Implementation of AES with Time Complexity Measurement for Various Input

Shraddha More[1]

[1] Mumbai university

Received: 11 February 2015 Accepted: 2 March 2015 Published: 15 March 2015

Abstract

Network Security has a major role in the development of data communication system, where more randomization in the secret keys increases the security as well as the complexity of the cryptography algorithms. In the recent years network security has become an important issue. Cryptography has come up as a solution which plays a vital role in the information security system against various attacks. This security mechanism uses the AES algorithm to scramble data into unreadable text which can only be decrypted with the associated key. The AES algorithm is limited only for text as an input. It also has, the more time complexity. So it suffers from vulnerabilities associated with another type of input and time constraints. So its challenge to implement the AES algorithm for various types of input and require less decryption time. The propose work demonstrate implementation of a 128-bit Advanced Encryption Standard (AES), which consists of both symmetric key encryption and decryption algorithms for input as a text, image and audio. It also gives less time complexity as compared to existing one. At the last stage comparing the time complexity for encryption and decryption process for all three types of input. This paper also demonstrates a side channel attack on the standard software implementation of the AES cryptographic algorithm.de

Index terms — side channel attack, aes, des, rsa, encryption, decryption, cryptography, network security..

1 ImplementationofAESwithTimeComplexityMeasurementforVariousInpu

Strictly as per the compliance and regulations of: Introduction ryptography plays an important role in the security of data transmission. Data Security is a challenging concern of data communications that focuses on many areas including secure communication channel and strong data encryption technique. The secure transmission of confidential data enclosed gets a great deal of attention because of the rapid development in information technology. The predictable methods of encryption can only maintain the data security. The development of computing technology imposes stronger requirements on the cryptography schemes. The rapidly growing number of wireless communication users has led to the increasing demand for security measures and devices to protect user data transmitted over wireless channels [1].

Two types of cryptographic systems have been developed for that purpose symmetric (secret key) and asymmetric (public key) cryptosystems. Symmetric cryptography, such as in the Data Encryption Standard (DES), 3DES, and Advanced Encryption Standard (AES) uses an identical key of the sender to encrypt the message text and receiver to decrypt the encrypted text. Asymmetric cryptography, such as the Rivest-Shamir-Adleman (RSA) uses different public keys for encryption and decryption, eliminating the key exchange problem. [2] Symmetric cryptography is more suitable for the encryption of a large amount of data. The Data Encryption Standard (DES) has been used by the U.S. government standard since 1977. However, now, it can be cracked quickly and inexpensively. The AES algorithm defined by the National Institute of Standards and Technology (NIST) of the United States has widely accepted to replace DES as the new symmetric encryption algorithm [3].

This above cryptographic algorithms are not more secure. To overcome the vulnerabilities in network security in 2000, the Advanced Encryption Standard (AES) replaced the DES to meet the ever-increasing requirements for security. In cryptography, the AES, also called as Rijndael, is a block cipher adopted as an encryption standard by the US government, which specifies an encryption algorithm capable of protecting sensitive information [4]. The Rijndael algorithm is a symmetric block cipher that can encrypt and decrypt information. Encryption converts data into an unintelligible form called cipher-text. Decryption of the cipher-text converts the data back into its unique form which is called plaintext. The AES algorithm supports 128, 192 and 256 bit key length to encrypt and decrypt data in blocks of 128 bits , hence the name AES-128, AES-192 and AES-256 respectively [5]. The hardware implementation of the AES algorithm can provide high performance, low cost for specific applications and trustworthiness compared to its software counterparts [6].

The organization of the paper is as follows, Section II describes the design overview of AES algorithm for both encryption and decryption. Section III presents implementation Details, Section IV is discussed on Experimental Results. Section V projects on future scope and conclusion.

2 II.

3 Design Overview of aes

AES is a symmetric block cipher with block length of 128 bits. It allows three different key lengths 128,192 and 256 bits. In encryption process processing of 128 bit keys required for 10 rounds, 192 bit keys required for 12 rounds and 256 bit keys required for 14 rounds which is shown in table1. AES is a round based algorithm. For encryption and decryption each round has four functions excepting last round. Last round required three functions. The encryption algorithm has four round functions SubByte( ), ShiftRows( ), MixColumn( )and AddRoundKey( ). The decryption, also has the same number of rounds with reverse transformation, order of round function is different i.e. InvShiftRow( ), InvSubByte( ), AddRoundKey( ) and InvMixColumn( ) [2]- [3].

4 a) AES Encryption Algorithm

The Encryption process consists of a number of different transformations applied consecutively over the data block bits in a fixed number of iterations which is called as rounds. The number of rounds depends on the length of the key used for the encryption process. 10 iterations are required for key length of 128 bits.

i. High-level description of the algorithm KeyExpansions -round keys are derived from the cipher key using Rijndael's key schedule. AES requires a separate 128-bit round key block for each round plus one more.

ii. InitialRound 1. AddRoundKey( )-Each byte of the state is combined with a block of the round key using bitwise xor. Rounds 2. SubBytes( )-A non-linear substitution step where each byte is replaced with another according to a lookup table. 3. ShiftRows( )-A transposition step where the last three rows of the state are shifted cyclically a certain number of steps.

5 MixColumns( )-A mixing operation which operates

on the columns of the state, combining the four bytes in each column. iii. Final Round (No MixColumns) SubBytes( ) ShiftRows( ) AddRoundKey( ).

Steps : These steps are used to encrypt128-bit block. 1. The set of round keys from the cipher key. 2. Initialize state array and add the initial round key to the starting state array. 3. Perform round = 1 to 9 : Execute Usual Round.

4. Execute Final Round.

6 Corresponding cipher text chunk output of Final Round

Step iv. Encryption process Each round consists of the following four steps: SubBytes Transformation: In this transformation, each of the byte in the state matrix is replaced with another byte as per the S-box (Substitution Box) [7]. The S-box is generated by firstly calculating the respective reciprocal of that byte in GF (2^8) and then affine transform is applied. ShiftRows Transformation: In this transformation, the bytes in the first row of the State do not change. The second, third, fourth and fifth rows shift cyclically to the left by one byte, two bytes, three bytes and four bytes respectively [7]. MixColumns Transformation: It is the operation that mixes the bytes in each column by the multiplication of the state with a fixed polynomial matrix [7]. It completely changes the scenario of the cipher even if all bytes look very similar. The Inverse Polynomial Matrix does exist in order to reverse the mix column transformation.

7 AddRoundKey

Transformation:

In AddRoundKey transformation, a roundkey is added to the State by bitwise Exclusive-OR (XOR) operation. AddRoundKey proceeds onecolumn at a time. AddRoundKey adds a roundkey word with each state column matrix.The operation performed in AddRoundKey is matrix addition.

8 b) AES Decryption Algorithm

Decryption is the process of extracting the plaintext from cipher text. For decryption the same process occurs simply in reverse order by taking the 128-bit block of cipher text and converting it to plaintext by the application of the inverse of the four operations. Decryption involves reversing all the steps taken in encryption using following inverse functions. InvSubBytes Transformation: InvSubBytes is the inverse transformation of SubBytes, in which the inverse S-box is applied to individual bytes in the State. The inverse Sbox is constructed by first applying the inverse of the affine transformation, then computing the multiplicative inverse in GF(2^8).

InvShiftRows Transformation: InvShiftRows is the inverse transformation of ShiftRows. In this transformation, the bytes in the first row of the State do not change. The second, third, and fourth and fifth rows are shifted cyclically by one byte, two bytes, three bytes and four bytes to the right respectively [2]. InvMixColumns Transformation: InvMixColumns is the inverse transformation of MixColumns. This is a complex procedure as it involves severely the byte multiplication under GF (2^8) [2].

9 Key Expansion (Keyexpansion Operation)

Keyexpansion refers to the process in which the 128 bits of the original key are expanded into eleven 128-bit round keys.

To compute round key (n+1) from round key (n) these steps are performed: 1. Compute the new first column of the next round key.

First all the bytes of the old fourth column have to be substituted using the Subbytes operation. These four bytes are shifted vertically by one byte position and then XORed to the old first column. The result of these operations is the new first column. The key expansion algorithm generates 128 bit key for each round and one more key for initial AddRoundKey function. The same expanded key is used for encryption and decryption except for decryption it reads in reverse order.

