoundKey (state_Matrix, W[0, Nb-

    1]) output = state_Matrix

end
```

Figure 3.22: Pseudo code for the decipher.

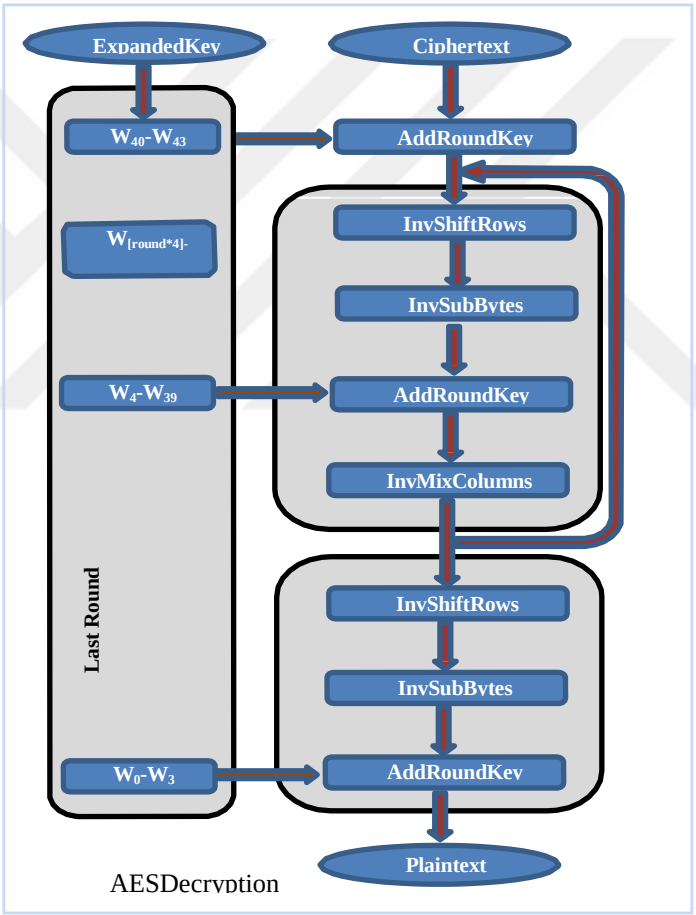


Figure 3.23: Decipher structure.

3.8.1 InvshiftRows transformation function

The InvShiftRows is the converse to the shift Row which recovers the permuted bytes to their authentic Location. The goal to this mission is for returning the propagation values amidst columns in the state. This kind of mission has the

influence to moveable bytes to top locations inside the row, whilst the top bytes wrap around into the down to the row. Watch Figures 3.29 and 3.30.

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

The process of inverse shift rows replacement the 4 bytes which followed for one column and distributed on all columns in the state to one column, see Figures 3.29 and 3.30.

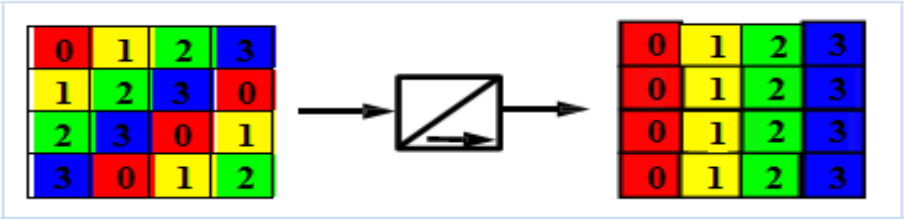
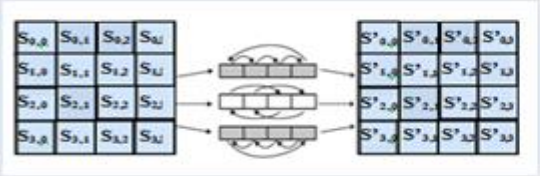


Figure 3.24: Inverse shift rows.

Figure 3.26 clarify for us of how a varied state value returning to its corresponding state utilizing In InShift Row technique.



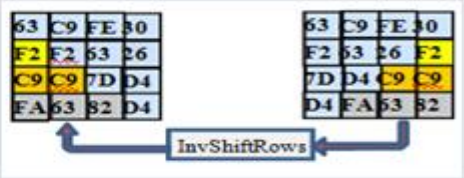


Figure 3.25: Inverse shift rows. **Figure 3.26:** An example of inverse shift rows.

3.8.2 Invsubbytes transformation function

The InvSubBytes function is the inverse of the replacement Byte function, that the inverse S-Box is applied for all byte of the State. The InvSubBytes function is too a non-linear byte replacement which deal with all byte to the state autonomous; with using Inverse S-Box table that too include 16 columns indexed from 0 to F in hexadecimal values and 16 rows indexed from 0 to F in hexadecimal values. This table includes values ranging from 0 to FF in hexadecimal values. The InvSubBytes process that takes input data of 8 bits uses the headmost nibble as index to row,

whilst the latest nibble is used as index to column, after that exchange the current value with intersection between row and column; as an outcome.

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

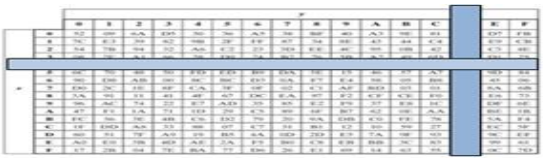


Figure 3.27: Inverse s-box values.

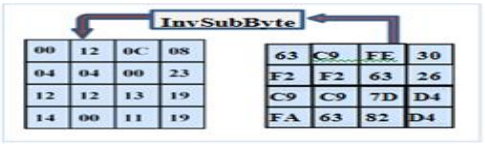


Figure 3.28: An example of inverse sub bytes.

The InvSubBytes function utilized the inverse S-box; this box is presented in hexadecimal format clarify in Figure3.32 and it doing as follows:-

For example, if input value S1,2 = {4d}, after that the new value could be specified by the intersection of the row with index '4' and the column with index 'd' (you can watch Figure 3.32). This could manufacture the output value S'1, 2 = {65}, that means it return to original value. The task applies the Inverse S-box for all byte of the State. for more clear you can see Figure 2.33 an example of how state values are exchanged return to the original utilizing the inverse S-Box technique.

