RAMAIAH UNIVERSITY
OF APPLIED SCIENCES

Assignment

Course Code	**19CSC315A**
Course Name	**Information Security and Protection**
Programme	**B.Tech**
Department	**Computer Science and Engineering**
Faculty	**Engineering and Technology**

Name of the Student	**Deepak R**
Reg. No.	**18ETCS002041**
Semester/Year	**6th/2021**
Course Leader(s)	**Prof. N. D. Gangadhar**

Declaration Sheet			
Student Name	**Deepak R**		
Reg. No	**18ETCS002041**		
Programme	**B.Tech**	**Semester/Year**	**6th/2021**
Course Code	**19CSC315A**		
Course Title	**Information Security and Protection**		
Course Date		**to**	
Course Leader	**Prof. N. D. Gangadhar**		

Declaration

The assignment submitted herewith is a result of my own investigations and that I have conformed to the guidelines against plagiarism as laid out in the Student Handbook. All sections of the text and results, which have been obtained from other sources, are fully referenced. I understand that cheating and plagiarism constitute a breach of University regulations and will be dealt with accordingly.

Signature of the Student		**Date**	
Submission date stamp (by Examination & Assessment Section)			

Signature of the Course Leader and date	**Signature of the Reviewer and date**

Faculty of Engineering and Technology			
Ramaiah University of Applied Sciences			
Department	Computer Science and Engineering	Programme	B. Tech. in CSE
Semester/Batch	6/2018		
Course Code	19CSC315A	Course Title	Information Security and Protection
Course Leader	Prof. N. D. Gangadhar		

Marking Scheme		Marks		
		Max Marks	First Examiner Marks	Moderator
1.1	Algorithm developed and Keys used	10		
1.2	C/Python Program	05		
1.3	Validation using Test Cases	05		
1.4	Conclusion	05		
	Part-A Max Marks	25		

Assignment-02			
Reg.No.	18ETCS002041	Name of Student	Deepak R

Course Marks Tabulation				
Assignment	First Examiner	Remarks	Moderator	Remarks
1				
Marks (out of 25)				

Encryption / Decryption (25 Marks)

Symmetric encryption is a type of encryption where only one secret key is used to both encrypt and decrypt electronic information. The entities communicating via symmetric encryption must exchange the key so that it can be used in the decryption process. It is an old and best-known technique that uses a secret key that can either be a number, a word or a string of random letters. Some of the encryption algorithms include Blowfish, AES, RC4, DES, RC5 etc. Select any algorithm and perform encryption/decryption on any sample text. Use any appropriate key size. Your report should include the following:

Solution for 1.1 Algorit-hm developed and Keys used

Encryption Steps of the Algorithm

The algorithm includes the following steps:

1. The algorithm takes the 64-bit plain text as input.
2. The text is parsed into a function called the Initial Permutation (IP) function.
3. The initial permutation (IP) function breaks the plain text into the two halves of the permuted block. These two blocks are known as Left Plain Text (LPT) and Right Plain Text (RPT).
4. The 16 round encryption process is performed on both blocks LPT and RPT. The encryption process performs the following:

1. Key Transformation
   2. Expansion Permutation
   3. S-Box Permutation
   4. P-Box Permutation
   5. XOR and Swap

5. After performing the encryption process, the LPT and RPT block are rejoined. After that, the Final Permutation (FP) is applied to the combined block.
6. Finally, we get the 64-bit ciphertext of the plaintext.

Decryption Step of the Algorithm

For decryption of the ciphertext, we use the same algorithm but in reverse order (step 4) of 16 round keys.

