

PROFESSIONAL TRAINING REPORT
at
Sathyabama Institute of Science and Technology
(Deemed to be University)

Submitted in partial fulfillment of the requirements for the award of
Bachelor of Technology in Information Technology

By

39120132
REG. NO. 39120132



DEPARTMENT OF INFORMATION TECHNOLOGY

SCHOOL OF COMPUTING

SATHYABAMA INSTITUTE OF SCIENCE AND TECHNOLOGY
JEPPIAAR NAGAR, RAJIV GANDHI SALAI,
CHENNAI – 600119, TAMILNADU

NOVEMBER 2021



SATHYABAMA
INSTITUTE OF SCIENCE AND TECHNOLOGY
(DEEMED TO BE UNIVERSITY)
Accredited with Grade "A" by NAAC
(Established under Section 3 of UGC Act, 1956)
JEPPIAAR NAGAR, RAJIV GANDHI SALAI
CHENNAI- 600119
www.sathyabama.ac.in



DEPARTMENT OF INFORMATION TECHNOLOGY

BONAFIDE CERTIFICATE

This is to certify that this Project Report is the Bonafide work of VASANTH P (Reg. No: 39120132) who carried out the project entitled "**IMAGE AND TEXT ENCRYPTION AND DECRYPTION USING AES ALGORITHM**" under my supervision from June 2021 to November 2021.

Internal Guide

Name of the Guide

Head of the Department

Submitted for Viva voce Examination held on _____

Internal Examiner

External Examiner

DECLARATION

I, **VASANTH P** hereby declare that the project report entitled **IMAGE AND TEXT ENCRYPTION AND DECRYPTION USING AES ALGORITHM** done by me under the guidance of **Dr.V.NIRMAL RANI,M.E., Ph.D.**, is submitted in partial fulfillment of the requirements for the award of Bachelor of Engineering Degree in Computer Science and Engineering.

DATE:

PLACE:

SIGNATURE OF THE CANDIDATE

ACKNOWLEDGEMENT

I am pleased to acknowledge my sincere thanks to **Board of Management** of **SATHYABAMA** for their kind encouragement in doing this project and for completing it successfully. I am grateful to them.

I convey my thanks to **Dr. T. Sasikala M.E., Ph.D**, Dean, School of Computing, **Dr. R. Subhashini, M.E., Ph.D. Head of the Department of Information Technology** for providing me necessary support and details at the right time during the progressive reviews.

I would like to express my sincere and deep sense of gratitude to my Project Guide **Dr. V.NIRMAL RANI M.E., Ph.D.**, for her valuable guidance, suggestions and constant encouragement paved way for the successful completion of my project work.

I wish to express my thanks to all Teaching and Non-teaching staff members of the **Department of Information Technology** who were helpful in many ways for the completion of the project.

TRAINING CERTIFICATE



ABSTRACT

In this modern digital world thanks to the internet technology, so much of data is transmitted. the number of unauthorized access to data increases. In the world of black hat hackers, due to modern apps so much of most important data is transmitted in form of images, text etc. data security is very major issue, there are many ways to prevent theft, one of the best way is encryption of the important data. There are many algorithms to solve this problem. One of the efficient method is to use AES (Advanced Encryption Standard) algorithm, the most notable and extensively used cryptographic algorithm because it is six times faster than 3-DES and much faster than RSA algorithm. With the development of information technology, image information has become the main content of network information transmission. With the development of image encryption technology, it is also about the development of image information theft technology. In order to cope with the evolving information theft technology, we must seek a better image encryption algorithm. Among many algorithms, due to the superiority of chaos technology, when the image is encrypted with chaos technology, the ciphertext presents a randomness, which makes the possibility of deciphering greatly reduced. Therefore, the research of digital image encryption algorithm based on chaos technology has become an important means of modern digital image encryption.

In this project i proposed an Text and image encryption and decryption algorithm using AES in which encryption contains a random image and decryption contains original image. The algorithm is implemented in java. The efficiency of AES using image and text and it is analyzed. The result thus shows that the sharing of information through image is much more reliable.

TABLE OF CONTENTS

CHAPTER No.	TITLE	PAGE No
	ABSTRACT	i
	LIST OF FIGURES	ii
	LIST OF TABLES	iii
	LIST OF ABBREVIATIONS	iv
1	INTRODUCTION	11
	1.1 AIM	11
	1.2 OBJECTIVE	11
	1.3 OVERVIEW	11
	1.4 AES ALGORITHM	13
2.	LITERATURE SURVEY	15
3.	METHODOLOGY	16
	3.1 ADVANCED ENCRYPTION STANDARD	16
	3.2 AES ENCRYPTION	16
	3.3 AES DECRYPTION	21
	3.4 IMPLEMENTATION	24
4.	RESULTS AND ANALYSIS	27
	4.1 RESULT	27
	4.2 ANALYSIS	28
	4.3 INFERENCE	33
5	CONCLUSION	33
	REFERENCES	34

APPENDIX	35
A.SCREENSHOTS	35
B. SOURCE CODE	37

LIST OF FIGURES

FIGURE NO.	FIGURE NAME	PAGE NO
1	PROCESS OF AES ENCRYPTION AND DESCRIPTIOM	19
2	MIX COLUMN- PROCESS	21
3	INVERSE SHIFROWS- PROCESS	23
4	AES- ENCRYPTION PROCESS	25
5	AES- DESCRIPTION PROCESS	26
6	FINAL OUTPUT-1	27
7	FINAL OUTPUT 2	29
8	COMPARISON TEXT AND IMAGE	32
9	SCREENSHOTS	35

LIST OF TABLES

TABLE NO.	TABLE NAME	PAGE NO.
1	AES PARAMETERS	18
2	S- Box	20
3	INVERSE S- BOX	22
4	TEXT TIME COMPLEXITY	28
5	TEXT TIME COMPLEXITY	30
6	TEXT AND IMAGE- TIME COMPLEXITY	32

INTRODUCTION

1.1 AIM

THE AIM OF THE PROJECT IS IMAGE AND TEXT ENCRYPTION AND SECRIPTION USING AES ALGORITHM.

1.2 OBJECTIVE

The objective criteria is that to get required data, to be encrypted in form text or images. then secret private key is given to algorithm. Then based on the key the text or image is encrypted. To decrypt the encrypted image, encrypted data (text) is given to the algorithm, Then the correct secret key is given, algorithm verify the secret key, then encrypted data, is decrypted to original form.

1.3 OVERVIEW

With the ever-increasing growth of multimedia applications, Important issue for communication and storage of images is security, and encryption is one the technique to ensure security. encryption techniques convert original image to another image that is hard to understand; to keep the image confidential between users, in other word, it is essential that nobody could get to know the confidential message without a key for decryption. Their different encryption techniques are used to protect the confidential message from unauthorized user. Encryption is a very common technique for promoting the information security. The evolution of encryption is moving towards a future of endless possibilities.

Encryption is the process of transforming the original data which called plaintext in to encrypted data called ciphertext. Different techniques are used to fulfill the data own features of each data type. Many encryption algorithms used to protect and ciphered text data such as classical cipher system. Digital images used in many communication applications, therefore the protection the content of these images become very important. Image encryption is a technique which coding the original image (plain image) to another un-understanding image (cipher image). This technique must be providing the decoding the cipher image to plain image without losing data or image properties. Divers set of applications ubiquitous depending on digital image encryption and used diver's algorithms to protect the content and information of original images from unauthorized users. There are different types of encryption algorithms according to plaintext message; some used for text data and not be

suitable for other multimedia data such as digital image. Others types used for images and not suitable with text data. Due to image powerful attribute such as vast data capacity, the great redundancy and great correlation among pixels of image. These algorithms can't used specifically actualized to image on the grounds that image size is practically. These algorithms need much time when used with image. The other problem is that the decrypted text must be equal to the original text. However, this requirement is not necessary for image data. Due to the characteristic of human perception, a decrypted image containing small distortion is usually acceptable. Textual encryption algorithms presented in this paper. These algorithms, such as Vigenere and Hill cipher systems; used to encrypted and decrypted color digital images after some enhancement to become suitable with digital image. The proposed system provide security so nobody can see or get the original image just approved individual and integrity is guarantee that the image has not altered and secrecy is guaranteed.

EVERY cryptographic process has two aspects: the algorithm and the key used for the encryption and decryption. The use of the keys makes the cryptographic process reliable. There are two types of cryptographic mechanisms: Symmetric key cryptography - uses same key for encryption and decryption process. Asymmetric key cryptography - uses two different keys for encryption and decryption process. Symmetric key algorithm is efficient, fast and easy to implement as compared to asymmetric key algorithm. The Advanced Encryption Standard (AES) was published by the National Institute of Standards and Technology (NIST) in 2001. AES is a subset of the Rijndael block cipher developed by two Belgian cryptographers, Vincent Rijmen and Joan Daemen. AES is a symmetric block cipher and it is believed to be the standard process, So, it replaces DES algorithm in most number of applications[1]. When compared to other symmetric and public key ciphers, the AES symmetric cipher is found to be complex.