3.8.3 Add round key transformation function

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

3.8.4 Invmixcolumns transformation function

The InvMixColumns function is the inverse of the Mix Columns function which restores the exchanged bytes to their original values. The goal of this mission is to restore the diffused values between columns in the state. all byte to a column in the state is processed separately and set it again into its original value which is rely on each four bytes in which column in combination with the inverse fixed matrix, for more info you can watch Figure 3.29.

Fixed matrix, for more info you can watch Figure 3.29.

It was designed as a matrix multiplication wherever all byte is treated as a constant inverse of polynomial a-1(x) which makes use of arithmetic over GF (28), the result

is a fixed inverse matrix for more info you can see in Figure 3.35. The fixed inverse matrix used is based on a linear code.

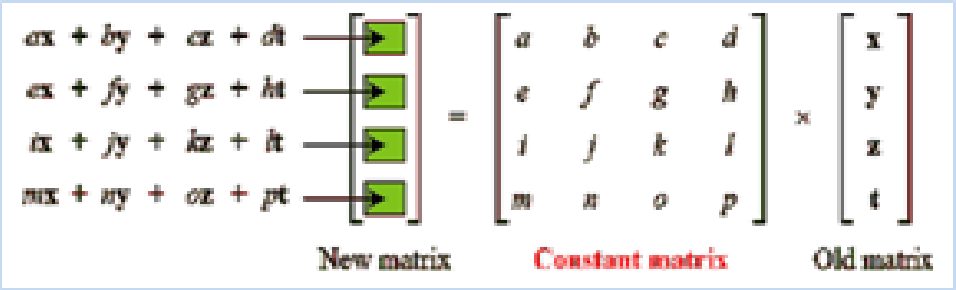


Figure 3.29: Mixing bytes using inv matrix multiplication.

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

Figure 3.30: Constant inverse matrix.

The InvMixColumns add up to an inter-byte transmutation which returns the bits into a byte, by virtue of the bits into the contiguous bytes. This return the diffusion at the bit level for more info you can watch figure 3.31. Wherever dot (.) operator: represent multiplication On GF (28). For more clear you can watch Figure 3.32 an example of how values are exchanged to the main values utilizing the InvMix Column technique.

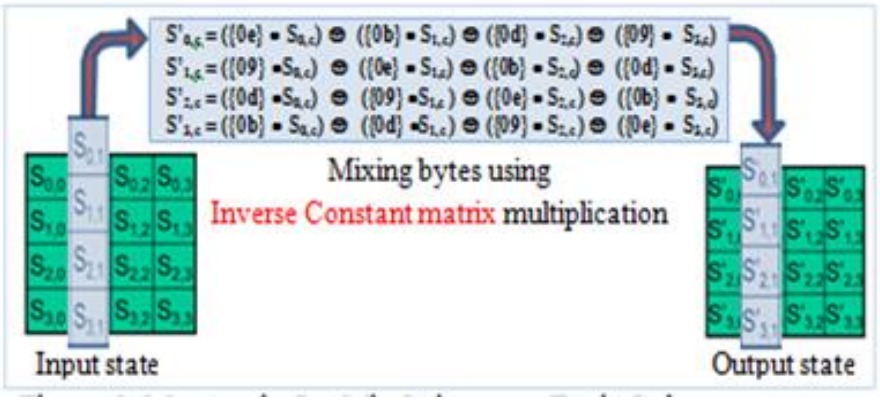


Figure 3.31: Apply inv mix column to each column.

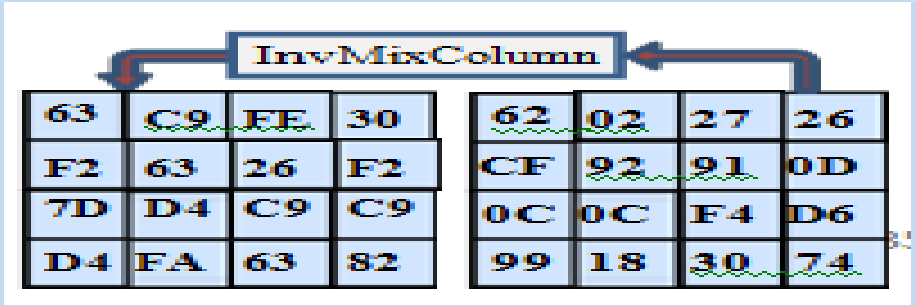


Figure 3.32: An example of inv mixcolumn.

4. IMPLEMENTATION

4.1 Project Aim

The purpose of this project is how it can encrypt -decrypt text and audio then comparison between them which is more secure and faster, Also which is more difficult to hack by hackers. To achieve these goals suggest a new plan to produce different encrypted for each session, even if you use the same key to encrypt the message**.** Teaching a group of different skills around how to make or organize MATLAB code to implement limited functions.