10 III.

11 Implementation Details

The system proposing aims to achieve network security by implementing appropriate countermeasures based on concept of constant time encryption against side channel timing attack to protect implementations of secret key cryptography. The contribution work includes implementing more suitable countermeasures against side channel attack.

The propose system, is intended to provide secure transmission of data over the network by implementing the appropriate countermeasures against side channel attack on AES implementation which is shown in Fig. 2. Here the work implementing AES 128bit algorithm using 10 rounds by taking input as text, image and audio. In AES encryption process, system performs round functions like SubByte( ), ShiftRows( ), MixColumn( ) and AddRoundKey( ). On the other side, the decryption processperforms round functions like InvShiftRow( ), InvSubByte( ), AddRoundKey( ) and InvMixColumn( ). After that the work implementing side channel attack on the AES implementation in such a way that the receiver cannot decrypt the encrypted data. After successful implementation of side channel attack, research work implementing some appropriate countermeasures against side channel attack on AES implementation and finally evaluating their performance and soundness to prevent possible vulnerabilities and develop more secure systems. The work implemented AES 128-bit, 10 rounds algorithm by taking input as text, image and audio.

12 Encryption Process when input as an Text file Decryption Process when input as an Text file

The work implemented 128 bit AES algorithm (10 round) decryption using text as an input by measuring performance parameter as time complexity which is shown in Fig. 4.Time required for decryption process is 2.128282 milliseconds.

13 Encryption Process when input as an audio file

The work implemented 128 bit AES algorithm (10 round) encryption using audio as an input by measuring performance parameter as time complexity which is shown in Fig. 5.Time required for encryption process is 13.899532 milliseconds. The work implemented 128 bit AES algorithm (10 round) encryption using text as an input by measuring performance parameter as time complexity which is shown in Fig. **??**.Time required for encryption process is 1.166557 milliseconds.

14 Decryption Process when input as an audio file

The work implemented 128 bit AES algorithm (10 round) decryption using audio as an input by measuring performance parameter as time complexity which is shown in Fig. 6.Time required for decryption process is 20.183485milliseconds.

15 Encryption Process when input as an Image file

The work implemented 128 bit AES algorithm (10 round) encryption using image as an input by measuring performance parameter as time complexity which is shown in Fig. 7. Time required for encryption process is 61.958627 milliseconds.

16 Decryption Process when input as an Image file

17 Experimental Results

In this section The work presented result graph of our proposed system, implementation of the AES algorithm by taking text, image and audio as input

18 Conclusion Andfuture Scope

Due to the increasing needs for secure communications a safer and more secured cryptographic algorithm has to be proposed and implemented. The Advanced Encryption Standard (AES-128bit) is widely used nowadays in many applications. In this paper, the work implemented an efficient AES128 bit encryption and decryption algorithm. The execution time for AES encryption and decryption is calculated by performing 10 round functions. The system presented an attack on AES software implementations. Future work will focus on investigating and implementing a number of countermeasures against side channel attack on AES implementation and have evaluated their performance and soundness to prevent possible vulnerabilities and develop more secure systems.

19 Global

2

Figure 1: 2 .

[1] [2]

[1] © 2015 Global Journals Inc. (US) 1

[2] © 2015 Global Journals Inc. (US)

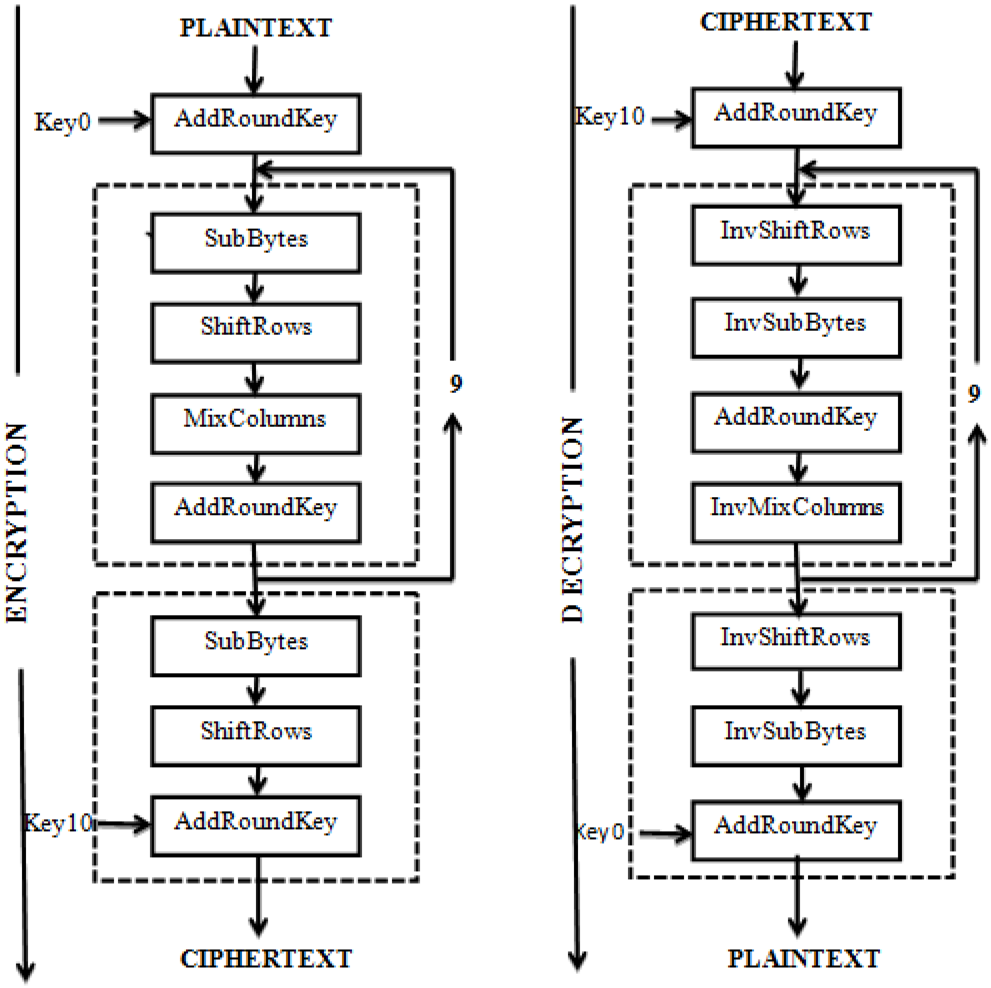
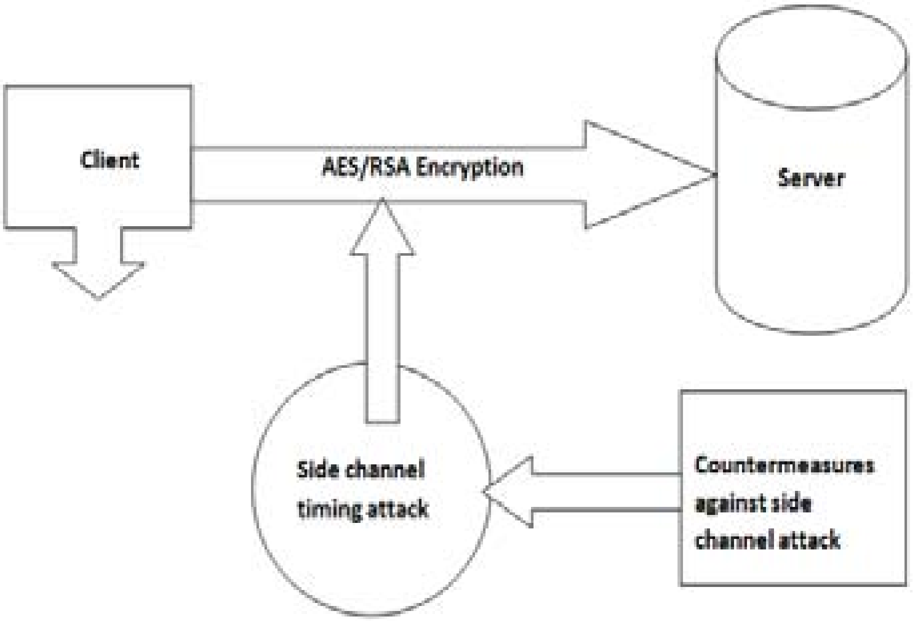
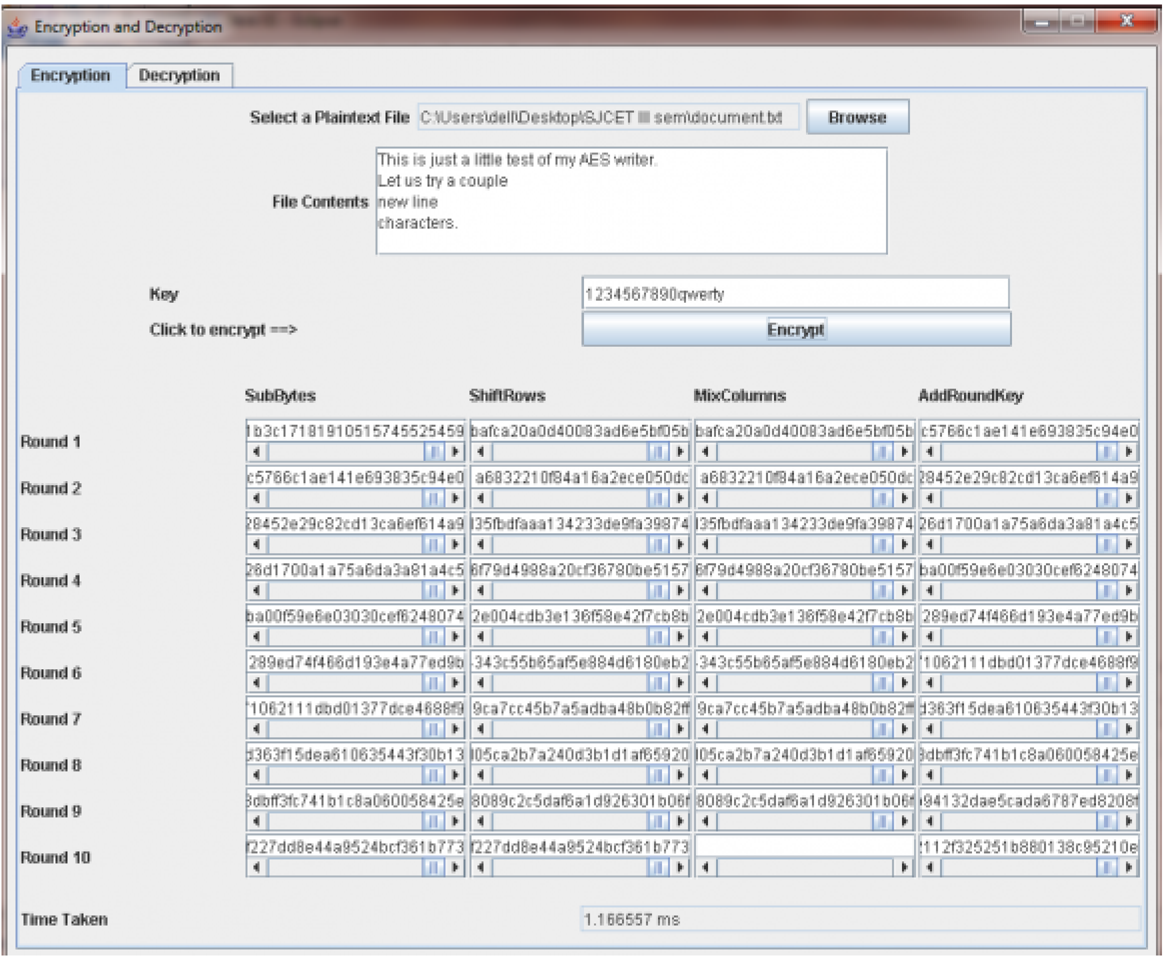