1.4 AES ALGORITHM

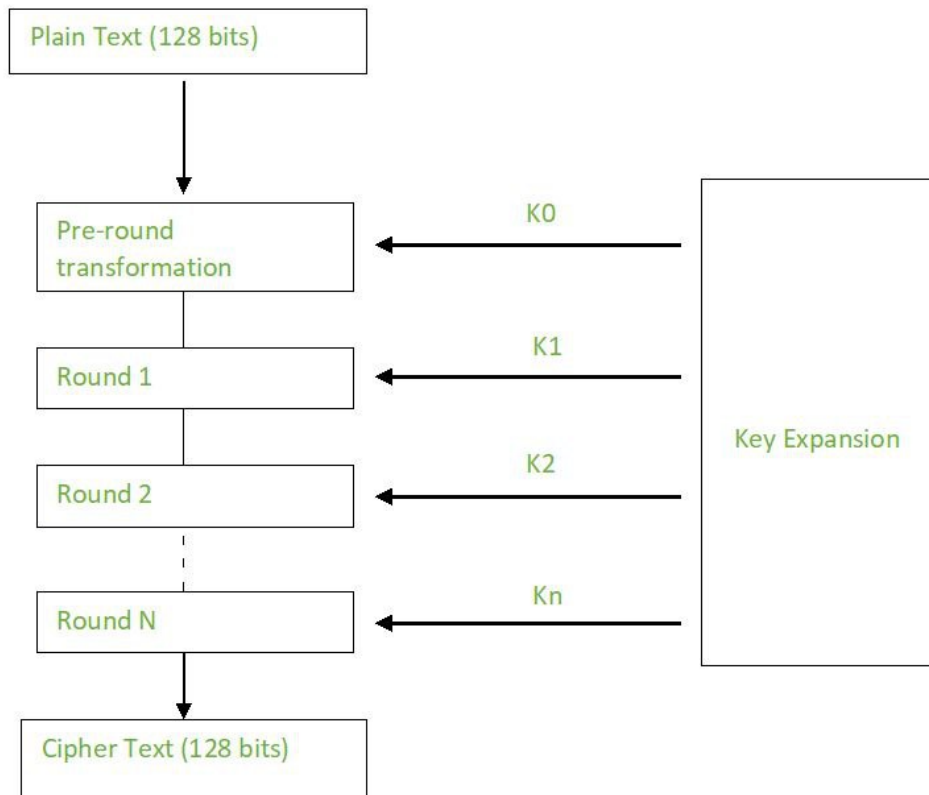
AES is short for Advanced Encryption Standard and is a United States encryption standard defined in Federal Information Processing Standard (FIPS) 192. AES is the most recent of the four current algorithms approved for federal use in the United States. AES is a symmetric encryption algorithm processing data in block of 128 bits. AES is symmetric since the same key is used for encryption and the reverse transformation, decryption. The only secret necessary to keep for security is the key. AES may be configured to use different key-lengths, the standard defines 3 lengths and the resulting algorithms are named AES-128, AES-192 and AES-256 respectively to indicate the length in bits of the key. The older standard, DES or Data Encryption Standard. DES is up to 56 bits only [4]. To overcome the disadvantages of DES algorithm, the new standard is AES algorithm. This standard explicitly defines the allowed values for the key length (N_k), block size (N_b), and number of rounds (N_r). For the AES algorithm, the length of the input block, the output block and the State is 128 bits. This is represented by $N_b = 4$, which reflects the number of 32-bit words (number of columns) in the State.

An implementation of the AES algorithm shall support at least one of the three key lengths: 128, 192, or 256 bits (i.e., $N_k = 4, 6, \text{ or } 8$, respectively). Implementations may optionally support two or three key lengths, which may promote the interoperability of algorithm implementations. For the AES algorithm, the length of the Cipher Key, K , is 128, 192 or 256 bits. The key length is represented by $N_k = 4, 6, \text{ or } 8$ which reflects the number of 32-bit words (number of columns) in the Cipher Key. For the AES algorithm, the number of rounds to be performed during the execution of the algorithm is dependent on the key size. The number of rounds is represented by N_r , where $N_r = 10$ when $N_k = 4$, $N_r = 12$ when $N_k = 6$, and $N_r = 14$ when $N_k = 8$. An implementation of the AES algorithm shall support at least one of the three key lengths: 128, 192, or 256 bits (i.e., $N_k = 4, 6, \text{ or } 8$, respectively). Implementations may optionally support two or three key lengths, which may promote the interoperability of algorithm implementations. For the AES algorithm, the length of the Cipher Key, K , is 128, 192 or 256 bits. The key length is represented by $N_k = 4, 6, \text{ or } 8$ which reflects the number of 32-bit words (number of columns) in the Cipher Key. For the AES algorithm, the number of rounds to be performed during the execution of the algorithm is dependent on the key size. The number of rounds

is represented by N_r , where $N_r = 10$ when $N_k = 4$, $N_r = 12$ when

Bit pattern	Key Length (N_K Words)	Block Size (N_B Words)	No Of Round s (N_R Words)
AES-128	4	4	10
AES-192	6	4	12
AES-256	8	4	14

For both its Cipher and Inverse Cipher, the AES algorithm uses a round function that is composed of four different byte-oriented transformations:
Byte substitution using a substitution table (S-box),
Shifting rows of the State array by different offsets,
Mixing the data within each column of the State array, and
Adding a Round Key to the State.



2. LITERATURE SURVEY

J. Daemen and V. Rijmen, The block cipher Rijndael, Smart Card research and Applications, LNCS 1820, Springer-Verlag, pp. 288-296.

J. Nechvatal, et. al., Report on the Development of the Advanced Encryption Standard (AES), National Institute of Standards and Technology, October 2, 2000.

—Specification for the Advanced Encryption Standard (AES), Federal Information Processing Standards Publication 197, Nov. 2001

A. Menezes, P. van Oorschot, and S. Vanstone, Handbook of Applied Cryptography, CRC Press, New York, 1997, p. 81-83.

C.-P. Su, T.-F. Lin, C.-T. Huang, and C.-W. Wu, —A high-throughput low-cost AES processor, IEEE Commun. Mag., vol. 41, no. 12, pp.86–91, Dec. 2003.

C.-P. Su, C.-L. Horng, C.-T. Huang, and C.-W. Wu, —A configurable AES processor for enhanced security,II in Proc. ASP-DAC, Shanghai, China, Jan. 2005, pp. 361–366.

Rachh, R.R.; Anami, B.S.; Ananda Mohan, P.V. —Efficient implementations of S-box and inverse S-box for AES algorithm,II in TENCON 2009 - 2009 IEEE Region 10 Conference , Nov. 2009, pp. 1–6.

Kaur, Swinder; Vig, Renu , II Efficient Implementation of AES Algorithm in FPGA Devicell in Conference on Computational Intelligence and Multimedia Applications, Nov 2007,pp. 179-187

2.METHODOLOGY

2.1 ADVANCED ENCRYPTION STANDARDS

2.1.1 AES PROCESS

The characteristics of AES algorithm are Uses common (Symmetric) key and so, it is called as Symmetric block cipher, The size of data is 128 bits, and so the key size varies from 128/192/256 bits.

It is highly robust and 6 times faster than 3-DES and much faster and powerful than RSA.

It uses single S-box for all rounds rather than DES, where it uses eight S-boxes.

The process involved in AES is iterative when compared with Fiestel cipher. AES is generally based on Substitution and Permutations operations. It

consists of sequence of linked operations at each round. Each round consists of the following process. 1) Interchanging of inputs by the outputs produced by the substitution process. 2) The bits are shifted in a cyclic manner. 3) The columns are mixed by transformation. 4) The key has been added with input using XOR operation. The computation of AES is performed at bytes level, so 128 bits of plain-text is considered as a block of 16 bytes which constitute a 4x4 matrix. Each round of AES depends on the key length. AES performs 10/12/14 rounds depending on the key size 128/192/256 respectively. So, AES is known to be AES-128, AES-192 and AES-256 algorithm based on key size. A unique 128 bit round key is given as an input at each round.

AES – Encryption and Decryption Specification-

AES is called AES-128, AES-192 and AES-256. This classification depends on the different key size used for cryptographic process. Those different key sizes are used to increase the security level. As, the key size increases the security level increases. Hence, key size is directly proportional to the security level. The input for AES process is a single block of 128 bits. The processing is carried out in several number of rounds where it depends on the key length: 16 byte key consists of 10 rounds, 24 byte key consists of 12 rounds, and 32 byte key consists of 14 rounds. The first round of encryption process consists of four distinct transformation functions:

- 1.Substitution Bytes
- 2.ShiftRows
- 3.MixColumns
- 4.AddRoundKey

The final round consists of only three transformation ignoring MixColumns. The Decryption method is the reverse of encryption and it consists of four transformations.