4.2 Encrypt-Decrypt Text By Using AES Algorithm

For encryption and decryption Text need to have key and plaintext ,or key and ciphertext then change binary code to Ascii code . After that cross all AES levels begin from subBytes step, shift row, mixcolumns and finaly Add round key. For encryption and decryption Text used five operation modes(ECB test of AES-128, CBC test of AES-128, OFB test of AES-128, CTR test of AES-128, CFB1 test of AES-128, CFB8 test of AES-128, CFB128 test of AES-128, CFB128 test of AES-256). It get different result, talk a bout all of them in detail, then compared between them to determine. which one is better, by using some performance metrics for all operation modes, And to know which is the best one for using with AES algorithm. After encrypt-decrypt text with AES algorithm and using Matlab programming . Which used all type of mode operation . Result are showen for only 20 characters and then acomparision is made amongst all of them. When used the first operation mode(ECB test of AES-128).the 20 Chars ORIGIN are(6bc1bee22e409f96e93d7e117393172aae2d8a57). After encrypted the result (3ad77bb4d7a3660a89ecaf32466ef97f5d3d585).after that decrypted 20 chars again and the output (6bc1bee22e409f96e93d7e117393172aae2d8a57). When used second operation mode (CBC test of AES-128) for 20 chars, the 20 Chars ORIGIN are (6bc1bee22e409f96e93d7e117393172aae2d8a57). After encrypted the result are (7649abac8119b246cee98e9b12e9197d5086cb9b). After that decrypted 20 chars again and the result (6bc1bee22e409f96e93d7e117393172aae2d8a57). When used

(OFB test of AES-128) for 20 chars, the 20 Chars ORIGIN are (6bc1bee22e409f96e93d7e117393172aae2d8a57).After encrypted the result are (3b3fd92eb72dad20333449f8e83cfb4a7789508d). After that decrypted 20 chars again and the result (6bc1bee22e409f96e93d7e117393172aae2d8a57). When used (CTR test of AES-128) for 20 chars, the 20 Chars ORIGIN are (6bc1bee22e409f96e93d7e117393172aae2d8a57). After encrypted the output are (874d6191b620e3261bef686499db6ce986f66b). After that decrypted 20 chars again and the result (6bc1bee22e409f96e93d7e117393172aae2d8a57). When used (CFB1 test of AES-128) for 20 chars, the 20 Chars ORIGIN are (6bc1bee22e409f96e93d7e117393172aae2d8a57). After encrypted the output are(68b3a264f838f5f8c3101070d1ab4c2e22e7f950). After that decrypted 20 chars again and the result (6bc1bee22e409f96e93d7e117393172aae2d8a57). When used (CFB8 test of AES-128) for 20 chars, the 20 Chars ORIGIN are (6bc1bee22e409f96e93d7e117393172aae2d8a57). After encrypted the result are(3b79424c9cdd436bace9eed4586a4f32b9ded5). After that decrypted 20 chars again and the result (6bc1bee22e409f96e93d7e117393172aae2d8a57). When used (CFB128 test of AES-128) for 20 chars, the 20 Chars ORIGIN are (6bc1bee22e409f96e93d7e117393172aae2d8a57). After encrypted the result are (3b3fd92eb72dad20333449f8e83cfb4ac8a64537). After that decrypted 20 chars again and the result (6bc1bee22e409f96e93d7e117393172aae2d8a57). When used (CFB128 test of AES-256) for 20 chars, the 20 Chars ORIGIN are (6bc1bee22e409f96e93d7e117393172aae2d8a57). After encrypted the result are (dc7e84bfda79164b7ecd8486985d386039ffed14). After that decrypted 20 chars again and the result (6bc1bee22e409f96e93d7e117393172aae2d8a57).

Table 4.1: Comparison between Encryption / Decryption of Each Operation Modes.

N	Operation Mode	20 Chars ORIGIN	20 Chars ENCRYPT	20 Chars DECRYPT
1	ECB test of AES-128	6bc1bee22e409f96e 93d7e117393172aae 2d8a57	3ad77bb4d7a366 0a89ecaf32466ef 97f5d3d585	6bc1bee22e409f96 e93d7e117393172a ae2d8a57
2	CBC test of AES-128	6bc1bee22e409f96e 93d7e117393172aae 2d8a57	7649abac8119b24 6cee98e9b12e919 7d5086cb9b	6bc1bee22e409f96 e93d7e117393172a ae2d8a57
3	OFB test of AES-128	6bc1bee22e409f96e 93d7e117393172aae 2d8a57	3b3fd92eb72dad2 0333449f8e83cfb 4a7789508d	6bc1bee22e409f96 e93d7e117393172a ae2d8a57
4	CTR test of AES-128	6bc1bee22e409f96e 93d7e117393172aae 2d8a57	874d6191b620e3 261bef686499db6 ce986f66b	6bc1bee22e409f96 e93d7e117393172a ae2d8a57
5	CFB1 test of AES-128	6bc1bee22e409f96e 93d7e117393172aae 2d8a57	68b3a264f838f5f 8c3101070d1ab4c 2e22e7f950	6bc1bee22e409f96 e93d7e117393172a ae2d8a57
6	CFB8 test of AES-128	6bc1bee22e409f96e 93d7e117393172aae 2d8a57	3b79424c9cdd43 6bace9eed4586a4 f32b9ded5	6bc1bee22e409f96 e93d7e117393172a ae2d8a57
7	CFB128 test of AES-128	6bc1bee22e409f96e 93d7e117393172aae 2d8a57	3b3fd92eb72dad2 0333449f8e83cfb 4ac8a64537	6bc1bee22e409f96 e93d7e117393172a ae2d8a57
8	CFB128 test of AES-256	6bc1bee22e409f96e 93d7e117393172aae 2d8a57	dc7e84bfda79164 b7ecd8486985d3 86039ffed14	6bc1bee22e409f96 e93d7e117393172a ae2d8a57

4.3 Operation Mode

AES is a block encryption algorithm to encrypt / decrypt a constant block size of 128 bits. Subsequently, after dividing an arbitrary message to constant block size of 128 bits, any and stuffing the latest block with Phantom characters .Depending on the goal, the encrypted message that utilizes these blocks requires several ways for enciphering. The best operation modes between them. The best mode is - depending on your requirements, the general overview of all mod.

4.3.1 ECB - Electronic code book.

Is simpler ways for process, where plaintext divided into separate blocks. This way encrypts any block autonomous of the other blocks and encrypts only one block at a time. As a outcome, the value is exchanged such as a codebook. Each blocks of plaintext are encrypted by utilized the same key. The operation may be encrypted/decrypted equivalent. The arithmetical formula to electronic codebook encryptions (Ci=EK (Pi)) and for decryptions (Pi=DK (Ci)). Figure4.4 explains more about ECB. It is convenient when one block or too small number of data blocks for sent, such as encrypted session key with a master key. This situation is not suitable to each amount of data that can be seen in repeated sections into the text of the message is encrypted. Especially with messages or graphics that change a little bit. The congruous message blocks are encrypted to congruous cipher text blocks. They are not concealment data type completely. Furthermore, while in the message blocks are shuffled, inserted or till omitted. This does not impact the encryption / decryption of all data block to be used encrypted blocks independently. This way is not advice to utilize in cryptographic protocols absolutely.

