Open University of Cyprus

Faculty of Pure and Applied Sciences

Information and Communication Systems

Master Thesis

Implementation and Performance Evaluation of Cryptographic Algorithms.

Nicolas Nicolaou

Supervisor
Dr. Petros Nicopolitidis

May 2017

Open University of Cyprus

Faculty of Pure and Applied Sciences

Information and Communication Systems

Master Thesis

Implementation and Performance Evaluation of Cryptographic Algorithms.

Νικόλας Νικολάου

Επιβλέπων Καθηγητής
Dr. Πέτρος Νικοπολιτίδης

Η παρούσα μεταπτυχιακή διατριβή υποβλήθηκε προς μερική εκπλήρωση των
απαιτήσεων για απόκτηση μεταπτυχιακού τίτλου σπουδών
Στα Πληροφοριακά και Επικοινωνιακά Συστήματα
από τη Σχολή Θετικών και Εφαρμοσμένων Επιστημών
του Ανοικτού Πανεπιστημίου Κύπρου.

May 2017

Summary

Network Security is a fast growing sector of computer networks. The rapid development of technology and especially in the field of networks requires research and study in several areas. The information data that pass through computer networks must be fast and secure to have a reliable system. Security is not only about protecting computer data, it is also important the availability and integrity of the network. While data is passing through a communication channel is more vulnerable to malicious attacks. To protect the data must have a strong encryption algorithm.

Encryption in modern times accomplished with the use of algorithms. To encrypt or decrypt a data algorithms have to use a secret key combination with a mathematical equation. The purpose of the key is to rearrange the relevant information into an unreadable data. The private key is the most important element in algorithms and must always remain secret.

An integrated online Information System requires a fast respond to exchange information between users. A direct access to an Information System is one of the essential elements of an organization's security (Confidentiality, Integrity, and Availability). A reliable Information System must have the right hardware for a fast respond and a strong algorithm to keep the data protected.

The purpose of this dissertation is to examine the various synchronous cryptographic algorithms and to evaluate their performance concerning their implementation in a particular software. The software compares the speed of the algorithms in different type scenarios and analyzes how fast can encrypt and decrypt the data.

Acknowledgments

I would like to express my gratitude to my supervisor, Professor Dr. Petros Nicopolitidis for the patience and trust that gave me the encouragement and help. Furthermore, I would like to thanks my family and my friends that support me and encourage me to accomplish the research.

Contents

Chapter 1
Introduction

It is amazing how networks and computer systems have advanced over the last few years. The network infrastructure has become more complex and the security control more challenging. Systems communicate with a different type of applications and the security of the network keeps the information safe. Malicious attacks can invade to the communication network collect data and obtain sensitive information. To ensure the reliability of the information, all the data that pass through the communication network must be encrypted.

Cryptography is an intersection of electrical engineering, computer science and mathematics. Cryptographic algorithms are complex mathematics that keep the system secure and protect the sensitive information. Over the years, different cryptographic algorithms have been developed to provide integrity, confidentiality and authentication of the data.

Cryptology is the art of science that studies the techniques for secure communication and information retrieval. The two main categories are Cryptography and Cryptanalysis. Cryptography came from the Greek "kryptós" it means "secret" and "graphein" it means "writing". In general, this subject studies the techniques of encryption and decryption that prevent third parties reading private messages. The other main category, Cryptanalysis is the study of analyzing the information systems and examine the hidden aspects of them from unauthorized entities.

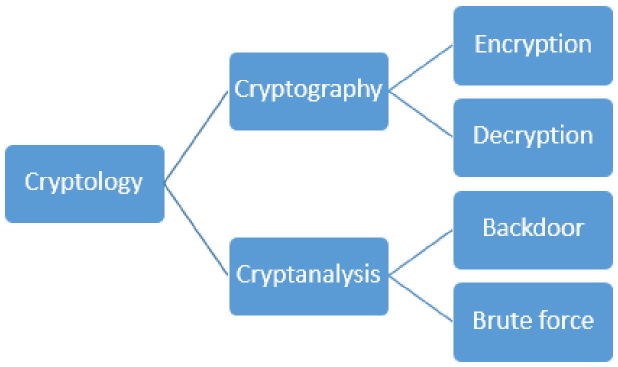


Figure 1. Cryptology Scheme

Encryption is the process of transforming a message into an unintelligible form using a cryptographic algorithm. The purpose of the encryption is to be unreadable by third parties. Decryption is the recovery process of the original message that performed by an authorized person. Encryption/decryption of a message performed based on a cryptographic algorithm and an encryption key. A cryptographic algorithm is a mathematical function or a method of transforming the data into an unreadable form. The key size is an important point for protecting the information data. Cryptographic algorithms are all about development of protocols that keep off any third party from accessing information, as well as the public against reading private information. Over the years, different cryptographic algorithms have been developed to address the issues of integrity, confidentiality and authentication of personal data. Cryptography is an intersection of electrical engineering, computer science and mathematics.

The encryption on computer systems can achieve through different types of algorithms. The algorithms can encrypt a plaintext or decrypt a ciphertext using a mathematical equation and a secret key. The main purpose of the key is to reclassify the data in such unique way that is incomprehensible to any intruder that would try to get access to the system. The key keeps the information secret and must always remain hidden.

A system must have a security level to keep the information safe. When two points A, B exchange information to the network the communication between them is safe if it satisfies the following properties:

- Confidentiality is all about giving the sensitive information to the right authorized users and prevent access to the wrong users.

- Integrity ensures that the content of the data cannot be altered by unauthorized people also maintains the accuracy, and trustworthiness of the data.

- Availability is to provide quality services to the users and keep the system available to provide the information data without any delay.

A system must have a number of different combination keys with a large cryptographic algorithm in order to avoid the intruders to have access. For example, if the cryptanalyst has the corresponding pair of original and encrypted text can try all the possible combination keys until he finds the right key. Otherwise, the cryptanalyst can decrypt the text with logical combinations until he gets the original and takes the right key. Typically, a key is a combination of bits in a series. Therefore, more bits in a key then become more complex. If a machine tests a key, every nanosecond can find all the possible combinations of 64-bit length key around 300 years. In the most cases, a strong cryptographic algorithm is the one is not able to break by conventional means in a reasonable time.

A modem information system with a high demand response has to exchange information instantly without any delay. A valuable network asset to the system must be a fast processing unit that can encrypt or decrypt the information. Direct access to the information is one of the three common basic policies for a secure organization (Confidentiality, Integrity, Availability). The performance of the system is not the only part of a fast processing, but equal emphasis must be the algorithm type, the key size and the cipher mode of the algorithm.

The purpose of this research is to study well-known symmetric algorithms with various cipher modes on a different type of data sizes and examine their performance in a particular program. The parameters will monitor the speed of the algorithms for encryption and decryption on a particular system.

The symmetric algorithms are with a single secret key that both sides use it to encrypt or decrypt the data. The security the algorithms always depends on the secret key. Symmetric cryptosystems use a secure communication channel to exchange the secret key. The two cryptosystems must have the secret key in a secure way and must keep the key protected. If an unauthorized system can discover the key and identifies the algorithm, all the data can be readable.

Chapter 2
Symmetric Algorithms

Cryptography has two main sections, symmetric cryptography that uses a single secret key and the asymmetric cryptography that uses two keys the public key, and the private key. The algorithms DES, 3DES, EAS, RC2, comes under the category of symmetric cryptography. The symmetric algorithms have an equation $E_{\{P, K\}}$ and use the plaintext (P) that comes with the secret key (K) to produce ciphertext (C) as shown in figure 2. The plaintext along with the secret key pass to the algorithm as an input and then the algorithm provide the ciphertext of the particular plaintext. The ciphertext(C) can be converted back to plaintext (P), with the use of the right secret key (K) along with the ciphertext (C) added to the reverse equation $D_{\{C, K\}}$.

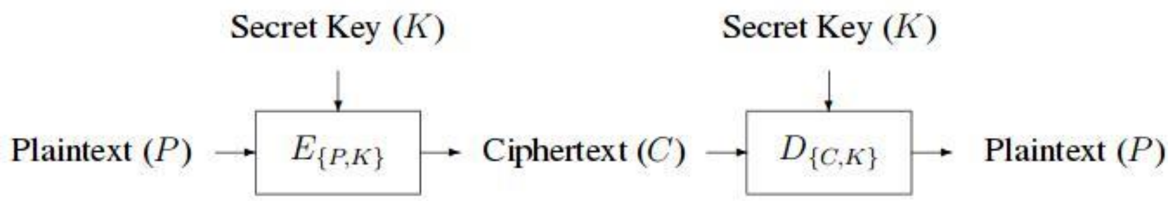


Figure 2. Symmetric Cryptography [1, p. 2]

2.1 Data Encryption Standard (DES)

Cryptography has two categories of ciphers, a block cipher, and a stream cipher. In the first category the data is encrypted or decrypted in small blocks, and in the second category the data is in the form of a stream and all bits are encrypted or decrypted individual. A stream cipher is not so often in use on modern cryptographic algorithms. DES algorithm is a 64-bit block cipher, therefore in the input of the algorithm, the data is divided into 64-bit blocks and moves to the encryption/decryption equation[1].

2.1.1 History of DES Algorithm

During 1960's, an IBM product division modified LUCIFER submitted the algorithm, referred to as the data encryption standard (DES)[2]. On 15 May 1973, National Bureau of Standards (NBS) published a request for creation of a standard encryption algorithm. On 27 August 1974, NBS again published an application for creation of a standard encryption algorithm. The International Business Machines (IBM) along with the help of National Security Agency (NSA) started creating an encryption algorithm. IBM modified and improved Lucifer to create an encryption algorithm with the name Data Encryption Standard. Therefore, Data Encryption Standard is a derivative of Lucifer cipher. On 17

March 1975, DES published for the purpose of comment in the Federal Register. Afterward, two workshops created the first in August 1976 and the second in September 1976 to discuss the mathematics used in DES, eventually become a US federal standard in November 1976. Next year on 15 January 1977 DES was published as a Federal Information Processing Standard (FIPS) is standard as FIPS PUB46 [3]. Federal Information Processing Standard are the standards made by US federal government, and they primarily used for computer security by US government contractors and agencies of US government which are not related to US Army. DES was reasserted first in 1983, then on 22 January 1988 as FIPS 46-1 and for the third time on 30 December 1993 as FIPS 46-2. In the coming years, DES broken multiple times. A project named DESCHALL succeeded in decrypting a text, which encrypted using DES publicly in June 1997. Later in July 1998, deep cracker by EFF successful break a key of DES in 56 hours. In January 1999, deep cracker along with a distributed network broke DES in 22 hours and 15 minutes. While on 25 October 1999 DES was reasserted for the fourth time as FIPS 46-3 and declare that Triple DES give preference over DES, but in the case of old systems the usage of simple DES was allowed [3].

Advanced Encryption Standard is known as AES published in Federal Information Processing Standard's (FIPS) standard FIPS 197 on 29 November 2001. Use of Advanced Encryption Standard (AES) started on 26 May 2002 effectively. As AES is much stronger than DES and is broken on 26 July 2004, a proposal made in Federal Register of US that DES FIPS 46-3 and removed from the Federal Register. Afterward, on 19 May 2005, National Institute of Standards and Technology (NIST) withdrew DES FIPS 46-3 [3]. National Bureau of Standards (NBS)'s name changed multiple times in history. Last time change in 1988 and it was as National Institute of Standards and Technology (NIST). In April 2006, a parallel machine based on Field Programmable Gate Arrays (FPGA) named COPACOBANA used to broke DES in almost nine days. In November 2008, another machine named RIVYERA, which came after COPACOBANA had much better performance in breaking DES and it broke DES in less than 24 hours [3]. Without any doubt, DES has been a very famous and widely used encryption standard in the past now AES is replacing it.

Lucifer uses a 128-bit key, while the key size of DES is 64-bit. From the 64-bit the 8-bit used for parity check, the rest 56-bit is for encryption. The size of 56-bit is small, and a brute force attacks can easily break DES algorithm. In the history of DES algorithm, the S-boxes made under dark conditions. The community discusses that National Security Agency made it in such way that they can keep a backdoor and can easily decrypt the data of the algorithm. The IBM denied the involvement of NSA in the creation of DES, but still today, it remains a controversy. United States Senate Select Committee on Intelligence, publish an unclassified detail in 1971 that NSA assured IBM for smaller key size use in DES was enough and the creation of S-boxes and further the finalize algorithm did not contain any mathematical and statistical loophole that can break DES algorithm. The IBM decide that a small key size for DES is appropriate for the commercial software. A declassified book by NSA on the history of cryptologic stated that NBS in 1973 gathered private companies for the creation of data encryption standard (DES). The initial offer was not so good as a result, NSA start to work on their encryption algorithm. The book further states that NSA's deputy director for research and engineering Howard Rosenblum found that Walter Tuchman was modifying Lucifer. Howard Rosenblum then brought Walter Tuchman to work with NSA for modifying Lucifer [8]. Thus, NSA starts working with IBM closely to improve the cryptographic algorithm. NSA made an effort to convince IBM to reduce the size of to 48 bit instead of 64 bit. In the end, both of them finalized 56 bit as the key size[4].

2.1.2 DES Structure

The Data Encryption Standard (DES) have 16 rounds with four different modes, to encrypt blocks individually or make each cipher block dependent on all the previous blocks. The initial and final permutations are straight Permutation boxes (Tables 1-2) that are inverses of each other. However, permutations they do not play any role for encryption or decryption. Are included to facilitate loading blocks in and out of mid-1970s 8-bit based hardware[5].

58	50	42	34	26	18	10	2
60	52	44	36	28	20	12	4
62	54	46	38	30	22	14	6
64	56	48	40	32	24	16	8
57	49	41	33	25	17	9	1
59	51	43	35	27	19	11	3
61	53	45	37	29	21	13	5
63	55	47	39	31	23	15	7

Table 1. Initial permutation [6, p. 10]

40	8	48	16	56	24	64	32
39	7	47	15	55	23	63	31
38	6	46	14	54	22	62	30
37	5	45	13	53	21	61	29
36	4	44	12	52	20	60	28
35	3	43	11	51	19	59	27
34	2	42	10	50	18	58	26
33	1	41	9	49	17	57	25

Table 2. Inverse Initial Permutation [6, p. 10]

Table 1 shows the Initial Permutation and how can be applied to the 64-bit block input. The first cell being in the table is the 58th bit of the input and the second cell is the 50th bit of input. The structure of the Final Permutation is the equivalent as the Initial Permutation, and the procedures are the same (Table 2).

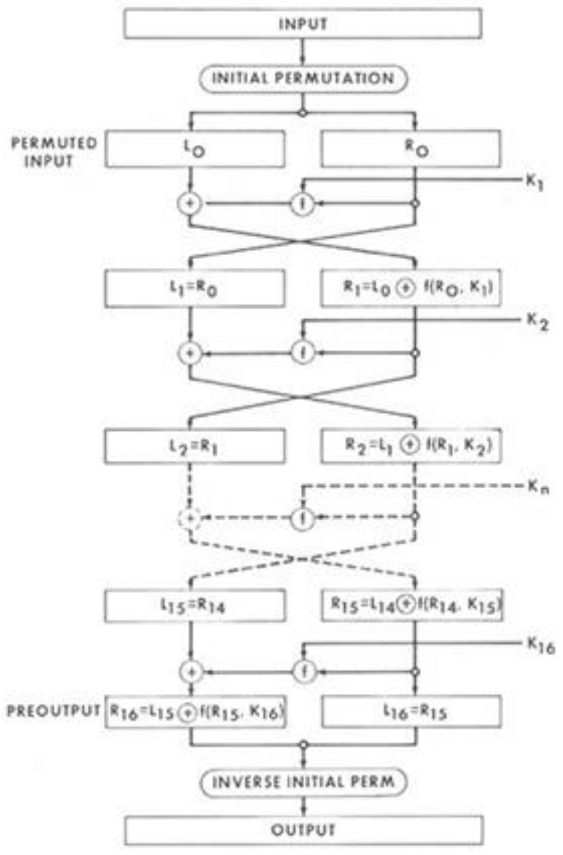


Figure 3. Enciphering computation [6, p. 9]

The 64-bit block of DES divided into two 32-bit blocks the left and the right respectively. The use of Feistel structure makes sure that both encryption and decryption are identical to each other. The only difference between encryption and decryption in DES is that all subkeys are in reverse order during the process of decryption. The symbol ⊕ is to show exclusive OR operation gate (XOR).

The Feistel function can work on 32-bit data at one time, and the structure consists four stages as follows:

1. Expansion
2. XOR with subkey
3. Substitution
4. Permutation

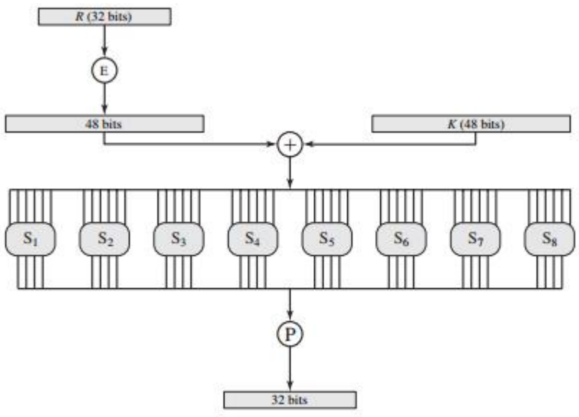


Figure 4. Feistel Structure [7, p. 83]

1-Expansion

An expansion permutation in "Table 3" is for expanding the 32-bit half block into 48-bit and prepared the duplication of the half bits of the input. The output of the 48-bit consists of 8 pieces where each piece contains 6-bit (8 * 6 = 48 bit). Out of this 6-bit, the 4-bit are respective input bits while the remaining 2-bit are a copy of an adjacent bit of input pieces present on either side.

32	1	2	3	4	5
4	5	6	7	8	9
8	9	10	11	12	13
12	13	14	15	16	17
16	17	18	19	20	21
20	21	22	23	24	25
24	25	26	27	28	29
28	29	30	31	32	1

Table 3. E BIT-Selection [6, p. 13]

2-XOR-with-subkey

There are sixteen rounds in DES algorithm, and for each round, uses a different subkey. The subkeys produced from the actual key.

3-Substitution

Substitution takes place in the S-boxes "Table 4", there are eight S-boxes in DES algorithm, each one takes 6-bit as an input and provide 4-bit as an output. The S-boxes are a vital and important part of DES security. The purpose of the S-boxes is to mix the cipher in such way to not remain linear and be secured. The table shows the substitution of eight S-boxes every table is different from the other.

S_1	**0000**	**0001**	**0010**	**0011**	**0100**	**0101**	**0110**	**0111**	**1000**	**1001**	**1010**	**1011**	**1100**	**1101**	**1110**	**1111**
00	14	4	13	1	2	15	11	8	3	10	6	12	5	9	0	7
01	0	15	7	4	14	2	13	1	10	6	12	11	9	5	3	8
10	4	1	14	8	13	6	2	11	15	12	9	7	3	10	5	0
11	15	12	8	2	4	9	1	7	5	11	3	14	10	0	6	13
S_2	**0000**	**0001**	**0010**	**0011**	**0100**	**0101**	**0110**	**0111**	**1000**	**1001**	**1010**	**1011**	**1100**	**1101**	**1110**	**1111**

00	15	1	8	14	6	11	3	4	9	7	2	13	12	0	5	10
01	3	13	4	7	15	2	8	14	12	0	1	10	6	9	11	5
10	0	14	7	11	10	4	13	1	5	8	12	6	9	3	2	15
11	13	8	10	1	3	15	4	2	11	6	7	12	0	5	14	9

S_3	**0000**	**0001**	**0010**	**0011**	**0100**	**0101**	**0110**	**0111**	**1000**	**1001**	**1010**	**1011**	**1100**	**1101**	**1110**	**1111**
00	10	0	9	14	6	3	15	5	1	13	12	7	11	4	2	8
01	13	7	0	9	3	4	6	10	2	8	5	14	12	11	15	1
10	13	6	4	9	8	15	3	0	11	1	2	12	5	10	14	7
11	1	10	13	0	6	9	8	7	4	15	14	3	11	5	2	12

S_4	**0000**	**0001**	**0010**	**0011**	**0100**	**0101**	**0110**	**0111**	**1000**	**1001**	**1010**	**1011**	**1100**	**1101**	**1110**	**1111**
00	7	13	14	3	0	6	9	10	1	2	8	5	11	12	4	15
01	13	8	11	5	6	15	0	3	4	7	2	12	1	10	14	9
10	10	6	9	0	12	11	7	13	15	1	3	14	5	2	8	4
11	3	15	0	6	10	1	13	8	9	4	5	11	12	7	2	14

S_5	**0000**	**0001**	**0010**	**0011**	**0100**	**0101**	**0110**	**0111**	**1000**	**1001**	**1010**	**1011**	**1100**	**1101**	**1110**	**1111**
00	2	12	4	1	7	10	11	6	8	5	3	15	13	0	14	9
01	14	11	2	12	4	7	13	1	5	0	15	10	3	9	8	6
10	4	2	1	11	10	13	7	8	15	9	12	5	6	3	0	14
11	11	8	12	7	1	14	2	13	6	15	0	9	10	4	5	3

S_6	**0000**	**0001**	**0010**	**0011**	**0100**	**0101**	**0110**	**0111**	**1000**	**1001**	**1010**	**1011**	**1100**	**1101**	**1110**	**1111**
00	12	1	10	15	9	2	6	8	0	13	3	4	14	7	5	11
01	10	15	4	2	7	12	9	5	6	1	13	14	0	11	3	8
10	9	14	15	5	2	8	12	3	7	0	4	10	1	13	11	6
11	4	3	2	12	9	5	15	10	11	14	1	7	6	0	8	13

S_7	**0000**	**0001**	**0010**	**0011**	**0100**	**0101**	**0110**	**0111**	**1000**	**1001**	**1010**	**1011**	**1100**	**1101**	**1110**	**1111**
00	4	11	2	14	15	0	8	13	3	12	9	7	5	10	6	1
01	13	0	11	7	4	9	1	10	14	3	5	12	2	15	8	6
10	1	4	11	13	12	3	7	14	10	15	6	8	0	5	9	2

11	6	11	13	8	1	4	10	7	9	5	0	15	14	2	3	12
S_8	**0000**	**0001**	**0010**	**0011**	**0100**	**0101**	**0110**	**0111**	**1000**	**1001**	**1010**	**1011**	**1100**	**1101**	**1110**	**1111**
00	13	2	8	4	6	15	11	1	10	9	3	14	5	0	12	7
01	1	15	13	8	10	3	7	4	12	5	6	11	0	14	9	2
10	7	11	4	1	9	12	14	2	0	6	10	13	15	3	5	8
11	2	1	14	7	4	10	8	13	15	12	9	0	3	5	6	11

Table 4. primitive functions S1,...,S8 [6, p. 17]

4-Permutation
The permutation, P-box reorganizes the 32-bit output that comes from the S-boxes in Table 5. The permutation in P-boxes is designed in such a way that the 4-bit output of each S-box will go to four different S-boxes in the next round.

16	7	20	21	29	12	28	17
1	15	23	26	5	18	31	10
2	8	24	14	32	27	3	9
19	13	30	6	22	11	4	25

Table 5. Permutation of half block of 32 bit [6]

Subkey-generation-from-main-key
The main 54-bit key includes the 8-bit that are only for parity check. The remaining 56-bit of the key are for the encryption or decryption of the data. The 56-bit use for permuted choice 1 in Table 6 and is divided into two halves of 28-bit. In all rounds, the halves rotated towards left by one or two bit. The rotation by one or two bit is different for each round "Table 8". A 48-bit subkey is by permuted choice to "Table 7". The 24-bit is from the left side, and the other 24-bit is from the right side. Different subkeys use for each round. The subkeys apply in the reverse order between encryption and decryption.

Left							Right						
57	49	41	33	25	17	9	63	55	47	39	31	23	15
1	58	50	42	34	26	18	7	62	54	46	38	30	22
10	2	59	51	43	35	27	14	6	61	53	45	37	29
19	11	3	60	52	44	36	21	13	5	28	20	12	4

Table 6. Permuted Choice 1 [6]

14	17	11	24	1	5
3	28	15	6	21	10
23	19	12	4	26	8
16	7	27	20	13	2
41	52	31	37	47	55
30	40	51	45	33	48
44	49	39	56	34	53
46	42	50	36	29	32

Table 7. Permuted Choice 2 [6]

Round number	Number of left rotations
1	1
2	1
3	2
4	2
5	2
6	2
7	2
8	2
9	1
10	2
11	2
12	2
13	2
14	2
15	2
16	1

Table 8. Rotation in rounds [6]

2.2 Triple Data Encryption Standard (3DES)

The Triple DES in figure 5 is derived from the DES algorithm and use the same structure as the simple DES. The difference is that in triple DES, the plaintext pass through the encryption process with key-1 then go to the decryption process with key-2 and then again pass for encryption with the key-3 for the final output. There are two types of Triple DES the "2 Key Triple DES" and "3 Key Triple DES". The "2-Key DES" contains two different keys k1 and k2. The k1 is to begin to encrypt the data, and then the k2 decrypt the data, and again k1 encrypt it back. The main key length for this process is 112-bit. The "3 Key

DES" contains three different keys k1, K2, and k3. In the beginning, the data encrypted with k1 then k2 for decryption and the last process use k3 to encrypt the data. In this procedure, the length of the main key is 168-bit.

3des use the DES three times to encrypt the data. The 3DES algorithm came out when DES algorithm becomes weaker due to advanced technology. Triple DES ensure backward compatibility with simple DES.

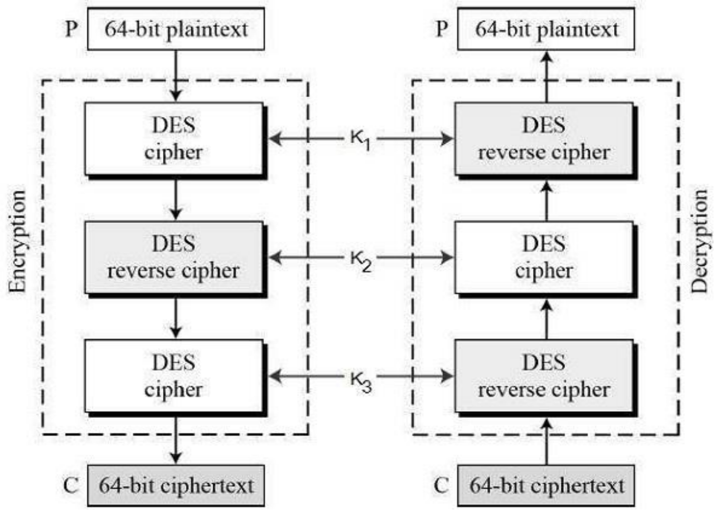


Figure 5. Triple DES

2.3 Advance Encryption Standard (AES)

The Advance Encryption Standard (AES) is a symmetric cryptography algorithm that uses a single secret key to encrypt the data and with the same secret key decrypt it back (figure6). AES is a secure method to communicate and transfer information by encrypting and decrypting the data by transmitting files in two points different protocols like HTTPS, FTPS, SFTP, WebDAVS, OFTP, the data can encrypt with AES algorithm.

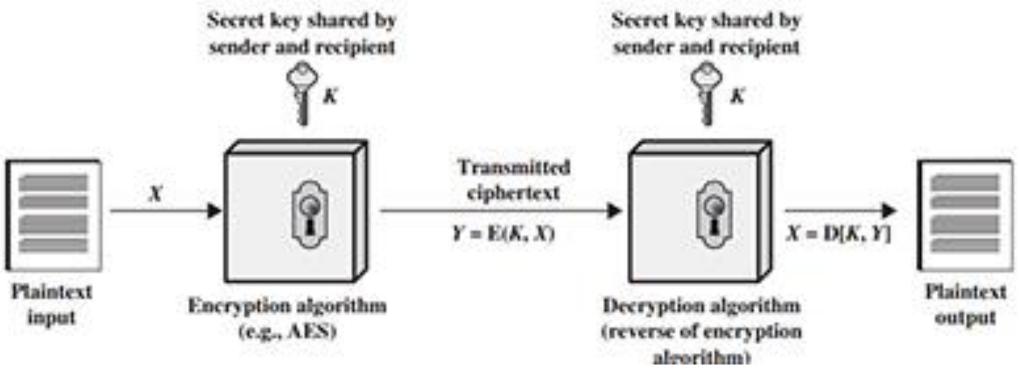

Secret key shared by sender and recipient
K
Secret key shared by sender and recipient
K
X
Transmitted ciphertext
Y = E(K, X)
X = D[K, Y]
Plaintext input
Encryption algorithm (e.g., AES)
Decryption algorithm (reverse of encryption algorithm)
Plaintext output

Figure 6. Simplified Model of Symmetric Encryption [7, p. 33]

AES is a block cipher that encrypts and decrypts the data in blocks of 128-bit with the use of cryptographic keys. The ciphers secret is the same key for encrypting and decrypting situation, so both times must know and use the same secret key. DES use fiestel structure while AES is using a substitution-permutation network. What the AES came from Square block cipher is invented by Joan Daemen and Vincent Rijmen, it has a block length and key length of 128 bits [8]. AES have three different key sizes 128 bit, 192 bit and 256 bit. Moreover, depend on the key size; AES can have different rounds 10, 12 and 14 respectively.

Throw time DES algorithm became more vulnerable to brute force attacks because of the small key size. It was time to create a new algorithm standard that can effectively replace "Data Encryption Standard" (DES). National Institute of Standards and Technology (NIST) start a process in 1997 to choose a new cryptographic algorithm that ended in 2000. On 2 January 1977, National Institute of Standards and Technology (NIST) announced the new algorithm "Advance Encryption Standard" (AES), that it would replace the DES algorithm. Along with a larger keyspace, AES had to be a 128-bit block cipher; that is, process 128-bit blocks of plaintext input at a time. AES also had to support 128-, 192-, and 256-bit key settings and be more efficient than DES.[9, p. 140]. On June 15, 1998, twenty-one algorithms submitted to NIST and after a review, NIST determined that 15 of these meet the minimum acceptability requirements. To discuss these submissions, NIST held a conference in August 1998 named as AES1. Different cryptographers gathered to focus on 15 cryptographic algorithms to check them for security and analysis[10]. They review each algorithm with a systematic evaluation of the factors of security, efficiency in speed and memory usage, flexibility on low and high-end smart cards and other issues discussed in public comments. Later during August 1999, NIST announced they had shortlisted five algorithms from a list of fifteen algorithms. These five algorithms termed as "AES finalists." During second conference "AES2", algorithms received votes.