Figure 2: ©


Client
AES/RSA Encryption
Server
Side channel timing attack
Countermeasures against side channel attack

1

Figure 3: Figure 1 :



Select a Plaintext File C:\Users\dell\Desktop\SJCET III sem\document.txt

File Contents
This is just a little test of my AES writer.
Let us try a couple
new line
characters.

Key 1234567890qwerty

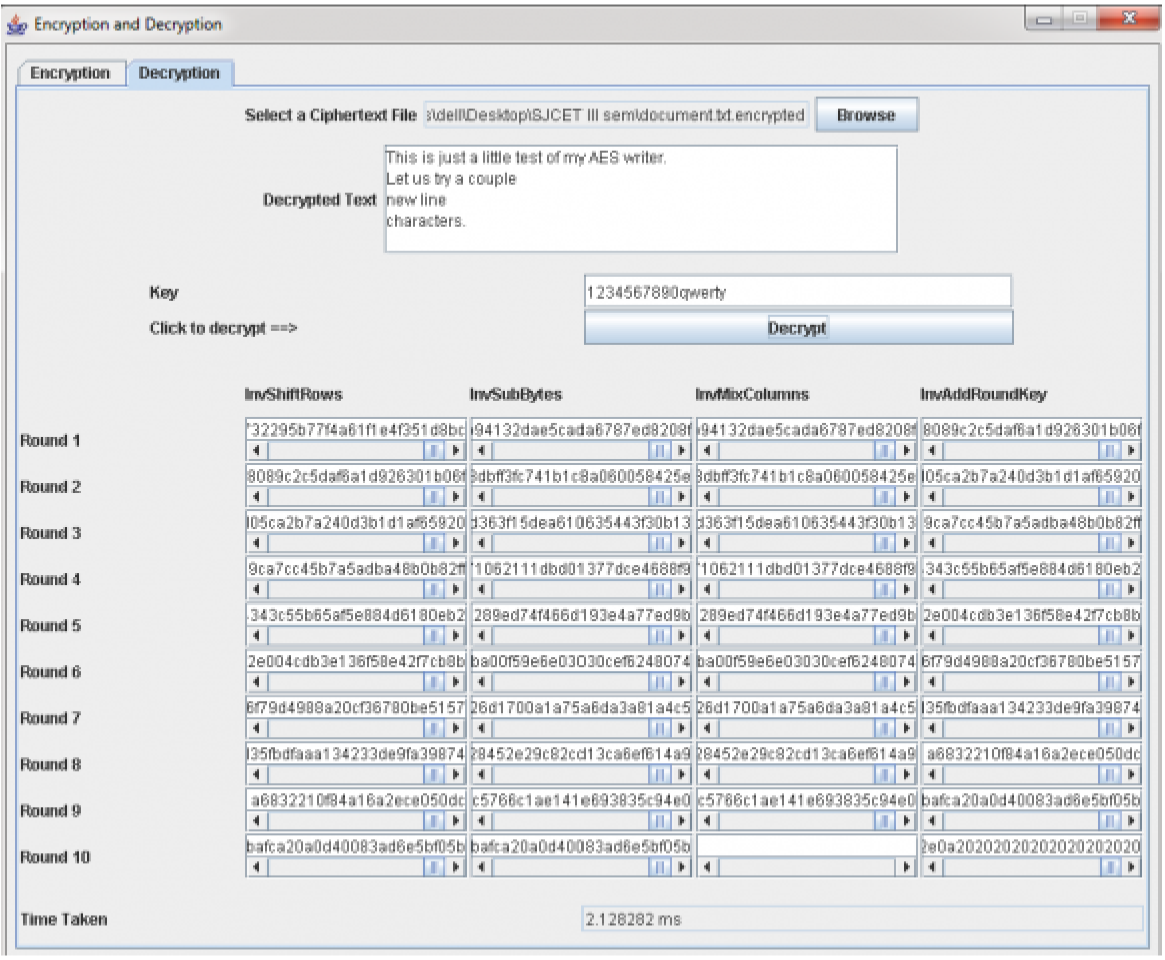
Click to encrypt ==>

	SubBytes	ShiftRows	MixColumns	AddRoundKey
Round 1	1b3c17181910515745525459	bafca20a0d40083ad6e5bf05b	bafca20a0d40083ad6e5bf05b	c5766c1ae141e693835c94e0
Round 2	c5766c1ae141e693835c94e0	a6832210f84a16a2ece050dc	a6832210f84a16a2ece050dc	28452e29c82cd13ca6ef614a9
Round 3	28452e29c82cd13ca6ef614a9	35fbdfaaa134233de9fa39874	35fbdfaaa134233de9fa39874	26d1700a1a75a6da3a81a4c5
Round 4	26d1700a1a75a6da3a81a4c5	6f79d4988a20cf36780be5157	6f79d4988a20cf36780be5157	ba00f59e6e03030cef6248074
Round 5	ba00f59e6e03030cef6248074	2e004cdb3e136f58e42f7cb8b	2e004cdb3e136f58e42f7cb8b	289ed74f466d193e4a77ed9b
Round 6	289ed74f466d193e4a77ed9b	343c55b65af5e884d6180eb2	343c55b65af5e884d6180eb2	1062111dbd01377dce4688f9
Round 7	1062111dbd01377dce4688f9	9ca7cc45b7a5adba48b0b82ff	9ca7cc45b7a5adba48b0b82ff	d363f15dea61063544330b13
Round 8	d363f15dea610635443f30b13	05ca2b7a240d3b1d1af65920	05ca2b7a240d3b1d1af65920	3dbff3fc741b1c8a060058425e
Round 9	3dbff3fc741b1c8a060058425e	8089c2c5daf6a1d926301b06f	8089c2c5daf6a1d926301b06f	94132dae5cada6787ed8208f
Round 10	f227dd8e44a9524bcf361b773	f227dd8e44a9524bcf361b773		112f325251b880138c95210e

Time Taken 1.166557 ms

2

Figure 4: Figure 2 :



Select a Ciphertext File s\dell\Desktop\SJCET III sem\document.txt.encrypted

Decrypted Text

This is just a little test of my AES writer.
Let us try a couple
new line
characters.