Advantages: too simple, encryption and decryption could be run in a same time.
Disadvantages: terribly insecure.

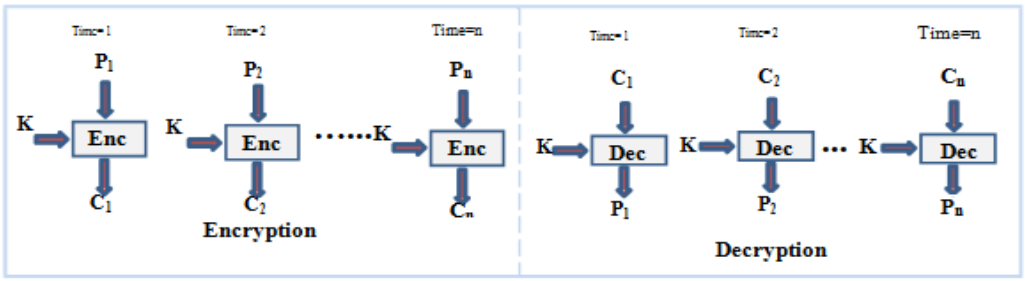


Figure 4.1: Electronic codebook (ECB) modes.

4.3.2 CBC - Cipher block chaining

The Cipher Block Chaining (CBC) makes the current block based on all cipher text blocks processing up to this point. To begin the operation, an Initial Vector (IV) it is utilized, which is generally a well known value: of ten times each 0's, otherwise agreed between the receiver and sender. The arithmetic formula for CBC encryptions$_i$=E_K (P_iXORC_{i-1}), C0=IV, while for decryptions$_i$=D_K (C_i) XORC_{i-1}, C0=IV. The CBC mode outperforms the problems of redundancy and requires

independence in ECB. Figure 4.5 shows the structure of CBC Mode. This mode is not suitable when you need to be encrypted and securely transfer large amounts of data, Provided that each data will be available in the beginning, like e-mail, Internet, etc. The sequence feature supplies an avalanche impact, this means that the encrypted message may not be alteration or reorder without completely and destruction of subsequent data. Likewise, all modulation affects each of the following blocks of cipher text. The primary obstacle is that the encryption is successive, whilst decryption could be parallelized well. Furthermore, It has the problem of that the IV, That should be known to the sender and recipient or either, is constant value or a one that should be sent encrypted in electronic codebook mode before the rest of the message. Likewise, the congruous message should be encrypted similarly except if the initialization vector is changed.

Advantage: It is secure, if utilized correctly, decryption parallel.

Disadvantages: encryption is not parallel, vulnerable for attacking if Validating are bad missing. However if using correctly, it's too good.

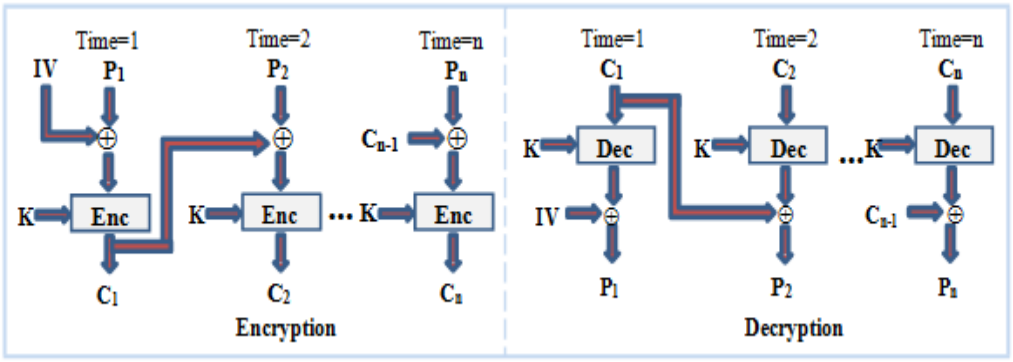


Figure 4.2: Cipher block chaining (CBC) modes.

4.3.3 Cipher feedback (CFB) mode

If data were only available as a bit / byte at a time, for example, a terminal session, the value of the sensor, etc., and then deal with the message as a bit stream. Then you have to use other methods to encrypt it so that no delay information. The idea is to use a block cipher mainly random number generator and the combination of these bits "random" with the message. Changing the block cipher to stream cipher concurrent. To begin the operation, an IV is utilized. After that use the cipher text as input to the next feeds each cipher block and return to the next stage. The procedure,

known as CFB-64orCFB-128mode (rely of the block size of the cipher utilized, e.g .,DES or AES,respectively).The arithmetical formula to CFB encryption is $C_i=E_K(C_{i-1})XORP_i$,while for decryptions $P_i=E_K(C_{i-1})XORC_i$.Figure4.6shows the structure of Cipher Feedback(CFB)Mode. This mode is suitable amounts of stream directed data to authentication use. This is an appropriate normal flow mode when reach data in bits / bytes. So long as it could keep up to the input, are encrypting every 8 bytes. It is used in the encryption mode of block ciphers, when both parties. It as well let the random arrival to cipher text and the congruous message is encrypted similarly except if the IV modifications. The potential issue lies in the fact that any damaged bit will destroy value in the current and future units. And therefore one should either use a more reliable network transmit layer or an OFB. Also encryption could not be parallelized whilst decryption may be parallelized.

Advantages: small footprint, parallel decryption.

Disadvantages: not ordinarily executed or utilized.