- 1.Inverse Substitution Bytes
- 2.Inverse ShiftRows
- 3.Inverse MixColumns
- 4.AddRoundKey

Table I. AES-parameters

Key size (words/bytes/bits)	4/16/128	6/24/192	8/32/256
Plaintext block size (words/bytes/bits)	4/16/128	4/16/128	4/16/128
Number of rounds	10	12	14
Round key size (words/bytes/bits)	4/16/128	4/16/128	4/16/128
Expanded key size (words/bytes)	44/176	52/208	60/240

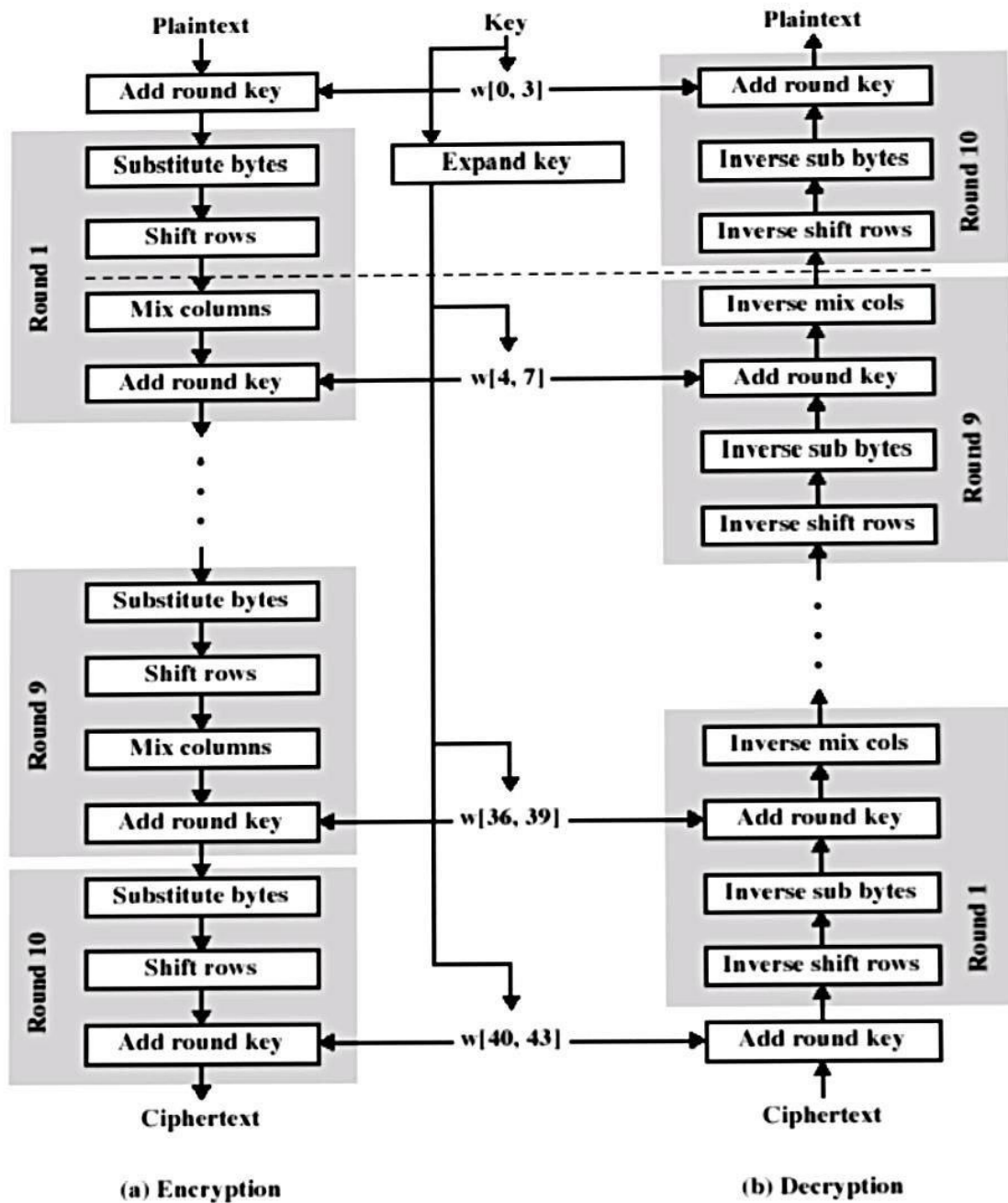


Fig 1. Process of AES Encryption and Decryption

2.2 AES ENCRYPTION

2.2.1 SUBSTITUTION BYTES :

The 16 byte plain-text substitutes the corresponding value from substitution table S-box. It is a non-linear method which performs in the following way:

Table II. S-Box

	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	6E	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

2.2.2 MIX COLUMNS:

MixColumns transformation performs by transforming each column of four bytes. It takes input as one column which is of 4 bytes and output as completely different 4 bytes by transforming the original column. The resultant matrix is same as the size of plain-text. MixColumn transformation will not be carried in the last round.

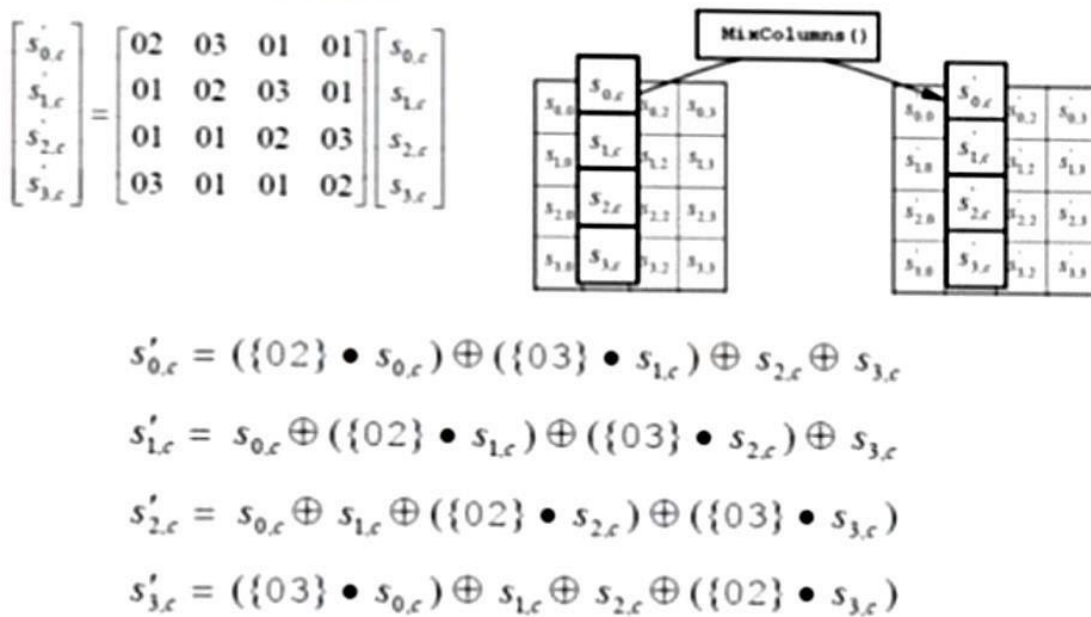


Fig 4. MixColumns-Process

2.2.3 ADD ROUND KEY

The 16 bytes which is produced from MixColumns is equal to 128 bits which is XORed with the round key of 128 bits. The above process has been repeated until final round to produce the corresponding cipher text.

2.3 AES DECRYPTION

Inverse Substitution Bytes:

Inverse Substitution Bytes is the inverse of the substitution byte transformation. This is performed through inverse S-box [6,7]. This is obtained by applying inverse of substitution bytes and by computing multiplicative inverse of Galois Field - GF (2^8).

Table III. Inverse S-Box

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

Inverse ShiftRows:

Inverse ShiftRows is the inverse of ShiftRows transformation. It carries out circular shifts in reverse direction for each last 3 rows and for the 2nd row, it performs one-byte circular shift to the right and it continues the process till (n-3)rd row.