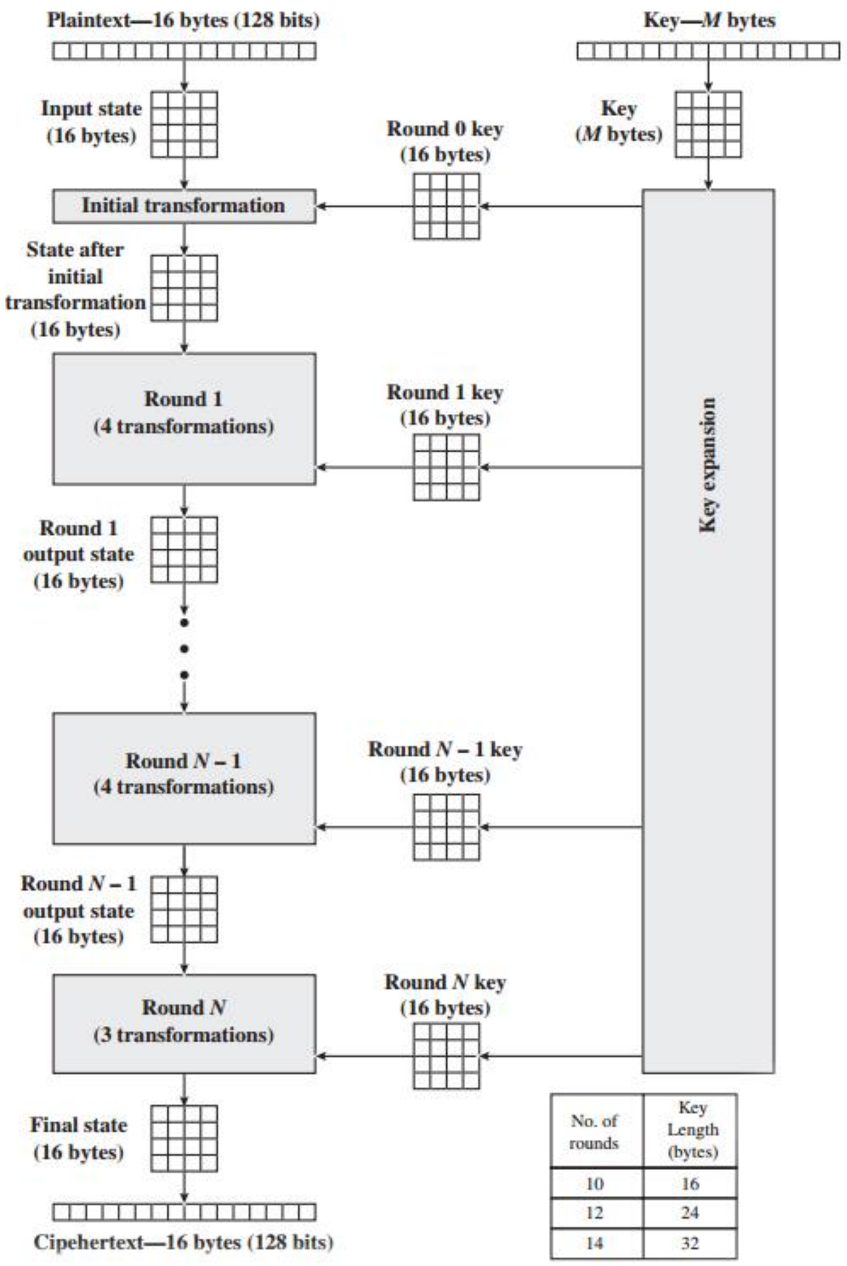
No	Algorithm Name	Submitters of algorithm	Positive votes	Negative votes
1	Rijndael	Joaen Daemen, Vincent Rijmen	86	10
2	Serpent	Ross Anderson, Eli Biham, Lars Knudsen	59	7
3	Twofish	Bruse Schneier, John Kelsey, Doug Whiting, David Wagner, Chris Hall, Niels Ferguson	31	21
4	RC6	RSA Laboratories	23	37
5	MARS	IBM	13	84

Table 9. "AES2" Algorithms with votes

On 2 October 2000, NIST announced that Rijndael block cipher selected as the Advance Encryption Standard (AES) and initiated the process to make it an official standard by publishing it in the Federal Register. Finally, the process completed on 26 November 2001 and AES approved as FIPS PUB 197. On 26 May 2002, AES became a standard of the US federal government and got approval from the secretary of commerce. Also, AES included in the ISO/IEC 18033-3 standard.

2.3.2 AES Structure

In “figure 7” shows the entire fabric of the AES encryption process. The cipher takes a plaintext block size of 128 bits or 16 bytes. The key length can be 128-bit, 192-bit, or 256-bit. Depends on the key size the algorithm can be as AES-128, AES-192, or AES-256.



Plaintext—16 bytes (128 bits)
Key—M bytes
Input state (16 bytes)
Key (M bytes)
Round 0 key (16 bytes)
Initial transformation
State after initial transformation (16 bytes)
Round 1 (4 transformations)
Round 1 key (16 bytes)
Key expansion
Round 1 output state (16 bytes)
Round N – 1 (4 transformations)
Round N – 1 key (16 bytes)
Round N – 1 output state (16 bytes)
Round N (3 transformations)
Round N key (16 bytes)
Final state (16 bytes)
Cipehertext—16 bytes (128 bits)

No. of rounds	Key Length (bytes)
10	16
12	24
14	32

Figure 7. AES Encryption Standard [7, p. 151]

The AES structure can separate into four sections “key expansion,” “add round key,” “rounds” and the “final round.”

1-Key-Expansion

In the first Section is the Rijndael Key Expansion that generates around key for each round plus one "16*(rounds + 1)", each round key is a 128- bit block. For example, if the original key is 128-bit then is equal to 16 bytes (8 bits = 1 byte) and can expand to "16*(rounds + 1)". If there are "10 rounds" then "16*(10 + 1) = 16 *11 = 176-bytes". Therefore, the cipher key expanded to a size of 176-bytes. This expanded key can be broke down into a round key for each round. Each Round key is added once at the start before the first round and then at the end of each round.

2-Add-Round-key

AES algorithm works with a 4x4 matrix, each cell has bytes in it and calls it as a state. The matrix has columns in a major order. Every cell of this matrix combines with a round key block using bitwise XOR operation.

3-Rounds

Each round contains four steps "Substitute Bytes," "Shift Rows," "Mix Columns" and "Add Round Key."

Step 1: Substitute Bytes, every cell replaced with another cell using the Rijndael S-box "Table 10". Suppose is the hexadecimal "2E", from the S-box the "2" goes to the y-axis, and "E" to the x-axis and the hexadecimal substitution is value "31". The S-box on "Table 10" is for the process of encryption and the inverse S-box "Table 11" is for the process of decryption.

	0	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**A**	**B**	**C**	**D**	**E**	**F**
0	63	7C	77	7B	F2	6B	6F	C5	30	01	67	2B	FE	D7	AB	76
1	CA	82	C9	7D	FA	59	47	F0	AD	D4	A2	AF	9C	A4	72	C0
2	B7	FD	93	26	36	3F	F7	CC	34	A5	E5	F1	71	D8	31	15
3	04	C7	23	C3	18	96	05	9A	07	12	80	E2	EB	27	B2	75
4	09	83	2C	1A	1B	6E	5A	A0	52	3B	D6	B3	29	E3	2F	84
5	53	D1	00	ED	20	FC	B1	5B	6A	CB	BE	39	4A	4C	58	CF
6	D0	EF	AA	FB	43	4D	33	85	45	F9	02	7F	50	3C	9F	A8
7	51	A3	40	8F	92	9D	38	F5	BC	B6	DA	21	10	FF	F3	D2
8	CD	0C	13	EC	5F	97	44	17	C4	A7	7E	3D	64	5D	19	73

9	60	81	4F	DC	22	2A	90	88	46	EE	B8	14	DE	5E	0B	DB
A	E0	32	3A	0A	49	06	24	5C	C2	D3	AC	62	91	95	E4	79
B	E7	C8	37	6D	8D	D5	4E	A9	6C	56	F4	EA	65	7A	AE	08
C	BA	78	25	2E	1C	A6	B4	C6	E8	DD	74	1F	4B	BD	8B	8A
D	70	3E	B5	66	48	03	F6	0E	61	35	57	B9	86	C1	1D	9E
E	E1	F8	98	11	69	D9	8E	94	9B	1E	87	E9	CE	55	28	DF
F	8C	A1	89	0D	BF	E6	42	68	41	99	2D	0F	B0	54	BB	16

Table 10. Rijndael S-box

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

Table 11. Rijndael inverse S-box

Step 2: Shift Rows, each row is a move to the left by n-1. The first row remains unchanged. The second, third and fourth rows shifted towards to the left by 1, 2 and three times respectively “figure 8”. This procedure happens when the key size is 128-bit or 192- bit. When the key is 256-bit, the first row is untouched and the shifting for the second, third and fourth row is a byte, 3 bytes, and 4 bytes respectively.

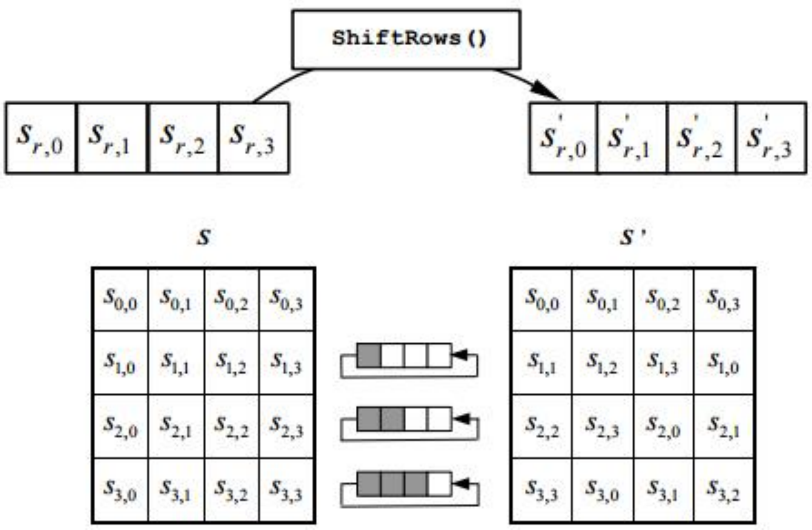


Figure 8. Shift Rows[11, p. 17]

Step 3: Mix Columns, every column from the state taken as a vector and multiply with a specific polynomial. Mix column gets an input of four bytes and provides an output of four bytes. The output replaces the input after multiplication in the state block. From this multiplication, the four bytes in a column superseded by the following:

$$s'_{0,c} \;\check{T}\; (\{0e\} \bullet s_{0,c}) \bullet (\{0b\} \bullet s_{1,c}) \bullet (\{0d\} \bullet s_{2,c}) \bullet (\{09\} \bullet s_{3,c})$$

$$s'_{1,c} \;\check{T}\; (\{09\} \bullet s_{0,c}) \bullet (\{0e\} \bullet s_{1,c}) \bullet (\{0b\} \bullet s_{2,c}) \bullet (\{0d\} \bullet s_{3,c})$$

$$s'_{2,c} \;\check{T}\; (\{0d\} \bullet s_{0,c}) \bullet (\{09\} \bullet s_{1,c}) \bullet (\{0e\} \bullet s_{2,c}) \bullet (\{0b\} \bullet s_{3,c})$$

$$s'_{3,c} \;\check{T}\; (\{0b\} \bullet s_{0,c}) \bullet (\{0d\} \bullet s_{1,c}) \bullet (\{09\} \bullet s_{2,c}) \bullet (\{0e\} \bullet s_{3,c})$$

Step 4: Add Round Key, the 128-bit subkey or 128-bit round key that came with the main key combined with the state using the XOR operation as in figure 9.

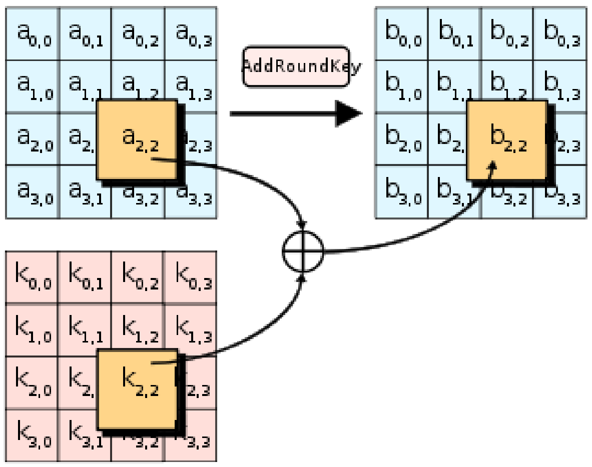


Figure 9. Round key

4- Final Round come with the three steps substitute bytes, shift rows and add round key, for decryption, the last operation would be the first as in “Figure 10”.

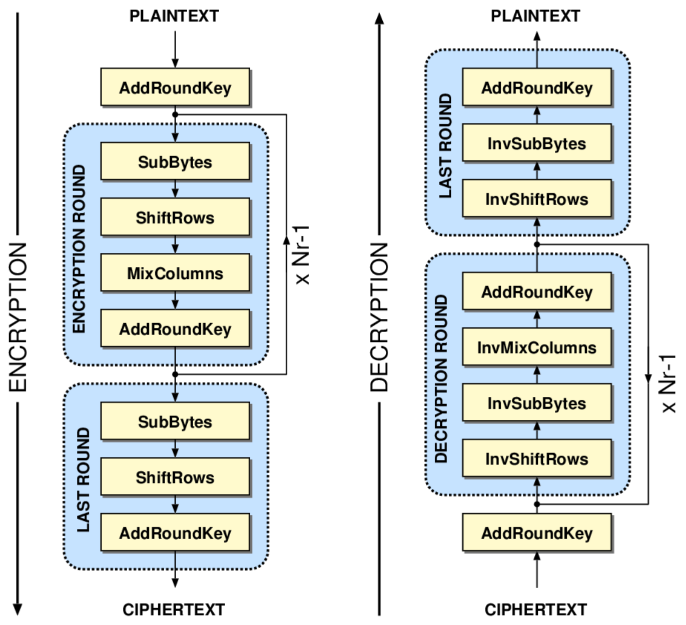


Figure 10. AES Encryption / Decryption

AES is a great secure algorithm, and the keys have a huge amount of possible combinations and are tough and difficult to break with a brute-force attack. The possible combinations for the 128-bit key are $3.4*10^{38}$, for the 192-bit key is $6.2*10^{57}$ and for the 256-bit key is $1.1*10^{77}$.

2.4 Rivest Cipher 2 (RC2)

The RC2 is a symmetric block cipher designed by Ron Rivest for RSA Data Security in 1989. Initially held as a confidential and proprietary algorithm, RC2 published as an Internet Draft during 1997. An important feature of RC2 is the flexibility offered to the user regarding the effective key size. Now, this feature becomes common in many block cipher proposals and it is a property that has proven to be important in commercial applications. Over the years, RC2 has very common, and it features prominently in the S/MIME secure messaging standard [12]. The RC2 cipher has a block size of 64- bit. The variable key size length comes from 8-bits up to 128-bits. The flexibility of the RC2 cipher can use it in many commercial applications.

2.4.1 History of Rivest Cipher 2

Ron Rivest was one of the co-founders of RSA. He initially created RC1, but he never published it, after a long research, he designed Rivest Cipher 2 (RC2) with a name ARC2. The design finish in 1987, the Lotus Software sponsored the project, and they include it in their software LotusNotes. After that, the National Security Agency (NSA) evaluate the cipher, and they provide suggestions for improvement. During 1988, Ron Rivest wrote an RFC (Request for Comments) document on RC2. In 1989, finally, RC2 approved for publishing it. In the establishment all the details of RC2 block cipher were secret. In 1996, RC2 source code leaked through Usenet. RC2 designed as a proposal to replace DES algorithm. The structure of RC2 it is so simple that can implement it on 16-bit microprocessors without any crash. The main purpose of the design of RC2 is to optimized for 16-bit microprocessors. The IBM AT machine can run RC2 algorithm two times faster than DES algorithm provided the same key[12].

3.3.2 RC2 Structure

The rc2 algorithm operates on 64-bit blocks and the blocks divided in four words where each word is sixteen bits. One byte is equal to 8-bit, and a word is 16-bit or two bytes. RC2 is an iterated block cipher, and the secret key is utilized to transform a plaintext into a ciphertext through a mixing and a mashing rounds. The mixing rounds in RC2 are sixteen while mashing rounds are only two. The main key is constructing a 64 -sub-keys, and in every mixing round, it uses a 16-bit subkey.

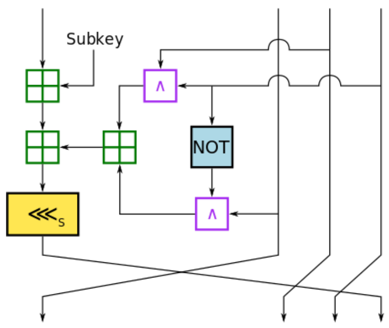


Figure 11. RC2 Diagram

In the explanation of RC2 given below there are different symbols for example “+” symbol stands for twos complement addition, the “&” symbol stands for bitwise “and” operation, the “~” symbol means bitwise supplement and the “^” symbol stands for exponentiation operation. The “XOR” means bitwise “exclusive-or” operation and the “MOD” means modulo operation[12].

The RC2 algorithm separate into three different parts:

1-Key-expansion, the algorithm have two different types of operations, the byte operations, and the word operations on the key buffer. For better identification to the explanation, the letter “K” is for word, and the letter “L” is for the byte operation. The relationship between the word and byte operation is “K[i] = L[2*i] + 256*L[2*i+1]”. The byte of a lower order of every K word is present before the higher order byte. Let us assume that a key has a “T” bytes and is in the range of T 1 to 128. For example T=8. The maximum effective key length in bits denoted T1. In this way, the smaller from 2^(8*T) or 2^T1 will consider as key space. The main task of key expansion is to change the key buffer in such a way that every bit of the expanded key is dependent in a complex manner on each and every bit of key which provided as an input.

First the key expansion algorithm the provide Tbytes of a key and are place in the form of bytes from L[0] to L[T-1] in the key buffer. The calculation effective key length is T8 in bytes “T8 = (T1+7)/8”. A mask TM calculates the effective key length in bits T1 “TM = 255 MOD 2^(8 + T1 - 8*T8)” TM have 8 - (8*T8 - T1) least significant bits set.

For example, if there is an effective key length of 64 bits then T1 = 64, T8 = 8 and TM = 0xff. With an effective key length of 63 bits, T1 = 63, T8 = 8 and TM = 0x7f.

From PITABLE[0] to PITABLE[255] is an array of random bytes which based on the digits of PI = 22/7 = 3.14159. The array “PITABLE” is made by randomly permuting values from 0 to 255. The “PITABLE” array in hexadecimal notation as in Table 12.

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

Table 12.”PITABLE” array [12]

The key expansion operation contains two loops and an intermediate step,

```
for i = T, T+1, ..., 127 do
L[i] = PITABLE[L[i-1] + L[i-T]];

L[128-T8] = PITABLE[L[128-T8] & TM];

 for i = 127-T8, ..., 0 do
L[i] = PITABLE[L[i+1] XOR L[i+T8]];
```

The valid key has values from L[128-T8] to L[127]. The intermediate step has bitwise "and" operation that decreases the search space for L[128-T8], this way the sufficient number of key bits is T1. The expand key depends entirely upon effective key bits, irrespective of the provided key K. As the expand key does not change during both encryption or decryption, as a result, can expand the key only one time during encrypting or decrypting a big block of data [13].

2-Encryption-Algorithm comprises of mixing and mashing operations. If there is a word “x” then “x rol k” means that a 16-bit word has rotated towards left by k-bits, while those bits that have been shifted outside from the upper end will enter from the bottom end[13].

The mixing operation can describe like, s[0] is 1, s[1] is 2, s[2] is 3, s[3] is 5, this index the array “R” that use as MOD 4. The “R” will always have a subscript from 0 to 3.

R[i] = R[i] + K[j] + (R[i-1] & R[i-2]) + ((~R[i-1]) & R[i-3]);
j = j + 1;
R[i] = R[i] rol s[i];
and the operations are Mix R[0], Mix R[1], Mix R[2], Mix R[3]

The Mashing operation have the mathematical operation R[i] = R[i] + K[R[i-1] & 63]; and the operations are Mash R[0], Mash R[1], Mash R[2], Mash R[3].

Steps of encryption operation

1. Words R[0], R[1], R[2] and R[3] Initialize by 64 bit input value.
2. Key expanded so that all words from K[0] to K[63] are defined.
3. J initialized by zero (j is a global variable of type integer, have effect only by mixing operation).
4. Mixing rounds perform five times.
5. Mashing round do one time.
6. Mixing rounds perform six times.
7. Mashing round do one time.
8. Mixing rounds perform five times.

3-Decryption-Algorithm can perform by reversing the mix and mash operations. The r-mix and r-mash use the “r” symbol to the reverse operations. For example is the word “x” then “x ror k” means that a 16-bit word rotates towards the right by k bits, while the bits are shifted outside from the bottom end will enter from the upper end.[13]

The Remixing round can perform with

R[i] = R[i] ror s[i];
R[i] = R[i] - K[j] - (R[i-1] & R[i-2]) - ((~R[i-1]) & R[i-3]);
j = j - 1;
and the operations are Mix R[3], Mix R[2], Mix R[1], Mix R[0]

The Mashing operation have the mathematical operation R[i] = R[i] + K[R[i-1] & 63]; and the operations are Mash R[3], Mash R[2], Mash R[1], Mash R[0].

Steps of decryption operation

1. Words R[0], R[1], R[2] and R[3] are Initialize by 64-bit ciphertext.
2. The key expands so that all words from K[0] to K[63] are defined.
3. J initialized by 63 (j is a global variable of type integer, it is effected only by mixing operation).
4. Reverse mixing rounds for five times.
5. Reverse mashing round for one time.
6. Reverse mixing rounds for six times.
7. Reverse mashing round for one time.
8. Reverse mixing rounds for five times.

2.5 Algorithm Modes

Block ciphers operate on blocks, and each block of plaintext is independent and is used to produce a ciphertext block of equal length. Typically, a block size is 64bit or 128bit, and an algorithm mode usually combines the basic cipher and simple operations. Transactions are simple because the security is a function of the underlying cipher and not the mode [5].

2.5.1 Electronic Codebook (ECB)

Electronic codebook (ECB) is the simplest encryption mode. A block of plaintext encrypts into a block of ciphertext straight forward. The specific block of plaintext always encrypts to the specific block of ciphertext. The encryption or decryption can do in parallel. The same blocks encrypt to identical blocks and though this cannot provide confidentiality to the data.

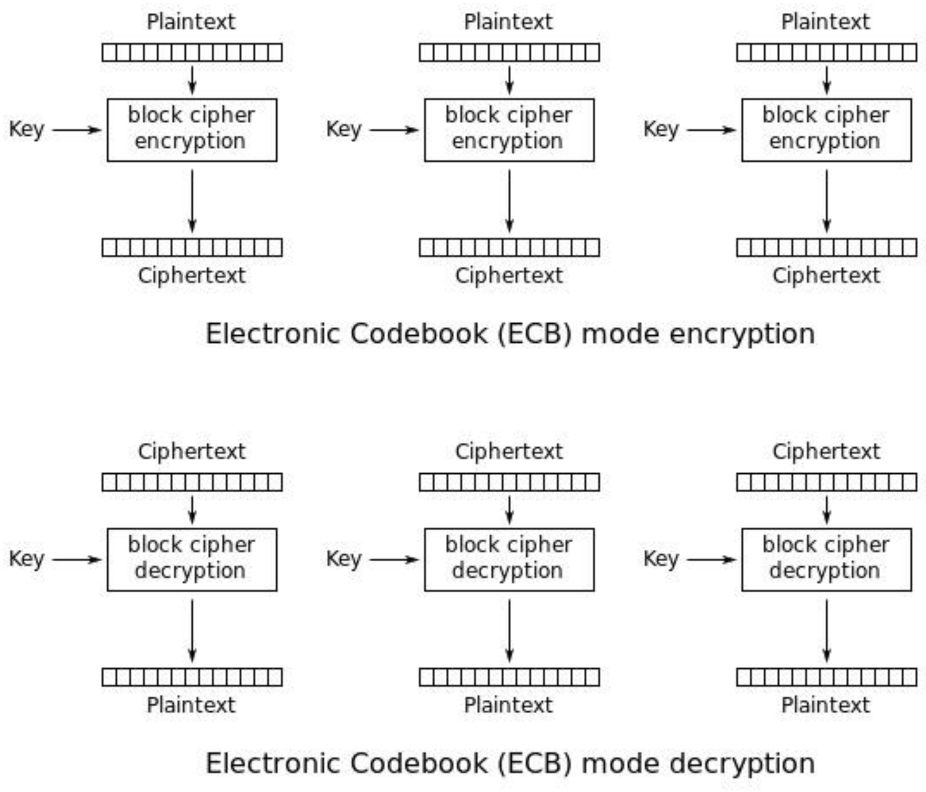


Figure 12. ECB Mode

2.5.2 Cipher Block Chaining (CBC)

On Cipher Block Chaining, the blocks are XOR from the previous block so depends on the previous one in the encryption. As shown in figure 13, every ciphertext block relies on all the plaintext blocks, prepared all the period of time. The initialization vector (IV) is to protect and prevents the reveal of the fist block that might give some useful information. Because each block depends on the previous one cannot support parallelizable encryption.

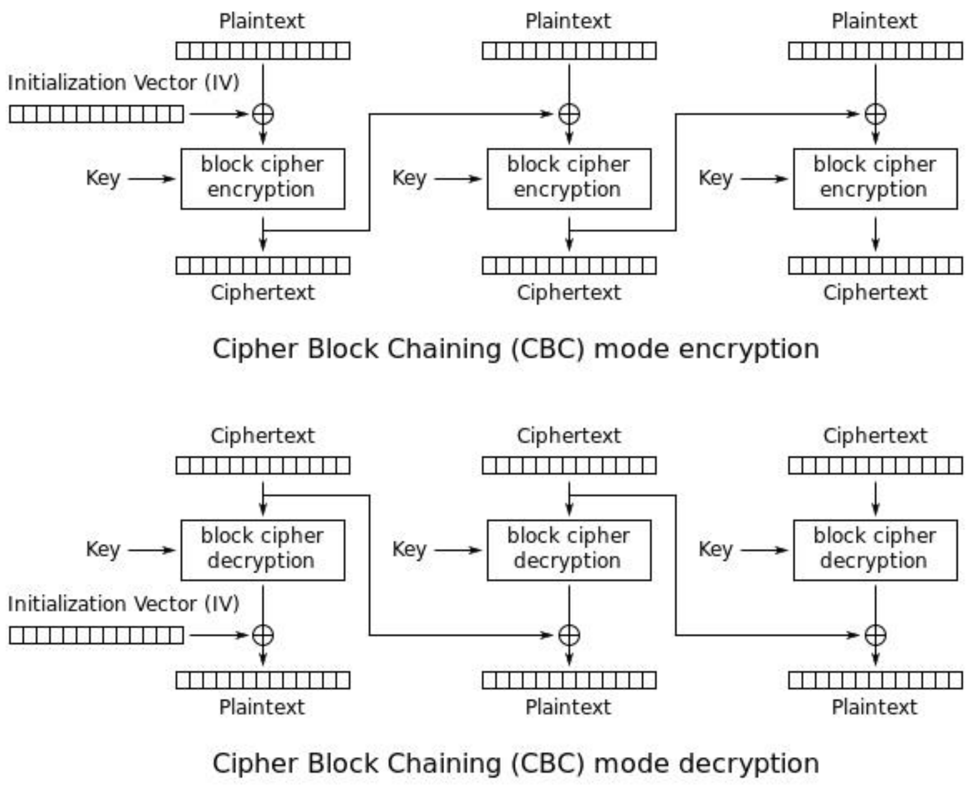


Figure 13. CBC Mode

2.5.3 Output Feedback (OFB)

The Output Feedback (OFB) mode makes a block cipher into a synchronous stream cipher. The encryption and decryption are the same because of the symmetry of the XOR operation. Then initialization vector (IV) is XOR with the plaintext, it generates the keystream block and with the key produce the cipher text. Each block operation depends from the previous blocks and cannot perform parallel encryption.

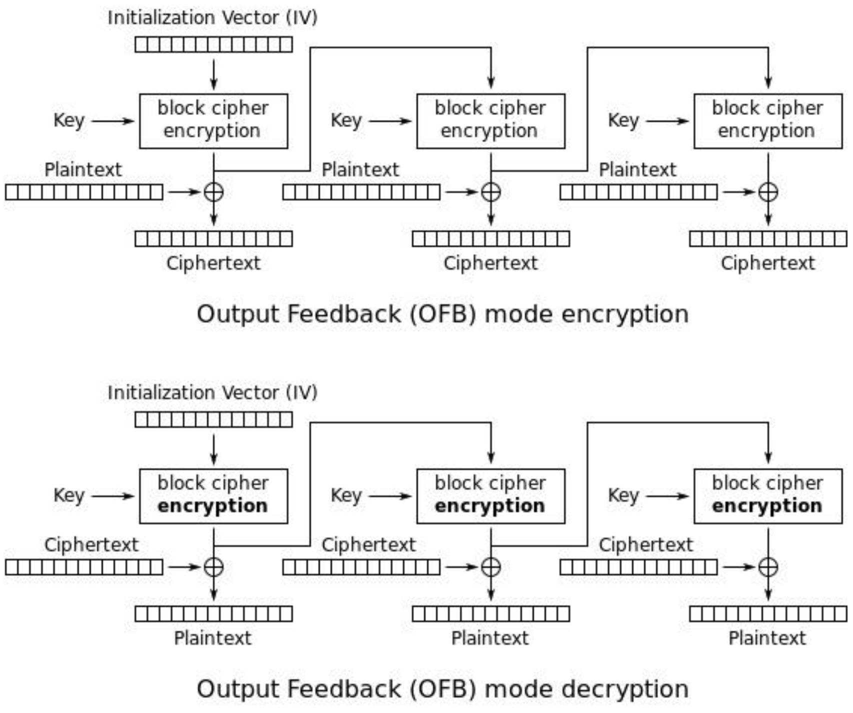

Initialization Vector (IV)
Key
block cipher encryption
Plaintext
Ciphertext
Output Feedback (OFB) mode encryption
Initialization Vector (IV)
Key
block cipher encryption
Ciphertext
Plaintext
Output Feedback (OFB) mode decryption

Figure 14. OFB Mode

2.5.4 Cipher Feedback (CFB)

The Cipher Feedback (CFB) mode is essentially like CBC mode, but it makes a block cipher into a self-synchronizing stream. The advantage of that is cipher is when a part of the ciphertext is lost is capable of continuing the decryption. This operation mode is associated to CBC and CFB decryption and is almost the same as CBC encryption but in reverse mode. Moreover, the Cipher Feedback has the problem of propagating; if an error occurs on a single bit, then all the procedure is completely corrupted.