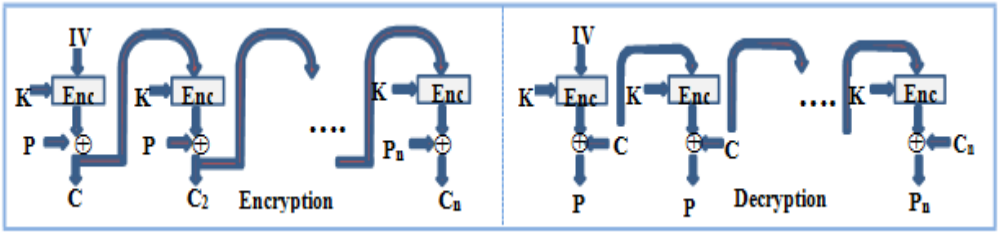


Figure 4.3: Cipher feedback (CFB) modes.

4.3.4 Output feedback (OFB) mode

Output Feedback. However don't have warranty which E (E (m, k), k) for each independently secure block cipher - there could be peculiar interactions between internal priorities which do not been studied correctly. If executed in a method which supplies partial blocks feedback (i.e. just part to the preceding block is purchased forward, with several constant or poorly random values to the other half) after that other troubles appear, like a short key stream cycle. Generally it must to avert Output Feedback (OFB) Mode. The arithmetical formula for encryption is $C_i=P_iXORO_i$, while for decryption is $Pi=C_iXORO_i$, where $O_i=E_K(O_{i-1})$, O0=IV.Figure 4.7shows the structure of OFB Mode.

Advantages: Key stream could be computed in advance, hardware implementations are very fast.

Disadvantages: The security model is questionable, several configurations lead to a short key stream cycles.

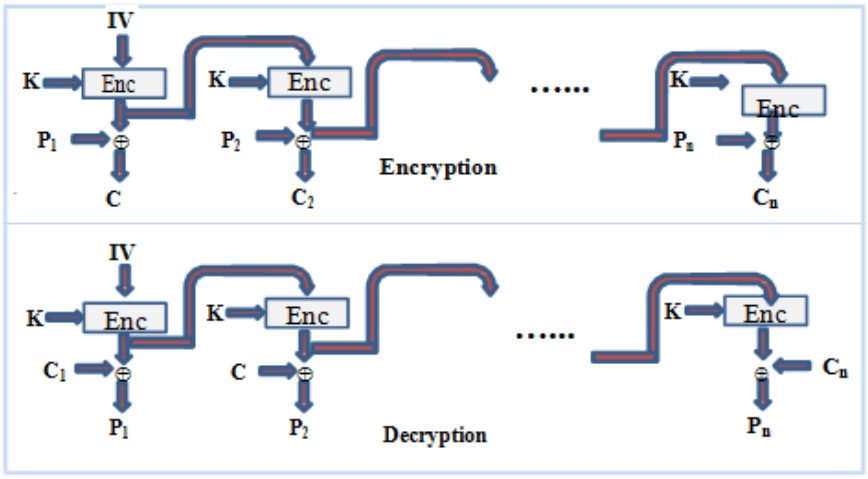


Figure 4.4: Output feedback (OFB) modes.

4.3.5 Counter (CTR) mode

This mode is like to Output Feedback (OFB) Mode however the value of the encryption counters, rather than the value of reactions. Though it was suggested few years ago, and but for decryption is Pi=(EK(i))XORCi. It is utilized for applications in asynchronous transfer mode (ATM), IP security (IPSec) and network security. The CTR may be all function that produces a concatenation, which are not secured to be iteration for a period time, while equal to the block size in order to use. The only condition is that the counter value should be various to any block. Usually, it is initialized counter to several values and after that Increasing by 1 to all subsequent block. Figure 4.8 see the structure of Counter Mode. The CTR mode is suitable to utilize at high-velocity network, may be executed in HW / SW and it works expeditiously in parallel, could before the operation the resulting values at the beginning of encryption needed, may get random reach for encrypted blocks, and just. This situation it is totally appropriate to work on a multi- processor machine where blocks may be enciphering in parallel. While such as OFB, it has the problem Never again the same key together with counter value.

Advantages: It is secure if using correctly, parallel decryption / encryption. Disadvantages: Not very secure model, however it's mostly considered to be secure.

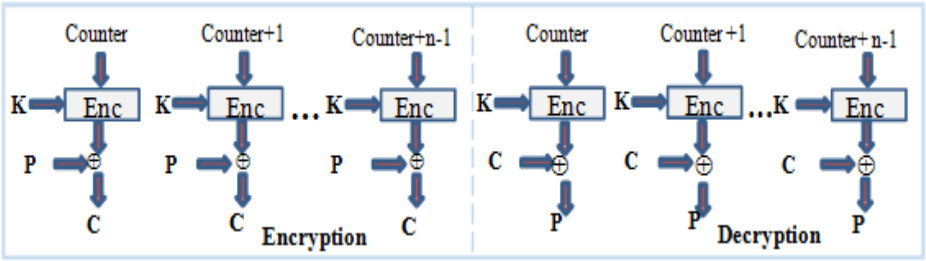


Figure 4.5: Counter (CTR) Mode.

For encryption and decryption audio. at the beginning used key type AES-128, not used key type AES-192 or AES-256 ,because it need a lot of time. It is used wave format for encryption\decryption voice by using AES algorithm; the standard sample rate used for audio is 44.1 kilohertz (44,100 hertz). This means that every second on the CD-ROM containing 44 100 individual samples. When samples are taken from an analog audio signal, such as sound, at a rate of tens of thousands of times per second, it may be digital recording is virtually identical to the original analog sound [30].it can used another sample rate not 44,1 kilohertz, but the voice will not be very clear therefore we used 44,1 kilohertz. Recording standard voice or record one person voice by using microphone .After that put this voice in function that responsible for saving voice in Matlab programming. Then encrypt that part of voice. And after that decrypt the same part of voice .When decrypt voice must that part of voice same the initial voice. Before we done encrypted. Then compare between them by using two shapes. In the shapes the blue line represented plain text data and green line represented cipher text data. if look at two shapes , it notes the second shape very changed if compare it with first shape .after used encryption process by using AES algorithm ,and this prove that the AES algorithm is very secure especially when used with audio and more secure than other algorithm like DES . Using AES algorithm, for this process with using Matlab code. Because, it is very difficult to hack by the hackers. It also faster than DES. Because DES has been hacked before in 1997 by the hackers. Selected Matlab programming instead of C++ and Java and other programs. Because Matlab programming is easy for using and also not need to beginning from zero. Rather than C++ or java needs to begin from zero. It has a big range of libraries and toolboxes.