Key 1234567890qwerty

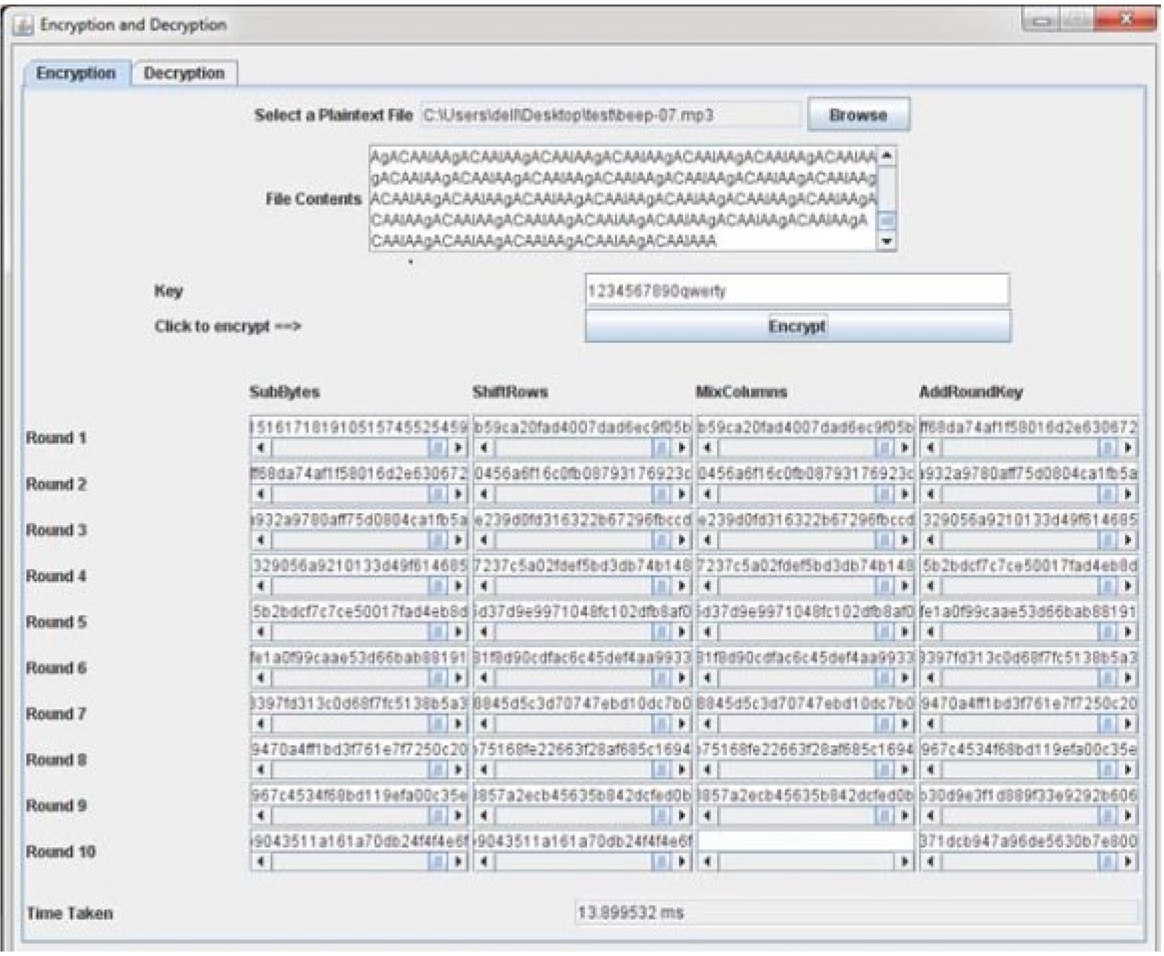
Click to decrypt ==>

	InvShiftRows	InvSubBytes	InvMixColumns	InvAddRoundKey
Round 1	32295b77f4a61f1e4f351d8bc	94132dae5cada6787ed8208f	94132dae5cada6787ed8208f	8089c2c5daf6a1d926301b06f
Round 2	8089c2c5daf6a1d926301b06f	3dbff3fc741b1c8a060058425e	3dbff3fc741b1c8a060058425e	05ca2b7a240d3b1d1af65920
Round 3	05ca2b7a240d3b1d1af65920	d363f15dea610635443f30b13	d363f15dea610635443f30b13	9ca7cc45b7a5adba48b0b82ff
Round 4	9ca7cc45b7a5adba48b0b82ff	1062111dbd01377dce4688f9	1062111dbd01377dce4688f9	343c55b65af5e884d6180eb2
Round 5	343c55b65af5e884d6180eb2	289ed74f466d193e4a77ed9b	289ed74f466d193e4a77ed9b	2e004cdb3e136f58e42f7cb8b
Round 6	2e004cdb3e136f58e42f7cb8b	ba00f59e6e03030cef6248074	ba00f59e6e03030cef6248074	6f79d4988a20cf36780be5157
Round 7	6f79d4988a20cf36780be5157	26d1700a1a75a6da3a81a4c5	26d1700a1a75a6da3a81a4c5	35fbdfaaa134233de9fa39874
Round 8	35fbdfaaa134233de9fa39874	28452e29c82cd13ca6ef614a9	28452e29c82cd13ca6ef614a9	a6832210f84a16a2ece050dc
Round 9	a6832210f84a16a2ece050dc	c5766c1ae141e693835c94e0	c5766c1ae141e693835c94e0	bafca20a0d40083ad6e5bf05b
Round 10	bafca20a0d40083ad6e5bf05b	bafca20a0d40083ad6e5bf05b		2e0a20202020202020202020

Time Taken 2.128282 ms

143

Figure 5: 14 GlobalFigure 3 :



Select a Plaintext File C:\Users\dell\Desktop\test\beep-07.mp3

File Contents

```
AgACAAIAAgACAAIAAgACAAIAAgACAAIAAgACAAIAAgACAAIAAgACAAIAA
gACAAIAAgACAAIAAgACAAIAAgACAAIAAgACAAIAAgACAAIAAgACAAIAAg
ACAAIAAgACAAIAAgACAAIAAgACAAIAAgACAAIAAgACAAIAAgACAAIAAgA
CAAIAAgACAAIAAgACAAIAAgACAAIAAgACAAIAAgACAAIAAgACAAIAAgA
CAAIAAgACAAIAAgACAAIAAgACAAIAAgACAAIAAA
```