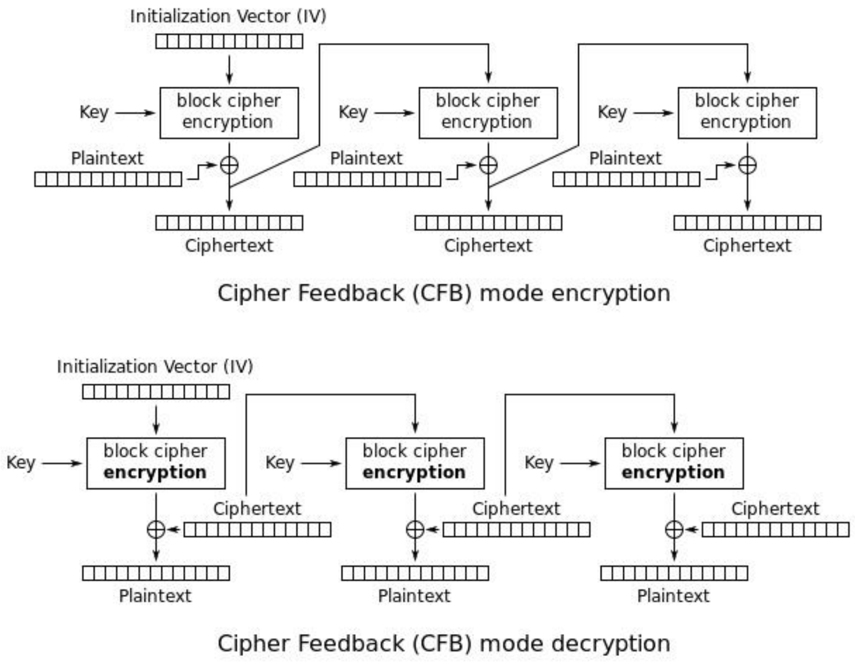

Initialization Vector (IV)
Key
block cipher encryption
Plaintext
Ciphertext
Cipher Feedback (CFB) mode encryption
Initialization Vector (IV)
Key
block cipher encryption
Ciphertext
Plaintext
Cipher Feedback (CFB) mode decryption

Figure 15.f CFB Mode

2.5.5 Ciphertext Stealing (CTS)

The Ciphertext stealing (CTS) is a cipher method that is encrypting and decrypting a plaintext to a block cipher, without expanding the message to a multiple of the block size. With CTS the ciphertext is the identical size as the data input the plaintext. The last incomplete block of the encryption can be filled in some reversible way and encrypted. For example, the reversible padding is appending a single one followed by zeroes until the length is a multiple of the block length. This padding operation results in a cryptogram that is longer than the message and can give storage or bandwidth problems if many relatively short messages must encrypt.

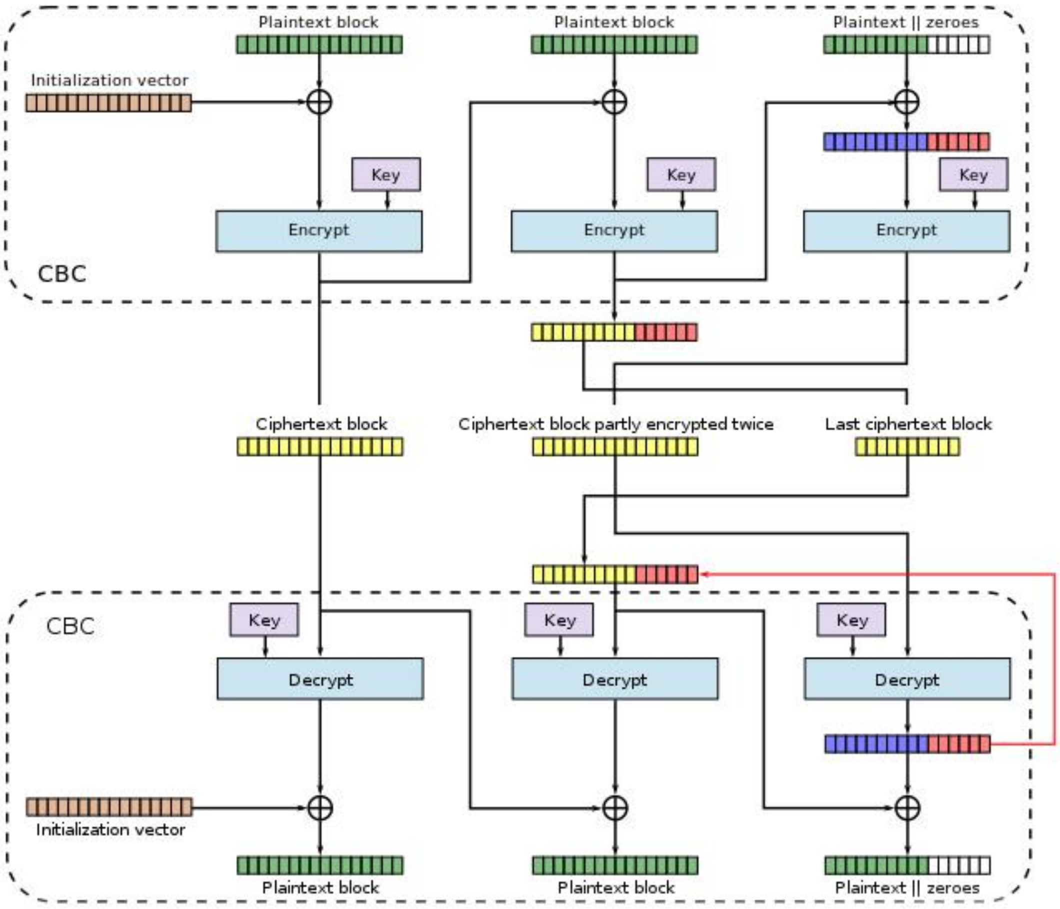


Figure 16. CTS Mode

Chapter 3
Algorithm Benchmark

A benchmark is a process of running a computer program tool, to measure a specific performance of an object and compare the results with the different past tests or different tools. Normally the common benchmark tests have to do with the comparison of the period and operations over time. The benchmark results have the characteristics of the hardware and operating system of the computer. To perform a benchmark testing have to perform on the same environment parameters and under the same conditions to have valid results. For the maximum proper results, the tests should be repeatable, other applications need to deactivated except the require processes, and the software /hardware components must be the same environment and conditions.

3.1 Simulation Implementation

The main objectives are to analyze and investigate the performance of different cryptographic algorithms over a different kind of data and hardware processors. The goal is to compare the measurements of the encryption and decryption processing time for each algorithm with the same data, hardware and "Cipher Mode." Processing time is the time that indicates the speed that takes an algorithm to complete the procedure of encrypting of decrypt. The simulation of the cryptographic algorithms designed under the dot net Framework.

3.1.1 Benchmark Program Development

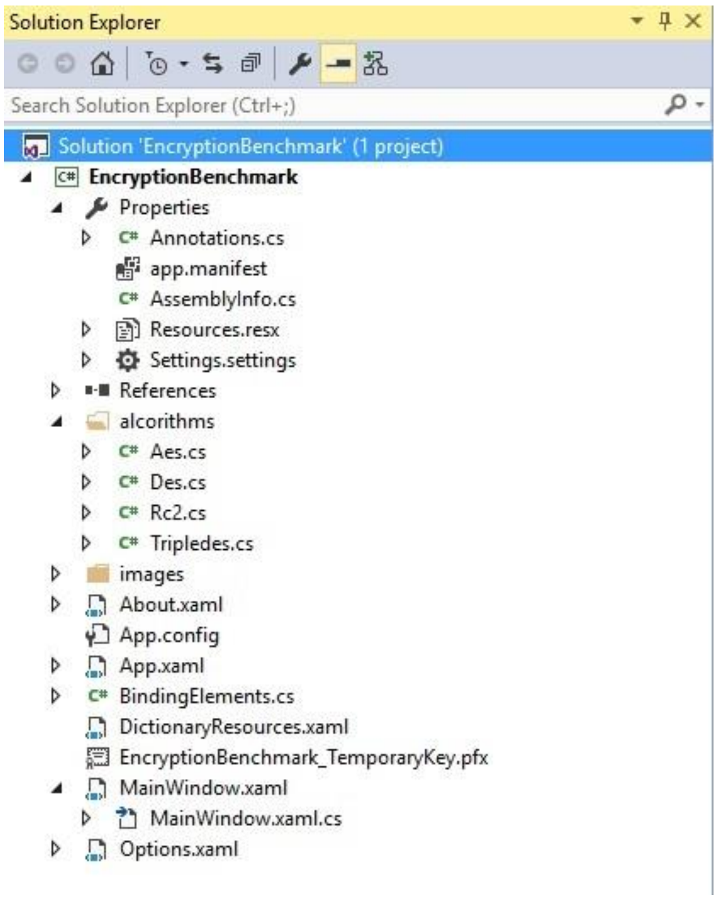
The dot net Framework is a software framework developed by Microsoft and runs under Windows operating system. The program developed under the latest Microsoft Framework 4 supports parallel programming and enable multiple threads to be executed simultaneously[14]. The simulation benchmark drawn up with the programming language C sharp and the graphic interface system is a rendering with Windows Presentation Foundation (WPF). The cryptography model designed through the CryptoStream class, which derives from the Stream class and all this give the feature to

Algorithms	Type	Block Bits
RC2	Block	64
DES	Block	64
3-DES	Block	64
AES	Block	128

add a hardware and software implementations [14].

Table 13. Algorithms supported by.NET

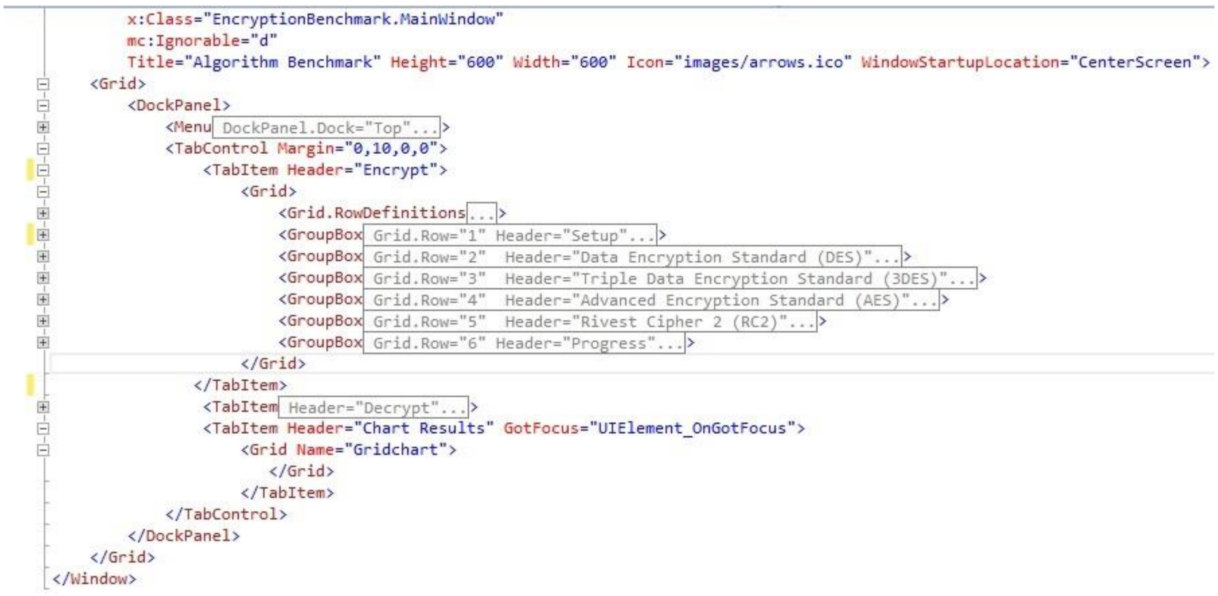
The structure design of the benchmark program comes down to three top parts as it shows to the figure 17 bellow. The user graphic interface, the algorithms classes and the "binding elements" that connect the two most important parts.



```
Solution Explorer
Search Solution Explorer (Ctrl+;)
Solution 'EncryptionBenchmark' (1 project)
  EncryptionBenchmark
    Properties
      Annotations.cs
      app.manifest
      AssemblyInfo.cs
      Resources.resx
      Settings.settings
    References
    alcorithms
      Aes.cs
      Des.cs
      Rc2.cs
      Tripledes.cs
    images
    About.xaml
    App.config
    App.xaml
    BindingElements.cs
    DictionaryResources.xaml
    EncryptionBenchmark_TemporaryKey.pfx
    MainWindow.xaml
      MainWindow.xaml.cs
    Options.xaml
```

Figure 17. Benchmark Solution Structure

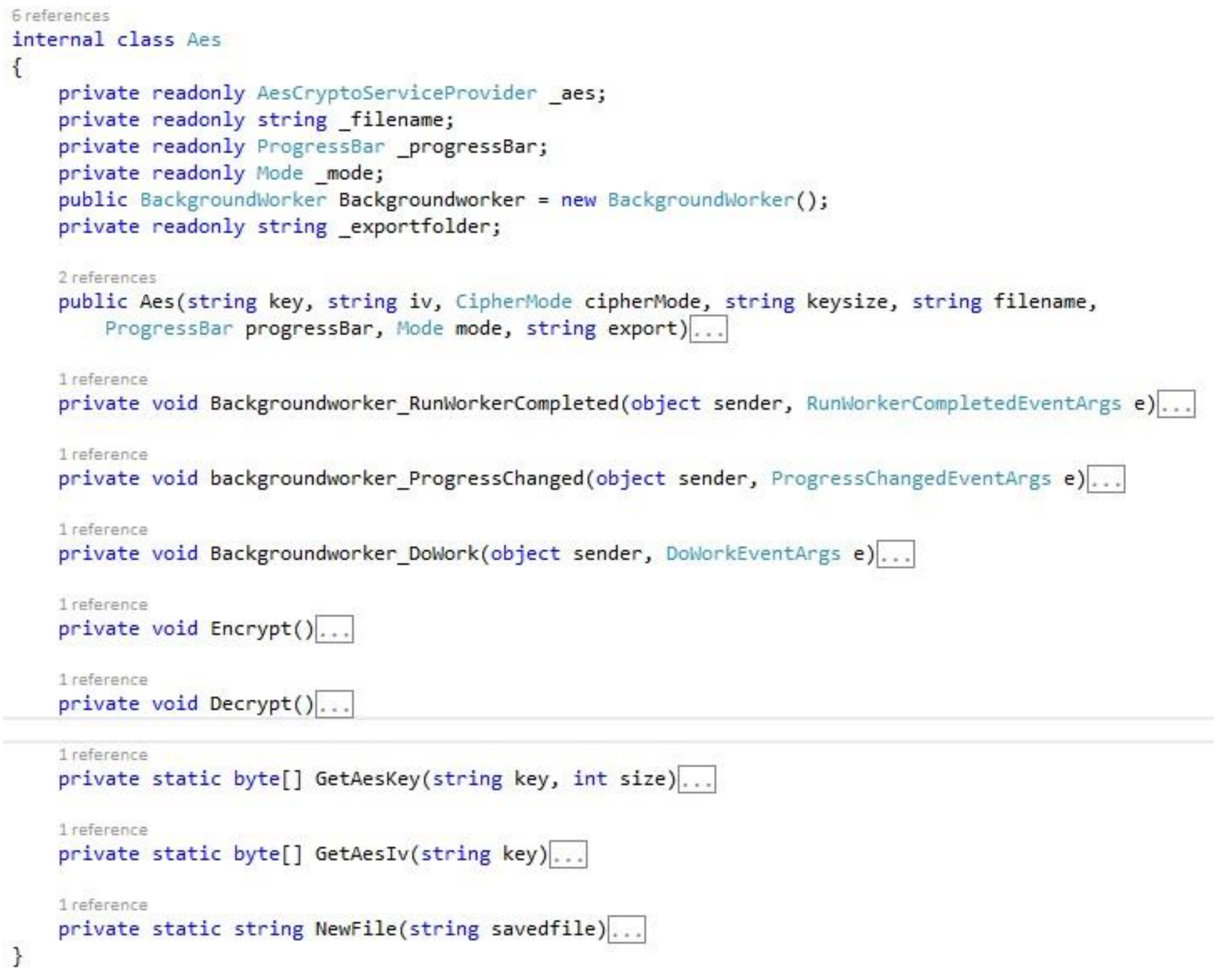
1. The GUI is a rendering user interface on Windows Presentation Foundation base on XML language for a standalone desktop program.



```
        x:Class="EncryptionBenchmark.MainWindow"
        mc:Ignorable="d"
        Title="Algorithm Benchmark" Height="600" Width="600" Icon="images/arrows.ico" WindowStartupLocation="CenterScreen">
    <Grid>
        <DockPanel>
            <Menu DockPanel.Dock="Top"...>
            <TabControl Margin="0,10,0,0">
                <TabItem Header="Encrypt">
                    <Grid>
                        <Grid.RowDefinitions...>
                        <GroupBox Grid.Row="1" Header="Setup"...>
                        <GroupBox Grid.Row="2"  Header="Data Encryption Standard (DES)"...>
                        <GroupBox Grid.Row="3"  Header="Triple Data Encryption Standard (3DES)"...>
                        <GroupBox Grid.Row="4"  Header="Advanced Encryption Standard (AES)"...>
                        <GroupBox Grid.Row="5"  Header="Rivest Cipher 2 (RC2)"...>
                        <GroupBox Grid.Row="6" Header="Progress"...>
                    </Grid>
                </TabItem>
                <TabItem Header="Decrypt"...>
                <TabItem Header="Chart Results" GotFocus="UIElement_OnGotFocus">
                    <Grid Name="Gridchart">
                       </Grid>
                    </TabItem>
            </TabControl>
        </DockPanel>
    </Grid>
</Window>
```

Figure 18. WPF Benchmark GUI

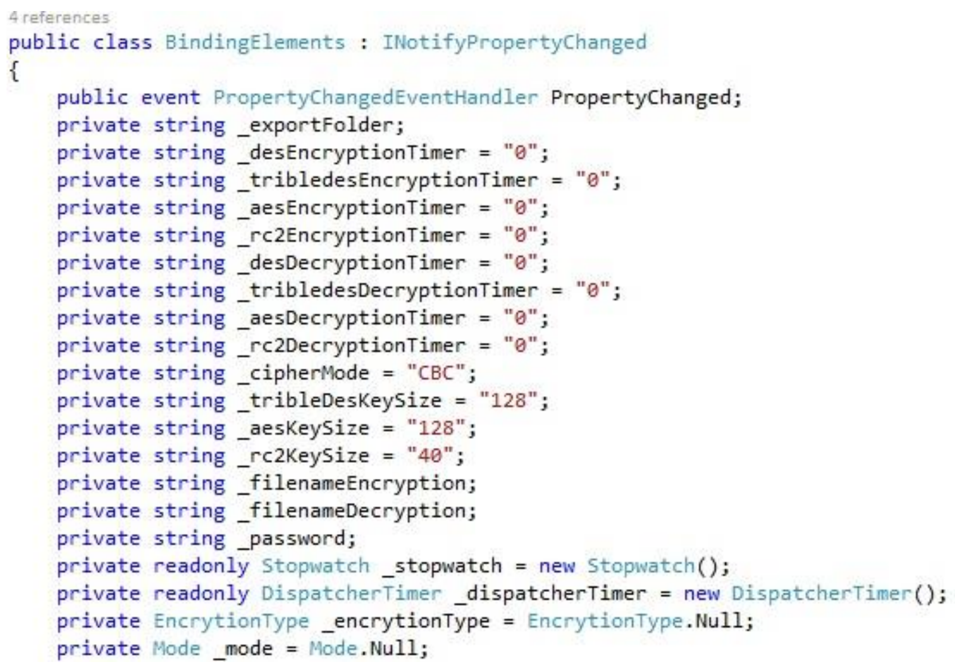
2. The second part of the program are the algorithms, and each one is a different Class with specific functions. The main public function for each class is the contractor that initialize the values of the algorithm (key, IV, cipher mode, key size, filename, progress bar, encryption, decryption mode, export folder). The background worker with a start the task of a specific process and call the private functions to encrypt or decrypt. Parallel with a background worker a stopwatch is running to calculate the time it takes to complete a task. The stopwatch starts from the time that a process starts on the input data. The private function for the Key size, Initialization Vector executed with the contractor.



```
6 references
internal class Aes
{
    private readonly AesCryptoServiceProvider _aes;
    private readonly string _filename;
    private readonly ProgressBar _progressBar;
    private readonly Mode _mode;
    public BackgroundWorker Backgroundworker = new BackgroundWorker();
    private readonly string _exportfolder;

    2 references
    public Aes(string key, string iv, CipherMode cipherMode, string keysize, string filename,
        ProgressBar progressBar, Mode mode, string export)...

    1 reference
    private void Backgroundworker_RunWorkerCompleted(object sender, RunWorkerCompletedEventArgs e)...

    1 reference
    private void backgroundworker_ProgressChanged(object sender, ProgressChangedEventArgs e)...

    1 reference
    private void Backgroundworker_DoWork(object sender, DoWorkEventArgs e)...

    1 reference
    private void Encrypt()...

    1 reference
    private void Decrypt()...

    1 reference
    private static byte[] GetAesKey(string key, int size)...

    1 reference
    private static byte[] GetAesIv(string key)...

    1 reference
    private static string NewFile(string savedfile)...
}
```

Figure 19. Benchmark AES Class

3. The binding element is a process that establishes the connection between the WPF graphic interface and the class functions of the algorithms. The binding has the data settings and provides the proper notifications when the data changes, the elements that are bound to the data reflect back automatically.



```
4 references
public class BindingElements : INotifyPropertyChanged
{
    public event PropertyChangedEventHandler PropertyChanged;
    private string _exportFolder;
    private string _desEncryptionTimer = "0";
    private string _tribledesEncryptionTimer = "0";
    private string _aesEncryptionTimer = "0";
    private string _rc2EncryptionTimer = "0";
    private string _desDecryptionTimer = "0";
    private string _tribledesDecryptionTimer = "0";
    private string _aesDecryptionTimer = "0";
    private string _rc2DecryptionTimer = "0";
    private string _cipherMode = "CBC";
    private string _tribleDesKeySize = "128";
    private string _aesKeySize = "128";
    private string _rc2KeySize = "40";
    private string _filenameEncryption;
    private string _filenameDecryption;
    private string _password;
    private readonly Stopwatch _stopwatch = new Stopwatch();
    private readonly DispatcherTimer _dispatcherTimer = new DispatcherTimer();
    private EncrytionType _encrytionType = EncrytionType.Null;
    private Mode _mode = Mode.Null;
```

Figure 20.Benchmark Binding Elements

3.1.2 Algorithms Private Functions

-The-password-key-size

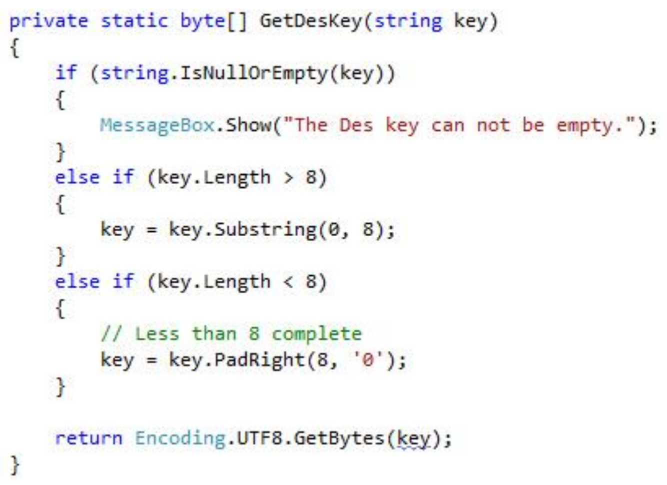
The key length is a critical security parameter that use to control the operation of the cipher. For each algorithm is a different level of cryptographic complexity and usual have different key sizes depend on the security level. The simulation generates keys from the string password and the user. Each letter on UTF8 (character encoding) code is equal with 8-bits, most of the times the password that a user enters is smaller than or greater that the key size. To get the correct key length each algorithm has to do a process to convert the password to the right size.

Letter	UTF-8	Binary
A	U+0041	01000001
B	U+0042	01000010
C	U+0043	01000011
D	U+0044	01000100

Table 14. UTF-8 encoding table and Unicode characters

-DES-key-function

The password string key enters the function, checks if is null and after that goes to the next statement and test the size. The DES algorithm has 64-bits (8+ parity bits) so if the string is bigger than the key cut down to the correct size. If the text is less than the key, a “0” bits are added to make the right size key.

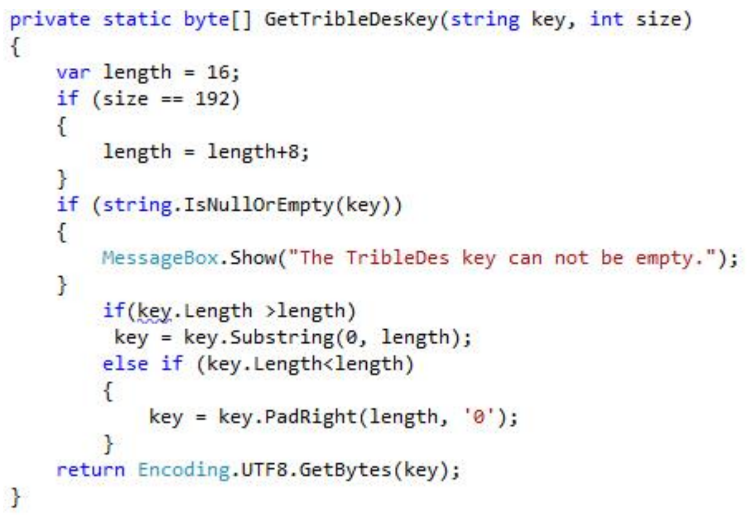


```
private static byte[] GetDesKey(string key)
{
    if (string.IsNullOrEmpty(key))
    {
        MessageBox.Show("The Des key can not be empty.");
    }
    else if (key.Length > 8)
    {
        key = key.Substring(0, 8);
    }
    else if (key.Length < 8)
    {
        // Less than 8 complete
        key = key.PadRight(8, '0');
    }

    return Encoding.UTF8.GetBytes(key);
}
```

Figure 21. DES key Function Simulation

-3DES-key-function

3DES have two types of key size length the 128-bits and 196-bits. The user can choose the size length from the user interface, and the 3DES key function computes the key. To calculate the key size the integer converts from bits to bytes length. If the size is 192-bits then is equal to the fist statement of the function, and the length becomes to 24-bytes. If the key is 128-bit then stays to the initial state of 16-bytes. Then if the text is more than the key is round up to the specific size, from the other side if the text is less than the text a 0-bits added to the end.

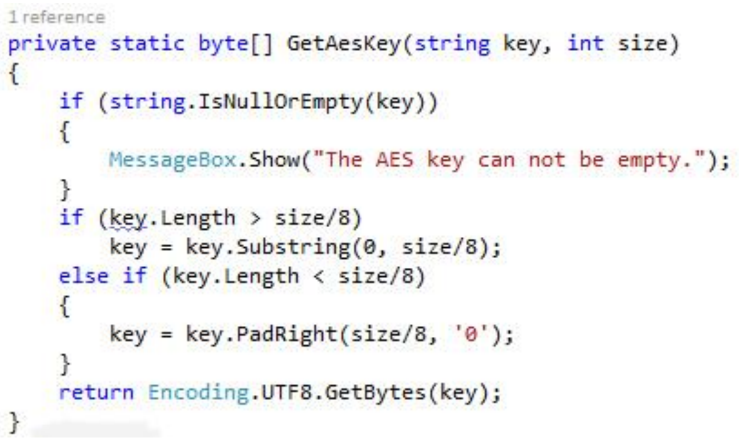


```
private static byte[] GetTribleDesKey(string key, int size)
{
    var length = 16;
    if (size == 192)
    {
        length = length+8;
    }
    if (string.IsNullOrEmpty(key))
    {
        MessageBox.Show("The TribleDes key can not be empty.");
    }
        if(key.Length >length)
         key = key.Substring(0, length);
        else if (key.Length<length)
        {
            key = key.PadRight(length, '0');
        }
    return Encoding.UTF8.GetBytes(key);
}
```

Figure 22. 3DES key Function Simulation

-AES-key-function

The AES algorithm comes with three types of the main length 128-bits, 192-bits, 256-bits the same with the rest function algorithms depend on the size from the input of the user. The size integer is divided with eight to convert it to bytes. After that, a statement checks the specific key length and is less or more than the size and add or remove the key.