4.4 Merits of Utilizing MATLAB Instead Of Other Programming like C, C# Or Java.

1. It has the ability to plot graphical comprehensive.

2. Utilizing MATLAB is more compatible than other programs like C# and Java, in order to it has a big collection of libraries and tools; ;moreover, for programming, it have not necessary for beginning from zero in order to MATLAB previously has functions for performing limited missions whilst some programs like C# should needs for beginning from zero.

3. Each MATLAB advantages give elasticity for processing encrypt -decrypt by using AES Algorithm.

4. And also used audio with AES algorithm encrypted and decrypted voice and the result as below.

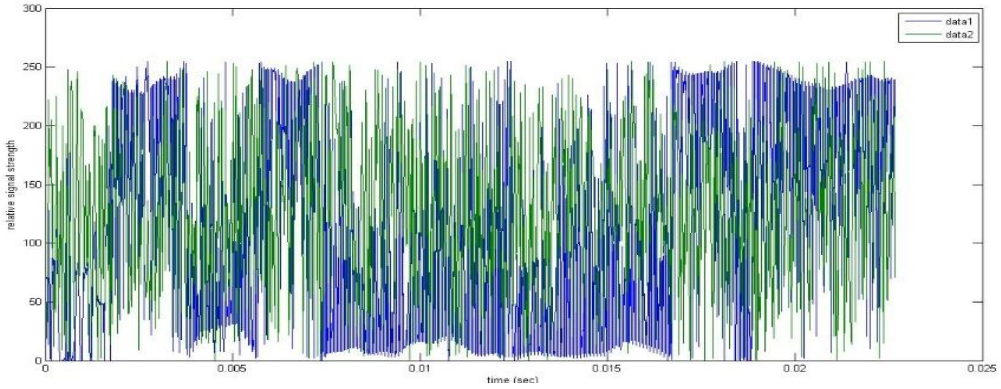


Figure 4.6: Before Encryption audio with AES algorithm.

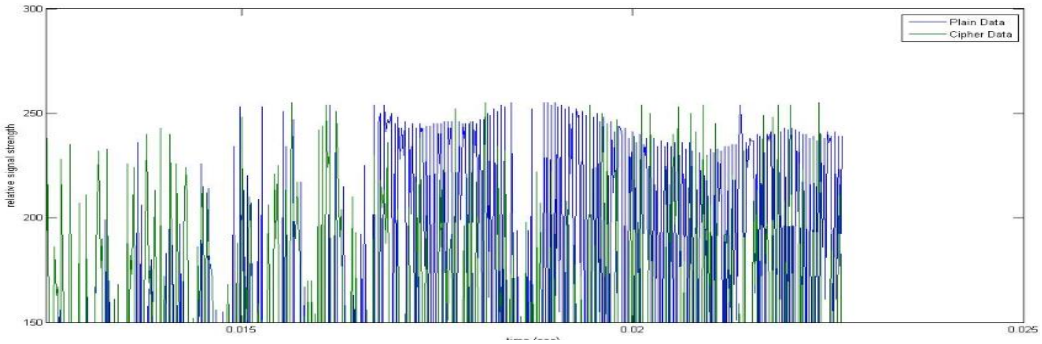


Figure 4.7: Encryption audio with AES algorithm.

Figure 4.7 shows after encryption process for audio, will note the result is very different if compared to the Figure 4.6 before encryption process, this prove that the voice is very difficult to hack by the hacker. Even more difficult than text.

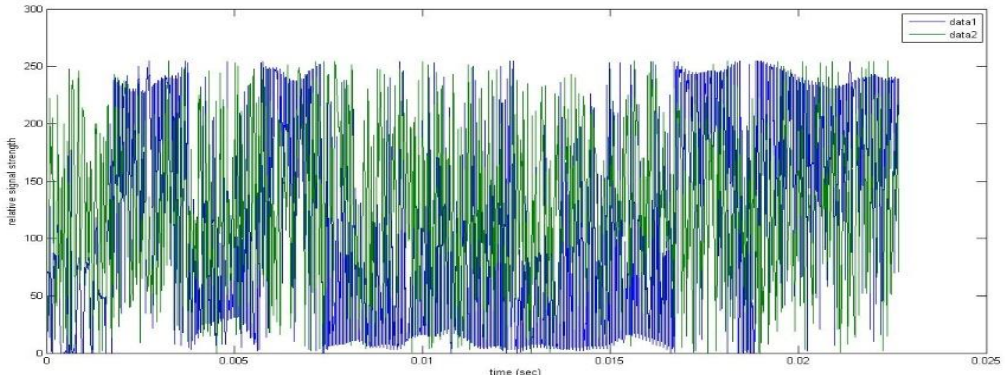


Figure 4.8: Decryption audio with AES algorithm.

4.5 A comparative Study of Encrypted-Decrypted Text and Voice Using AES Algorithm According to:

1- Speed.

2- Security.

3- Time Duration.

According to speed for encryption / decryption text is faster than audio because for encryption and decryption audio have many data. But text has less data if compared with voice (Nalini et al., 2009). According to security audio is more secure than text and also more difficult for hacking by the pirates (Muhammad Usman and Usman Akram, 2014). According to time duration text is better than audio .because text need less time for encryption and decryption if compared to voice (Gong et al., 2012).

Table 4.2: Comparison between Encryption\Decryption Texts and voice.

N	encryption / decryption Text by using AES	encryption / decryption Voice by using AES
1-	Less data if compare it with Voice.	More data if compare it with Text.