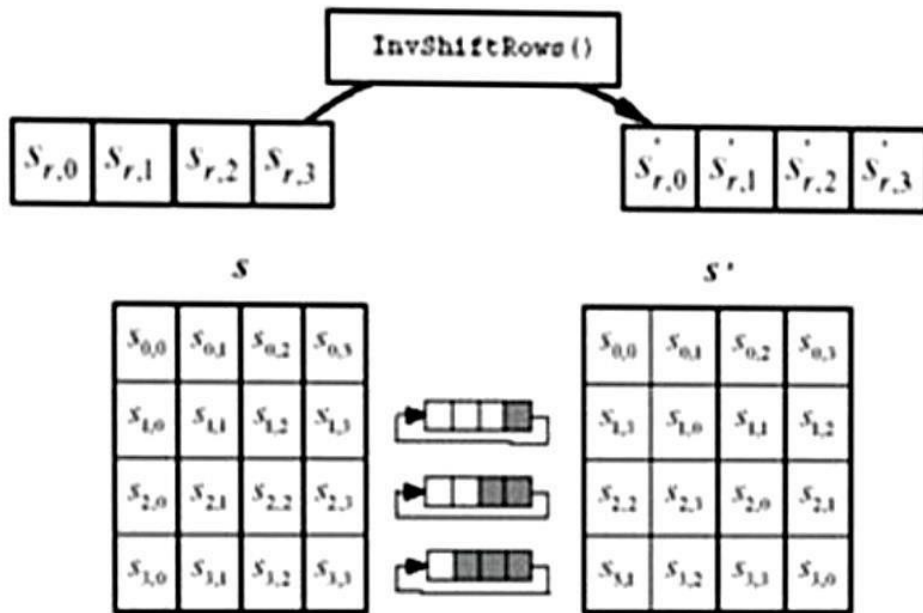


Fig 6. Inverse ShiftRows-Process

Inverse MixColumns:

Inverse MixColumns is the inverse of Mixcolumns transformation. It carries out operations on a matrix by column-wise. Resultant columns are in the form of polynomials

2.4 IMPLEMENTATION

2.4.1 AES-ENCRYPTION ALGORITHM

The implementation of AES-encryption is done in java. The input given to the encryption process is a Plain-text (or) a Plain-image. The image is divided into 4*4 matrix and so there are 10 rounds performed for this process. So, there are totally nine rounds performed with 4 transformations (i.e.) subbytes, shiftrows, mixcolumns and addroundkey. And 10th round consists of three transformations ignoring Mixcolumns [9,10]. The original image will be encrypted and the encrypted image will be displayed as another random image to the sender. This encryption process also applied on text and the encryption process is same and the result is obtained

2.4.2 AES DECRYPTION ALGORITHM

The AES decryption is an inverse of encryption process. The below figure shows the decryption process. The decryption process takes the cipher-image (or) cipher-text as input and it performs the four transformations (i.e.) Inverse subbytes, Inverse Shiftrows, Inverse Mixcolumns and Addroundkey until 9 rounds and the 10th round performs three transformation ignoring Inverse Mixcolumns . The output of decryption process will produce the original image to the receiver.

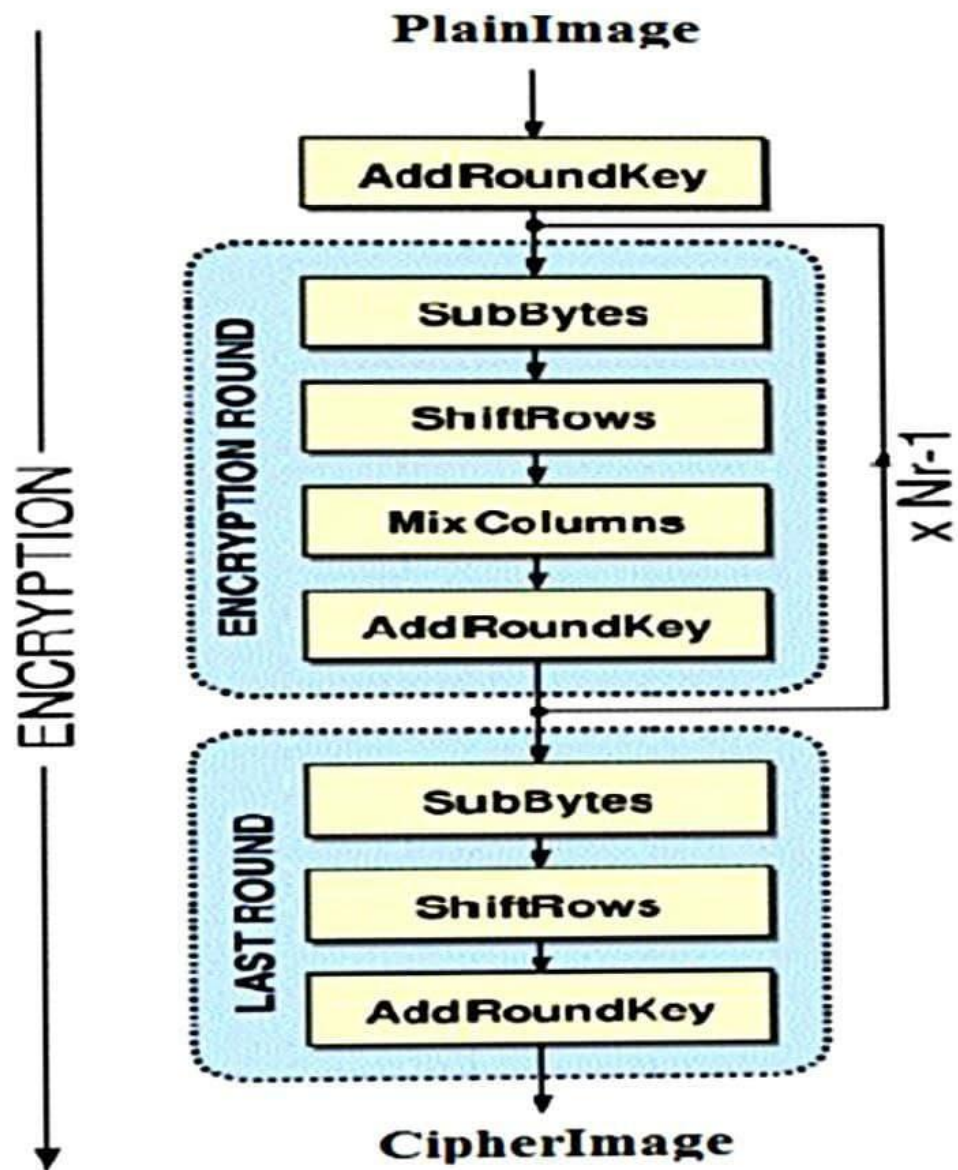


Fig 8. AES – Encryption process

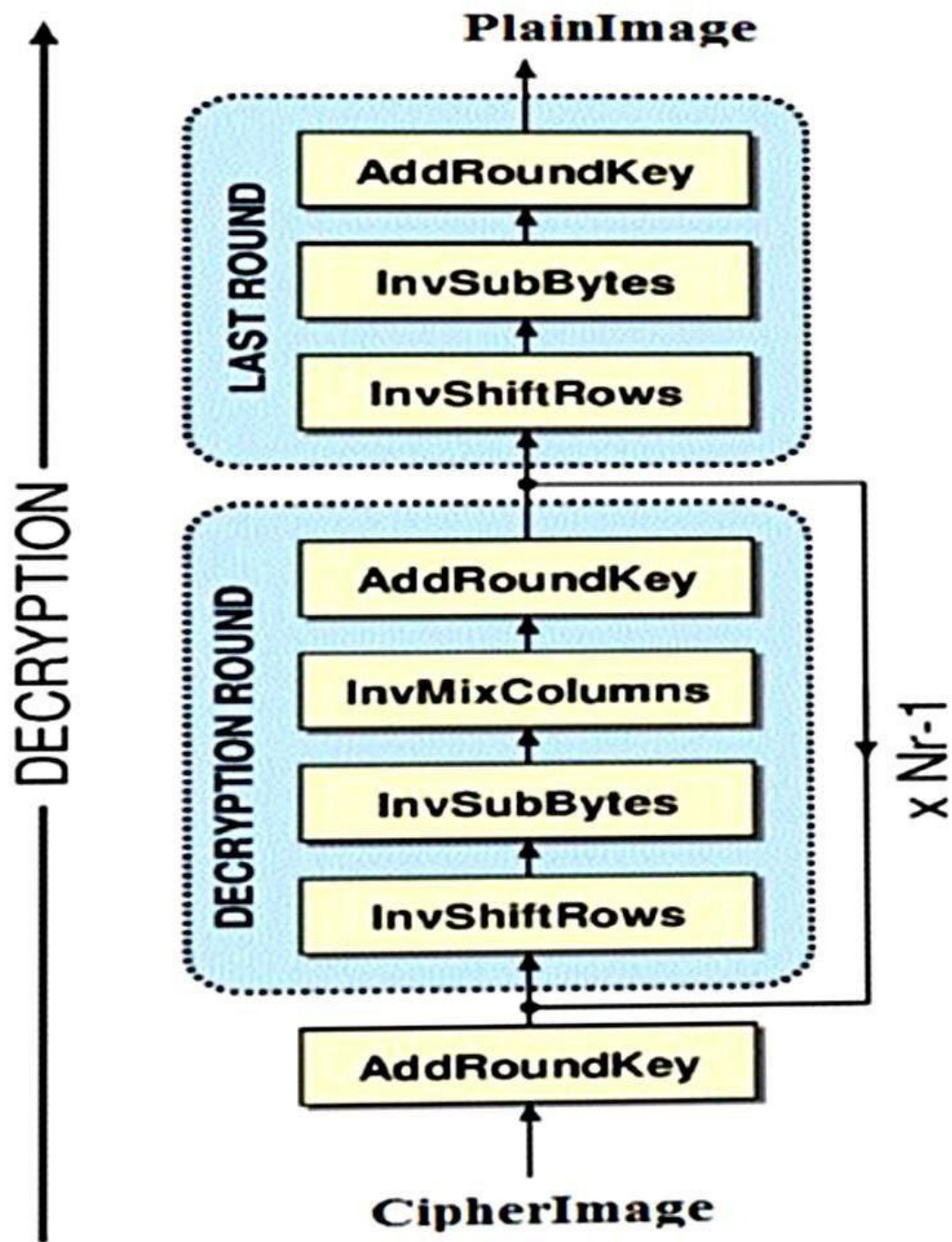


Fig 9. *AES-Decryption Process*

4.RESULT AND ANALYSIS

4.1 RESULT

IMAGE

The input-image given to the AES algorithm is of JPG/PNG format. The original image is given to the encryption process, produced the cipher-image and decryption process is enhanced by providing the cipher-image as input, produced the plain-image. Both encryption and decryption utilize the same key.

same key.