Generating Keys

The algorithm performs 16 rounds of encryption and for each round, a unique key is generated. Before moving to the steps, it is important to know that in plaintext the bits are labeled from 1 to 64 where 1 is the most significant bit and 64 is the least significant bit. The process of generating keys are as follows:

1. First, we compress and transpose the given 64-bit key into a 48-bit keys by using the table given below:

```
int pc1[56] = {
 57,49,41,33,25,17,9,
 1,58,50,42,34,26,18,
 10,2,59,51,43,35,27,
 19,11,3,60,52,44,36,
 63,55,47,39,31,23,15,
 7,62,54,46,38,30,22,
 14,6,61,53,45,37,29,
 21,13,5,28,20,12,4
};
```

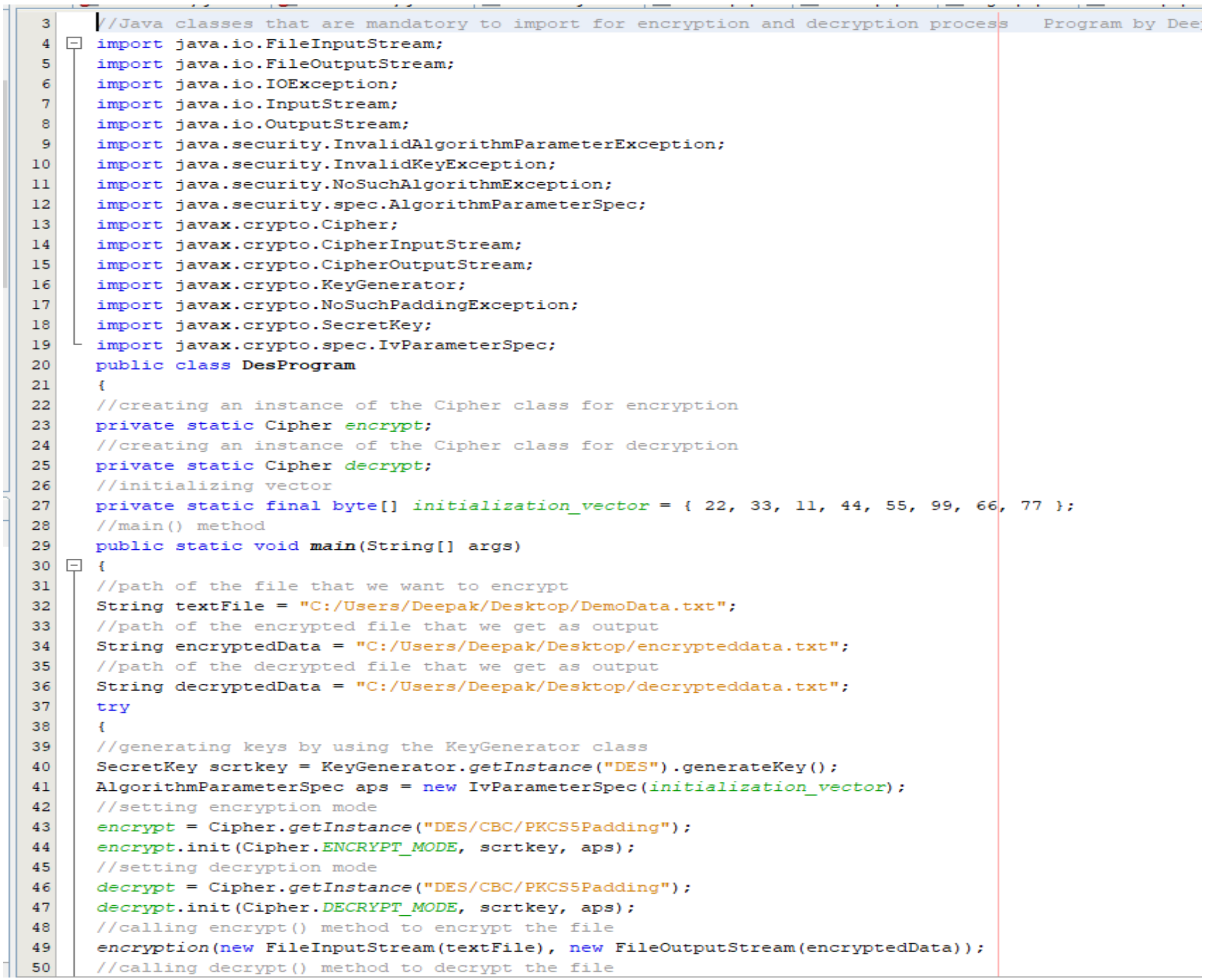
2. Separate the result into two equal parts i.e. C and D.