Key 1234567890qwerty

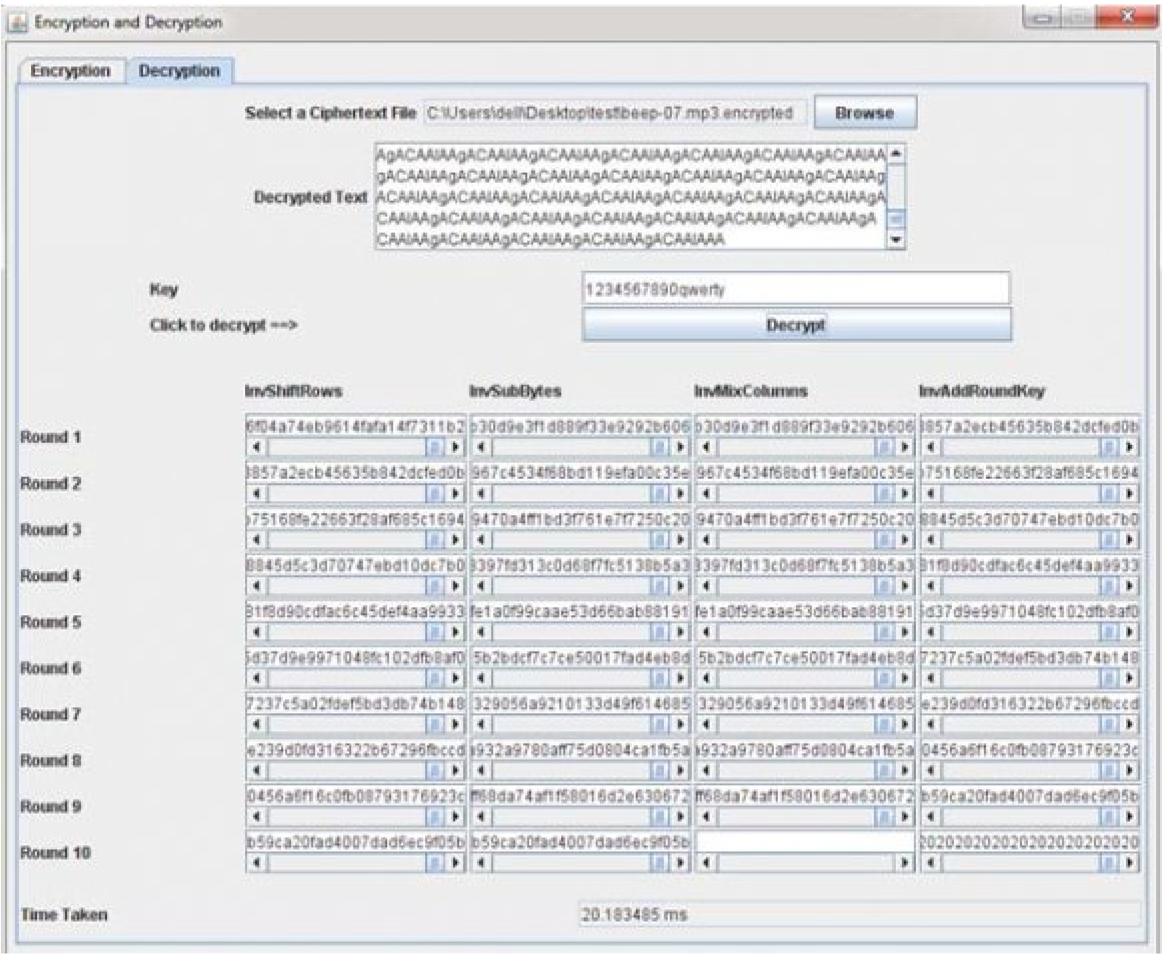
Click to encrypt ==>

	SubBytes	ShiftRows	MixColumns	AddRoundKey
Round 1	15161718191051574552545 9	b59ca20fad4007dad6ec9f05b	b59ca20fad4007dad6ec9f05b	ff68da74af1f58016d2e630672
Round 2	ff68da74af1f58016d2e630672	0456a6f16c0fb08793176923c	0456a6f16c0fb08793176923c	932a9780aff75d0804ca1fb5a
Round 3	932a9780aff75d0804ca1fb5a	e239d0fd316322b67296fbccd	e239d0fd316322b67296fbccd	329056a9210133d49f614685
Round 4	329056a9210133d49f614685	7237c5a02fdef5bd3db74b148	7237c5a02fdef5bd3db74b148	5b2bdcf7c7ce50017fad4eb8d
Round 5	5b2bdcf7c7ce50017fad4eb8d	d37d9e9971048fc102dfb8af0	d37d9e9971048fc102dfb8af0	fe1a0f99caae53d66bab88191
Round 6	fe1a0f99caae53d66bab88191	31f8d90cdfac6c45def4aa9933	31f8d90cdfac6c45def4aa9933	3397fd313c0d68f7fc5138b5a3
Round 7	3397fd313c0d68f7fc5138b5a3	8845d5c3d70747ebd10dc7b0	8845d5c3d70747ebd10dc7b0	9470a4ff1bd3f761e7f7250c20
Round 8	9470a4ff1bd3f761e7f7250c20	75168fe22663f28af685c1694	75168fe22663f28af685c1694	967c4534f68bd119efa00c35e
Round 9	967c4534f68bd119efa00c35e	857a2ecb45635b842dcfed0b	857a2ecb45635b842dcfed0b	b30d9e3f1d889f33e9292b606
Round 10	9043511a161a70db24f4f4e6f	9043511a161a70db24f4f4e6f		371dcb947a96de5630b7e800

Time Taken 13.899532 ms

4

Figure 6: Figure 4 :



Select a Ciphertext File C:\Users\dell\Desktop\test\beep-07.mp3.encrypted

Decrypted Text AgACAAIAAgACAAIAAgACAAIAAgACAAIAAgACAAIAAgACAAIAAgACAAIAAgACAAIAA
gACAAIAAgACAAIAAgACAAIAAgACAAIAAgACAAIAAgACAAIAAgACAAIAAg
ACAAIAAgACAAIAAgACAAIAAgACAAIAAgACAAIAAgACAAIAAgACAAIAAgA
CAAIAAgACAAIAAgACAAIAAgACAAIAAgACAAIAAgACAAIAAgACAAIAAgA
CAAIAAgACAAIAAgACAAIAAgACAAIAAgACAAIAAA