Original image



Encrypted image



Decrypted image

Fig 11. Final Output1

TEXT

The Plain-text is given as input to the AES algorithm, performs the process of encryption, thus produced the cipher-text and the decryption process is enhanced by providing the cipher-text as input, produced the plain-text.

Original Text : hello world

Encrypted Text : s207+VdCkD6p/T9HwcwHGg==

DeCrypted Text : hello world

4.2 ANALYSIS

TEXT TIME COMPLEXITY-

The time complexity of encryption and decryption for text has been calculated using AES algorithm and the following results are obtained.

Table IV. Text-time Complexity

Size(kb)	Encryption(ms)	Decryption(ms)
20	838	917
25	911	989
50	941	1128
75	1154	1453
100	1704	2117
150	1815	2305

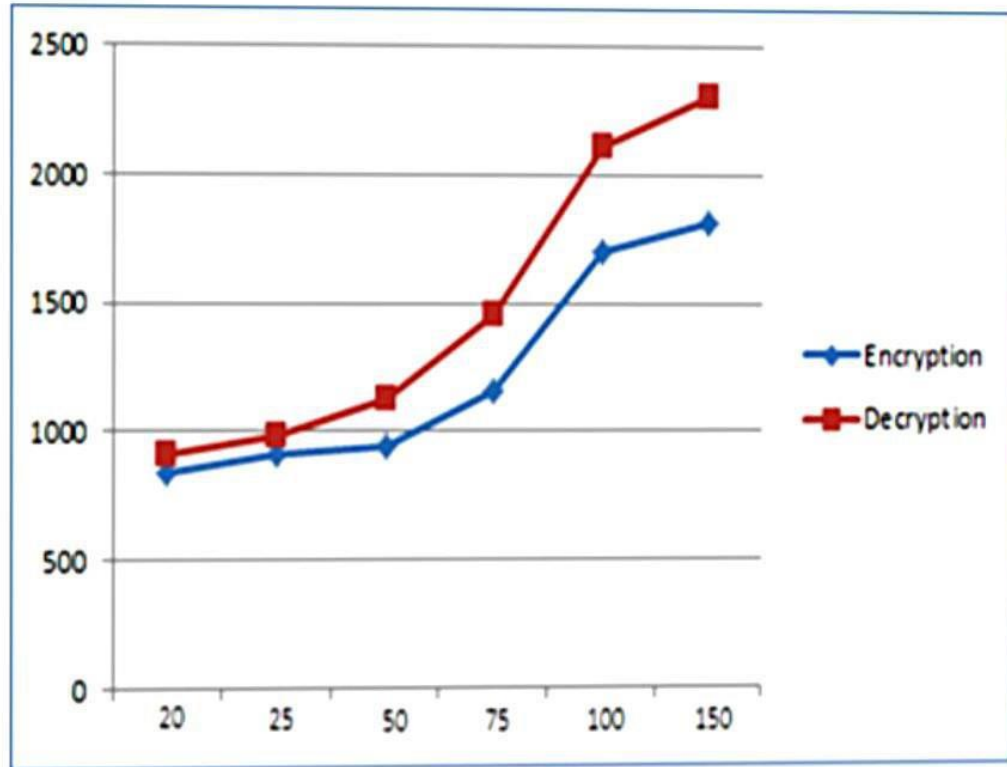


Fig 13. Final Output1

IMAGE-TEXT COMPLEXITY

The time complexity of encryption and decryption for image has been calculated using AES algorithm and the following results are obtained. The Table 5 displays the time complexity of image for cryptographic process. The result shows that —as the size increases the encryption time increases and decryption time also increases. But, to be noted the encryption and decryption time for image is lesser than time complexity of text.

Table V. *Text-time Complexity*

Size(kb)	Encryption(ms)	Decryption(ms)
20	17	25
25	21	34
55	40	58
75	57	98
100	77	132
150	108	188

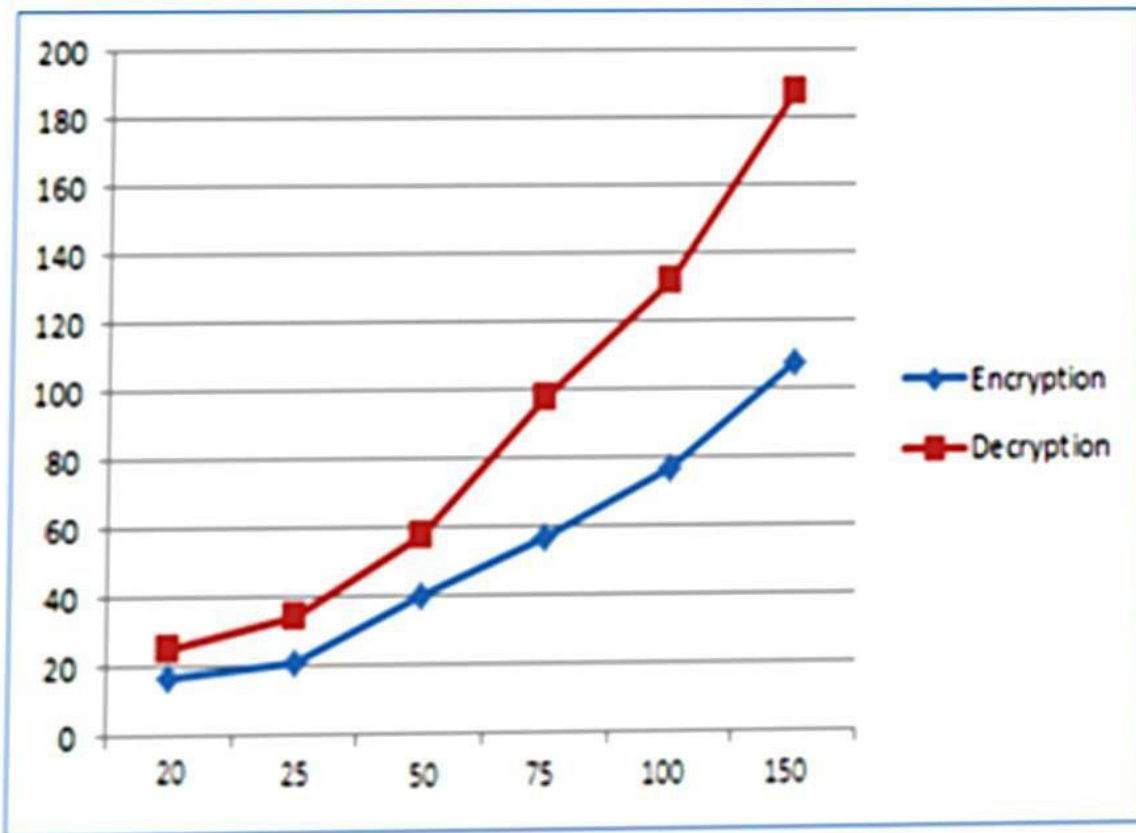


Fig 14. Final Output2

COMPARISON OF TEXT AND IMAGE

The time complexity of encryption and decryption for both text image has been compared by using the above results