2-	For encryption\decryption Text need small time duration if compare it with Voice.	For encryption\decryption Voice need more time duration if compare it with Text.
3-	Not need microphone for encryption \ decryption	need microphone for encryption \decryption
4-	It is difficult to hack by the hacker	It is more difficult than Text to hack by the hacker because has a lot of data more than Text

5. CONCLUSION

5.1 Conclusion

In this project voice and text has been encrypted \decrypted by using AES algorithm, It used five operation modes .AES - CTR mode of -128, AES-ECB-mode of -128,AES -CBC mode of -128 and AES -OFB mode of -128,AES-CFB - mode of 128.after that we compared between them ,we cannot say which one is better than other .because each of them has advantages and disadvantages ,but after we used five operation and we got different results. The AES-Counter mode of-128 is better than other mode for using text. Used Matlab code in this project because it has the ability to plot graphical comprehensive. It is more compatible than other programs like C# and Java, in order to it has a big collection of libraries and tools; moreover, for programming, it have not necessary for beginning from zero in order to MATLAB previously has functions for performing limited missions whilst some programs like C# should needs for beginning from zero. Used each mode operations in this thesis, get different results from them. First used ECB - Electronic Code Book. This mode is too simple for, encryption and decryption could be run in a same time. However, the disadvantage about this mode operation is widely insecure. But the second one CBC - Cipher Block Chaining. This mode is Secure when utilized properly, parallel decryption. But the bad points about this mode are, No parallel encryption, Vulnerable to attacks when validating are bad/ missing. However when using right, it is too perfect. The third one OFB - Output Feedback. The advantages Key stream could be computed in advance, hardware implementations are very fast. However the bad things about this mode are Security model is doubtful, because the algorithm is not very difficult and several configurations lead to short key stream cycles. The forth operation mode CFB - Cipher Feedback. Another stream cipher mode, similar completely to CBC performed backwards. The advantages about this mode are, Small footprint, and parallel decryption. But the disadvantages are not ordinarily executed or utilized. The last operation mode is CTR - Counter Mode. It is very good for Secure while done right, parallel decryption and encryption. But the bad points about this mode are not a lot of. Some question the security of the "concerning

plaintext" model however it's mostly considered to be secure. When encrypted and decrypted audio by using the AES algorithm. It used key type AES-128 Obtained result but needed more time than text for encryption and decryption. Because audio has a lot of data. Therefore, need more time than text for encryption and decryption (Cyberwarzone 2015). According to speed for encrypt and decrypt text is faster than audio because for encryption and decryption audio have a lot of data. But text have less data if compare with voice. Then little data need small time. According to security audio is more secure than text and also more difficult to hack by hackers. The Completion of this project is suggest a new plan to produce different encrypted for each session, even if you use the same key to encrypt the message. Teaching a group of different skills around how to make or organize MATLAB code to implement limited functions.

5.2 Future Work

1-Apply AES to video encryption and decryption then compared to voice.

2- Using many standards for comparing between encryption\decryption texts with encryption\decryption voice, because in this project we used only three standards speed, security and time duration. If we have used many standards the result will be more accurate and easier to comparative between audio and text. It is easy to know which one are the best mode operations among them by using AES algorithm.

3- The proposed scheme will be used with some other scenario, like if changed the first digit in key or plaintext or last digit in key or plaintext, or some others scenarios.

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

REFRENCES

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

Asghar, M. N., Ghanbari, M., Fleury, M., and Reed, M. J, 2015.Sufficient encryption based on entropy coding syntax elements of wave audio. Multimedia Tools and Applications, 74, 23, 10215-10241.

Abomhara, M. A. S, 2010. Enhancing selective encryption for wave audio using advanced encryption standard. International Journal of Computer Theory and Engineering, 2, 2, 223-229.

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

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

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

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

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

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

Haskell, B. G., Puri, A., and Netravali, A. N, 1996. Digital audio: an introduction to Wave format. Springer Science and Business Media, 7, 1, 265-266.

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

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

Kulkarni, A. K, 2012. Implementation of fast inter-prediction mode decision in wave format audio encoder.

Mishra, D. C., Sharma, R. K., Dawar, M., and Hanmandlu, M, 2015. Two Layers of Security for Text and Voice by Matrix Affine Cipher with Two-Dimensional Discrete Wavelet Transform, 23, 4.

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

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

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

Richardson, I. E, 2004. Wave format Audio compression: audio coding for next-Wiley ISBN 0-470-84837-5, United Kingdom generation multimedia.

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

Rao, K. R., Kim, D. N., and Hwang, J. J, 2013. Voice coding standards: AVS China, H264/MPEG-4 PART 10, HEVC, VP6, DIRAC and VC-1. Springer Science and Business Media, Netherland.

Rijkse, K, 1996. Mp3 Voice coding for low-bit-rate communication. IEEE Communications magazine, 34, 1 2, 42-45.

Shi, C., and Bhargava, B, 1998. A fast Wave format voice encryption algorithm. In Proceedings of the sixth ACM international conference on Multimedia, 81-88.

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

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

Sikora, T, 1997. Wave format digital voice-coding standards. IEEE signal processing magazine, 14, 5, 82-100.

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

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

Varalakshmi, L. M, 2015. Studies on Security Enhancement Schemes for Wave format Voice (Doctoral dissertation).

Yi, K., Lee, Y. H., and Joo, J. H. 2016. Fast Voice Decoding by Bit Order Reversing and Its Application to Real-time w Encryption. International Journal of Multimedia and Ubiquitous Eng Wave format audio neering, 11, 3, 45-56 .

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

.

APPENDIX LIST

APPENDIX A: Appendix of the proposed code for ecoding voice and text.

APPENDIX B: Appendix of the proposed code of interface for voice encryption.

APPENDIX A: Appendix of the proposed code for encoding voice and text.