3. The part C and D are left-shifted circularly. For encryption, the 1st, 2nd, 9th, and 16th round is responsible that shifts a bit to the left by 1 bit, circularly. All the rest rounds are shifted to the left by 2-bit circularly.

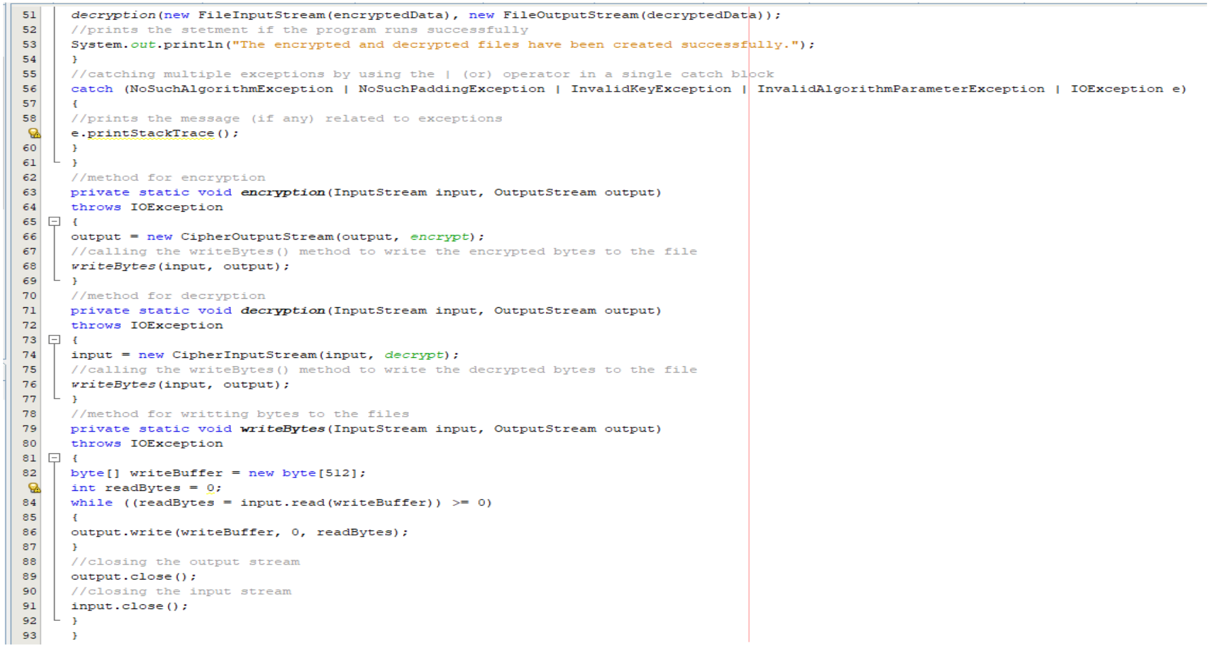
4. After that, the result is compressed to 48-bits with the help of the following table.

```
int pc2[48] = {
 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
};
```