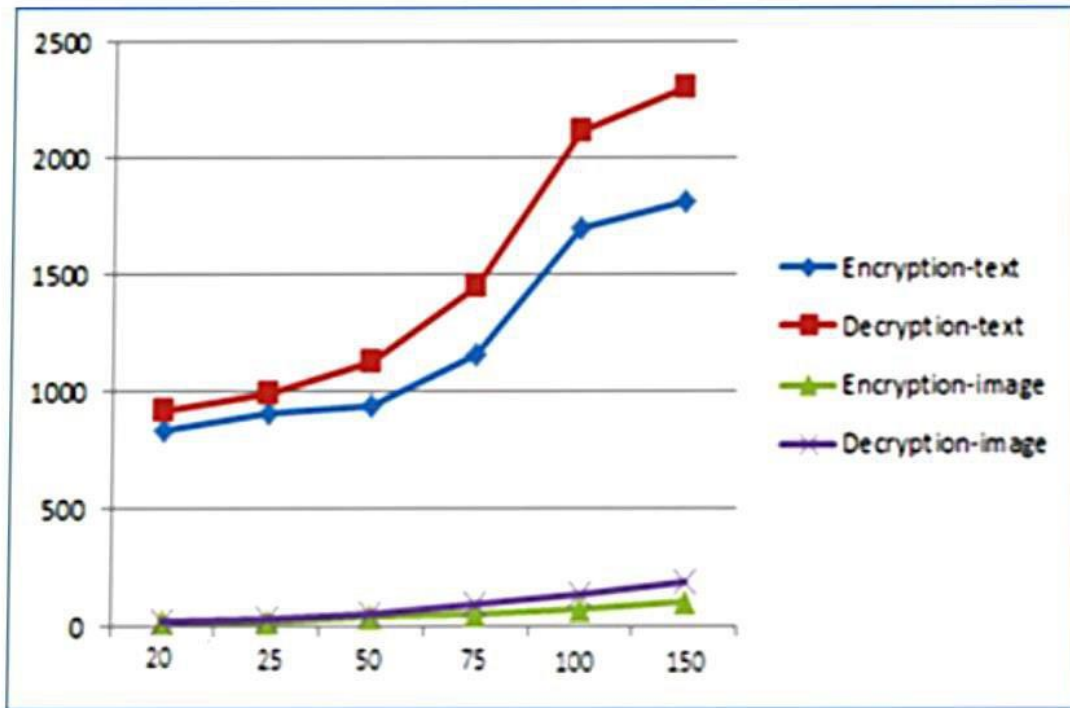


Fig 15. Comparison – Text & Image

Table VI. Text & Image – Time Complexity

Size (kb)	Text		Image	
	Encryption (ms)	Decryption (ms)	Encryption (ms)	Decryption (ms)
20	838	917	17	25
25	911	989	21	34
50-55	941	1128	40	58
75	1154	1454	57	98
100	1704	2117	77	132
150	1815	2305	108	188

4.3 INFERENCE

- 1.AES algorithm works well for both encryption and decryption.
- 2.From Figure 15 the decryption time is larger than encryption time for both the text and image.
- 3.In Figure 13 and Figure 14 the encryption and decryption time increases with increase in file size.
- 4.But while comparing both text and image from Figure 15 the decryption time of image is lesser than text. So, it is efficient than text.

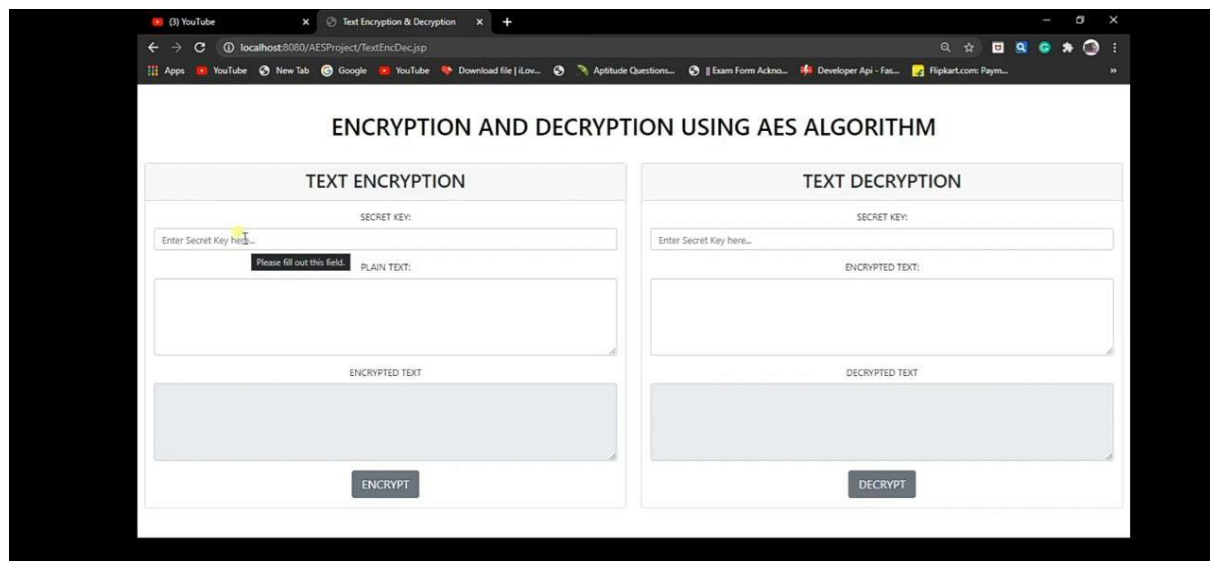
5. CONCLUSION

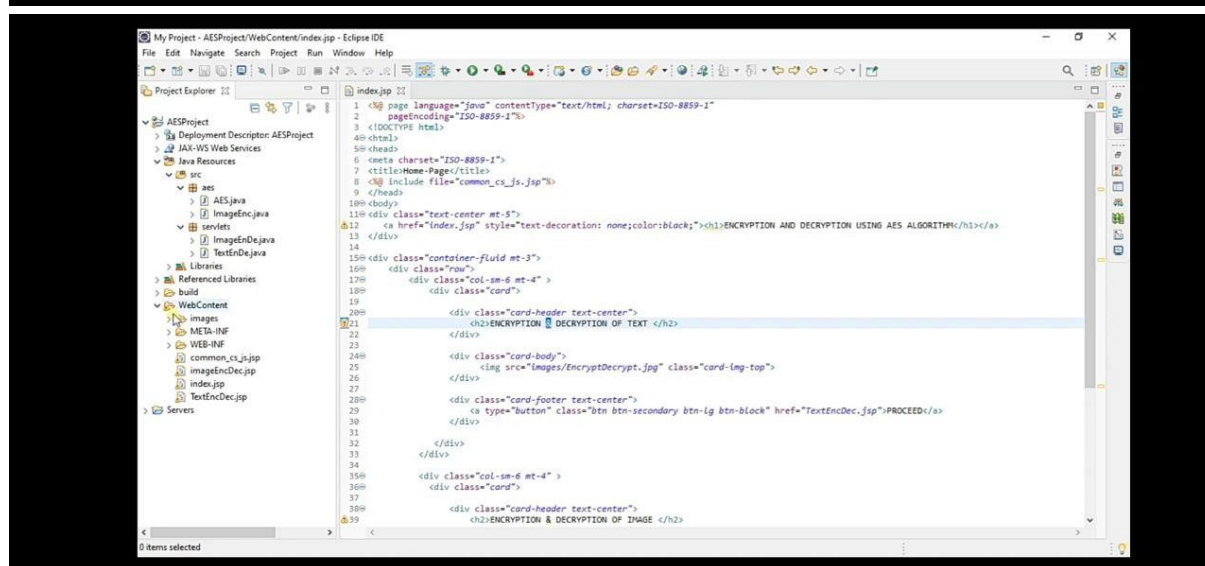
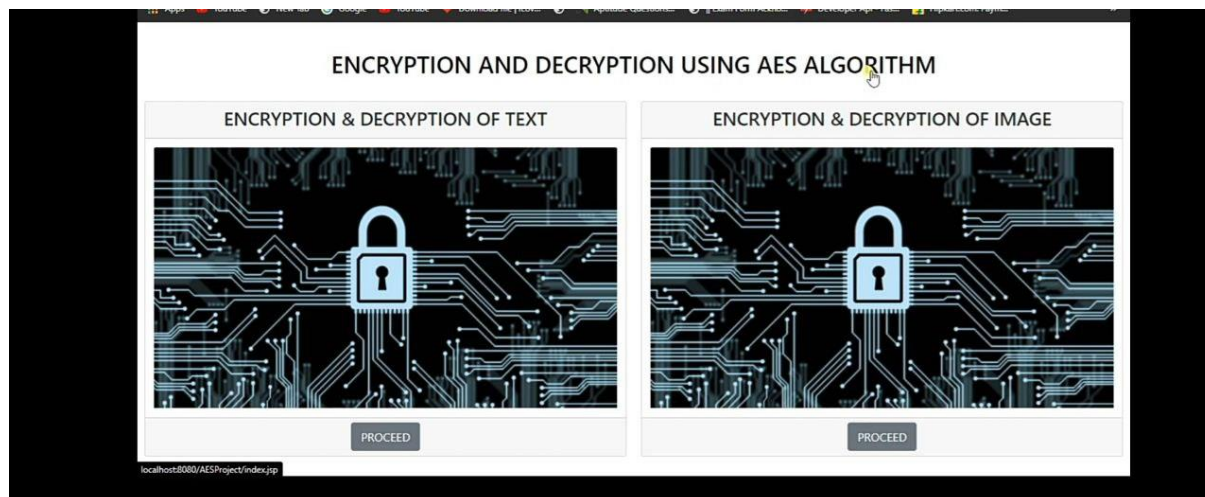
The proposed work makes use of AES algorithm to encrypt and decrypt the image and text. It makes use of 128 bit key for encryption which makes AES secure and faster than DES.As the key size is larger, it helps to overcome several attacks such as brute force attack and man in the middle attack. In our proposed system, encryption image doesn't remain the same. The encryption image is chosen in random. So, it is difficult for intruder to differentiate the encrypted image and the original image. So, AES algorithm is most suited for image encryption in real time applications. As a future work, we are planning for a different encryption keys in each round to perform encryption.