Key 1234567890qwerty

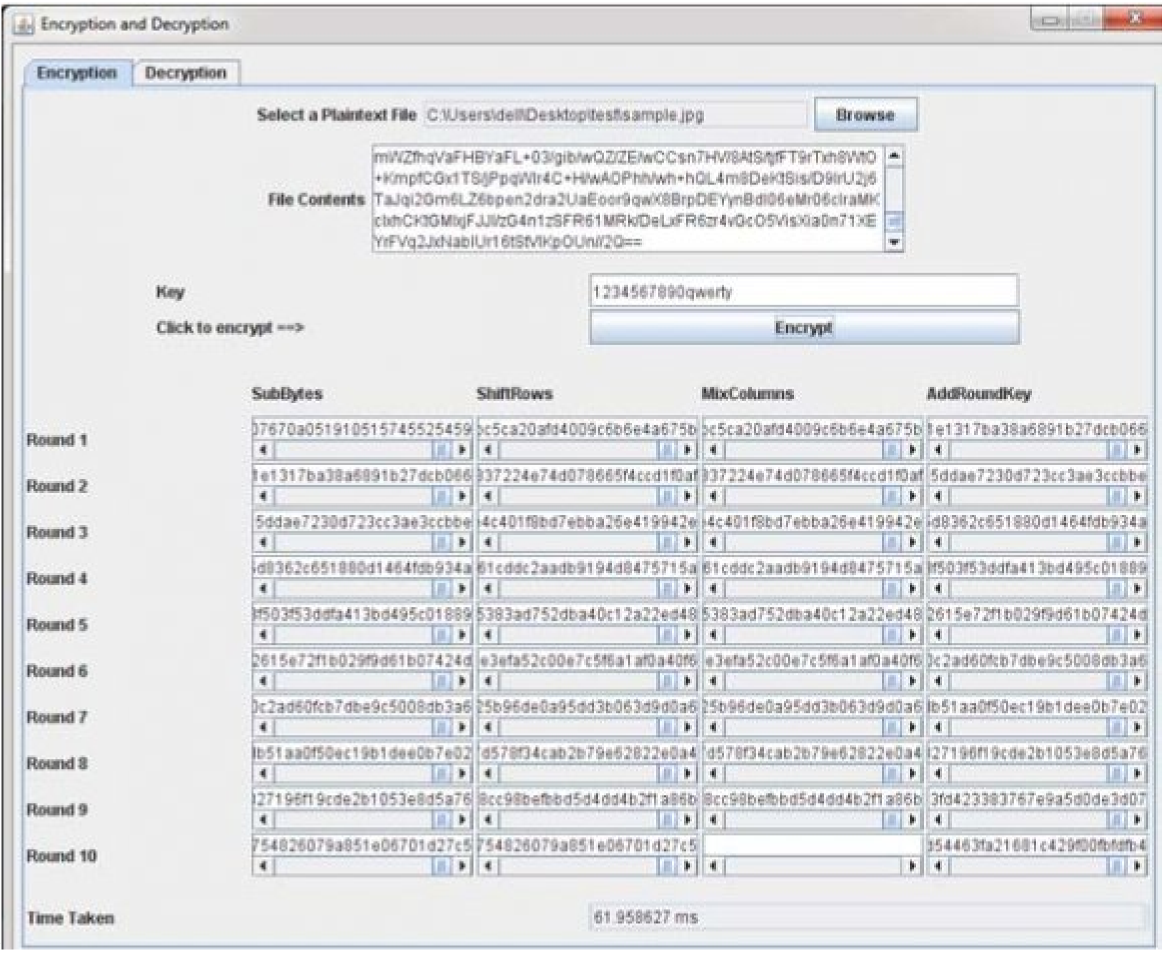
Click to decrypt ==>

	InvShiftRows	InvSubBytes	InvMixColumns	InvAddRoundKey
Round 1	6f04a74eb9614fafa14f7311b2	b30d9e3f1d889f33e9292b606	b30d9e3f1d889f33e9292b606	3857a2ecb45635b842dcfed0b
Round 2	3857a2ecb45635b842dcfed0b	967c4534f68bd119efa00c35e	967c4534f68bd119efa00c35e	75168fe22663f28af685c1694
Round 3	75168fe22663f28af685c1694	9470a4ff1bd3f761e7f7250c20	9470a4ff1bd3f761e7f7250c20	8845d5c3d70747ebd10dc7b0
Round 4	8845d5c3d70747ebd10dc7b0	3397fd313c0d68f7fc5138b5a3	3397fd313c0d68f7fc5138b5a3	81f8d90cdfac6c45def4aa9933
Round 5	81f8d90cdfac6c45def4aa9933	fe1a0f99caae53d66bab88191	fe1a0f99caae53d66bab88191	5d37d9e9971048fc102dfb8af0
Round 6	5d37d9e9971048fc102dfb8af0	5b2bdcf7c7ce50017fad4eb8d	5b2bdcf7c7ce50017fad4eb8d	7237c5a02fdef5bd3db74b148
Round 7	7237c5a02fdef5bd3db74b148	329056a9210133d49f614685	329056a9210133d49f614685	e239d0fd316322b67296fbccd
Round 8	e239d0fd316322b67296fbccd	a932a9780aff75d0804ca1fb5a	a932a9780aff75d0804ca1fb5a	0456a6f16c0fb08793176923c
Round 9	0456a6f16c0fb08793176923c	ff68da74af1f58016d2e630672	ff68da74af1f58016d2e630672	b59ca20fad4007dad6ec9f05b
Round 10	b59ca20fad4007dad6ec9f05b	b59ca20fad4007dad6ec9f05b		202020202020202020202020

Time Taken 20.183485 ms

6

Figure 7: Figure 6 :



Encryption and Decryption

Select a Plaintext File C:\Users\dell\Desktop\test\sample.jpg

File Contents

```
mWZfhqVaFHBYaFL+03/gib/wQZ/ZE/wCCsn7HV8AtS/tjfFT9rTxh8WtO
+KmpfCGx1TSjPpqWlr4C+H/wAOPhh/wh+hQL4m8DeKtSis/D9IrU2j6
TaJqi2Gm6LZ6bpen2dra2UaEoor9qwX8BrpDEYynBdI06eMr06clraMK
clxhCKtGMlxjFJJlzG4n1zSFR61MRk/DeLxFR6zr4vGcO5VisXia0n71XE
YrFVq2JxNabIUr16tStVKpOUn//2Q==
```

Key 1234567890qwerty

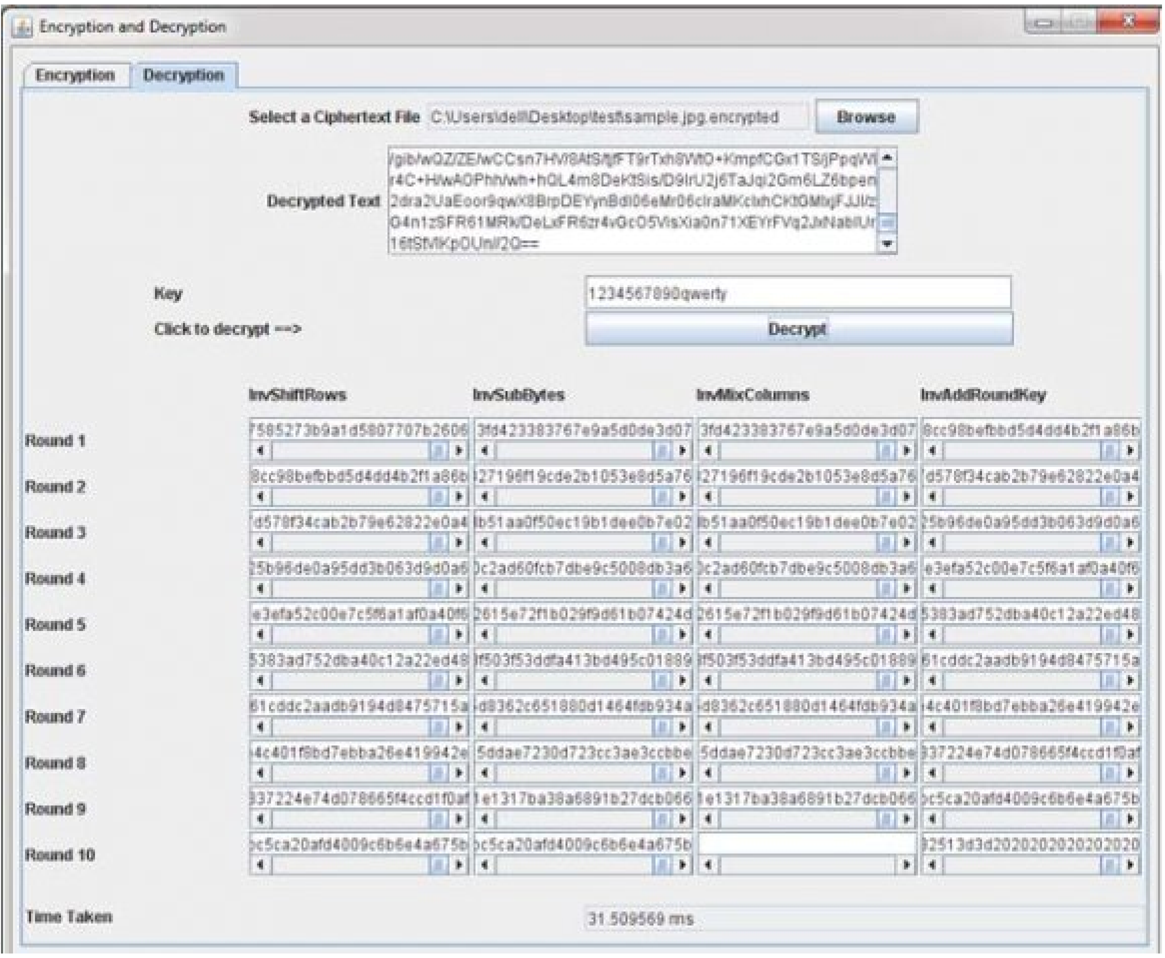
Click to encrypt ==>

	SubBytes	ShiftRows	MixColumns	AddRoundKey
Round 1	07670a051910515745525459	c5ca20afd4009c6b6e4a675b	c5ca20afd4009c6b6e4a675b	1e1317ba38a6891b27dcb066
Round 2	1e1317ba38a6891b27dcb066	337224e74d078665f4ccd1f0af	337224e74d078665f4ccd1f0af	5ddae7230d723cc3ae3ccbbe
Round 3	5ddae7230d723cc3ae3ccbbe	4c401f8bd7ebba26e419942e	4c401f8bd7ebba26e419942e	d8362c651880d1464fdb934a
Round 4	d8362c651880d1464fdb934a	61cddc2aadb9194d8475715a	61cddc2aadb9194d8475715a	f503f53ddfa413bd495c01889
Round 5	f503f53ddfa413bd495c01889	5383ad752dba40c12a22ed48	5383ad752dba40c12a22ed48	2615e72f1b029f9d61b07424d
Round 6	2615e72f1b029f9d61b07424d	e3efa52c00e7c5f6a1af0a40f6	e3efa52c00e7c5f6a1af0a40f6	0c2ad60fcb7dbe9c5008db3a6
Round 7	0c2ad60fcb7dbe9c5008db3a6	25b96de0a95dd3b063d9d0a6	25b96de0a95dd3b063d9d0a6	b51aa0f50ec19b1dee0b7e02
Round 8	b51aa0f50ec19b1dee0b7e02	d578f34cab2b79e62822e0a4	d578f34cab2b79e62822e0a4	27196f19cde2b1053e8d5a76
Round 9	27196f19cde2b1053e8d5a76	8cc98befbbd5d4dd4b2f1a86b	8cc98befbbd5d4dd4b2f1a86b	3fd423383767e9a5d0de3d07
Round 10	754826079a851e06701d27c5	754826079a851e06701d27c5		354463fa21681c429f00fbfdfb4