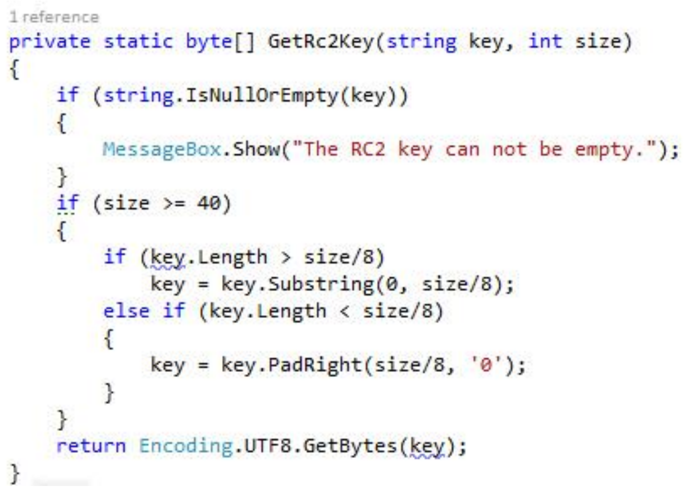


```
1 reference
private static byte[] GetAesKey(string key, int size)
{
    if (string.IsNullOrEmpty(key))
    {
        MessageBox.Show("The AES key can not be empty.");
    }
    if (key.Length > size/8)
        key = key.Substring(0, size/8);
    else if (key.Length < size/8)
    {
        key = key.PadRight(size/8, '0');
    }
    return Encoding.UTF8.GetBytes(key);
}
```

Figure 23.AES Function key Simulation

-RC2-key-function

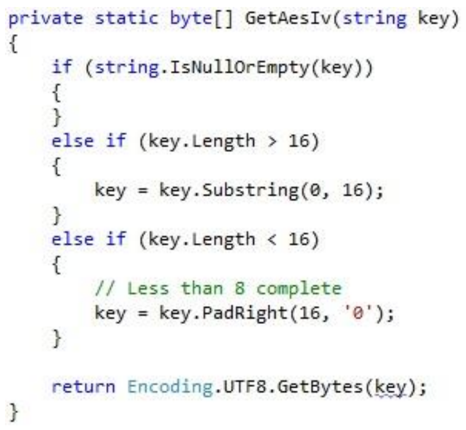
The RC2 algorithm supports key lengths from 40-bits to 1024-bits in increments of 8-bits. If the key size is smaller than 40-bits, the procedure cannot continue, and the key return to the function as it is. If the size is greater than 40-bits, then go to the next function and compares the key length with the size. For example, the key size is larger; then letters have to remove the password to get the correct size. From the other side if the key is less than the size then 0-bits round up the key length to the right.



```
1 reference
private static byte[] GetRc2Key(string key, int size)
{
    if (string.IsNullOrEmpty(key))
    {
        MessageBox.Show("The RC2 key can not be empty.");
    }
    if (size >= 40)
    {
        if (key.Length > size/8)
            key = key.Substring(0, size/8);
        else if (key.Length < size/8)
        {
            key = key.PadRight(size/8, '0');
        }
    }
    return Encoding.UTF8.GetBytes(key);
}
```

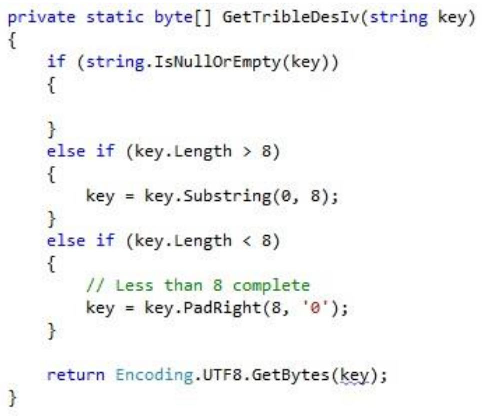
-Benchmark-Initialization-Vector-(IV)

Each symmetric algorithm has an initialization vector (IV) and is a binary number that uses at the beginning of the session of the encryption. The initialization vector is not necessary for the encryption but is more vulnerable to dictionary attacks. The initialization vector size depends on the block size of the algorithm. The block size of the algorithms is eight times bigger than the Initialization Vector. The AES algorithm has 16-bits initialization vector, and the rest benchmark algorithms have 8-bits. The program function takes the key password and checks the length, for example, is 3DES the key must be 8-bit and add or subtract to complete the 8-bits. Each bit represents a letter from the key and completes the initialization vector.



```
private static byte[] GetAesIv(string key)
{
    if (string.IsNullOrEmpty(key))
    {
    }
    else if (key.Length > 16)
    {
        key = key.Substring(0, 16);
    }
    else if (key.Length < 16)
    {
        // Less than 8 complete
        key = key.PadRight(16, '0');
    }

    return Encoding.UTF8.GetBytes(key);
}
```

Figure 24. AES Initialization Vector

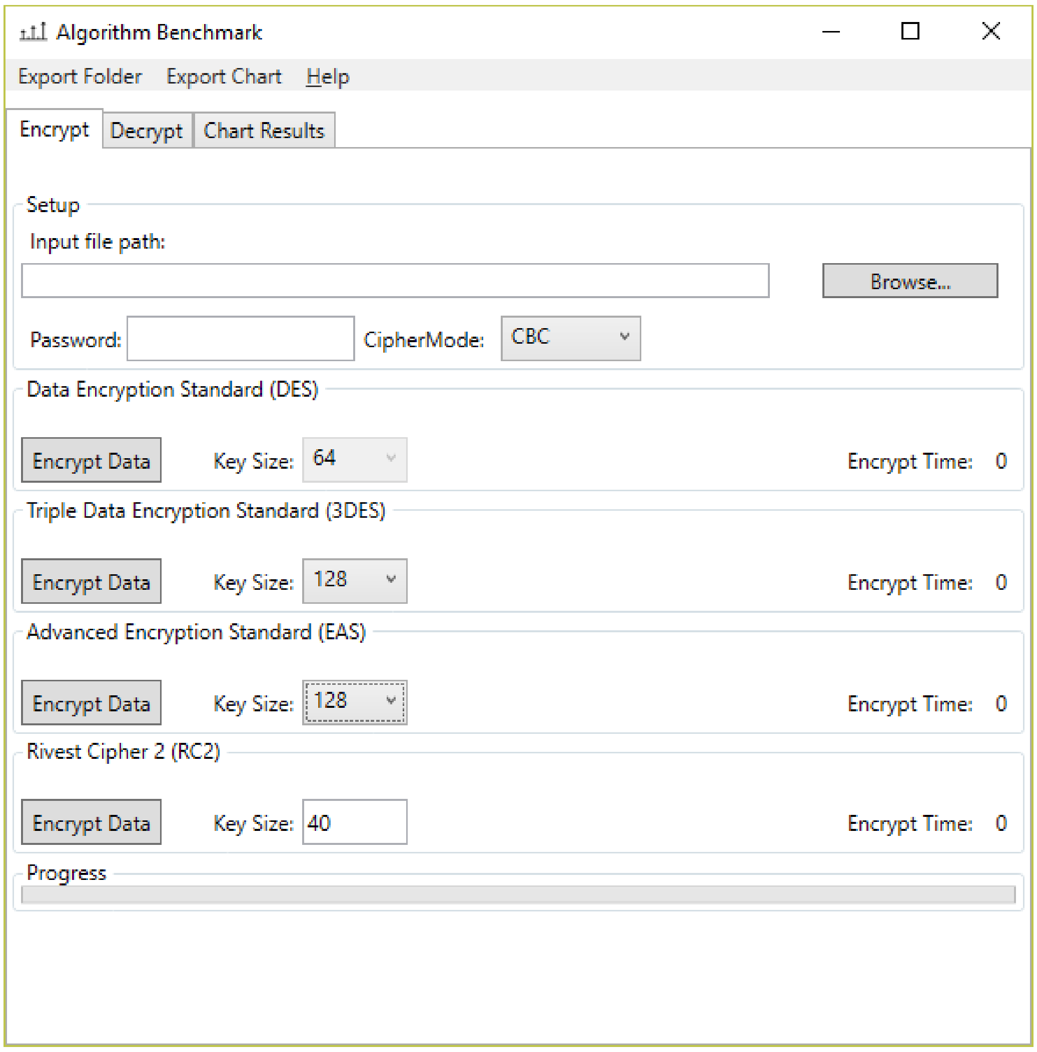


```
private static byte[] GetTribleDesIv(string key)
{
    if (string.IsNullOrEmpty(key))
    {

    }
    else if (key.Length > 8)
    {
        key = key.Substring(0, 8);
    }
    else if (key.Length < 8)
    {
        // Less than 8 complete
        key = key.PadRight(8, '0');
    }

    return Encoding.UTF8.GetBytes(key);
}
```

Figure 25. 3DES Initialization Vector

3.1.3 User Graphic Interface

The Simulation Algorithm Benchmark contains three main taps the Encrypt tap, the Decrypt tap and the Chart Results tap. For the Encryption/Decryption tap, can browse the input file, enter the password key and choose the Cipher Mode. Can encrypt or decrypt each symmetric algorithm by selecting the key size and start the process. Afterward, a process complete and get the exact encryption or decryption time. The last tap is the Chart Results that compares the cryptographic algorithms through time is a single graph.



Input file path:

Password: CipherMode: CBC

Key Size: 64 Encrypt Time: 0

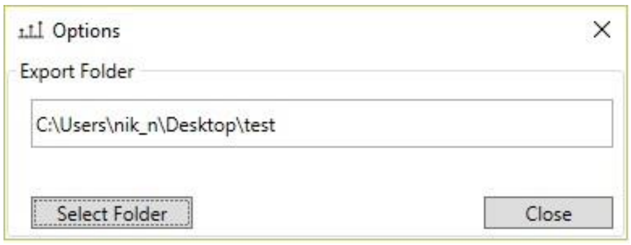
Key Size: 128 Encrypt Time: 0

Key Size: 128 Encrypt Time: 0

Key Size: 40 Encrypt Time: 0

Figure 26. Algorithm Benchmark Simulation

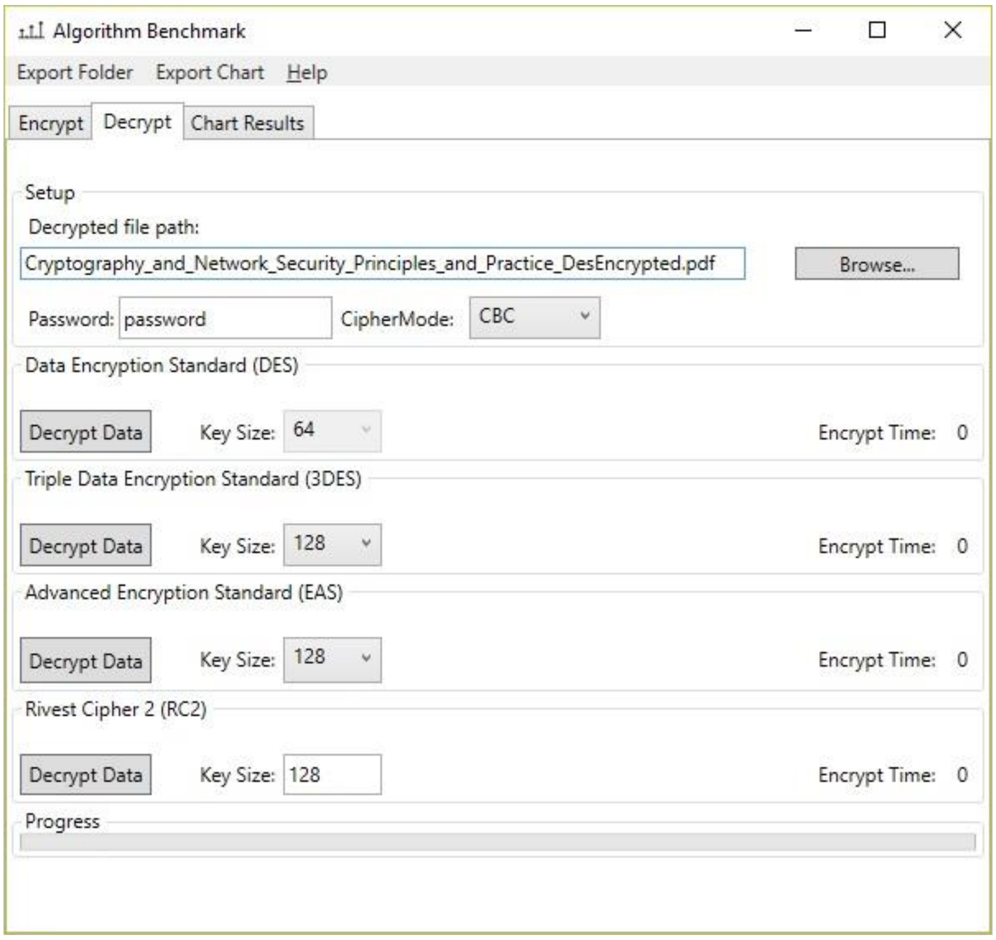
The data encrypted file save to a specific folder with the same name as the input file and the encryption type. Can choose and select the specific folder for the main menu "Export Folder."



C:\Users\nik_n\Desktop\test

Figure 27. Benchmark "Export Folder."

The user has to choose the correct file name from the specific export folder before the encryption process. To have a successful decryption, have to choose the right algorithm, password, cipher mode and the right key size. At the end of each encrypted file name, can find the encryption type. For example, "figure 28" show the file that encrypted with DES algorithm you can recognize the name "DesEncrypted."



Decrypted file path: Cryptography_and_Network_Security_Principles_and_Practice_DesEncrypted.pdf

Password: password CipherMode: CBC

Data Encryption Standard (DES): Key Size: 64 Encrypt Time: 0

Triple Data Encryption Standard (3DES): Key Size: 128 Encrypt Time: 0

Advanced Encryption Standard (EAS): Key Size: 128 Encrypt Time: 0

Rivest Cipher 2 (RC2): Key Size: 128 Encrypt Time: 0

Figure 28. Algorithm Decryption Benchmark

When the process complete, the benchmark program has all the time results. The last tap is the "Chart Results" you can check the results for all the algorithms briefly and compare them. The graph compares the algorithms base the time that takes to encrypt and decrypt on the same file and system. The Chart Results can export to an image with the "Export Chart" button on the main menu.

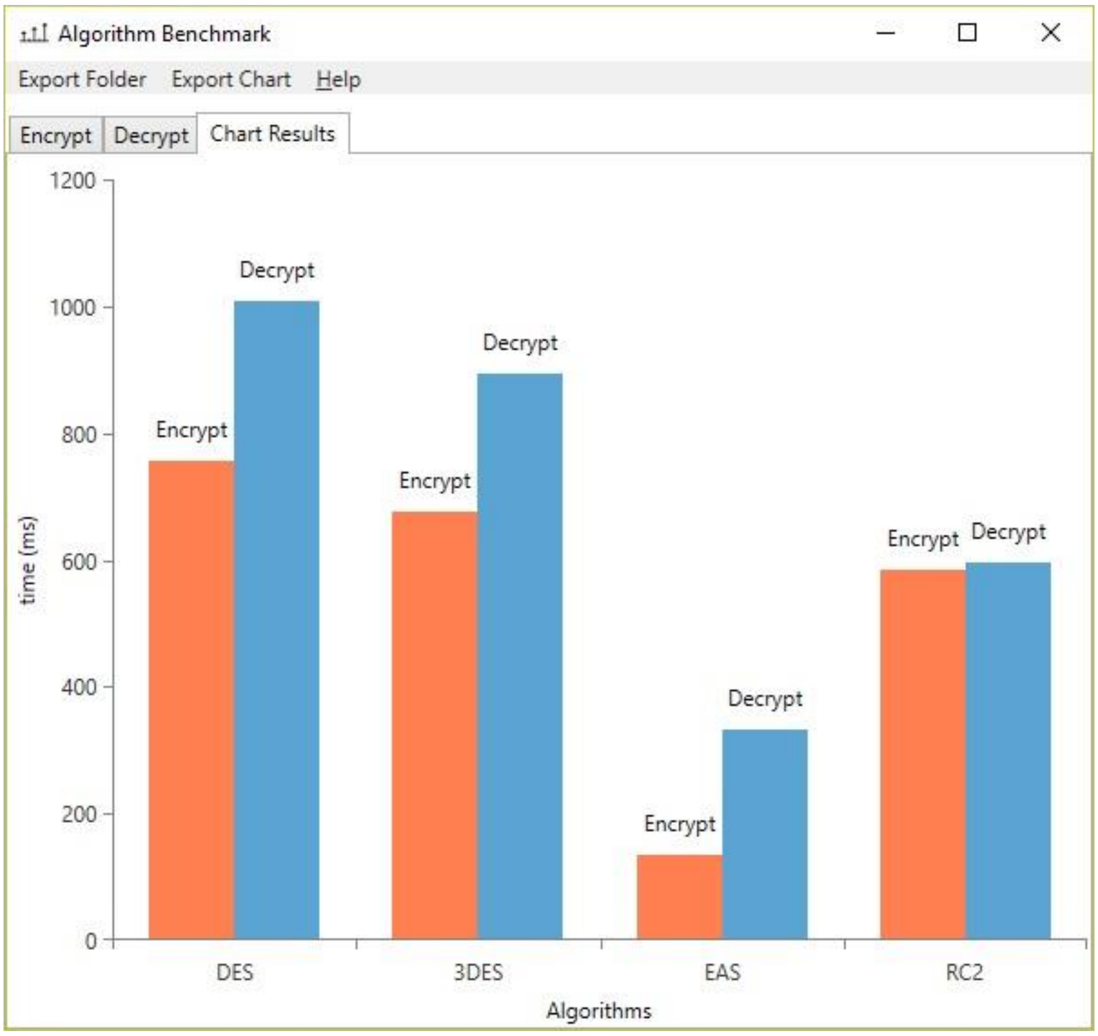


Figure 29. Algorithm Benchmark Chart Results

Chapter 4
Benchmark Results

The goal of this project is to measure the processing time of each algorithm for encryption and decryption on a different files and systems. A time watch counter runs parallel with the procedure to calculate the progress time. The calculation speed of the algorithm is the data file divide by the total time that takes the particular procedure to finish. All the experiments for the synchronous cryptographic algorithms are under Windows interface with dot net framework. For the experiments, we use the data files as shown in Tables 15, 16, 17 bellows.

File Data Sample 1	
Filename	Sample1.jpg
Size	3299 Kbyte
Type	Image
Dimensions	5760 * 3840 pixels

Table 15. Data Sample 1 Details

File Data Sample 2	
Filename	Sample2.jpg
Size	2886 Kbyte
Type	Image
Dimensions	5760 * 3840 pixels

Table 16. Data Sample 2 Details

File Data Sample 3	
Filename	ISAW0022.MP4
Size	617717 Kbyte
Type	Video
Dimensions	1920 * 1080 pixels
Frame rate	60 frames/second

Table 17. Data Sample 3 Details

File Data Sample 4	
Filename	ISAW0506.MP4
Size	110902 Kbyte
Type	Video
Dimensions	1920 * 1080 pixels
Frame rate	60 frames/second

Table 18. Data Sample 4 Details

Benchmark User Password		
Algorithms	Key Size (bits)	Key password
DES	64	sample
3DES	128	sampleimage00000
AES	128	sampleimage00000
RC2	128	sampleimage00000

Table 19. Password key size

4.1 Dual Core Experiment

The first experiment was under Intel Dual-Core laptop with a combination of all encryption and decryption algorithms with a different size range of files. The benchmark process a different type of photos around 3-5 Mbytes and a video files format from 100MB up to 620MB with the same key. For every example, the benchmark program runs several times to get the most accurate results for every example, and every cipher mode try.

Hardware components:

- Mainboard: R700-X.AP12HS
- Central Processing Unit (CPU): Intel(R) Core(TM)2 Duo CPU T8300
- Physical Memory 4GB
- Hard Disk: Corsair Force LS SSD ATA Device (111.8GB)

CPU Performance T8300	
Cores	2
Processor Base Frequency	2.40 GHz
Cache	3 MB L2
Bus Speed	800 MHz FSB
FSB Parity	No
Thermal Design Power (TDP)	35W
VID Voltage Range	1.000V-1.250V

Table 20. Performance Intel® Core™2 Duo Processor T8300

4.1.1 Benchmark Cipher Mode ECB

The first examination of the experiment was with the simplest ECB cipher with the minimum key length of 128bit and 64bit for the DES (max length key). Four different file size takes place to the experiment. From the first results as shown on figures below the AES algorithm is the faster one, the RC2 algorithm is a second more quickly and is the only one that in the most cases the decryption is quicker than the encryption. The 3DES algorithm is almost the same results with the DES algorithm.

Sample 1

Benchmark Encryption ECB Sample 1 (Size:3299 Kbyte)			
Type	Time (seconds)	Speed (byte/second)	
DES	0.284	11616K	12M
3DES	0.247	13356K	13M
AES	0.103	32029K	32M
RC2	0.308	10711K	11M

Table 21.ECB Dual Core encryption Sample1

Benchmark Decryption ECB Sample 1 (Size:3299 Kbyte)			
Type	Time (seconds)	Speed (byte/second)	
DES	0.366	9014K	9M
3DES	0.447	7380K	7M
AES	0.136	24257K	24M
RC2	0.212	15561K	16M

Table 22. ECB Dual Core decryption Sample1

The graph below shows the speed of algorithms of encryption and decryption in ECB mode with a sample data size 3299 Kbyte. DES algorithm encryption is slightly faster than decryption also 3DES remain to the same structure but is gently slower in encryption and slightly slower in decryption. The AES is significantly quicker from the other algorithms with a bit difference between encryption/decryption. The RC2 is the second faster algorithm and decryption is swifter than encryption. In this sample, the 3DES algorithm is not 3 times faster than DES because the file disassemble in a two smaller buffer size of 2000KB and 1299KB in a parallel process. The encryption is not the same with decryption because the results are in a real situation and the environment of the system change due to the different address array pointer.

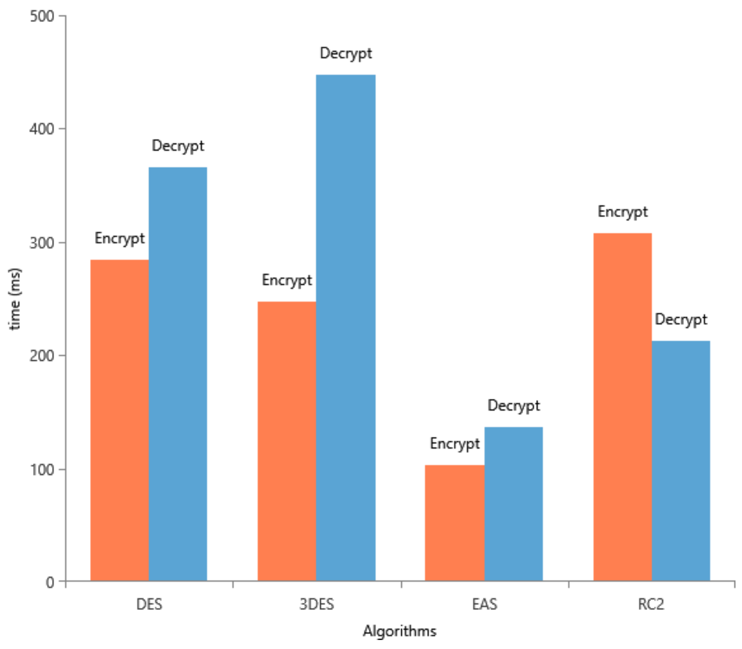


Figure 30. ECB Dual Core Sample1 Chart

Sample 2

Benchmark Encryption ECB Sample 2 (Size:2886 Kbyte)			
Type	Time (seconds)	Speed (byte/second)	
DES	0.232	12440K	12M
3DES	0.213	13549K	14M
AES	0.059	48915K	49M
RC2	0.153	18863K	19M

Table 23. ECB Dual Core encryption Sample2

Benchmark Decryption ECB Sample 2 (Size:2886 Kbyte)			
Type	Time (seconds)	Speed (byte/second)	
DES	0.233	12386K	12M
3DES	0.290	9952K	10M
AES	0.099	29152K	29M
RC2	0.130	22200K	22M

Table 24. ECB Dual Core Decryption Sample2

The graph below shows the speed of algorithms of encryption and decryption in ECB mode with a sample data size 2886 Kbyte. DES algorithm remained the same between encryption and decryption from the other hand 3DES decryption significantly increases. The AES is steeply quicker from the other algorithms with a moderately difference between encryption/decryption. The RC2 is the second faster algorithm with the encryption bit slower than decryption. This sample is the smaller file from the experiments and again the data buffer size separate into two pieces of 2000KB and 886KB and the parallel process gives close results to the DES and 3DES.

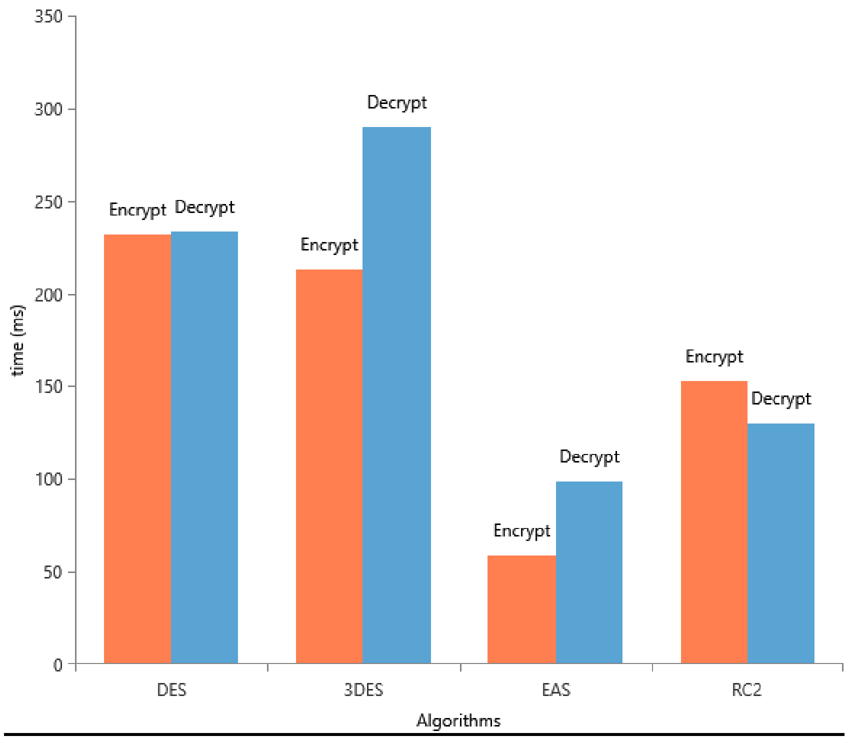


Figure 31. ECB Dual Core Sample2 Chart

Sample 3

Benchmark Encryption ECB Sample 3 (Size: 617717 Kbyte)			
Type	Time (seconds)	Speed (byte/second)	
DES	72.295	8544K	9M
3DES	76.715	8052K	8M
AES	18.335	33691K	34M
RC2	55.282	11174K	11M

Table 25. ECB Dual Core encryption Sample3

Benchmark Decryption ECB Sample 3 (Size: 617717 Kbyte)			
Type	Time (seconds)	Speed (byte/second)	
DES	78.635	7855K	8M
3DES	77.665	7954K	8M
AES	25.705	24031K	24M
RC2	48.423	12757K	13M

Table 26. ECB Dual Core decryption Sample3

The graph below shows the speed of algorithms of encryption and decryption in ECB mode with a sample data size 617717 Kbyte. Both DES, 3DES algorithms stabilized to the same levels of encryption and decryption. The AES is steeply quicker from the other algorithms with a moderately difference between encryption/decryption. The RC2 is again the second faster algorithm with the encryption slightly delayed in a relation of the decryption.

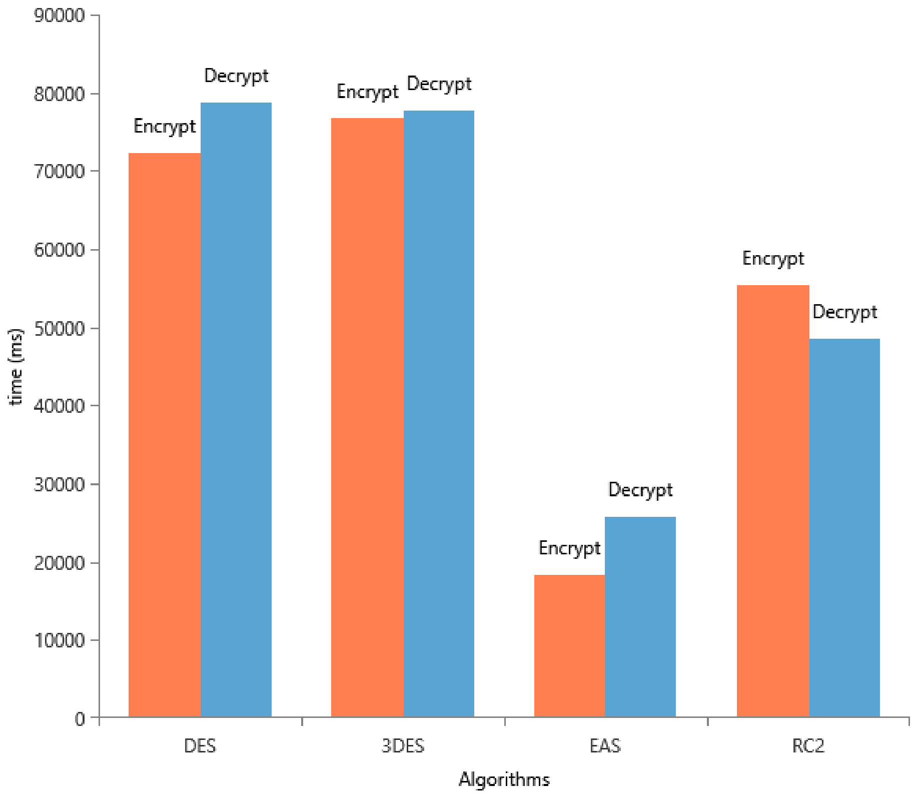


Figure 32. ECB Dual Core Sample3 Chart

Sample 4

Benchmark Encryption ECB Sample 4 (Size: 110902 Kbyte)			
Type	Time (seconds)	Speed (byte/second)	
DES	11.889	9328K	9M
3DES	11.787	9409K	9M
AES	2.965	37404K	37M
RC2	8.773	12641K	13M

Table 27. ECB Dual Core encryption Sample4

Benchmark Decryption ECB Sample 4 (Size: 110902 Kbyte)			
Type	Time (seconds)	Speed (byte/second)	
DES	13.027	8513K	9M
3DES	15.275	7260K	7M
AES	4.636	23922K	24M
RC2	9.162	12105K	12M

Table 28. ECB Dual Core decryption Sample4

The graph below shows the speed of algorithms of encryption and decryption in ECB mode with a sample data size 110902 Kbyte. Both DES, 3DES algorithms stabilized to the same levels of encryption the 3DES is slightly slower on decryption. The AES the faster algorithm with little slower on decryption. The RC2 is a faster algorithm with almost to the same encryption and decryption. The two big files of sample 3, 4 can easy understand the fast algorithm with parallel threads. The encryption and decryption are close in all the algorithms but the decryption is slightly slower than the encryption direction due to asymmetries in the key schedule.

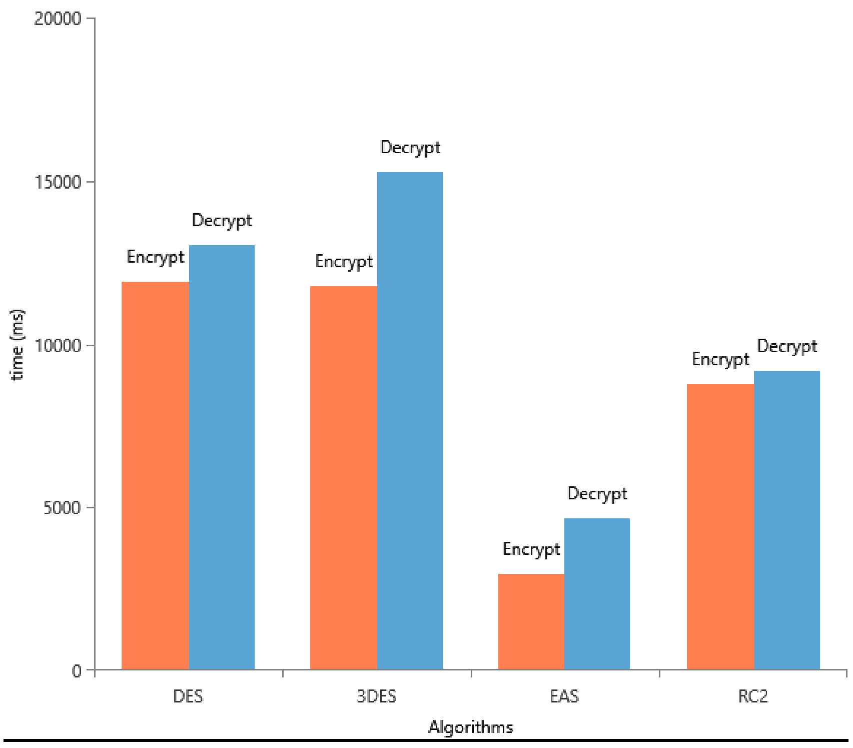


Figure 33. ECB Dual Core Sample4 Chart

4.1.2 Benchmark Cipher Mode CBC

The second try of the benchmark program is with the CBC mode of operation that uses a XOR gate with the previous ciphertext. Again, on this mode, AES is the faster then RC2 comes second, and DES with 3DES are almost the same.