Solution for 1.2 Program Source Code



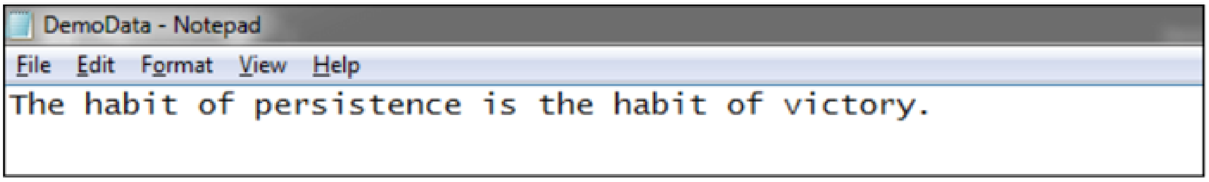
```java
//Java classes that are mandatory to import for encryption and decryption process    Program by Dee
import java.io.FileInputStream;
import java.io.FileOutputStream;
import java.io.IOException;
import java.io.InputStream;
import java.io.OutputStream;
import java.security.InvalidAlgorithmParameterException;
import java.security.InvalidKeyException;
import java.security.NoSuchAlgorithmException;
import java.security.spec.AlgorithmParameterSpec;
import javax.crypto.Cipher;
import javax.crypto.CipherInputStream;
import javax.crypto.CipherOutputStream;
import javax.crypto.KeyGenerator;
import javax.crypto.NoSuchPaddingException;
import javax.crypto.SecretKey;
import javax.crypto.spec.IvParameterSpec;
public class DesProgram
{
//creating an instance of the Cipher class for encryption
private static Cipher encrypt;
//creating an instance of the Cipher class for decryption
private static Cipher decrypt;
//initializing vector
private static final byte[] initialization_vector = { 22, 33, 11, 44, 55, 99, 66, 77 };
//main() method
public static void main(String[] args)
{
//path of the file that we want to encrypt
String textFile = "C:/Users/Deepak/Desktop/DemoData.txt";
//path of the encrypted file that we get as output
String encryptedData = "C:/Users/Deepak/Desktop/encrypteddata.txt";
//path of the decrypted file that we get as output
String decryptedData = "C:/Users/Deepak/Desktop/decrypteddata.txt";
try
{
//generating keys by using the KeyGenerator class
SecretKey scrtkey = KeyGenerator.getInstance("DES").generateKey();
AlgorithmParameterSpec aps = new IvParameterSpec(initialization_vector);
//setting encryption mode
encrypt = Cipher.getInstance("DES/CBC/PKCS5Padding");
encrypt.init(Cipher.ENCRYPT_MODE, scrtkey, aps);
//setting decryption mode
decrypt = Cipher.getInstance("DES/CBC/PKCS5Padding");
decrypt.init(Cipher.DECRYPT_MODE, scrtkey, aps);
//calling encrypt() method to encrypt the file
encryption(new FileInputStream(textFile), new FileOutputStream(encryptedData));
//calling decrypt() method to decrypt the file
```



```java
decryption(new FileInputStream(encryptedData), new FileOutputStream(decryptedData));
//prints the stetment if the program runs successfully
System.out.println("The encrypted and decrypted files have been created successfully.");
}
//catching multiple exceptions by using the | (or) operator in a single catch block
catch (NoSuchAlgorithmException | NoSuchPaddingException | InvalidKeyException | InvalidAlgorithmParameterException | IOException e)
{
//prints the message (if any) related to exceptions
e.printStackTrace();
}
}
//method for encryption
private static void encryption(InputStream input, OutputStream output)
throws IOException
{
output = new CipherOutputStream(output, encrypt);
//calling the writeBytes() method to write the encrypted bytes to the file
writeBytes(input, output);
}
//method for decryption
private static void decryption(InputStream input, OutputStream output)
throws IOException
{
input = new CipherInputStream(input, decrypt);
//calling the writeBytes() method to write the decrypted bytes to the file
writeBytes(input, output);
}
//method for writting bytes to the files
private static void writeBytes(InputStream input, OutputStream output)
throws IOException
{
byte[] writeBuffer = new byte[512];
int readBytes = 0;
while ((readBytes = input.read(writeBuffer)) >= 0)
{
output.write(writeBuffer, 0, readBytes);
}
//closing the output stream
output.close();
//closing the input stream
input.close();
}
}
```