REFERENCE

- [1] Priya Deshmukh, —An Image Encryption and Decryption Using AES Algorithm, International Journal of Scientific & Engineering Research(IJSER), Vol.7, Issue.2, pp.210-213, 2016.
- [2] J. Daemen and V. Rijmen, The block cipher Rijndael, Smart Card research and Applications, LNCS 1820, Springer-Verlag, pp. 288-296.
- [3] J. Nechvatal, et. al., Report on the Development of the Advanced Encryption Standard (AES), National Institute of Standards and Technology, October 2, 2000.
- [4] —Specification for the Advanced Encryption Standard (AES), Federal Information Processing Standards Publication 197, Nov. 2001
- [5] A. Menezes, P. van Oorschot, and S. Vanstone, Handbook of Applied Cryptography, CRC Press, New York, 1997, p. 81-83.
- [6] C.-P. Su, T.-F. Lin, C.-T. Huang, and C.-W. Wu, —A high-throughput low-cost AES processor, IEEE Commun. Mag., vol. 41, no. 12, pp.86–91, Dec. 2003.
- [7] C.-P. Su, C.-L. Horng, C.-T. Huang, and C.-W. Wu, —A configurable AES processor for enhanced security, in Proc. ASP-DAC, Shanghai, China, Jan. 2005, pp. 361–366.
- [8] Rachh, R.R.; Anami, B.S.; Ananda Mohan, P.V. —Efficient implementations of S-box and inverse S-box for AES algorithm, in TENCON 2009 - 2009 IEEE Region 10 Conference , Nov. 2009, pp. 1–6.
- [9] Kaur, Swinder; Vig, Renu , Efficient Implementation of AES Algorithm in FPGA Device in Conference on

SCREEN SHOTS





SOURCE CODE

AES TEXT ENCRYPTION-

```
Package aes;
import java.io.UnsupportedEncodingException;
import java.security.MessageDigest;
import java.security.NoSuchAlgorithmException;
import java.util.Arrays;
import java.util.Base64;

import javax.crypto.Cipher;
import javax.crypto.spec.SecretKeySpec;

public class AES {

    private static SecretKeySpec secretKey;
    private static byte[] key;

    public static void setKey(String myKey)
    {
        MessageDigest sha = null;
        try {
            key = myKey.getBytes("UTF-8");
            sha = MessageDigest.getInstance("SHA-1");
            key = sha.digest(key);
            key = Arrays.copyOf(key, 16);
            secretKey = new SecretKeySpec(key, "AES");
        }
        catch (NoSuchAlgorithmException e) {
            e.printStackTrace();
        }
    }
}
```

```

        catch (UnsupportedEncodingException e) {
            e.printStackTrace();
        }
    }

    public static String encrypt(String strToEncrypt, String secret)
    {
        try
        {
            setKey(secret);
            Cipher cipher = Cipher.getInstance("AES/ECB/PKCS5Padding");
            cipher.init(Cipher.ENCRYPT_MODE, secretKey);
            return
Base64.getEncoder().encodeToString(cipher.doFinal(strToEncrypt.getBytes("UTF-8")));
        }
        catch (Exception e)
        {
            System.out.println("Error while encrypting: " + e.toString());
        }
        return null;
    }

    public static String decrypt(String strToDecrypt, String secret)
    {
        try
        {
            setKey(secret);
            Cipher cipher = Cipher.getInstance("AES/ECB/PKCS5PADDING");
            cipher.init(Cipher.DECRYPT_MODE, secretKey);
            return new String(cipher.doFinal(Base64.getDecoder().decode(strToDecrypt)));
        }
        catch (Exception e)
        {
            System.out.println("Error while decrypting: " + e.toString());
        }
        return null;
    }
}

```

IMAGE ENCRYPTION-

```
package aes;
```

```

import java.io.FileInputStream;
import java.io.FileOutputStream;
import javax.crypto.Cipher;
import javax.crypto.CipherOutputStream;

```

```

import javax.crypto.spec.SecretKeySpec;

public class ImageEnc
{
    public static void EncryptImage(String imagePath,String secretKey)
    {
        try
        {
            FileInputStream file= new FileInputStream(imagePath);
            FileOutputStream outStream = new
FileOutputStream("C:/Users/Vostro.MURADALIMJ/Desktop/EncDecImages/E
ncrypt.jpg");
            byte k[]= secretKey.getBytes();
            SecretKeySpec key= new SecretKeySpec(k, "AES");
            Cipher enc = Cipher.getInstance("AES");
            enc.init(Cipher.ENCRYPT_MODE, key);
            CipherOutputStream cos= new CipherOutputStream(outStream, enc);
            byte[] buf= new byte[1024];
            int read;
            while((read=file.read(buf))!=-1){
                cos.write(buf,0,read);
            }
            file.close();
            outStream.flush();
            cos.close();

        } catch (Exception ex)
        {
            System.out.println(ex);
        }
    }

    public static void DecryptImage(String imagePath,String secretKey)
    {
        try
        {
            FileInputStream file= new FileInputStream(imagePath);
            FileOutputStream outStream = new
FileOutputStream("C:/Users/Vostro.MURADALIMJ/Desktop/EncDecImages/D
ecrypt.jpg");
            byte k[]= secretKey.getBytes();
            SecretKeySpec key= new SecretKeySpec(k, "AES");
            Cipher enc = Cipher.getInstance("AES");
            enc.init(Cipher.DECRYPT_MODE, key);

```

```

        CipherOutputStream cos= new CipherOutputStream(outStream, enc);
        byte[] buf= new byte[1024];
        int read;
        while((read=file.read(buf))!=-1){
            cos.write(buf,0,read);
        }
        file.close();
        outStream.flush();
        cos.close();

    } catch (Exception ex)
    {
        System.out.println(ex);
    }
}
}

```

SERVLET CODE FOR IMAGE ENCRYPTION AND DECRYPTION

```
package servlets;
```

```
import java.io.IOException;
```

```
import javax.servlet.RequestDispatcher;
import javax.servlet.ServletException;
import javax.servlet.annotation.MultipartConfig;
import javax.servlet.annotation.WebServlet;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;
import javax.servlet.http.Part;
import aes.ImageEnc;
```

```
/**
```

```
 * Servlet implementation class ImageEnDe
```

```
 */
```

```
@MultipartConfig
```

```
@WebServlet("/ImageEnDe")
```

```
public class ImageEnDe extends HttpServlet {
```

```
    private static final long serialVersionUID = 1L;
```

```
    /**
```

```

    * @see HttpServlet#HttpServlet()
    */
    public ImageEnDe() {
        super();
        // TODO Auto-generated constructor stub
    }

    /**
     * @see HttpServlet#doGet(HttpServletRequest request,
     HttpServletResponse response)
     */
    protected void doGet(HttpServletRequest request, HttpServletResponse
    response) throws ServletException, IOException {
        // TODO Auto-generated method stub
        //response.getWriter().append("Served at:
    ").append(request.getContextPath());
    }

    /**
     * @see HttpServlet#doPost(HttpServletRequest request,
     HttpServletResponse response)
     */
    protected void doPost(HttpServletRequest request,
    HttpServletResponse response) throws ServletException, IOException {
        // TODO Auto-generated method stub
        //doGet(request, response);

        RequestDispatcher rd;
        String action=request.getParameter("action");
        System.out.println(action);
        if(action.equals("imageE"))
        {
            String seKey=request.getParameter("seKey");
            Part part =request.getPart("image");
            String imageFileName=part.getSubmittedFileName();
            String
path="C:/Users/Vostro.MURADALIMJ/Desktop/MyFolder/"+imageFileName;
            ImageEnc.EncryptImage(path, seKey);
            request.setAttribute("resultEnc","Selected Image Enrypted
Successfully");
            rd=request.getRequestDispatcher("imageEncDec.jsp");
            rd.forward(request,response);
        }
    }

```

```

        if(action.equals("imageD"))
        {
            String deKey=request.getParameter("sdKey");
            System.out.println(deKey);
            Part part =request.getPart("image");
            String imageFileName=part.getSubmittedFileName();
            String
path="C:/Users/Vostro.MURADALIMJ/Desktop/EncDecImages/"+imageFileNa
me;

            ImageEnc.DecryptImage(path, deKey);
            request.setAttribute("resultDec","Selected Image Decrypted
Successfully");
            rd=request.getRequestDispatcher("imageEncDec.jsp");
            rd.forward(request,response);
        }

    }

}