Sample 1

Benchmark Encryption CBC Sample 1 (Size:3299 Kbyte)			
Type	Time (seconds)	Speed (byte/second)	
DES	0.236	13979K	14M
3DES	0.253	13040K	13M
AES	0.057	57877K	58M
RC2(64,128)	0.177/0.205	18638K/16092K	19M/16M

Table 29. . CBC Dual Core encryption Sample1

Benchmark Decryption CBC Sample 1 (Size:3299 Kbyte)			
Type	Time (seconds)	Speed (byte/second)	
DES	0.277	11910K	12M
3DES	0.320	10309K	10M
AES	0.073	45192K	45M
RC2(64,128)	0.138/0.206	23906K/16014K	24M/16M

Table 30. CBC Dual Core decryption Sample1

The graph below shows the speed of algorithms of encryption and decryption in CBC mode with a sample data size 3299 Kbyte. DES algorithm encryption is slightly faster than encryption the 3DES is different in encryption/decryption structure but is gently slower. The AES is significantly quicker from the other algorithms with a bit difference between encryption/decryption. The RC2 is a fast algorithm and decryption is swifter than encryption.

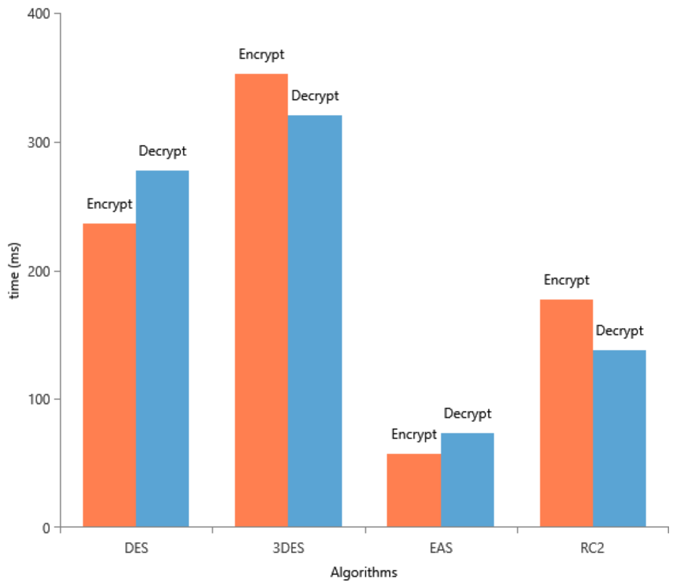


Figure 34. CBC Dual Core Sample1 Chart

Sample 2

Benchmark Encryption CBC Sample 2 (Size:2886 Kbyte)			
Type	Time (seconds)	Speed (byte/second)	
DES	0.200	14430K	14M
3DES	0.433	6665K	7M
AES	0.069	41826K	42M
RC2(64/128)	0.159/0.166	18151K/17385K	18M/17M

Table 31. CBC Dual Core encryption Sample2

Benchmark Decryption CBC Sample 2 (Size: 2886 Kbyte)			
Type	Time (seconds)	Speed (byte/second)	
DES	0.273	10571K	11M
3DES	0.377	7655K	8M
AES	0.078	37000K	37M
RC2(64/128)	0.132/0.157	21864K/18382K	22M/18M

Table 32.CBC Dual Core decryption Sample2

The graph below shows the speed of algorithms of encryption and decryption in CBC mode with a sample data size 2886 Kbyte. DES algorithm encryption is faster than decryption the 3DES algorithm is slower with gently slower in encryption and slightly slower in decryption. The AES is significantly quicker from the other algorithms and almost the same encryption/decryption. The RC2 is the second faster algorithm and decryption is swifter than encryption.

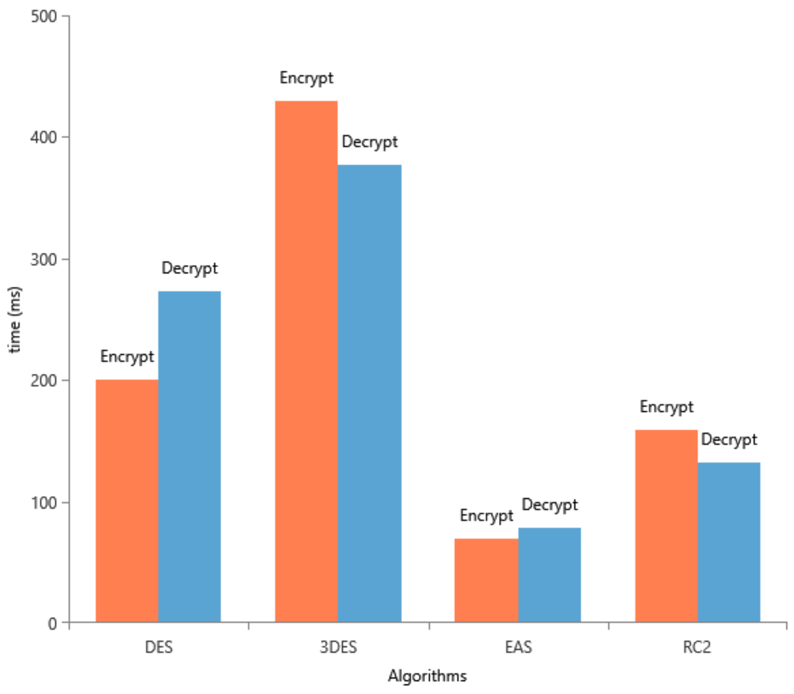


Figure 35. CBC Dual Core Sample2 Chart

<u>Sample 3</u>

Benchmark Encryption CBC Sample 3 (Size: 617717 Kbyte)			
Type	Time (seconds)	Speed (byte/second)	
DES	70.551	8756K	9M
3DES	65.903	9373K	9M
AES	17.921	34469K	34M
RC2(64/128)	53.104/54.345	11632K/11366k	12M/11M

Table 33. CBC Dual Core encryption Sample3

Benchmark Decryption CBC Sample 3 (Size: 617717 Kbyte)			
Type	Time (seconds)	Speed (byte/second)	
DES	87.653	7047K	7M
3DES	93.131	6633K	7M
AES	25.666	24068K	24M
RC2(64/128)	52.574/53.567	11749K/11531	12M/11M

Table 34. CBC Dual Core decryption Sample3

The graph below shows the speed of algorithms of encryption and decryption in CBC mode with a sample data size 617717 Kbyte. DES algorithm encryption is quicker than decryption the 3DES is almost the same result than the DES. The AES is a fast algorithm with a bit faster on encryption. The RC2 is the second faster algorithm and decryption is the same with encryption.

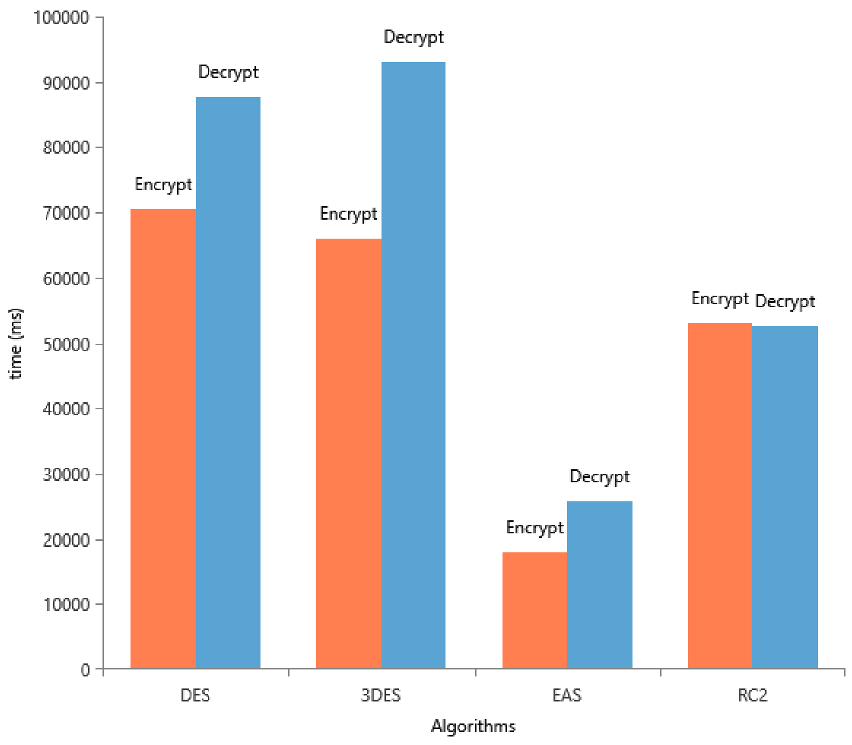


Figure 36. CBC Dual Core Sample3 Chart

<u>Sample 4</u>

Benchmark Encryption CBC Sample 4 (Size: 110902 Kbyte)			
Type	Time (seconds)	Speed (byte/second)	
DES	13.444	8249K	8M
3DES	14.026	7907K	8M
AES	3.608	30738K	31M
RC2(64/128)	9.158/10.086	12110K/10995K	12M/11M

Table 35. CBC Dual Core encryption Sample4

Benchmark Decryption CBC Sample 4 (Size: 110902 Kbyte)			
Type	Time (seconds)	Speed (byte/second)	
DES	15.088	7350K	7M
3DES	15.372	7215K	7M
AES	4.195	26437K	26M
RC2(64/128)	9.595/9.765	11558K/11357K	12M/11M

Table 36. CBC Dual Core decryption Sample4

The graph below shows the speed of algorithms of encryption and decryption in CBC mode with a sample data size 110902 Kbyte. Both DES, 3DES algorithms stabilized to the same levels of encryption the 3DES is on the same results as DES. The AES the faster algorithm with little slower on decryption. The RC2 is a faster algorithm with bit faster on encryption.

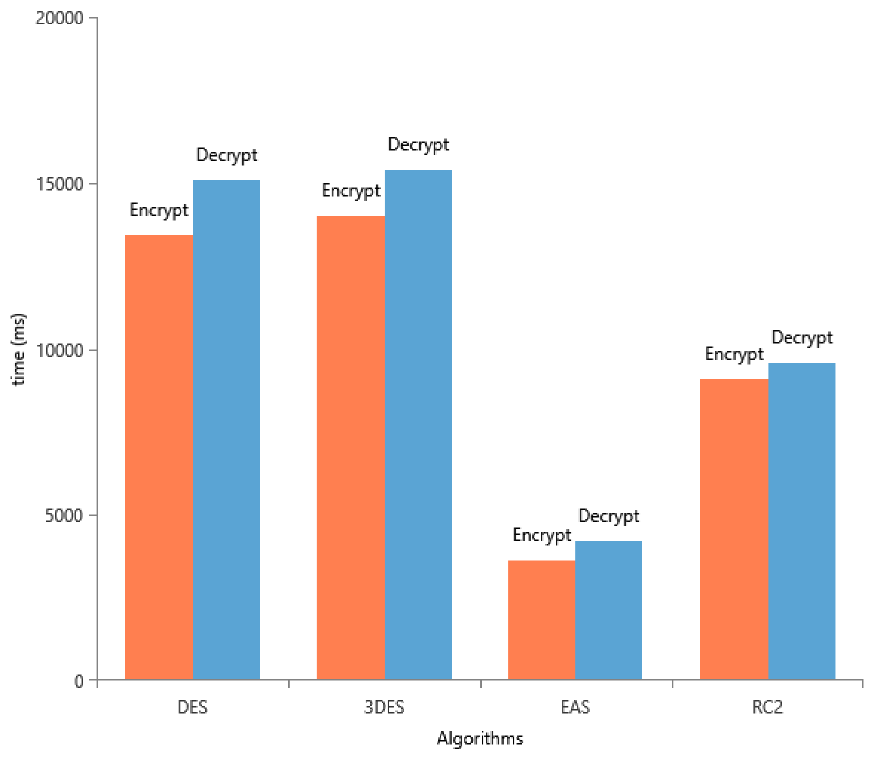


Figure 37. CBC Dual Core Sample4 Chart

4.1.3 Benchmark Cipher Mode CFB

The last investigation is the most complex cipher mode than the two previous tests. In this trial, the algorithms are slower than the others, but the algorithms stay in the same order as the previous trials.

<u>Sample1</u>

Benchmark Encryption CFB Sample 1 (Size:3299 Kbyte)			
Type	Time (seconds)	Speed (byte/second)	
DES	19.44	170K	0.170M
3DES	19.51	169K	0.169M
AES	0.668	4939K	5M
RC2	15.06	219K	0.219M

Table 37. CFB Dual Core encryption Sample1

Benchmark Decryption CFB Sample 1 (Size:3299 Kbyte)			
Type	Time (seconds)	Speed (byte/second)	
DES	22.26	148K	0.148M
3DES	20.18	163K	0.163M
AES	0.729	4525K	5M
RC2	18.17	182K	0.182M

Table 38. CFB Dual Core decryption Sample1

The graph below shows the speed of algorithms of encryption and decryption in CFB mode with a sample data size 3299 Kbyte. DES algorithm encryption is faster than encryption; the 3DES remain to the same levels but is gently slower in decryption and slightly faster in encryption. The AES is a fast algorithm with a bit difference between encryption/decryption. The RC2 is slow in this cipher mode and decryption is slower than encryption.

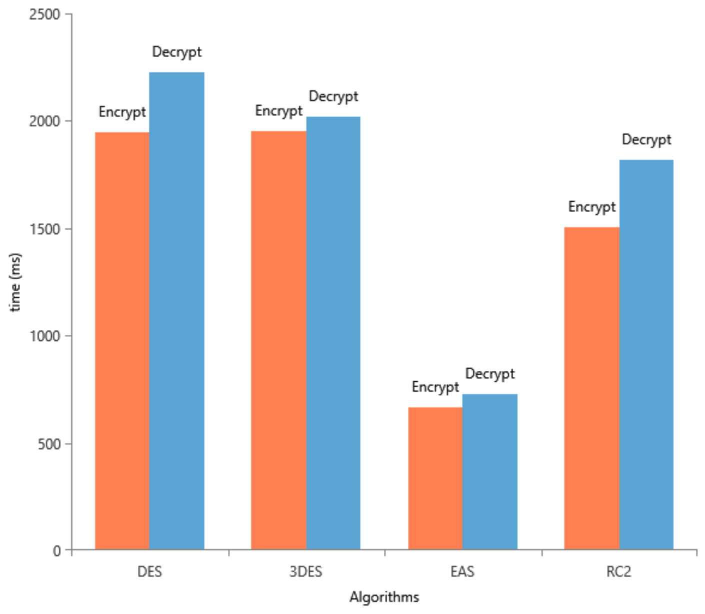


Figure 38. CFB Dual Core Sample1 Chart

Sample2

Benchmark Encryption CFB Sample 2 (Size: 2886 Kbyte)			
Type	Time (seconds)	Speed (byte/second)	
DES	22.98	126K	0.126M
3DES	19.67	147K	0.147M
AES	0.644	4481K	4.000M
RC2(64/128)	15.22	190K	0.190M

Table 39. CFB Dual Core encryption Sample2

Benchmark Decryption CFB Sample 2 (Size: 2886 Kbyte)			
Type	Time (seconds)	Speed (byte/second)	
DES	21.51	134K	0.134M
3DES	16.74	172K	0.172M
AES	0.623	4632K	5.000M
RC2(64/128)	14.59	198K	0.198M

Table 40. CFB Dual Core decryption Sample2

The graph below shows the speed of algorithms of encryption and decryption in CFB mode with a sample data size 2886 Kbyte. DES is slower than 3DES both are slightly slower on encryption. The AES the faster algorithm with both encryption/decryption on the same level. The RC2 is gently faster algorithm than DES and 3DES.

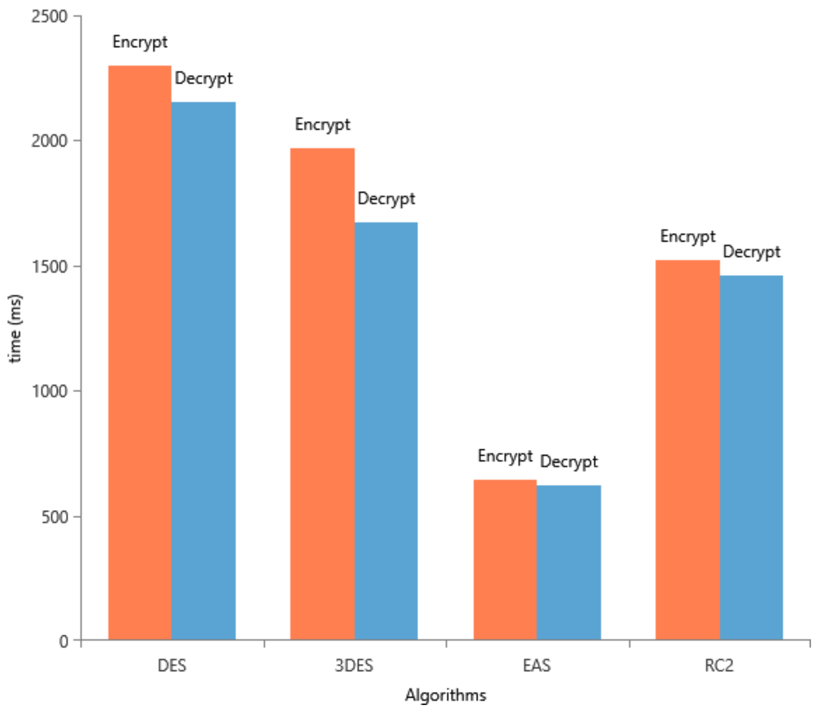


Figure 39. CFB Dual Core Sample2 Chart

Sample3

Benchmark Encryption CFB Sample 3 (Size: 617717 Kbyte)			
Type	Time (seconds)	Speed (byte/second)	
DES	507.162	1218K	1M
3DES	503.173	1228K	1M
AES	162.909	3792K	4M
RC2(64/128)	366.225	1687K	2M

Table 41. CFB Dual Core encryption Sample3

Benchmark Decryption CFB Sample 3 (Size: 617717 Kbyte)			
Type	Time (seconds)	Speed (byte/second)	
DES	498.887	1238K	1M
3DES	619.583	997K	1M
AES	210.466	2935K	3M
RC2(64/128)	431.247	1432K	1M

Figure 40. CFB Dual Core decryption Sample3

The graph below shows the speed of algorithms of encryption and decryption in CFB mode with a sample data size 617717 Kbyte. DES remain the same, as 3DES but the decryption is slower. The AES is the faster algorithm with encryption step faster. The RC2 is gently faster algorithm than DES and 3DES.

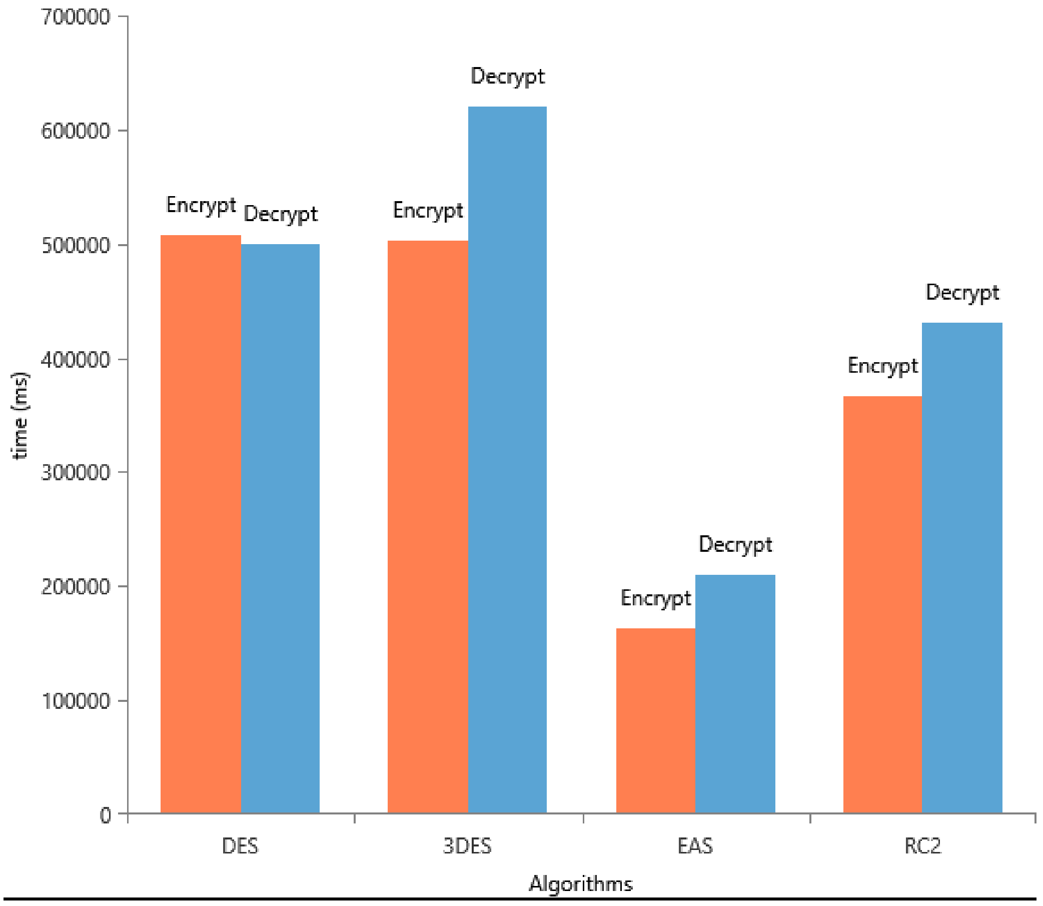


Figure 41. CFB Dual Core Sample3 Chart

Sample 4

Benchmark Encryption CFB Sample 4 (Size: 110902 Kbyte)			
Type	Time (seconds)	Speed (byte/second)	
DES	89.863	1234K	1M
3DES	95.540	1161K	1M
AES	29.477	3762K	4M
RC2(64/128)	68.834	1611K	2M

Table 42. CFB Dual Core encryption Sample4

Benchmark Decryption CFB Sample 4 (Size: 110902 Kbyte)			
Type	Time (seconds)	Speed (byte/second)	
DES	90.934	1220K	1M
3DES	92.917	1194K	1M
AES	30.615	3622K	4M
RC2(64/128)	66.048	1679K	2M

Figure 42. CFB Dual Core decryption Sample4

The graph below shows the speed of algorithms of encryption and decryption in CFB mode with a sample data size 110902 Kbyte. Both DES, 3DES algorithms stabilized to the same levels of encryption the 3DES is slightly slower on encryption. The AES the faster algorithm with little slower on decryption. The RC2 is a faster algorithm with a little bit slower to encryption.

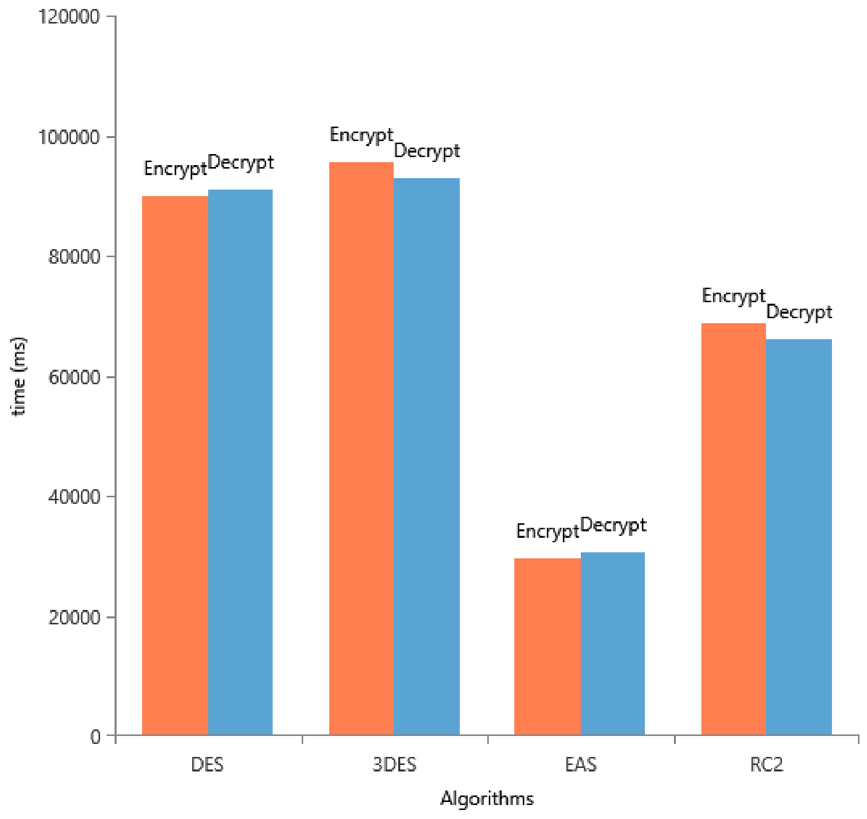


Figure 43. CFB Dual Core Sample4 Chart

4.1.4 Dual Core Benchmark Overview

A summary of the previous results is easier to identify that AES is a fast algorithm. The ECB mode of operation is the simpler one the AES algorithm process the data faster in CBC mode. The rest of the algorithms compare them from ECB versus CBC mode is not a big difference between those two cipher modes. The ECB is a basic operation but is faster than the others for the reason that same block repeats it will result the same ciphertext. The CFB mode very slow compares the other ones the process speed reduce dramatically. An important performance parameter is the memory utilization, the file size increases memory size is drastically increased in AES means for extra-large files, it need a system with good memory and more CPU.