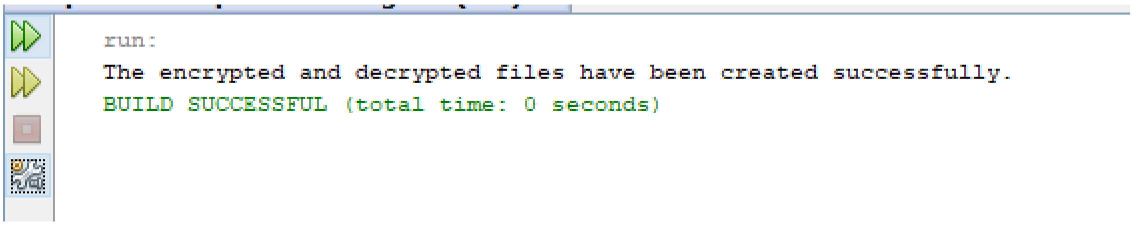
Solution for 1.3 Validation using Test Cases

DemoData.txt



```
The habit of persistence is the habit of victory.
```

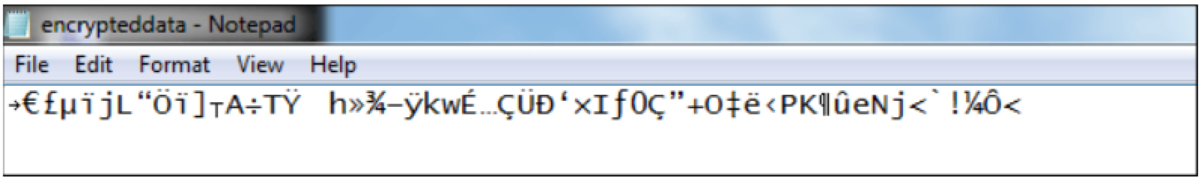
Output



```
run:
The encrypted and decrypted files have been created successfully.
BUILD SUCCESSFUL (total time: 0 seconds)
```

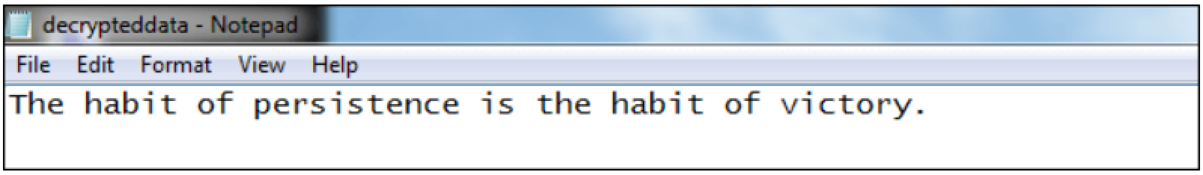
When we run the above program, it generates the two files encrypteddata.txt and deecrypteddata.txt at the specified location. Let's see what inside the encrypted and decrypted file.

encrypteddata.txt



```
→€£µïjL“Öï]┬A÷TŸ  h»¾–ÿkwÉ…ÇÜÐ‘xIƒ0Ç”+O‡ë‹PK¶ûeNj<`!¼Ô<
```

deecrypteddata.txt



```
The habit of persistence is the habit of victory.
```

we see that data is decrypted into the same text as we had written in the DemoData.txt file.

Solution for 1.4 Conclusion

DES is broken; however, 3DES is currently considered a secure cipher. DES does have the desirable properties of confusion and diffusion: each bit of ciphertext is based upon multiple bits of the key and changing a single bit of plaintext changes, on average, half of the bits of ciphertext.

Due to its Feistel structure and uncomplicated logic, DES is relatively easy to implement. However, it uses eight distinct S-Boxes, which increases its footprint (AES uses a single S-Box).

The main disadvantage to DES is that it is broken using brute-force search. However, using 3DES mitigates this issue at the cost of increasing execution time.

DES is also vulnerable to attacks using linear cryptanalysis. However, it takes 247 known plaintexts to break DES in this manner.

BIBILOGRAPHY

1. BishopSullivanRuppel-Computer Security_ Art And Science (2019) Text Book
2. WhitmanMattord-Principles of Information Security-Cengage Learning (6ed, 2017) Text Book