```

SERVLET TEXT ENCRYPTION AND DECRYPTION

```

package servlets;

import java.io.IOException;

import javax.servlet.RequestDispatcher;
import javax.servlet.ServletException;
import javax.servlet.annotation.WebServlet;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;

import aes.AES;

/**
 * Servlet implementation class TextEnDe
 */
@WebServlet("/TextEnDe")
public class TextEnDe extends HttpServlet {
    private static final long serialVersionUID = 1L;

    /**

```

```

    * @see HttpServlet#HttpServlet()
    */
    public TextEnDe() {
        super();
        // TODO Auto-generated constructor stub
    }

    /**
     * @see HttpServlet#doGet(HttpServletRequest request,
     HttpServletResponse response)
     */
    protected void doGet(HttpServletRequest request, HttpServletResponse
    response) throws ServletException, IOException {
        // TODO Auto-generated method stub
        //response.getWriter().append("Served at:
    ").append(request.getContextPath());
    }

    /**
     * @see HttpServlet#doPost(HttpServletRequest request,
     HttpServletResponse response)
     */
    protected void doPost(HttpServletRequest request,
    HttpServletResponse response) throws ServletException, IOException
    {
        // TODO Auto-generated method stub
        //doGet(request, response);

        String action=request.getParameter("action");

        if(action.equals("TextE"))
        {
            String plainText=request.getParameter("plainText");
            final String secretKey=request.getParameter("secretKey");

            String encryptedString = AES.encrypt(plainText,
secretKey);

            RequestDispatcher rd;
            request.setAttribute("eText",encryptedString);
            request.setAttribute("pText",plainText);
            request.setAttribute("sText",secretKey);
            rd=request.getRequestDispatcher("TextEncDec.jsp");
            rd.forward(request, response);

```

```

    }

    if(action.equals("TextD"))
    {
        final String secKey=request.getParameter("secKey");
        String encText=request.getParameter("encText");

        String decryptedString = AES.decrypt(encText, secKey);

        RequestDispatcher rd;
        request.setAttribute("secKey",secKey);
        request.setAttribute("encText",encText);
        request.setAttribute("decryptedString",decryptedString);
        rd=request.getRequestDispatcher("TextEncDec.jsp");
        rd.forward(request, response);
    }
}
}

```

WEB PAGE TEXT &IMAGE CODES

```

<%@ page language="java" contentType="text/html; charset=ISO-8859-1"
    pageEncoding="ISO-8859-1"%>
<!DOCTYPE html>
<html>
<head>
<meta charset="ISO-8859-1">
<title>Image Encryption & Decryption</title>
<%@ include file="common_cs_js.jsp"%>
</head>
<body>

<div class="text-center mt-5">
    <a href="index.jsp" style="text-decoration:
none;color:black;"><h1>ENCRYPTION AND DECRYPTION USING AES
ALGORITHM</h1></a>
</div>

```

```

<div class="container-fluid mt-3">
  <div class="row">
    <div class="col-sm-6 mt-4" >
      <div class="card text-center">

        <div class="card-header text-center">
          <h2>IMAGE ENCRYPTION</h2>
        </div>

        <div class="card-body">
          <form action="ImageEnDe" method="post"
enctype="multipart/form-data" >
            <input type="hidden" name="action" value="imageE">

<%
  String message=(String)request.getAttribute("resultEnc");
  if(message!=null)
  {

%>
    <div class="alert alert-success alert-dismissible fade show text-center"
role="alert">
      <strong><%=message %></strong>
    </div>
<%
  }
%>

      <div class="mb-3">
        <label for="exampleInputEmail1" class="form-
label">SECRET KEY:</label>
        <input class="form-control" type="text" name="seKey"
placeholder="Enter Secret Key here..." required>
      </div>

      <div class="mb-3">
        <label for="exampleInputEmail1" class="form-
label">SELECT IMAGE:</label>
        <input type="file" class="form-control" name="image"
required>
      </div>

      <button type="submit" class="btn btn-secondary btn-lg btn-
block">ENCRYPT</button>

```

```

        </form>
    </div>

</div>
</div>

<div class="col-sm-6 mt-4" >
    <div class="card text-center">

        <div class="card-header text-center">
            <h2>IMAGE DECRYPTION</h2>
        </div>

        <div class="card-body">
            <form action="ImageEnDe" method="post"
enctype="multipart/form-data">
                <input type="hidden" name="action" value="imageD">

<%
    String message1=(String)request.getAttribute("resultDec");
    if(message1!=null)
    {

%>
        <div class="alert alert-success alert-dismissible fade show text-center"
role="alert">
            <strong><%=message1 %></strong>
        </div>
<%
    }
%>

        <div class="mb-3">
            <label for="exampleInputEmail1" class="form-
label">SECRET KEY:</label>
            <input class="form-control" type="text" name="sdKey"
placeholder="Enter Secret Key here..." required>
        </div>

        <div class="mb-3">
            <label for="exampleInputEmail1" class="form-
label">SELECT ENCRYPTED IMAGE:</label>
            <input type="file" class="form-control" name="image"

```

```

required>
        </div>

        <button type="submit" class="btn btn-secondary btn-lg btn-
block">DECRYPT</button>

    </form>
</div>

</div>
</div>
</div>
</div>

</body>
</html>

```

TEXT-

```

<%@ page language="java" contentType="text/html; charset=ISO-8859-1"
    pageEncoding="ISO-8859-1"%>
<!DOCTYPE html>
<html>
<head>
<meta charset="ISO-8859-1">
<title>Text Encryption & Decryption</title>
<%@ include file="common_cs_js.jsp"%>
</head>
<body>

<div class="text-center mt-5">
    <a href="index.jsp" style="text-decoration:
none;color:black;"><h1>ENCRYPTION AND DECRYPTION USING AES
ALGORITHM</h1></a>
</div>

<div class="container-fluid mt-3">
    <div class="row">
        <div class="col-sm-6 mt-4" >
            <div class="card text-center">

                <div class="card-header text-center">
                    <h2>TEXT ENCRYPTION</h2>

```

```

</div>

<%
    String eText=(String)request.getAttribute("eText");
    String pText=(String)request.getAttribute("pText");
    String sText=(String)request.getAttribute("sText");
%>

<div class="card-body">
    <form action="TextEnDe" method="post">
        <input type="hidden" name="action" value="TextE">
        <div class="mb-3">
            <label for="exampleInputEmail1" class="form-
label">SECRET KEY:</label>
            <input class="form-control" type="text" name="secretKey"
placeholder="Enter Secret Key here..." value="<% if(sText!=null){%>
<%=sText %> <% } %>" required>
            </div>
            <div class="mb-3">
                <label for="exampleInputEmail1" class="form-label">PLAIN
TEXT:</label>
                <textarea class="form-control" rows="5" name="plainText"
required><% if(pText!=null){%> <%=pText %> <% } %> </textarea>
                </div>
                <div class="mb-3">
                    <label for="exampleInputPassword1" class="form-
label">ENCRYPTED TEXT</label>
                    <textarea class="form-control" rows="5"
name="encryptText" readonly><% if(eText!=null){%> <%=eText %> <% }
%></textarea>
                    </div>

                    <button type="submit" class="btn btn-secondary btn-lg btn-
block">ENCRYPT</button>
                </form>
            </div>

        </div>
    </div>

    <div class="col-sm-6 mt-4" >
        <div class="card text-center">

```

```

<div class="card-header text-center">
  <h2>TEXT DECRYPTION</h2>
</div>

<%
  String secKey=(String)request.getAttribute("secKey");
  String encText=(String)request.getAttribute("encText");
  String
decryptedString=(String)request.getAttribute("decryptedString");
%>

<div class="card-body">
  <form action="TextEnDe" method="post">
    <input type="hidden" name="action" value="TextD">
    <div class="mb-3">
      <label for="exampleInputEmail1" class="form-
label">SECRET KEY:</label>
      <input class="form-control" type="text" name="secKey"
placeholder="Enter Secret Key here..." value="<% if(secKey!=null){%>
<%=secKey %> <% } %>" required>
    </div>
    <div class="mb-3">
      <label for="exampleInputEmail1" class="form-
label">ENCRYPTED TEXT:</label>
      <textarea class="form-control" rows="5" name="encText"
required><% if(encText!=null){%> <%=encText %> <% } %></textarea>

    </div>
    <div class="mb-3">
      <label for="exampleInputPassword1" class="form-
label">DECRYPTED TEXT</label>
      <textarea class="form-control" rows="5" id="decText"
readonly><% if(decryptedString!=null){%> <%=decryptedString %> <% }
%></textarea>
    </div>

    <button type="submit" class="btn btn-secondary btn-lg btn-
block">DECRYPT</button>
  </form>
</div>

</div>

```

```
        </div>
    </div>
</div>

</body>
</html>
```