Time Taken 61.958627 ms

5

Figure 8: Figure 5 :



Select a Ciphertext File C:\Users\dell\Desktop\test\sample.jpg.encrypted

Decrypted Text

```
/gib/wQZ/ZE/wCCsn7HV8AtS/tjfFT9rTxh8WtO+KmpfCGx1TS/jPpqWI
r4C+H/wAOPhh/wh+hQL4m8DeKtSis/D9IrU2j6TaJqi2Gm6LZ6bpen
2dra2UaEoor9qwX8BrpDEYynBdl06eMr06clraMKclxhCKtGMlxjFJJl/z
G4n1zSFR61MRk/DeLxFR6zr4vGcO5VisXia0n71XEYrFVq2JxNablUr
16tSMKpOUnl/2Q==
```

Key 1234567890qwerty

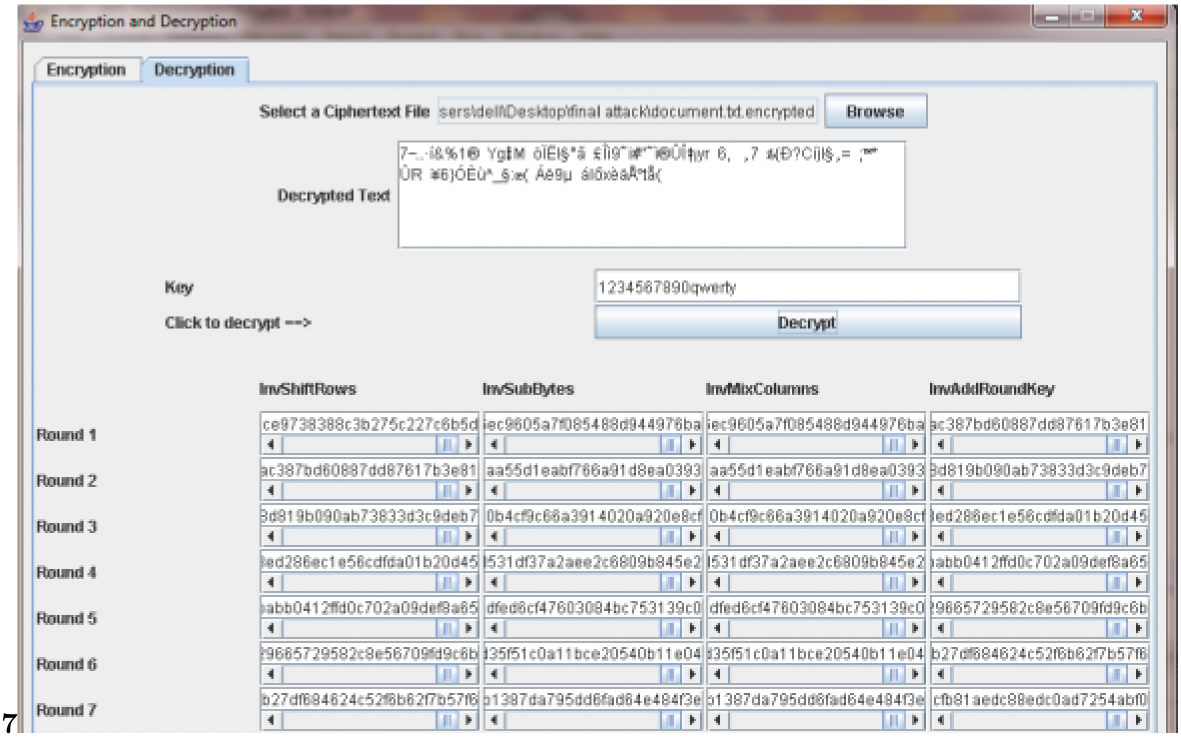
Click to decrypt -->

	InvShiftRows	InvSubBytes	InvMixColumns	InvAddRoundKey
Round 1	7585273b9a1d5807707b2606	3fd423383767e9a5d0de3d07	3fd423383767e9a5d0de3d07	8cc98befbbd5d4dd4b2f1a86b
Round 2	8cc98befbbd5d4dd4b2f1a86b	27196f19cde2b1053e8d5a76	27196f19cde2b1053e8d5a76	d578f34cab2b79e62822e0a4
Round 3	d578f34cab2b79e62822e0a4	b51aa0f50ec19b1dee0b7e02	b51aa0f50ec19b1dee0b7e02	25b96de0a95dd3b063d9d0a6
Round 4	25b96de0a95dd3b063d9d0a6	0c2ad60fcb7dbe9c5008db3a6	0c2ad60fcb7dbe9c5008db3a6	e3efa52c00e7c5f6a1af0a40f6
Round 5	e3efa52c00e7c5f6a1af0a40f6	2615e72f1b029f9d61b07424d	2615e72f1b029f9d61b07424d	5383ad752dba40c12a22ed48
Round 6	5383ad752dba40c12a22ed48	f503f53ddfa413bd495c01889	f503f53ddfa413bd495c01889	61cddc2aadb9194d8475715a
Round 7	61cddc2aadb9194d8475715a	d8362c651880d1464fdb934a	d8362c651880d1464fdb934a	4c401f8bd7ebba26e419942e
Round 8	4c401f8bd7ebba26e419942e	5ddae7230d723cc3ae3ccbbe	5ddae7230d723cc3ae3ccbbe	337224e74d078665f4ccd1f0af
Round 9	337224e74d078665f4ccd1f0af	1e1317ba38a6891b27dcb066	1e1317ba38a6891b27dcb066	c5ca20afd4009c6b6e4a675b
Round 10	c5ca20afd4009c6b6e4a675b	c5ca20afd4009c6b6e4a675b		2513d3d2020202020202020

Time Taken 31.509569 ms

89

Figure 9: Figure 8 :Fig. 9 .



Select a Ciphertext File sers\dell\Desktop\final attack\document.txt.encrypted

Decrypted Text

```
7–.·i&%1® Yg‡M öÏÉI§*ã ¢Ìî9˜¥ "®ÜÎ¶yr 6, ,7 ¥(Ð?Ci|I§,= ;™
ÚR ¥6)ÓÈù^_§:æ( Áè9µ á6xèåÅ*1å(
```

Key 1234567890qwerty

Click to decrypt -->

	InvShiftRows	InvSubBytes	InvMixColumns	InvAddRoundKey
Round 1	ce9738388c3b275c227c6b5d	ec9605a7f085488d944976ba	ec9605a7f085488d944976ba	ac387bd60887dd87617b3e81
Round 2	ac387bd60887dd87617b3e81	aa55d1eabf766a91d8ea0393	aa55d1eabf766a91d8ea0393	3d819b090ab73833d3c9deb7
Round 3	3d819b090ab73833d3c9deb7	0b4cf9c66a3914020a920e8cf	0b4cf9c66a3914020a920e8cf	ed286ec1e56cdfda01b20d45
Round 4	ed286ec1e56cdfda01b20d45	531df37a2aee2c6809b845e2	531df37a2aee2c6809b845e2	abb0412ffd0c702a09def8a65
Round 5	abb0412ffd0c702a09def8a65	dfed6cf47603084bc753139c0	dfed6cf47603084bc753139c0	9665729582c8e56709fd9c6b
Round 6	9665729582c8e56709fd9c6b	35f51c0a11bce20540b11e04	35f51c0a11bce20540b11e04	b27df684624c52f6b62f7b57f6
Round 7	b27df684624c52f6b62f7b57f6	1387da795dd6fad64e484f3e	1387da795dd6fad64e484f3e	cfb81aedc88edc0ad7254abf0

7

Figure 10: Figure 7 :