Average Algorithms Speed ECB Mode				
Algorithm	Encryption Speed bytes/second		Decryption Speed bytes/second	
DES	10482K	10M	9442K	9M
3DES	11091.5K	11M	8136.5K	8M
AES	38009.75K	30M	25340.5K	25M
RC2	13347.25K	13M	15655.75K	15M

Table 43. Overview Speed ECB Mode

Average Algorithms Speed CBC Mode				
Algorithm	Encryption Speed bytes/second		Decryption Speed bytes/second	
DES	11354K	11M	9220K	9M
3DES	9246K	9M	7953K	8M
AES	41228K	41M	33174K	33M
RC2(64/128)	15133K/99407K	15M/10M	17269K/14321K	17M/14M

Table 44. Overview Speed CBC Mode

Average Algorithms Speed CFB Mode				
Algorithm	Encryption Speed bytes/second		Decryption Speed bytes/second	
DES	687K	0.687M	685K	0.685M
3DES	676K	0.676M	632K	0.632M
AES	4244K	4M	3929K	4M
RC2	927K	0.927M	873K	0.873M

Table 45. Overview Speed CFB Mode

The figures below show the progress of each algorithm as the data file increase in the different benchmark scenarios. In the first ECB scenario, the two algorithms AES, RC2 have a starting point in a high peak and immediately after the size increases the speed decreases vertically. The CBC mode is slightly different the algorithms elevated in a minor and reduced but is more stable until the end. The last mode is CFB and is the slower one, and even here the AES start from high to low but with not a big difference. The latency for individual tasks is slow in cryptographic contexts CFB mode requires the complete processing for each block before the next can be processed.[15, p. 392]

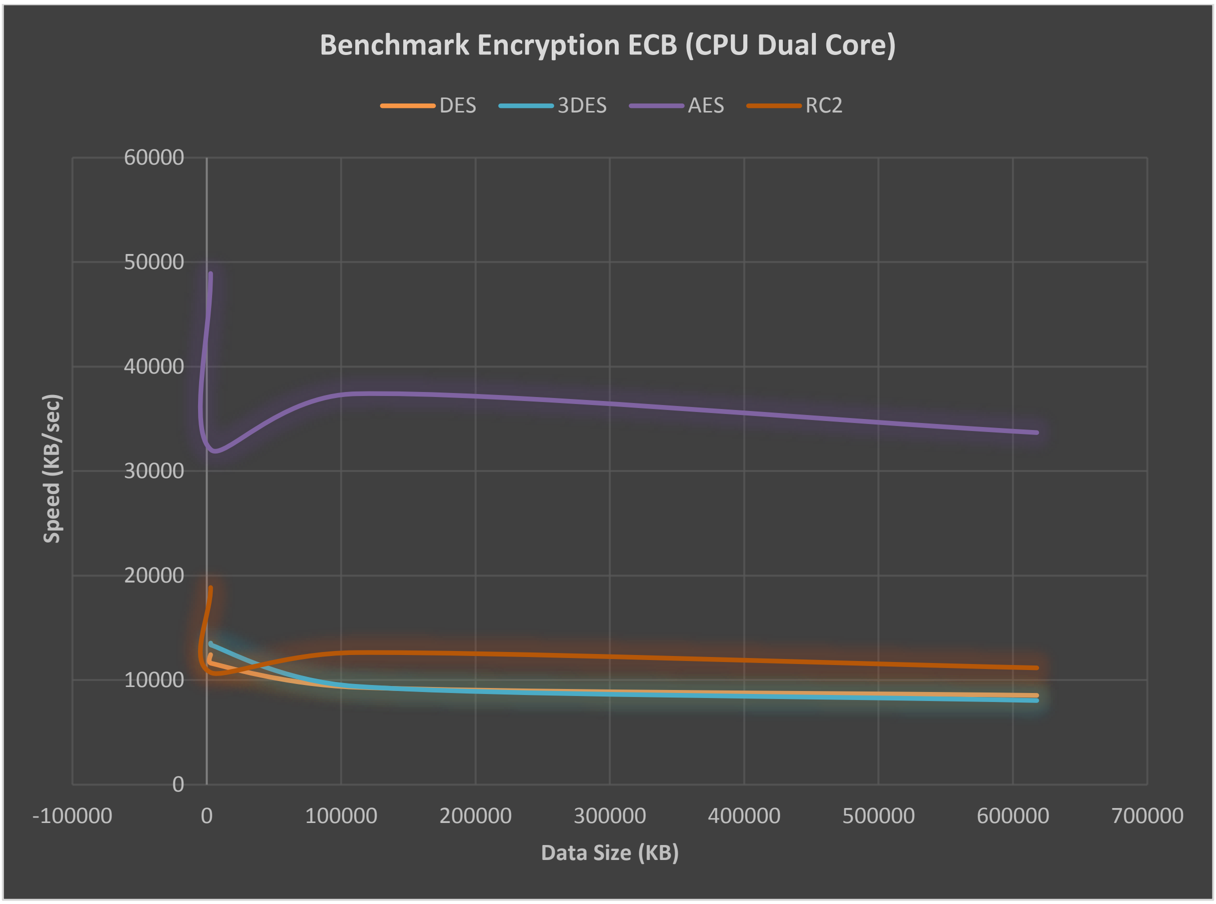


Figure 44. Encryption Speed ECB (CPU Dual core)

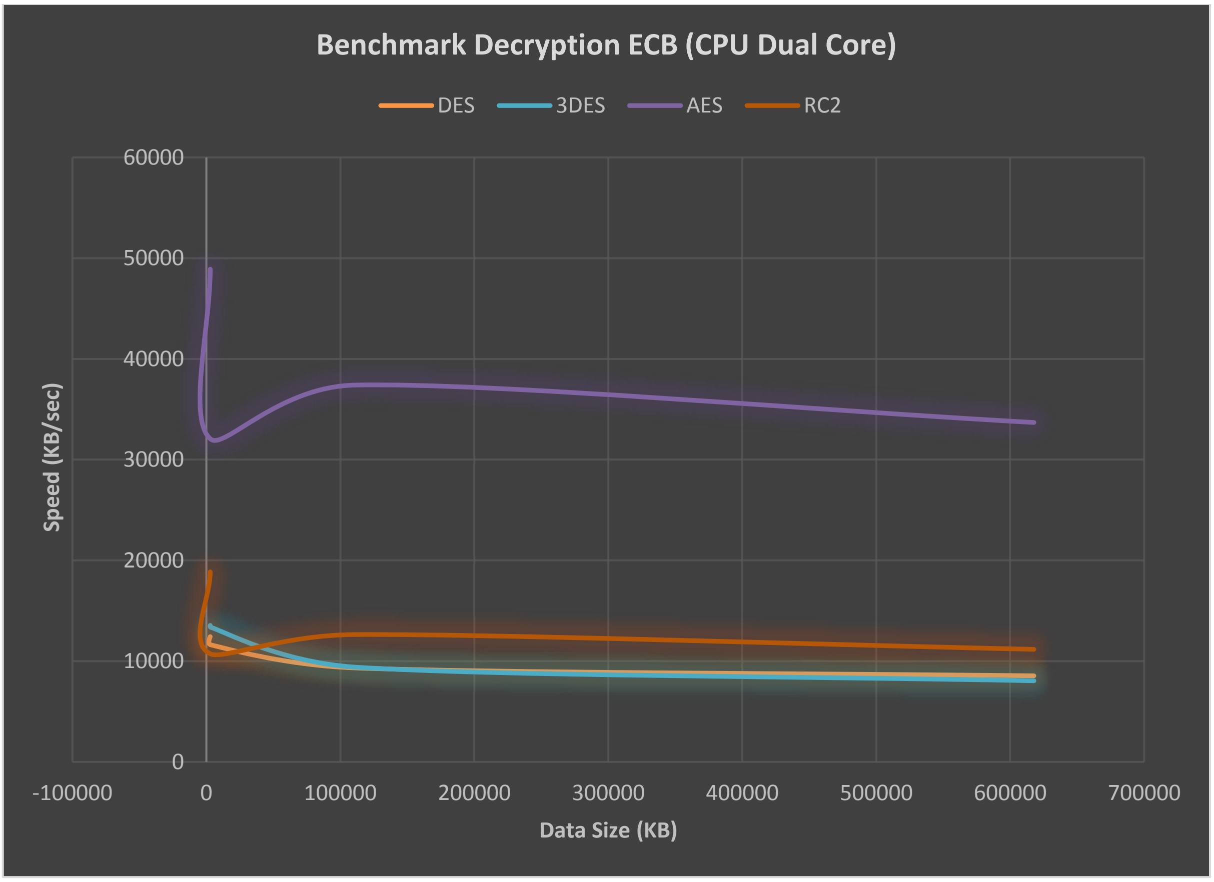


Figure 45. Decryption Speed ECB (CPU Dual core)

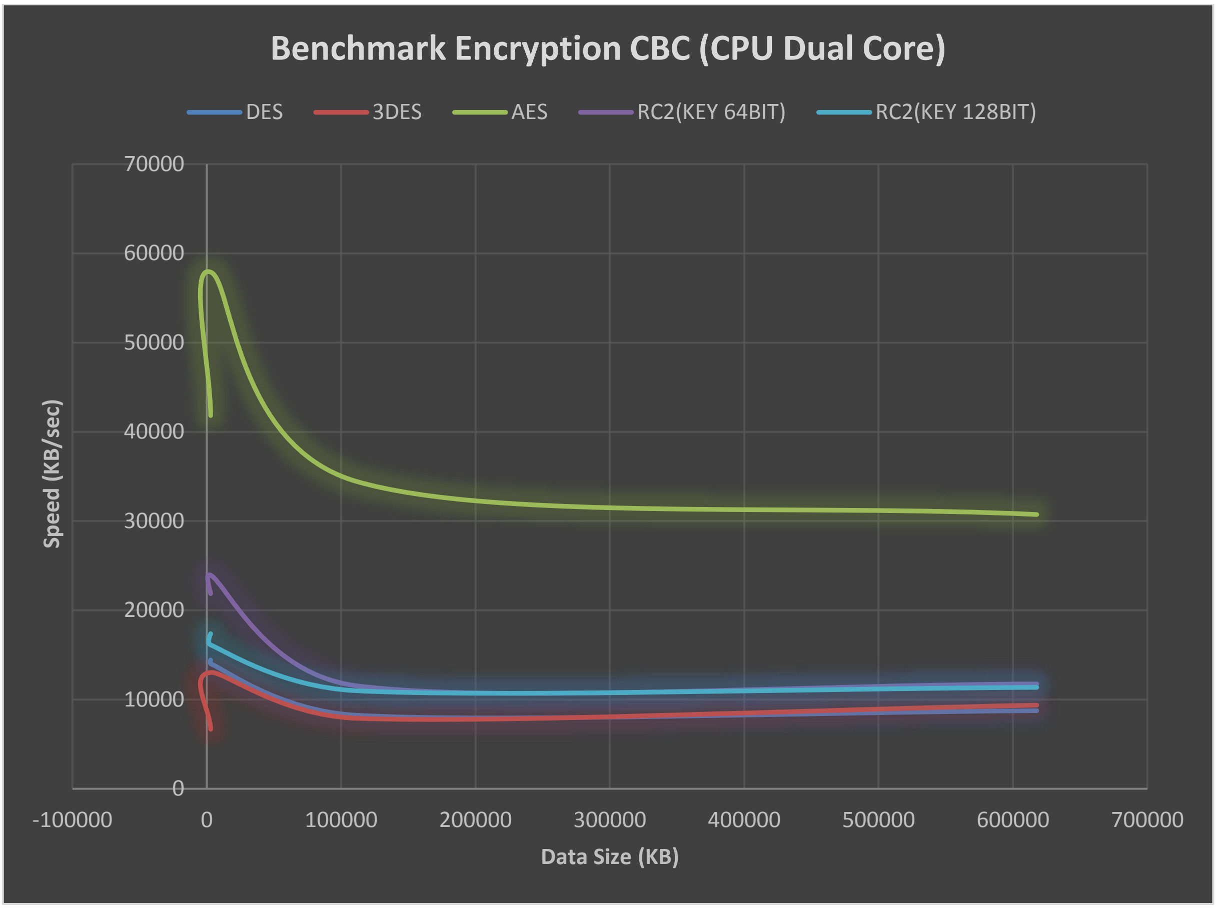


Figure 46. Encryption Speed CBC (CPU Dual core)

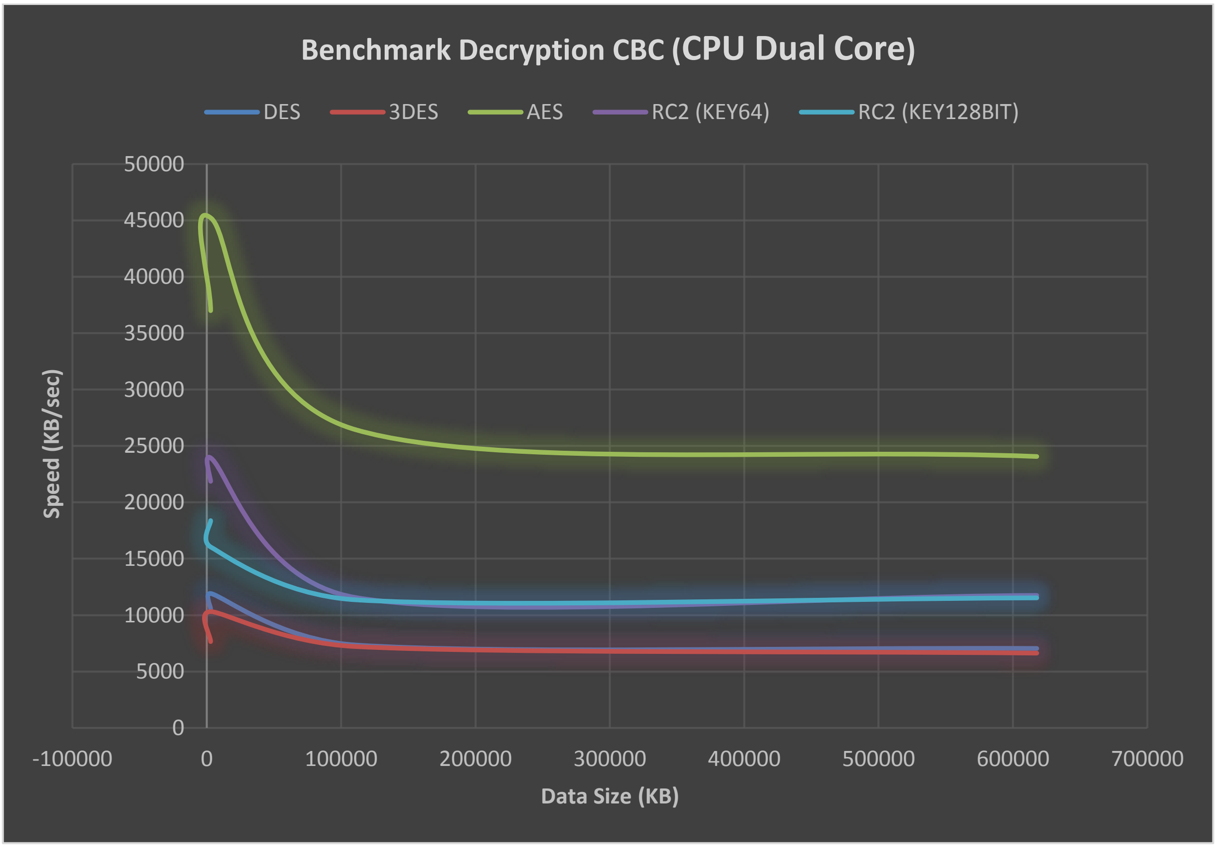


Figure 47. Decryption Speed CBC (CPU Dual core)

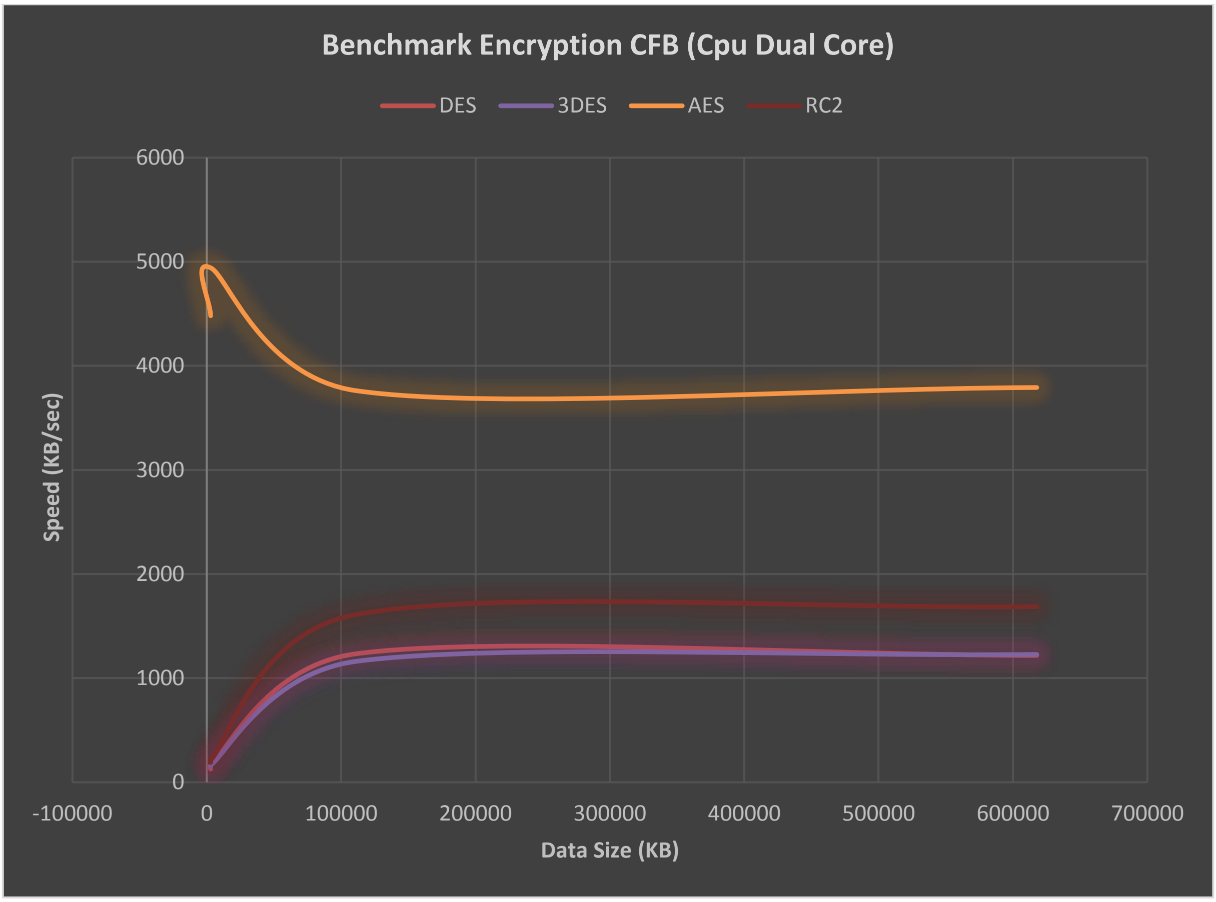


Figure 48. Encryption Speed CFB (CPU Dual core)

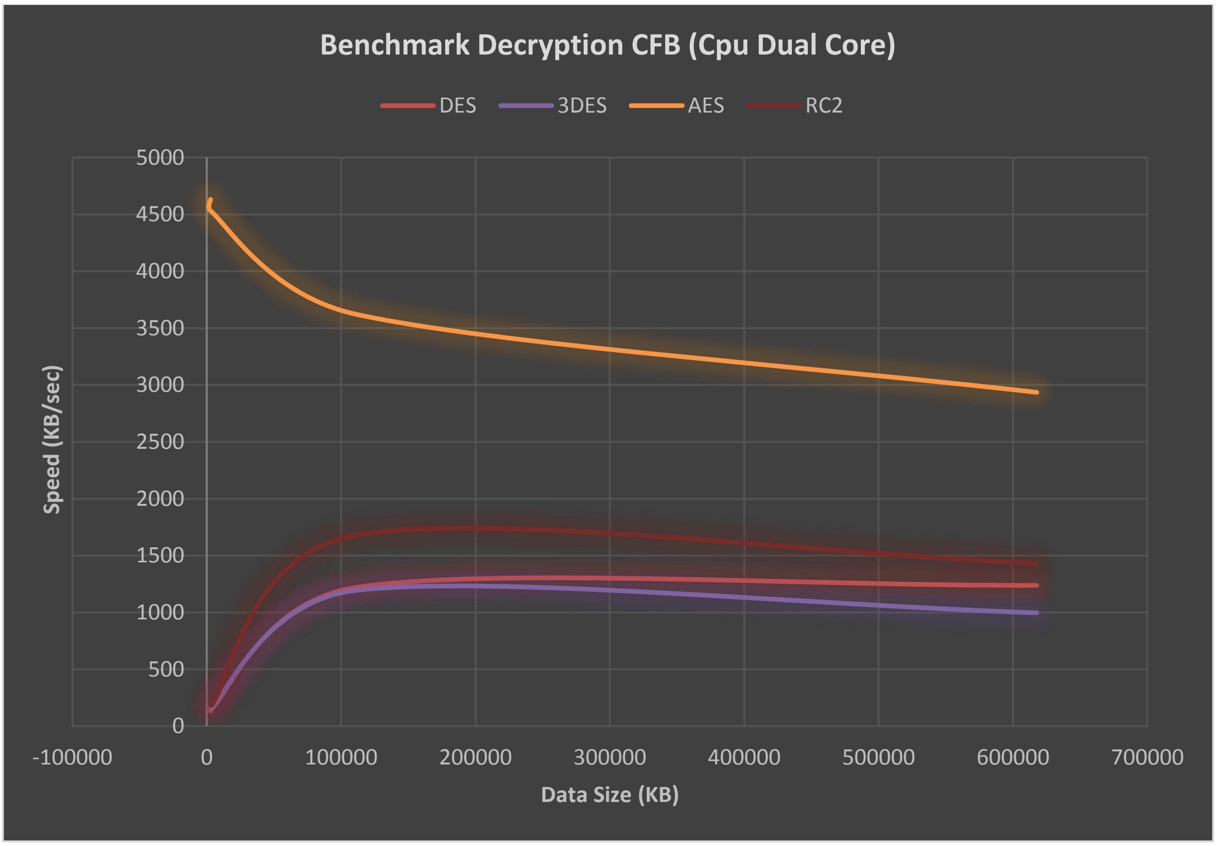


Figure 49. Decryption Speed CFB (CPU Dual core)

4.2 Intel Xeon Experiment

For the second experiment, we use a Google Cloud Platform that can enable to launch virtual machines on different operating systems. The virtual machines have a feature to customize the needs. The software and the data files are the same of the first experiment.

Hardware components:

- Mainboard: Google Compute Engine
- Central Processing Unit (CPU): Intel(R) Xeon(R) CPU @ 2.30GHz [Family 6 Model 63 Stepping 0]
- Hard Disk: Google Persistent SCSI Disk Device

Intel(R) Xeon(R) CPU	
Cores	8
Processor Base Frequency	2.40 GHz
Cache	12 MB Smart Cache
Bus Speed	5.86 GT/s QPI
Thermal Design Power (TDP)	80W
VID Voltage Range	0.750V-1.350V

Table 46. Intel(R) Xeon(R) CPU [Family 6 Model 63 Stepping 0]

4.2.1 Benchmark Cipher Mode ECB

The first analysis was with the simplest ECB and from the first results; there is a big difference between AES algorithm. Compare the results with the Dual Core CPU on table 21, 22 the DES and 3DES is not even double the speed. From the other side, RC2 is two times faster, and AES increases five times faster.

Sample 1

Benchmark Encryption ECB Sample 1 (Size:3299 Kbyte)			
Type	Time (seconds)	Speed (byte/second)	
DES	0.183	18027K	18M
3DES	0.207	15937K	16M
AES	0.021	157095K	157M
RC2	0.138	23906K	24M

Table 47.ECB Intel Xeon encryption Sample1

Benchmark Decryption CBC Sample 1 (Size:3299 Kbyte)			
Type	Time (seconds)	Speed (byte/second)	
DES	0.177	18638K	19M
3DES	0.190	17363K	17M
AES	0.035	94257K	94M
RC2	0.099	33323K	33M

Table 48. ECB Intel Xeon decryption Sample1

The graph below shows the speed of algorithms of encryption and decryption in CBC mode with a sample data size 3299 Kbyte. DES algorithm decryption is slightly faster than encryption also 3DES remain to the same structure but is gently slower in encryption and slightly slower in decryption. The AES is significantly quicker from the other algorithms and the encryption is faster than decryption. The RC2 is the second faster algorithm and decryption is faster in this case than encryption.

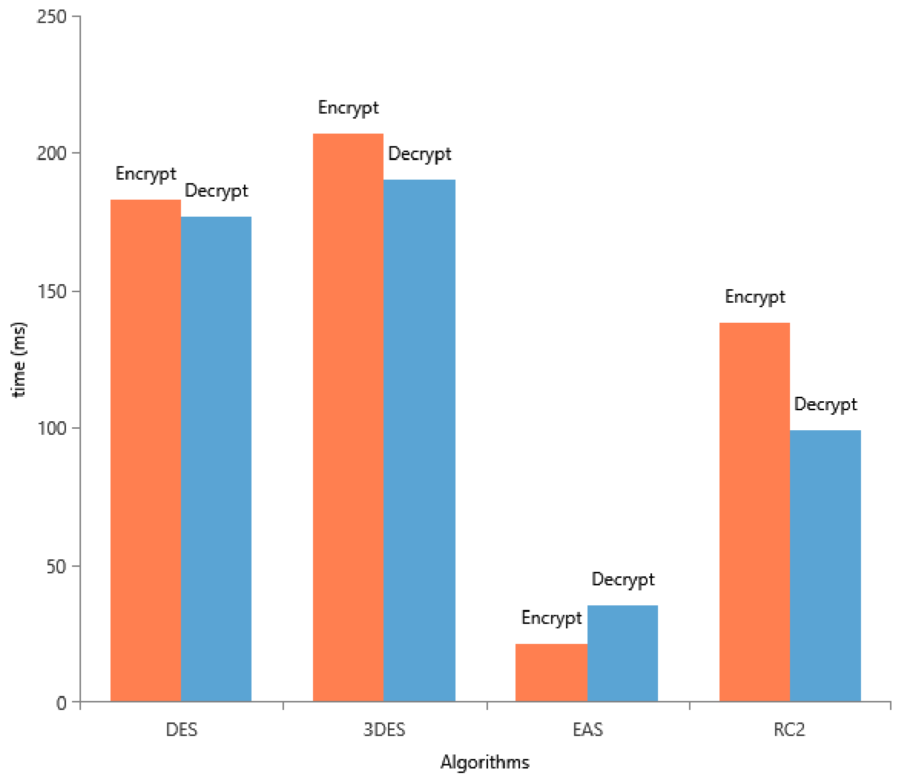


Figure 50. ECB Intel Xeon Sample1 Chart

<u>Sample 2</u>

Benchmark Encryption ECB Sample 2 (Size:2886 Kbyte)			
Type	Time (seconds)	Speed (byte/second)	
DES	0.144	20042K	20M
3DES	0.145	19903K	20M
AES	0.014	206143K	206M
RC2	0.101	28574K	29M

Table 49. ECB Intel Xeon encryption Sample2

Benchmark Decryption ECB Sample 2 (Size:2886 Kbyte)			
Type	Time (seconds)	Speed (byte/second)	
DES	0.149	19369K	19M
3DES	0.149	19369K	19M
AES	0.026	111000K	111M
RC2	0.083	34771K	35M

Figure 51. ECB Intel Xeon decryption Sample2

The graph below shows the speed of algorithms of encryption and decryption in ECB mode with a sample data size 2886 Kbyte. DES and 3DES algorithm stabilized to the same level. The AES is sharply quicker from the other algorithms and the encryption is more rapidly than decryption. The RC2 is the second faster algorithm and decryption is faster in this case than encryption.

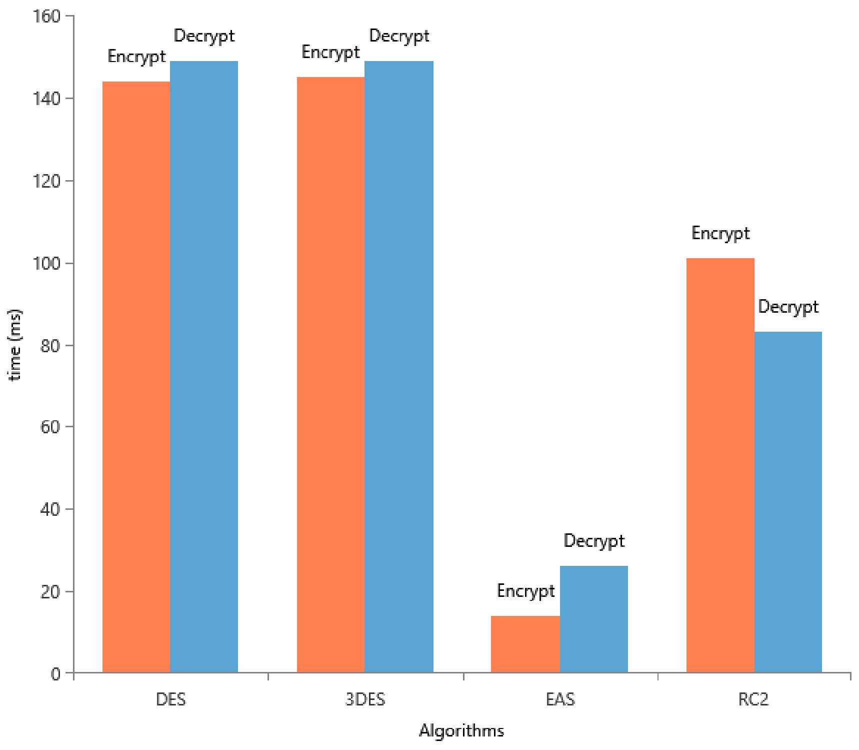


Figure 52.ECB Intel Xeon Sample2 Chart

Sample 3

Benchmark Encryption ECB Sample 3 (Size: 617717 Kbyte)			
Type	Time (seconds)	Speed (byte/second)	
DES	30.327	20369K	20M
3DES	29.896	20662K	21M
AES	2.241	275643K	276M
RC2	20.622	29954K	30M

Table 50. ECB Intel Xeon encryption Sample3

Benchmark Decryption ECB Sample 3 (Size: 617717 Kbyte)			
Type	Time (seconds)	Speed (byte/second)	
DES	31.152	19829K	20M
3DES	31.021	19913K	20M
AES	3.605	171350K	171M
RC2	16.189	38157K	38M

Table 51. ECB Intel Xeon decryption Sample3

The graph below shows the speed of algorithms of encryption and decryption in ECB mode with a sample data size 617717 Kbyte. DES and 3DES algorithm remained constant to the same level. The AES is sharply the faster algorithm and the encryption is more rapidly than decryption. The RC2 is the second faster algorithm and decryption is faster in this case than encryption.

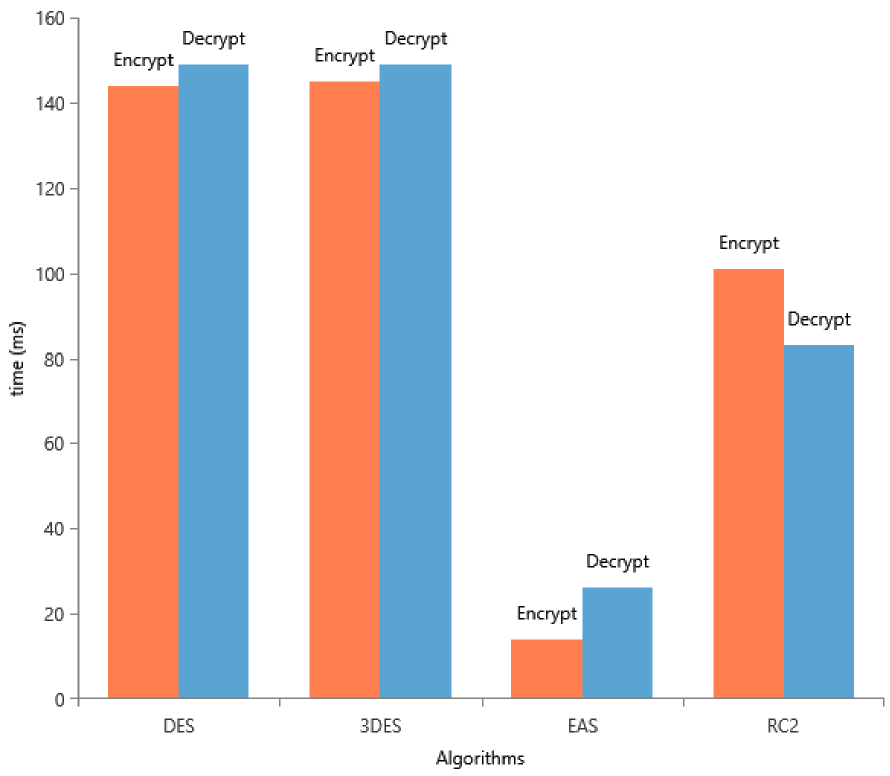


Figure 53. ECB Intel Xeon Sample3 Chart

Sample 4

Benchmark Encryption ECB Sample 4 (Size: 110902 Kbyte)			
Type	Time (seconds)	Speed (byte/second)	
DES	5.392	20568K	21M
3DES	5.258	21092K	21M
AES	0.258	429853K	430M
RC2	3.707	29917K	30M

Table 52. CBC Intel Xeon encryption Sample4

Benchmark Decryption ECB Sample 4 (Size: 110902 Kbyte)			
Type	Time (seconds)	Speed (byte/second)	
DES	5.453	20338K	20M
3DES	5.256	21100K	21M
AES	0.558	198749K	199M
RC2	2.852	38886K	39M

Figure 54. ECB Intel Xeon encryption Sample4

The graph below shows the speed of algorithms of encryption and decryption in ECB mode with a sample data size 110902 Kbyte. Both DES, 3DES algorithms stabilized to the same levels. The AES the faster algorithm with little slower on decryption. The RC2 is a faster algorithm with a faster decryption.