function [output] = aes(s, oper, mode, input, iv, sbit)

This is entry point function which is called all other function, you can use this function to encrypt and decrypt message using oper argument which indicates operation encrypted or decrypted

Arguments:

```
% s:            AES structure (generated by aesinit)
% oper:         operation:
%               'e', 'enc', 'encrypt', ‘E’, = encrypt
%               'd', 'dec', 'decrypt', ‘D’, = decrypt
% mode:         operation mode
%               'ecb' = Electronic Codebook Mode
%               'cbc' = Cipher Block Chaining Mode
%               'cfb' = Cipher Feedback Mode
%               'ofb' = Output Feedback Mode
%               'ctr' = Counter Mode
%         for counter mode you need external AES_GET_COUNTER ()
%               counter function.
% input:        plaintext/ciphertext byte-vector with length
%               multiple of 16
% IV:           initialize vector - some modes need it
%               ending initialize vector is stored in s.iv, so you
%    can use aes () repetitively to encode/decode
%               large vector:
%               out = aes(s, 'enc', 'cbc', input1, iv);
%               out = [out aes(s, 'enc', 'cbc', input1, s.iv)];
% sbit:         bit-width parameter for CFB mode
% output:           cipher text/plaintext byte-vector Example of calling this function:
```

Example

```
ct = aes(s, 'enc', 'ecb', pt);  %enc
 pt2 = aes(s, 'dec', 'ecb', ct);  %dec
```

```
1.  function s = aesinit(key)
```

AESINIT Generate structure with s-boxes expanded key, etc.

% key: 16 (AES-128), 24 (AES-192), and 32 (AES-256)

% items array with bytes of key

% s: AES structure for AES parameters and tables

```
2.  function [out] = aesencrypt(s, in)
```

Arguments:

% s: AES structure

% in: input 16-bytes vector (plaintext)

% out: output 16-bytes vector (ciphertext)

Example

```
pth = {'6b' 'c1' 'be' 'e2' '2e' '40' '9f' '96'...
       'e9' '3d' '7e' '11' '73' '93' '17' '2a'...
       'ae' '2d' '8a' '57' '1e' '03' 'ac' '9c'...
       '9e' 'b7' '6f' 'ac' '45' 'af' '8e' '51'...
       '30' 'c8' '1c' '46' 'a3' '5c' 'e4' '11'...
       'e5' 'fb' 'c1' '19' '1a' '0a' '52' 'ef'...
       'f6' '9f' '24' '45' 'df' '4f' '9b' '17'...
       'ad' '2b' '41' '7b' 'e6' '6c' '37' '10'};

pt = hex2dec(pth);
  keyh = {'2b' '7e' '15' '16' '28' 'ae' 'd2' 'a6'...
          'ab' 'f7' '15' '88' '09' 'cf' '4f' '3c'};
 key = hex2dec(keyh);
  s = aesinit(keyh);
```

output(idx) = aesencrypt(s, **pth**(1:16));

3. function [out] = aesdecrypt(s, in)

% s: AES structure

% in: input 16-bytes vector (ciphertext)

% out: output 16-bytes vector (plaintext)

Example

4. s.keyexp = key_expansion(s.key, s.s_box, s.rounds, s.mod_pol, s.aes_logt, s.aes_ilogt);

When aesinit(keyh) is called , it creates key expanasion round and save for encryption and decryption , the above function is used to create expanation key and store in s.keyexp;

5. function out = shift_rows(in, dir)

This function is used to shift row either left or right depended on the encryption and decryption method

Arguments

in: input 16-bytes vector

dir : 0 for left and 1 for right

6. function out = mix_columns(in, s)

This function is used to mixed column

Arguments

% s: AES structure, result of (ase init) function

dir : 0 for left and 1 for right

In: input 16-bytes vector

7. function p = poly_mult(a, b, mod_pol, aes_logt, aes_ilogt)

% Multiplication in a finite field

The following segment of code is used to iterate round which is required for encryption

```
for i = 1:(s.rounds - 1)
% SubBytes - lookup table
   state = s.s_box(state + 1);
   % ShiftRows
    state = shift_rows(state, 0);
    % MixColumns
   state = mix_columns(state, s);
   % AddRoundKey keyexp(i*4 + (1:4))
    state = bitxor(state, (s.keyexp((1:4) + 4*i, :))');
   end
```

APPENDIX B: Appendix of the proposed code of interface for voice encryption/decryption.

Code Explaining

The following is segment of codes which are used to do encrypt and decrypt on plain file.

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

Figure B.1: process to encrypted file

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

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

The following is a body this function

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

Figure B.2: The body of process encrypted file

Code in line 111 call a function AES_Encrypt, this function is response for encryption process. This function takes three parameters, first Bytes intend to encrypt, second password

is used for encryption and decryption process and final is a key length which is selected from UI.

Function signature

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

Body of the function

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

Figure B.3: The body of process encryption and decryption with final key length.

The above function is used to encrypt plain file and return array of encrypted byte. Same as EncrypFile, the is DecrypFile function which is used to decrypt file which encrypted by above function, the following is signature of Decrypt File function.

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

Body of the function

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

Figure B.4: The of process decrypted file.

In line 117 the is a function called AES_Decrypt which is response for decryption process. This function has three parameters, first byte to decrypt, second password which has been used for encryption and final is key length, and the following is signature of the function.

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

Body of the function

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

Figure B.5: The body of process decrypted file

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

CURRICULUM VITAE

Name and Surname: SABAH SALIH HUSSEIN

Birth Date and Place: 22.06.1986 DAHUK\IRAQ

E-mail address: **sabahgermavy@gmail.com**

EDUCATION:

Bachelor's Degree: Dahuk University, 2005-2009

Department of Computer science

Dahuk/ IRAQ

Master degree: Aksaray University, 2015-2017

Department Electrical Electronic and Computer Engineering

Aksaray/ TURKEY