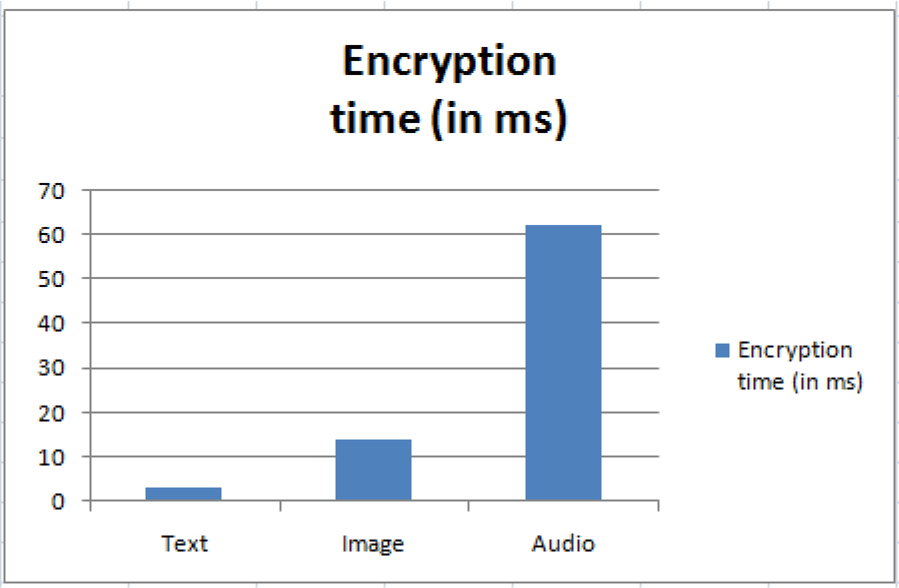


Figure 11:

9

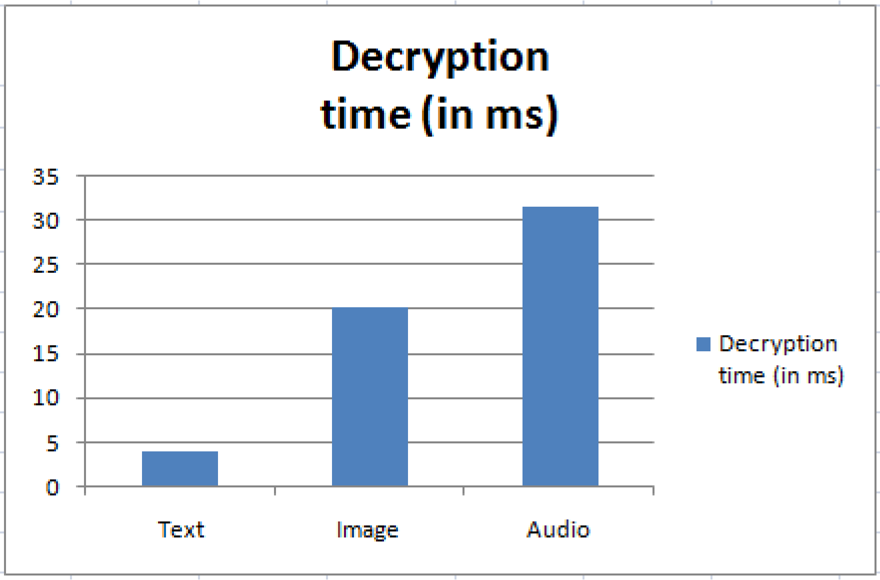


Figure 12: Figure 9 :

1

AES PARAMETERS	AES-128	AES-192	AES-256
Key Size (Bits)	128	192	256
Number of rounds	10	12	14
Plaintext box size (Bits)	128	128	128

Figure 13: Table 1 :

[Daemen and Rijmen] , J Daemen , V Rijmen . *Rijndael* (2) .

[Su et al. ()] ‘A high throughput low cost AES processor’. Chih-Pin Su , Tsung-Fu Lin , Chih-Tsun Huang , Cheng-Wen Wu . *IEEE Communications Magazine*, 2003.

[Dr et al. ()] ‘A Study of Encryption Algorithms AES, DES and RSA forSecurity’. Dr , Prerna Mahajan & Abhishek , Sachdeva . *Global Journal of Computer Science and Technology Network*, 2013. 13 p. .

[Hiremath and Suma] ‘Advanced Encryption Standard Implemented on FPGA’. S Hiremath , M S Suma . *IEEE Inter*,

[Jain et al. (2014)] ‘AES Algorithm Using 512 Bit Key Implementationfor Secure Communication’. Rishabh Jain , Rahul Jejurka2 , Shrikrishna Chopade , Someshwar Vaidya , Mahesh Sanap . *International Journal of Innovative Research in Computerand Communication Engineering* March 2014. 2 p. .

[Gnanambika et al. (2013)] ‘AES-128 Bit Algorithm Using Fully Pipelined Architecture for Secret Communication’. M Gnanambika , S Adilakshmi , Dr Noorbasha . *International Journal of Engineering Research and Applications* March -April 2013. 3 p. .

[Kuo and Verbauwhede (2001)] ‘Architectural optimization for a 1.82 Gbits/sec VLSIimple mentation of the AES Rijndael algorithm’. H Kuo , I Verbauwhede . *Proc. CHES*, (CHESParis, France) 2001. May 2001. p. .

[Khatri et al. (2012)] ‘Comparison of power consumption and strict avalanche criteria at encryption/Decryption side of Different AES standards’. Navraj Khatri , Rajeev Dhanda , Jagtar Singh . *International Journal Of Computational Engineering Research* August 2012. 2.

[Conf. Comp Elec Engin. (IECEE) (2009)] *Conf. Comp Elec Engin. (IECEE)*, Dec.2009. 02 p. .

[Stallings] *Cryptography and network security*, W Stallings .

[Soumya and Shyam Kishore (2013)] ‘Design and Implementation of Rijndael Encryption Algorithm Based on FPGA’. K Soumya , G Shyam Kishore . *International Journal of Computer Science and Mobile Computing* September 2013. 2 (9) p. .

[Ritika and Kuldeep] *Efficiency and Security of Data with Symmetric Encryption*, Chehal Ritika , Singh Kuldeep .

[Pahal and Kumar (2013)] ‘Efficient Implementation of AES’. Ritu Pahal , Vikas Kumar . *International Journal of Advanced Research in Computer Science and Software Engineering* July 2013. 3 p. .

[Federal Information Processing Standards Publication 197 (2001)] *Federal Information Processing Standards Publication 197*, Nov. 2001. (Advanced Encryption Standard (AES))

[Zhang and Parhi ()] ‘Implementation approaches for the advanced encryption standard algorithm’. Xinmiao Zhang , Keshab K Parhi . *IEEE Transactions*, 2002.

[Vinayak Bajirao Patil et al. (2013)] ‘Implementation of AES algorithm on ARM processor for wireless network’. Prof Vinayak Bajirao Patil , Dr , L Uttam , Pallavi Bombale , Hemant Dixit . *International Journal of Advanced Research in Computer and Communication Engineering* August 2013. 2 p. .

[Vinayak Bajirao Patil et al. (2013)] ‘Implementation of AES algorithm on ARM processor for wireless network’. Prof Vinayak Bajirao Patil , Dr , L Uttam , Pallavi Bombale , Hemant Dixit . *International Journal of Advanced Research in Computer and Communication Engineering* August 2013. 2 p. .

[Kaushik and Singhal (2012)] ‘Network Security Using Cryptographic Techniques’. Sumedha Kaushik , Ankur Singhal . *International Journal of Advanced Research in Computer Science and Software Engineering* December 2012. 2 p. .

[Debasis and Rajiv (2011)] ‘Programmable Cellular Automata Based Efficient Parallel AES Encryption Algorithm’. Das Debasis , Misra Rajiv . *International Journal of Network Security & Its Applications (IJNSA)* November 2011. 3 (6) p. 204.

[Xinjie et al. (2008)] ‘Robust First Two Rounds Access Driven Cache Timing Attack on AES’. Z Xinjie , W Tao , M Dong , Z Yuanyuan , L Zhaoyang . *IEEE InternationalConference on Computer Science and Software Engineering*, (Wuhan, Hubei, China) Dec. 2008. p. .

[Kocher ()] ‘Timing attacks on implementations of Diffie-Hellman, RSA, DSS and other systems’. P C Kocher . *16th Annual International Cryptology Conference on Advances in Cryptology (CRYPTO)*, (London UK) 1996. Springer-Verlag. p. .