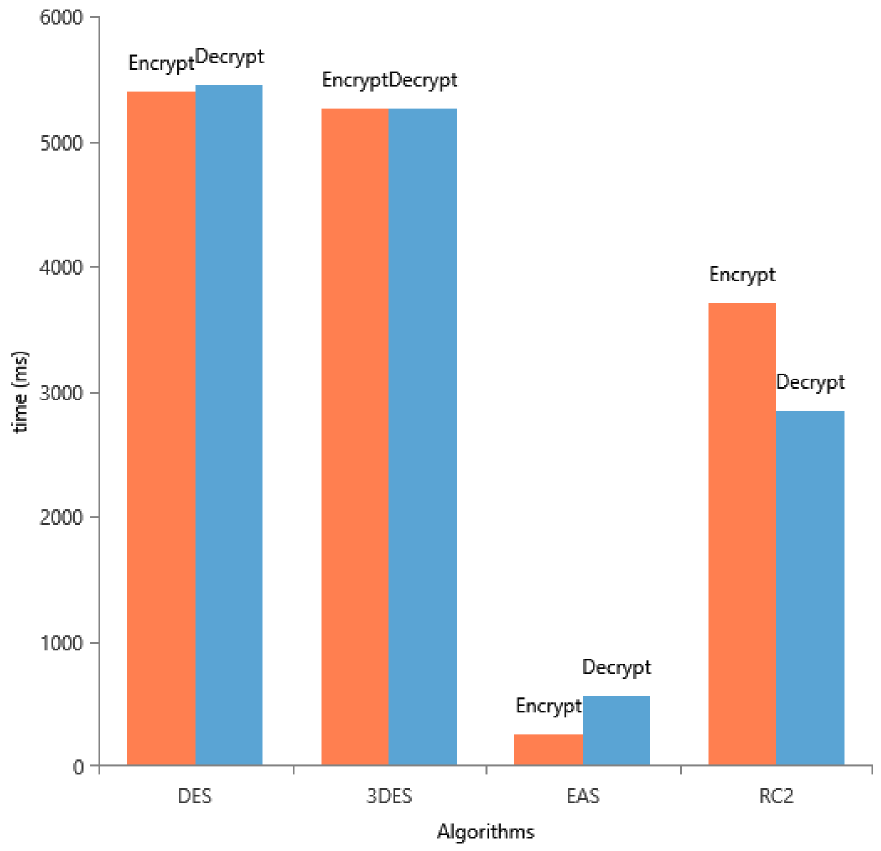


Figure 55. ECB Intel Xeon Sample4 Chart

4.2.2 Benchmark Cipher Mode CBC

The second try with the CBC mode of operation shows that is not so fast on small files. The process of AES algorithm is going faster on a large data files. Again, on this mode, AES is the faster then RC2 comes second, and DES with 3DES are almost the same.

Sample 1

Benchmark Encryption CBC Sample 1 (Size:3299 Kbyte)			
Type	Time (seconds)	Speed (byte/second)	
DES	0.236	13979K	14M
3DES	0.353	9346K	9M
AES	0.057	57877K	58M
RC2	0.177	18638K	19M

Table 53.ECB Intel Xeon encryption Sample1

Benchmark Decryption CBC Sample 1 (Size:3299 Kbyte)			
Type	Time (seconds)	Speed (byte/second)	
DES	0.277	11910K	12M
3DES	0.320	10309K	10M
AES	0.073	45192K	45M
RC2	0.138	23906K	24M

Table 54. ECB Intel Xeon decryption Sample1

The graph below shows the speed of algorithms of encryption and decryption in CBC mode with a sample data size 3299 Kbyte. DES algorithm is different between encryption and decryption from the other hand 3DES is slower and decryption is faster. The AES is very quick with a moderately difference between encryption/decryption. The RC2 is the next faster algorithm with the encryption bit slower than decryption.

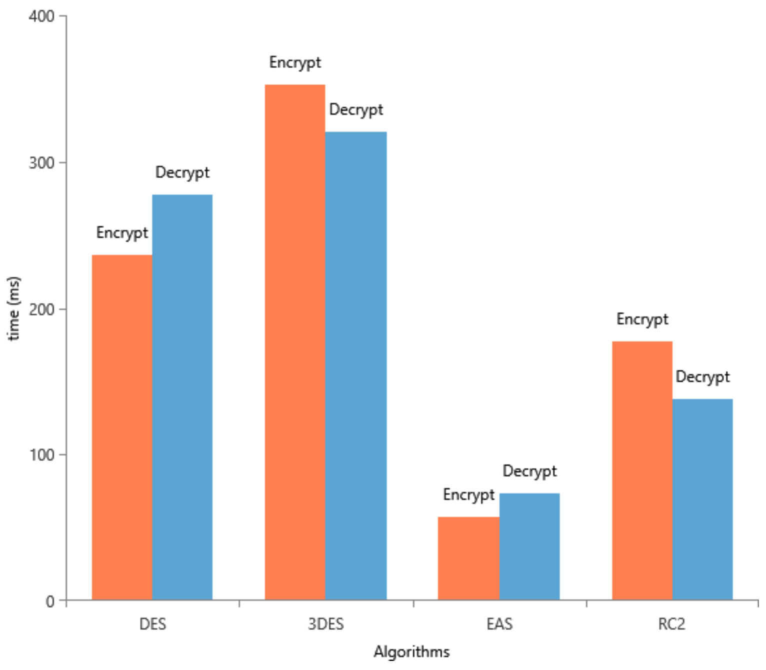


Figure 56.CBC Intel Xeon Sample1 Chart

Sample 2

Benchmark Encryption CBC Sample 2 (Size:2886 Kbyte)			
Type	Time (seconds)	Speed (byte/second)	
DES	0.154	18740K	19M
3DES	0.145	19903K	20M
AES	0.017	169765K	170M
RC2	0.106	27226K	27M

Table 55. . CBC Intel Xeon encryption Sample1

Benchmark Decryption CBC Sample 2 (Size:2886 Kbyte)			
Type	Time (seconds)	Speed (byte/second)	
DES	0.165	17491K	17M
3DES	0.154	18740K	19M
AES	0.014	206143K	206M
RC2	0.082	35195K	35M

The graph below shows the speed of algorithms of encryption and decryption in CBC mode with a sample data size 2886 Kbyte. DES algorithm is a little bit slower than 3DES. The AES is the faster algorithm with a moderately difference between encryption/decryption. The RC2 is the next faster algorithm with the encryption bit slower than decryption.

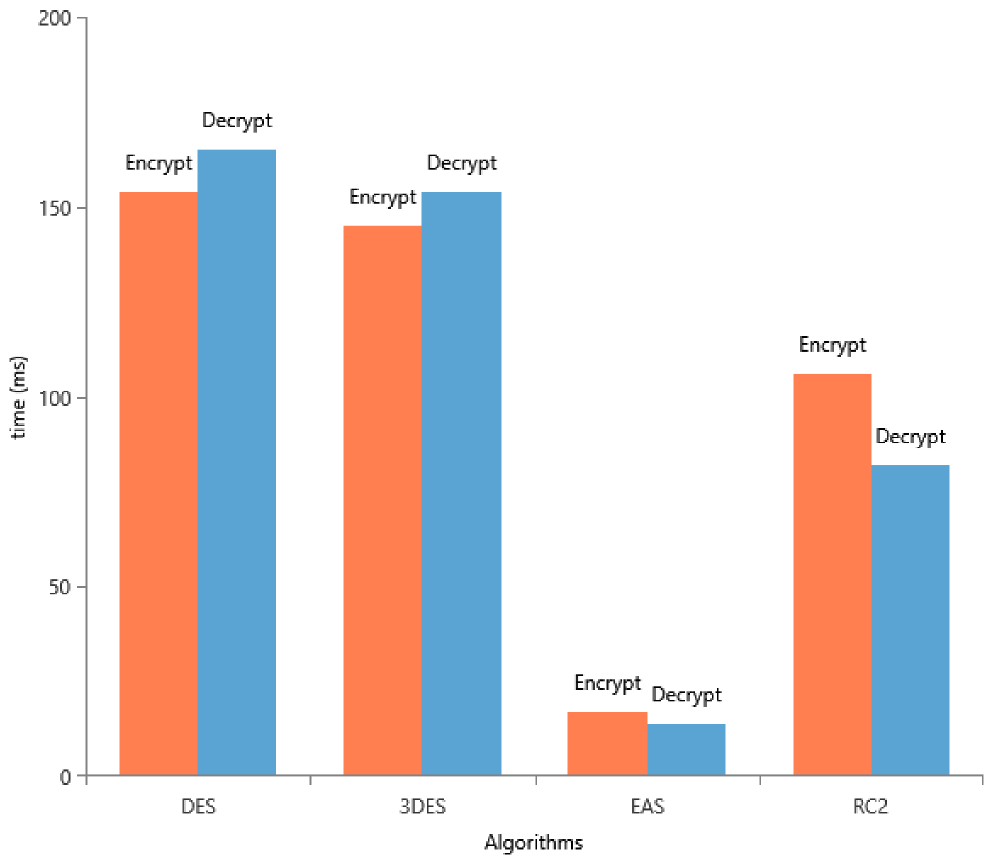


Figure 57. CBC Intel Xeon Sample2 Chart

Sample 3

Benchmark Encryption CBC Sample 3 (Size: 617717 Kbyte)			
Type	Time (seconds)	Speed (byte/second)	
DES	31.444	19645K	20M
3DES	31.463	19633K	20M
AES	2.501	246988K	247M
RC2	22.190	27838K	28M

Table 56. ECB Intel Xeon encryption Sample3

Benchmark Decryption CBC Sample 3 (Size: 617717 Kbyte)			
Type	Time (seconds)	Speed (byte/second)	
DES	33.432	18477K	18M
3DES	32.801	18832K	19M
AES	3.280	188328K	188M
RC2	17.827	34651K	35M

Table 57. ECB Intel Xeon decryption Sample3

The graph below shows the speed of algorithms of encryption and decryption in CBC mode with a sample data size 617717 Kbyte. DES algorithm is a remained constant with 3DES. The AES is the faster algorithm with a moderately difference between encryption/decryption. The RC2 is the next faster algorithm with the encryption bit slower than decryption.

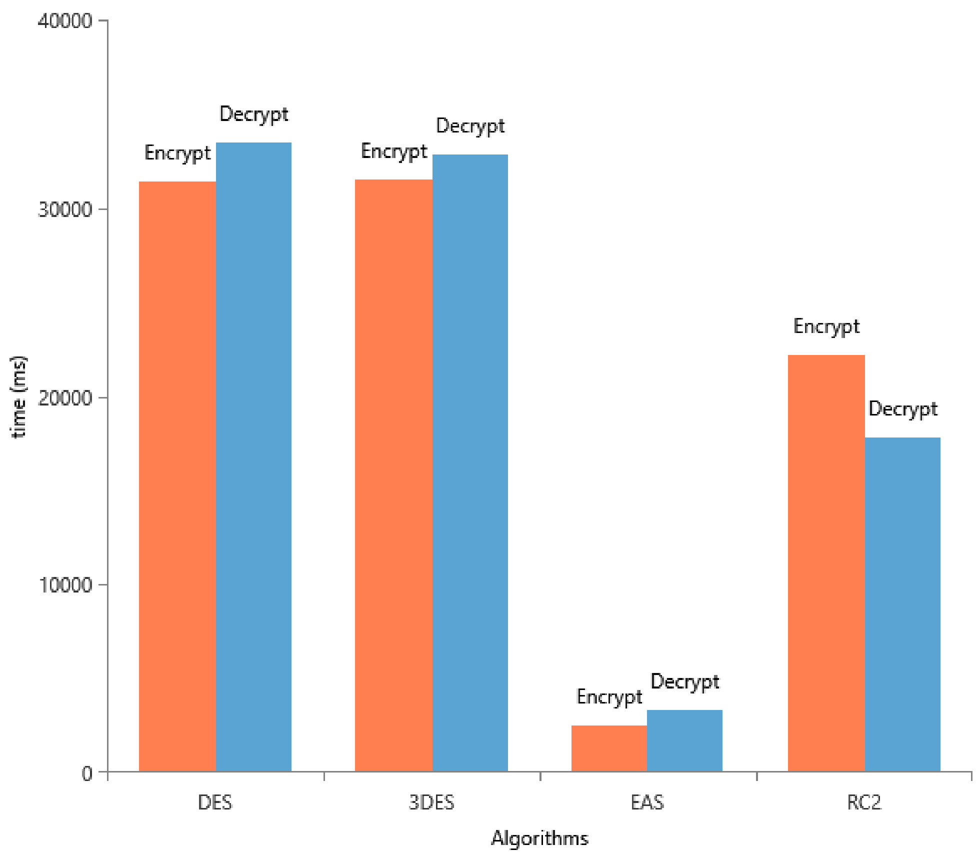


Figure 58. CBC Intel Xeon Sample3 Chart

Sample 4

Benchmark Encryption CBC Sample 4 (Size: 110902 Kbyte)			
Type	Time (seconds)	Speed (byte/second)	
DES	5.744	19307K	19M
3DES	5.595	19822K	20M
AES	0.430	257912K	258M
RC2	3.866	28686K	29M

Table 58. CBC Intel Xeon encryption Sample4

Benchmark Decryption CBC Sample 4 (Size: 110902 Kbyte)			
Type	Time (seconds)	Speed (byte/second)	
DES	5.657	19604K	20M
3DES	5.737	19331K	19M
AES	0.658	168544K	169M
RC2(64/128)	3.114	35614K	36M

Table 59. CBC Intel Xeon decryption Sample4

The graph below shows the speed of algorithms of encryption and decryption in CBC mode with a sample data size 110902 Kbyte. Both DES, 3DES algorithms stabilized to the same levels. The AES is the faster algorithm with a big difference. The RC2 is a little bit faster than DES and 3DES.

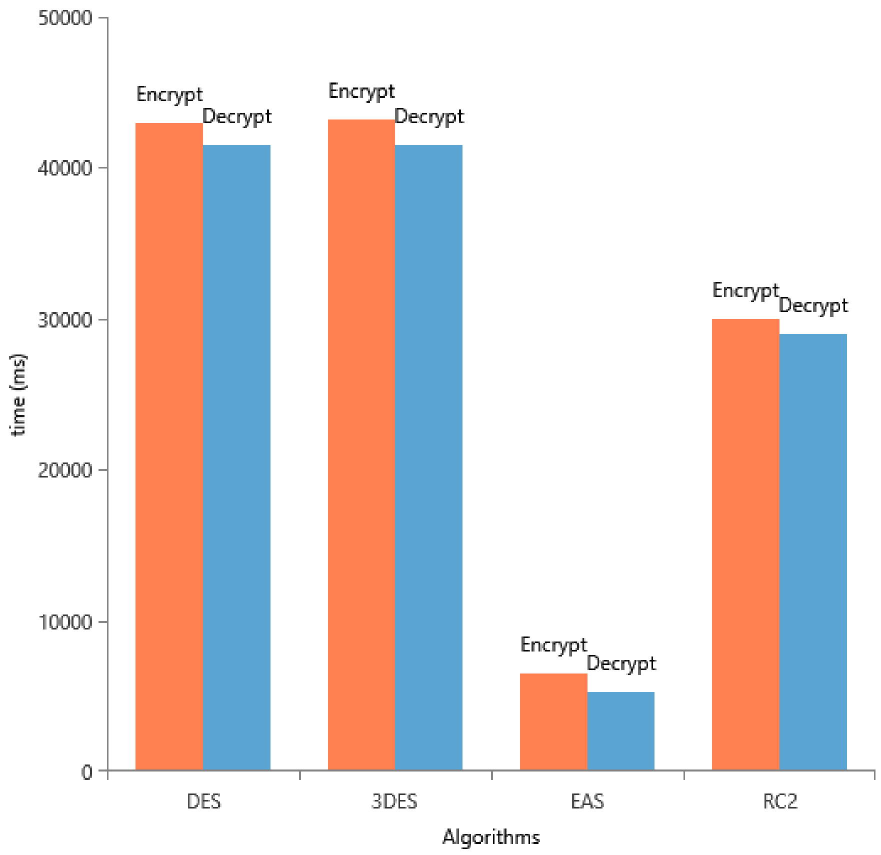


Figure 59. CBC Intel Xeon Sample4 Chart

4.2.3 Benchmark Cipher Mode CFB

The CFB mode even if it has common characteristics of CBC, the algorithms are slower and takes a lot of time to encrypt or decrypt a large data files. In all the samples, it was a very low speed, and the Xeon CPU did not play a significant role in the process.

Sample 1

Benchmark Encryption CFB Sample 1 (Size:3299 Kbyte)			
Type	Time (seconds)	Speed (byte/second)	
DES	1.944	1697K	2M
3DES	1.951	1691K	2M
AES	0.668	4939K	5M
RC2	1.506	2191K	2M

Table 60.CFB Intel Xeon encryption Sample1

Benchmark Decryption CFB Sample 1 (Size:3299 Kbyte)			
Type	Time (seconds)	Speed (byte/second)	
DES	2.226	1482K	1M
3DES	2.018	1635K	2M
AES	0.729	4525K	5M
RC2	1.817	1816K	2M

Table 61. CFB Intel Xeon decryption Sample2

The graph below shows the speed of algorithms of encryption and decryption in CFB mode with a sample data size 3299 Kbyte. DES algorithm is n the same levels with 3DES. The AES is a fast algorithm with a bit difference between encryption/decryption. The RC2 is slow in this cipher mode and decryption is slower than encryption.

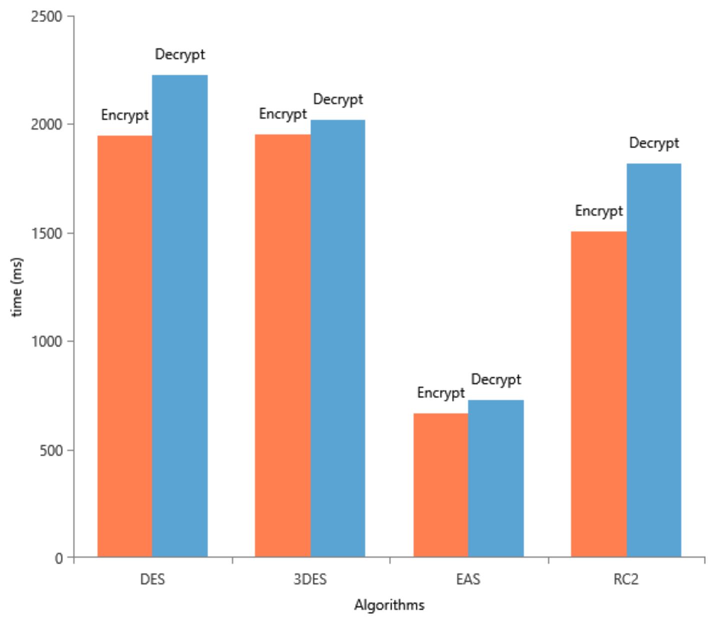


Figure 60. CFB Intel Xeon Sample1 Chart

Sample 2

Benchmark Encryption CFB Sample 2 (Size:2886 Kbyte)			
Type	Time (seconds)	Speed (byte/second)	
DES	1.129	2556K	3M
3DES	1.115	2588K	3M
AES	0.164	17598K	18M
RC2	0.775	3724K	4M

Table 62. CFB Intel Xeon encryption Sample1

Benchmark Decryption CFB Sample 2 (Size:2886 Kbyte)			
Type	Time (seconds)	Speed (byte/second)	
DES	1.079	2675K	3M
3DES	1.099	2626K	3M
AES	0.144	20042K	20M
RC2	0.761	3792K	4M

Table 63. CFB Intel Xeon decryption Sample1

The graph below shows the speed of algorithms of encryption and decryption in CFB mode with a sample data size 2886 Kbyte. DES and 3DES algorithms are almost the same time speed level. The AES is a fast algorithm with a bit difference between encryption/decryption. The RC2 is slow in this cipher mode and decryption/encryption are almost on the same levels.

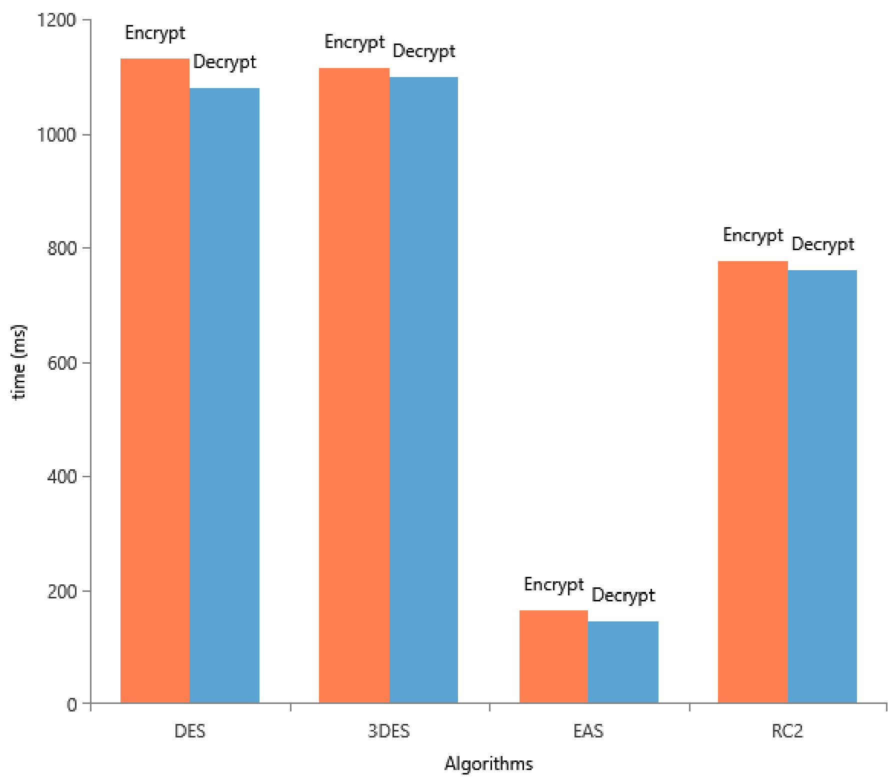


Figure 61. CFB Intel Xeon Sample2 Chart

Sample 3

Benchmark Encryption CFB Sample 3 (Size: 617717 Kbyte)			
Type	Time (seconds)	Speed (byte/second)	
DES	244.641	2525K	3M
3DES	244.010	2532K	3M
AES	36.206	17061K	17M
RC2	167.273	3693K	4M

Table 64. ECB Intel Xeon encryption Sample3

Benchmark Decryption CFB Sample 3 (Size: 617717 Kbyte)			
Type	Time (seconds)	Speed (byte/second)	
DES	241.235	2561K	3M
3DES	238.347	2592K	3M
AES	32.852	18803K	19M
RC2	164.217	3762K	4M

Table 65. ECB Intel Xeon decryption Sample3

The graph below shows the speed of algorithms of encryption and decryption in CFB mode with a sample data size 617717 Kbyte. DES algorithm is at the same levels with 3DES. The AES is a fast algorithm with a bit difference between encryption/decryption. The RC2 is slower than AES and encryption/decryption remain the same.

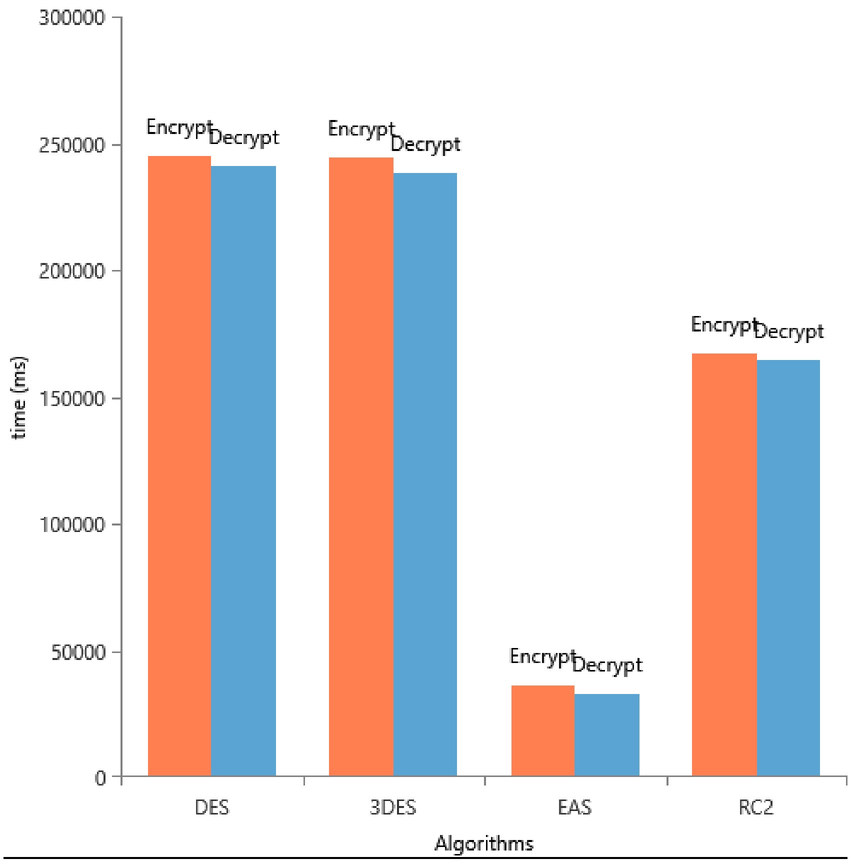


Figure 62. CFB Intel Xeon Sample2 Chart

Sample 4

Benchmark Encryption CFB Sample 4 (Size: 110902 Kbyte)			
Type	Time (seconds)	Speed (byte/second)	
DES	43.005	2579K	3M
3DES	43.136	2571K	3M
AES	6.506	17046K	17M
RC2(64/128)	30.027	3693K	4M

Table 66. CFB Intel Xeon encryption Sample4

Benchmark Decryption CFB Sample 4 (Size: 110902 Kbyte)			
Type	Time (seconds)	Speed (byte/second)	
DES	41.474	2674K	3M
3DES	41.524	2671K	3M
AES	5.204	21311K	21M
RC2(64/128)	28.925	3834K	4M

Table 67. CFB Intel Xeon encryption Sample4

The graph below shows the speed of algorithms of encryption and decryption in CFB mode with a sample data size 110902 Kbyte. Both DES, 3DES algorithms stabilized to the same points. The AES a fast algorithm compare to the others. The RC2 is the second faster algorithm. The CFB mode is a factor n/m times slower than the CBC mode, since only m bits are used per encryption operation.[16, p. 793]

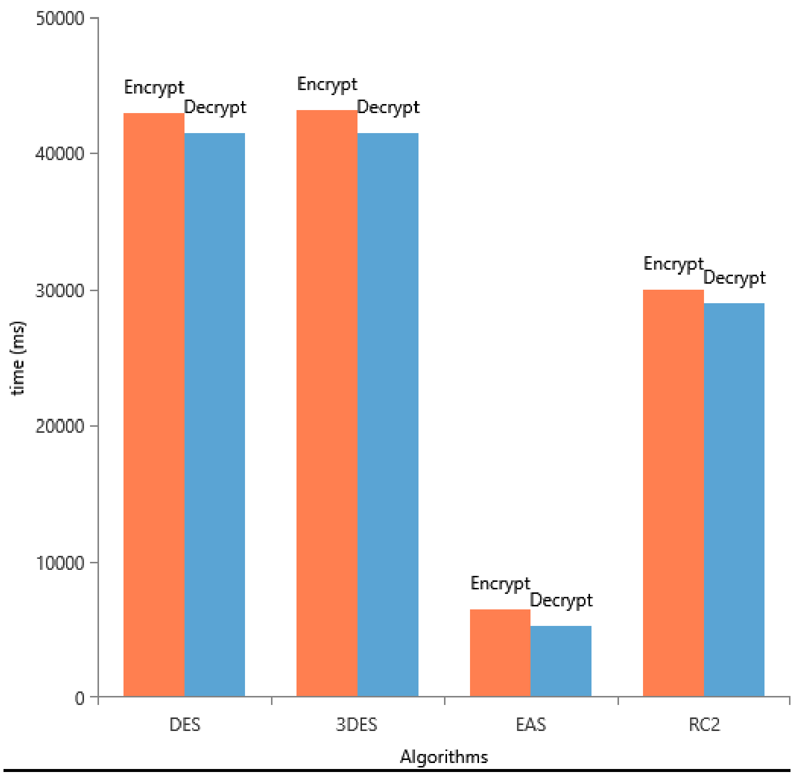


Figure 63. CFB Intel Xeon Sample4 Chart

4.1.4 Intel Xeon Benchmark Overview

A summary of the results CBC mode is the faster because is more simple procedure mode. The AES algorithm is the faster on all the average results. Compare the two processors the Intel Xeon and the Intel Dual-Core the algorithms DES, 3DES, RC2 are two times faster, and AES is almost six times faster. From these results, we can recognize the parallel cores architecture of the specific CPUs. The cipher mode CFB is the most complex one and again the process is slow even if the processor is eight times faster.

Average Algorithms Speed ECB Mode				
Algorithm	Encryption Speed bytes/second		Decryption Speed bytes/second	
DES	197512K	20M	19544K	19M
3DES	19399K	19M	19436K	19M
AES	267184K	267M	143839K	143M
RC2	28088K	28M	36284K	36M

Table 68. ECB Intel Xeon Overview

Average Algorithms Speed CBC Mode				
Algorithm	Encryption Speed bytes/second		Decryption Speed bytes/second	
DES	17918K	18M	16871K	17M
3DES	17176K	17M	16803K	17M
AES	183136	183M	152052K	152M
RC2(64/128)	25597K	26M	32342K	32M

Table 69. CBC Intel Xeon Overview

Average Algorithms Speed CFB Mode				
Algorithm	Encryption Speed bytes/second		Decryption Speed bytes/second	
DES	2340K	2M	2348K	2M
3DES	2346K	2M	2381K	2M
AES	14161K	14M	16170K	16M
RC2	3325K	3M	3301K	3M

Table 70. CFB Intel Xeon Overview

The figures show the progress of each algorithm as the data file increase in the different benchmark scenarios. In the first ECB scenario, the algorithms DES, 3DES, RC2 are stable, and AES as the data increases the process going faster until it reaches the high peak after that it reduce as the time pass. The graph of CBC and CFB is the same with different scales AES when it reaches the high peak keep the speed stable. The rest of the algorithms take specific speed time without any dramatical change. As the time, pass the AES algorithm decreases in ECB mode. For the AES algorithm there is always a peak time at that time the CPU goes to the maximum threads of the program.

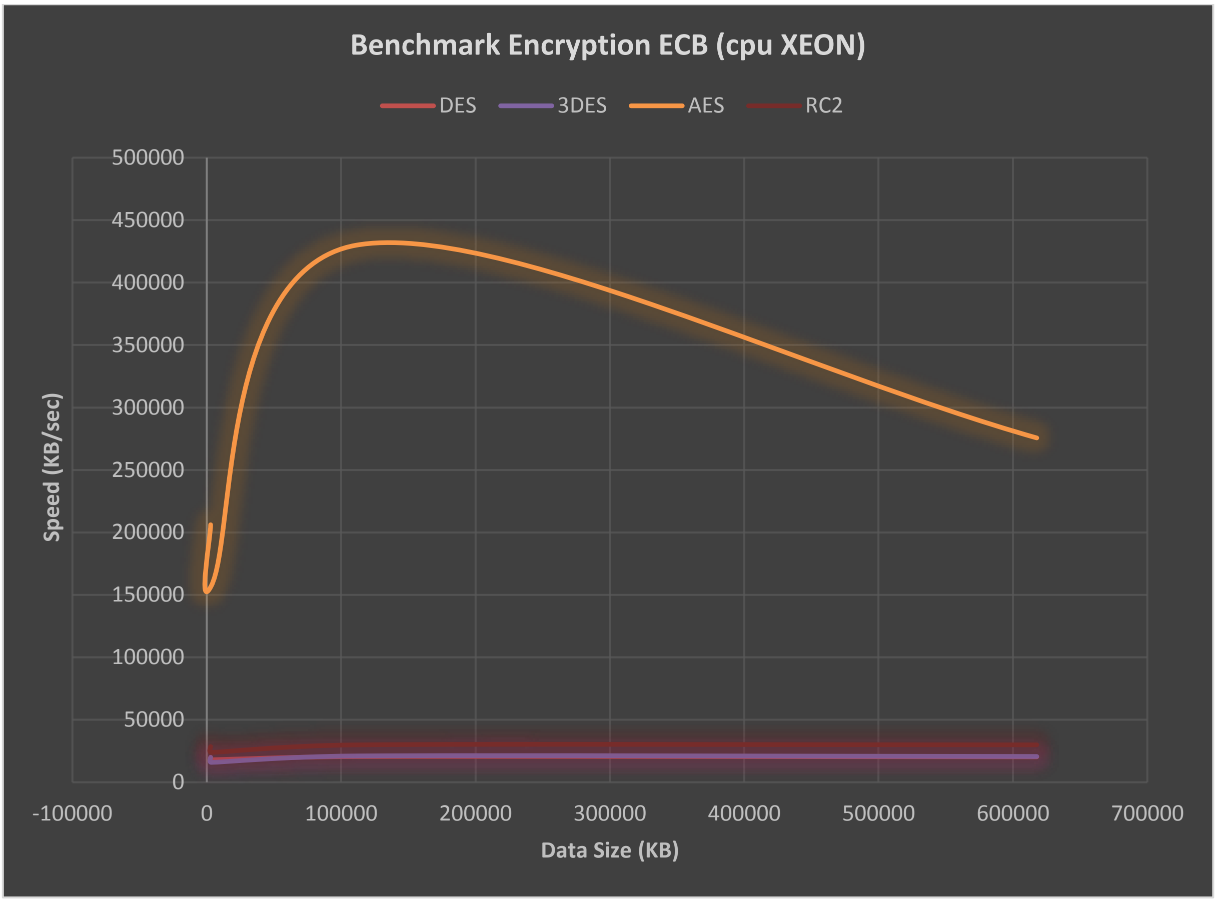


Table 71. Encryption Speed ECB (XEON CPU)

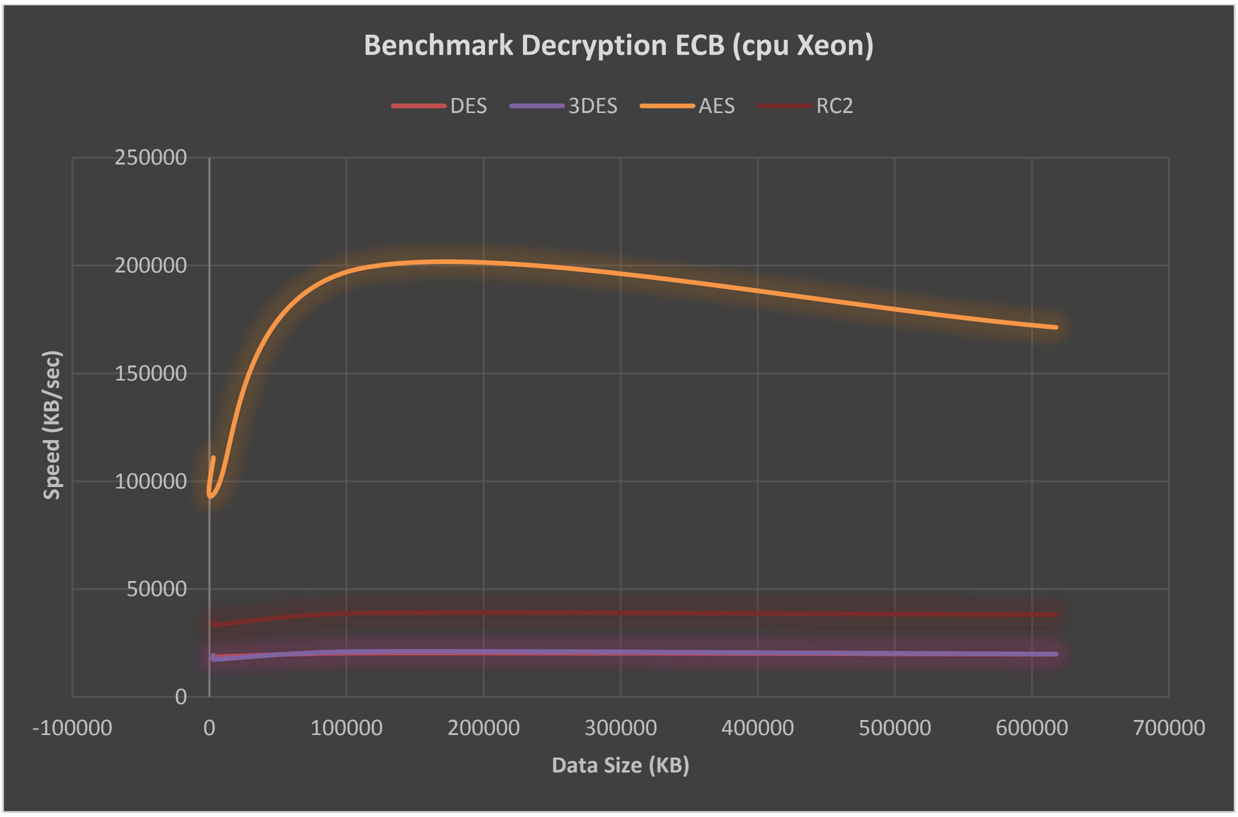


Table 72. Decryption Speed ECB (XEON CPU)

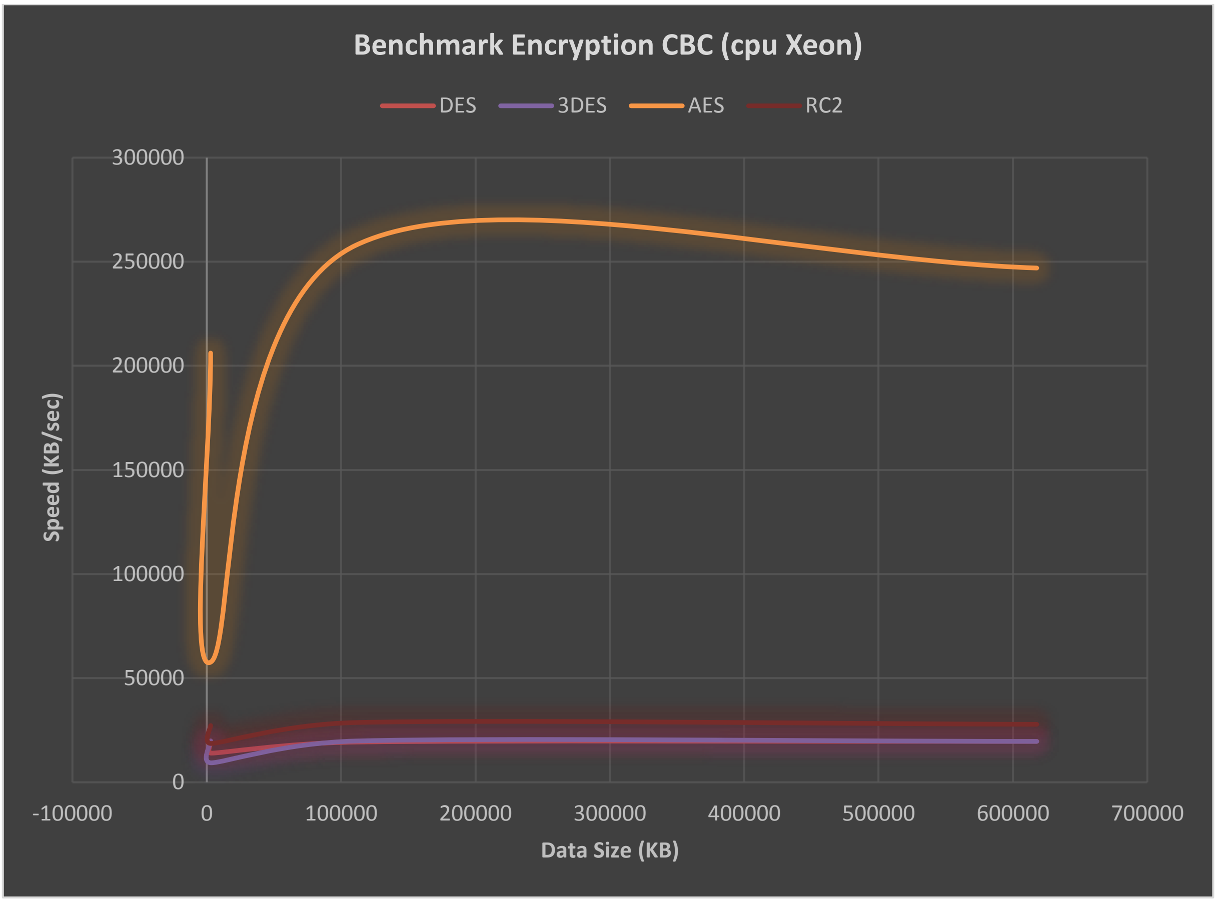


Table 73. Encryption Speed CBC (XEON CPU)

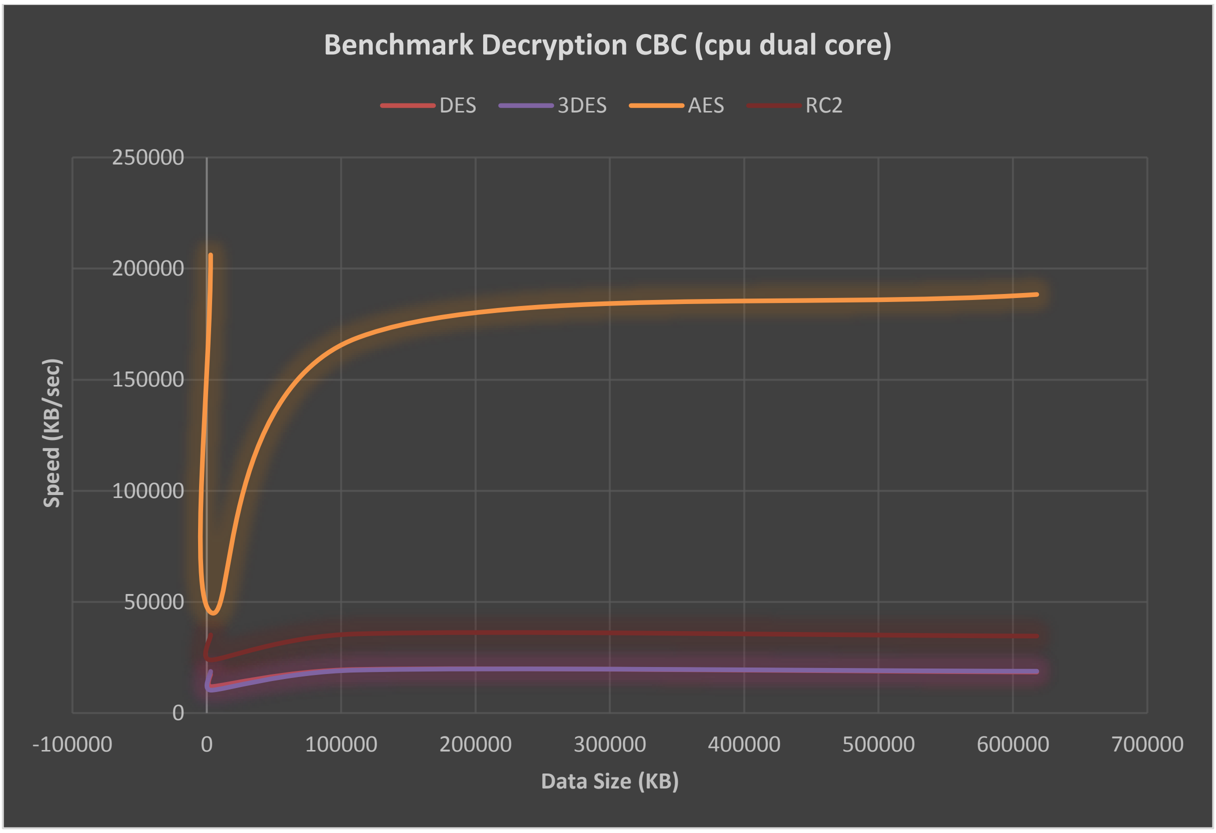


Table 74.. Encryption Speed CBC (XEON CPU)

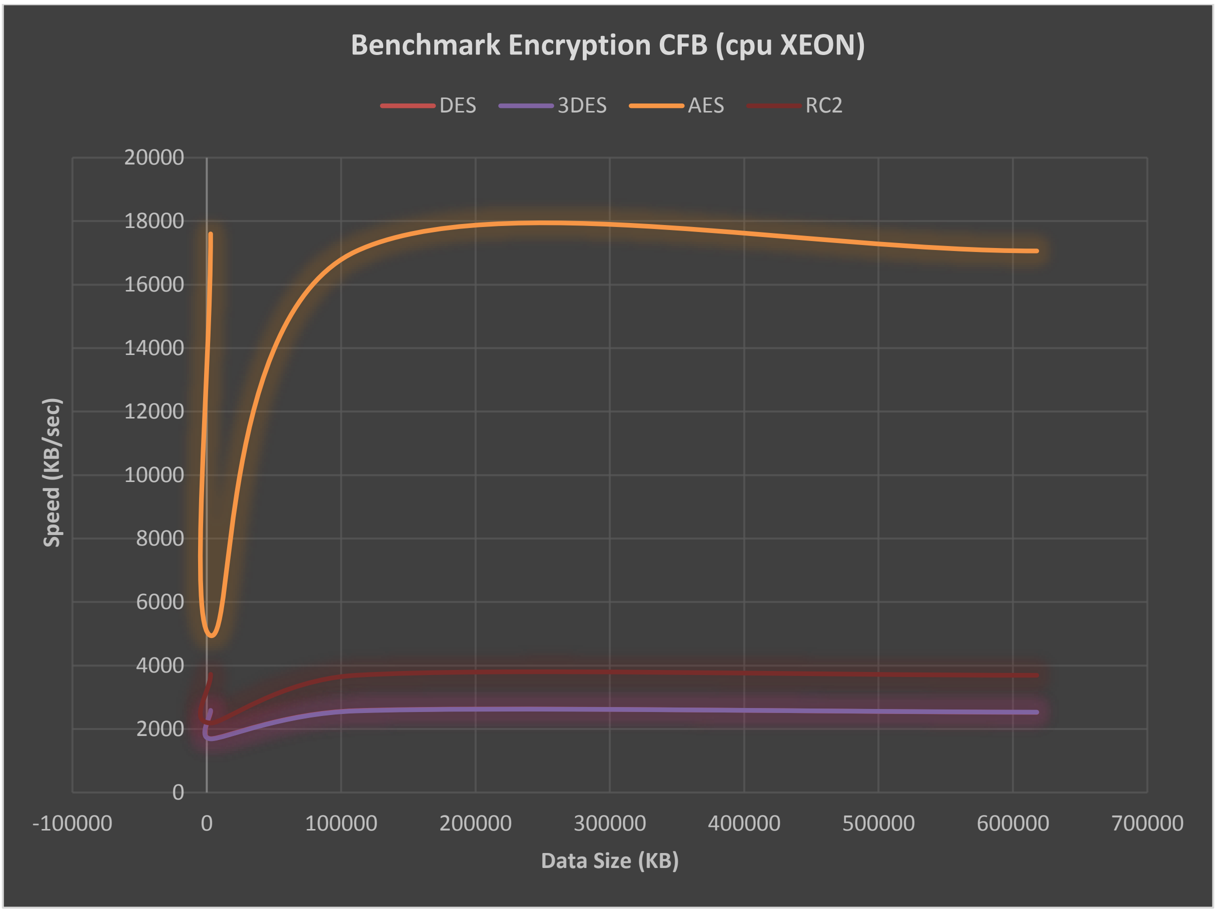


Table 75. Encryption Speed CFB (XEON CPU)

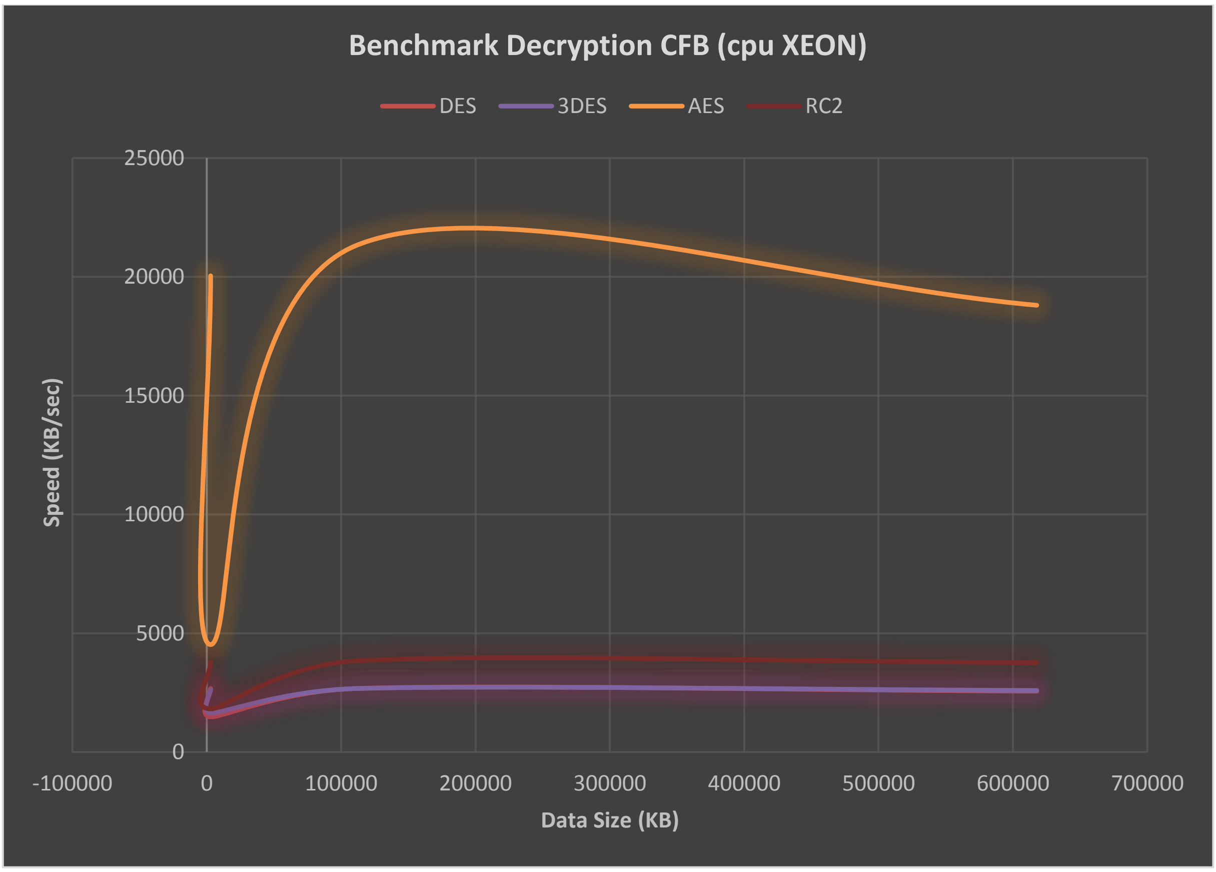


Table 76. Decryption Speed CFB (XEON CPU)

4.3 Compare Results

Block ciphers like DES, 3DES, AES and RC2, ideal decryption is equal to encryption and therefore takes the same time. The minor difference is encryption may require generating a unique Initialization Vector (IV) and might take a small amount of extra time.

The encryption and decryption on block cipher in ECB mode are the same but in a different direction. From the overview results on both hardware, the decryption is slower than encryption but is possible the different point on the memory address. The system hardware environment plays a significant role in the results. The results cannot be the same because the speed of the hard disk and the address of the random access memory is not always the same.

The CBC mode decryption is using the block cipher in inverse mode. Block ciphers that are using an opposite direction may be slower than the forward direction due to asymmetries in the key schedule. Therefore, systems like Xeon and Dual Core that support parallel threads, CBC decryption is faster than encryption, but on systems that do not support it may be slightly slower.

The block ciphers on CFB mode, the encryption of each block depends on the previous one, but the decryption is parallel. Therefore, on a big data files with a multi-core implementation like the eight core Xeon, the CFB decryption is typically faster than encryption also the memory address changes each time and never will be the same in both cases.

3DES is a three-key method, which the ciphertext is encrypted three times in a sequence. The ciphertext is encrypted with the first key, then that text is decrypted using the second key, and finally, the third key encrypts the last ciphertext. From the experiments, the 3DES is almost the same with the DES because of the parallel blocks of the dot NET software and hardware multicores. The below figures show a test experiment of a small 2KB file (ECB Mode) that can't implement a parallel process, the results are close to the background theory of DES algorithms.

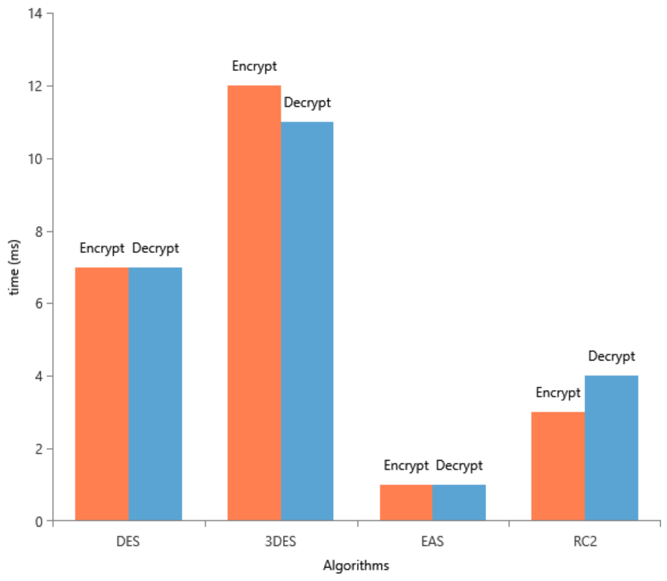
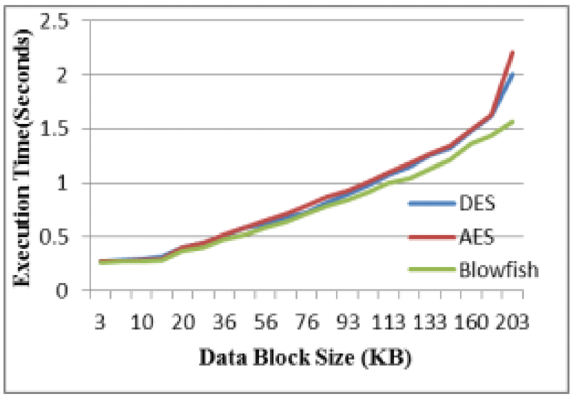


Figure 64. Dual Core ECB 2KB File

According Jawahar Thakur and Nagesh Kuma[17] provide a comparison performance of DES, AES implemented through a Java environment with a JCE framework. The experiments made using AMD Sempron processor with 2GB RAM. [17]



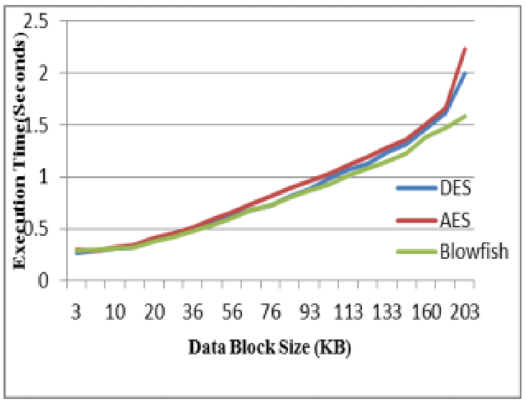


Figure 65. ECB Results[17]

Figure 66. CBC Results[17]

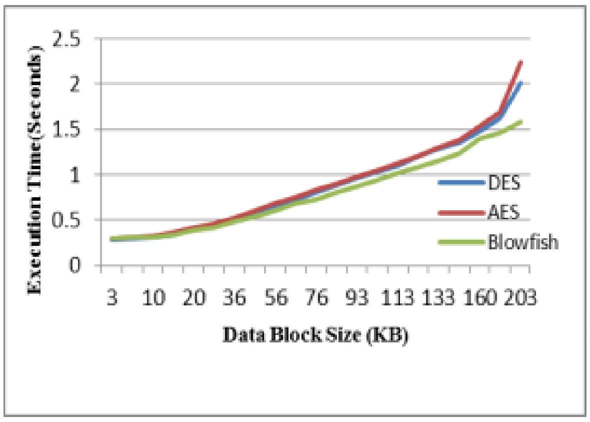


Figure 67. CFB Results[17]

The conclusion is using ECB mode shows that AES consumes more resources when the data block size is relatively big[17, p. 5]. From this dissertation, the results show that big data files AES is faster, and DES algorithm is not even close to that speed. According to Jawahar Thakur and Nagesh Kuma[17], the CBC require more processing time than ECB because of its key-chaining nature. The results indicate that the extra time added is not significant for many applications, knowing that CBC is much better than ECB regarding protection. Compare to that the average speed of the algorithms from Dual Core CPU tables 43, 44 and Intel Xeon Tables 68, 69 show that CBC is faster that CBC and AES algorithm have a huge difference on the speed. The CFB results are almost the same as the rest cipher modes but with an overview in this experiment is a very slow block cipher and takes a lot of process time to complete a data file even data around 1~2 Mbytes.

Chapter 5
Conclusion

This dissertation presents the performance evaluation of selected symmetric algorithms. The selected algorithms AES, DES, 3DES, and RC2, examine the process of the encryption and decryption time. A particular benchmark program record the performance of the algorithms for different files and systems. How fast can a system process a Symmetric algorithm? Is the parallelizable encryption with Electronic Codebook (ECB) faster than Cipher Block Chaining (CBC) that is not parallelizable? There is a need to examine how an information system can manage a process of encrypted data.

For a secure network or application, a strong algorithm provides a safe system. A system that is safe and slow is not so reliable to the users because it takes time to get the information. Encryption must be fast and secure to satisfy all the requirements. The simulation results show that an expensive hardware is not always the key to a fast encryption or decryption process. The Intel Xeon processor is an expensive x86 architecture that targets workstations and servers but compares to an Intel Dual core processor the expectations are high. The results of Cipher Feedback (CFB) show that is a complex cipher block and the speed of both experiments is very slow. If the cost matters the usage of a fast cipher mode like CBC compare to CFB, it would be a wise choice.

A safe and fast access to a data information is the key asset for a security system. The most critical element in a service is the high level of the availability. A system that provides control services for applications and devices the interruption of services is considered a significant disadvantage. When a service is working with difficulties results in financial loss, low productivity, and customer downtime. In today's times, computers have evolved into multiple and parallel processors. This technology makes it possible to process parallel commands together with the appropriate encryption software design. Some standard algorithms can take advantage of this fact and make encryption faster.

The technology and information security are demanding, a fast process of the data is an important feature. On different kind of sizes and processors, the algorithms correspond differently. On this area, Network Security requires more research, cryptographic algorithms have to manage massive files of data on various kind scenarios.

Bibliography

[1] N. Kumar and B. V. Gopal, “VLSI Implementation of Data Encryption Standard Algorithm,” no. 6, pp. 106–110, 2012.

[2] A. G. Konheim, *Computer security and cryptography*. Wiley-Interscience, 2007.

[3] E. B. William C. Barker, “Data Encryption Algorithm,” *Recomm. Triple Data Encryption Algorithm Block Cipher*, no. January, 2012.

[4] “National Security Agency Releases History of Cold War Intelligence Activities.” [Online]. Available: http://nsarchive.gwu.edu/NSAEBB/NSAEBB260/. [Accessed: 12-Mar-2017].

[5] B. Schneier, *Applied cryptography: Protocols, algorithm, and source code in C*, vol. 13, no. 3. 1996.

[6] U.S. DEPARTMENT OF COMMERCE/National Institute of Standards and Technology, “Data Encryption Standard (DES),” *Fips Pub 46-3*, vol. 3, 1999.

[7] W. Stallings, *Cryptography and Network Security*, vol. 139, no. 3. Pearson, 2011.

[8] J. Daemen, L. Knudsen, and V. Rijmen, “The Block Cipher SQUARE,” *Lect. Notes Comput. Sci.*, vol. 1267, pp. 149–165, 1997.

[9] T. St Denis and S. Johnson, “Advanced Encryption Standard,” in *Cryptography for Developers*, Elsevier, 2007, pp. 139–202.

[10] Edward Roback and and Morris Dworkin, “First advanced encryption standard (AES) candidate conference Ventura, CA August 20-22, 1998,” vol. 104, no. 1.

[11] F. Information and P. S. P. 197, “Announcing the ADVANCED ENCRYPTION STANDARD (AES),” 2001.

[12] L. R. Knudsen, V. Rijmen, R. L. Rivest, and M. J. B. Robshaw, “On the Design and Security of RC2,” Springer, Berlin, Heidelberg, 1998, pp. 206–221.

[13] R. Rivest, “A Description of the RC2(r) Encryption Algorithm,” *MIT Lab. Comput. Sci.*

RSA Data Secur. Inc., pp. 1–12, 1998.

[14] “Parallel Programming in the .NET Framework.” [Online]. Available: https://msdn.microsoft.com/en-us/library/dd460693(v=vs.110).aspx. [Accessed: 19-May-2017].

[15] S. Hauck and A. DeHon, *Reconfigurable computing : the theory and practice of FPGA-based computation*. Morgan Kaufmann, 2008.

[16] S. J. Henk C.A. van Tilborg, *Encyclopedia of Cryptography and Security - Google Books*. .

[17] J. Thakur and N. Kumar, “DES, AES and Blowfish: Symmetric Key Cryptography Algorithms Simulation Based Performance Analysis,” *Int. J. Emerg. Technol. Adv. Eng. Website www.ijetae.com*, vol. 1, no. 2, 2250.