Batch: A3 Roll No: 16010121045

Experiment No. 2

Title: Implementation of Symmetric key cryptography algorithm (DES/AES)

Objective:

Implement the Symmetric key cryptography algorithm (DES/AES) and understand various execution steps in detail

Expected Outcome of Experiment:

CO	Outcome
CO2	Demonstrate various cryptographic algorithms for securing systems

Books/ Journals/ Websites referred:

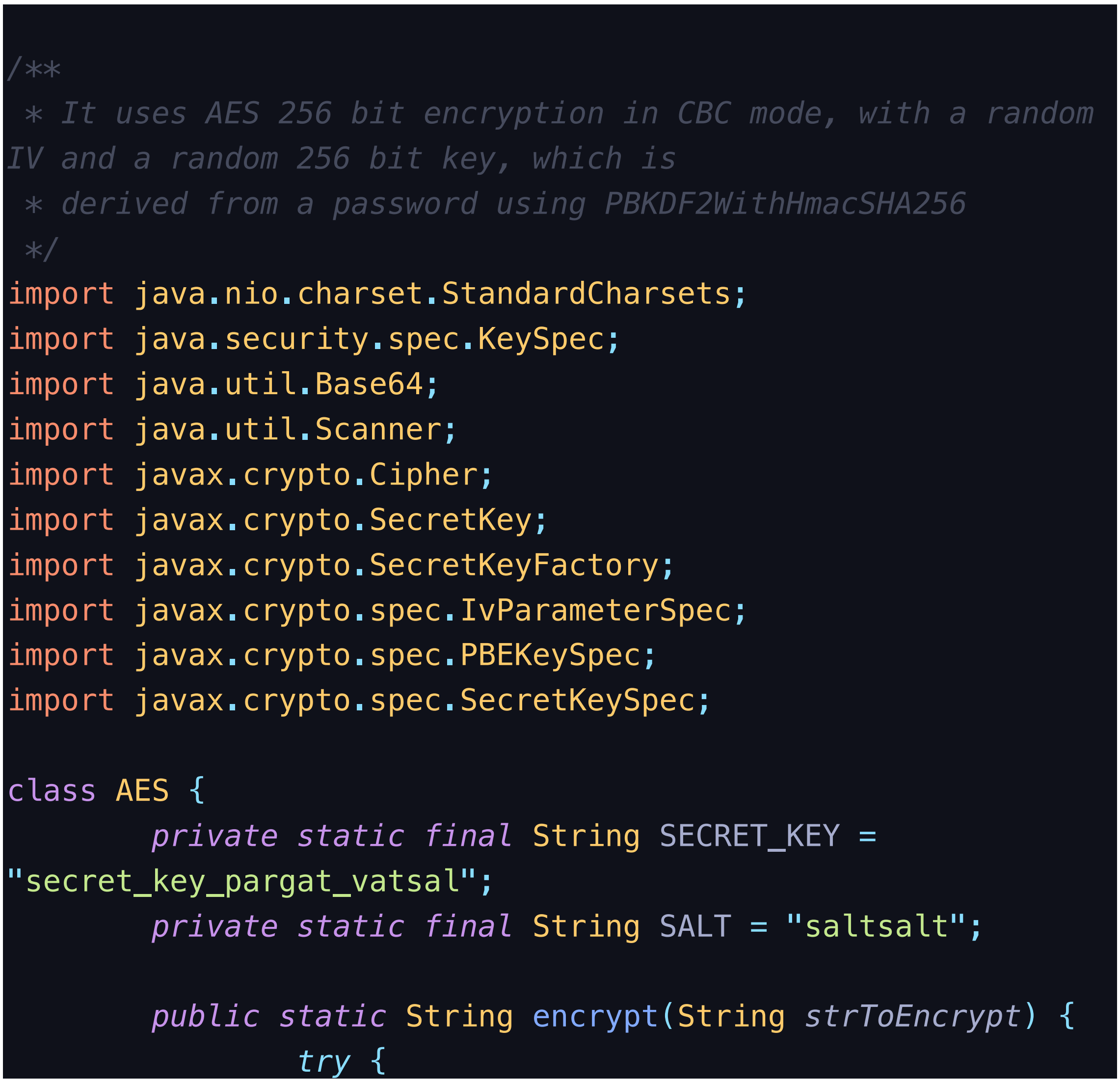
Abstract:-

Symmetric key cryptography

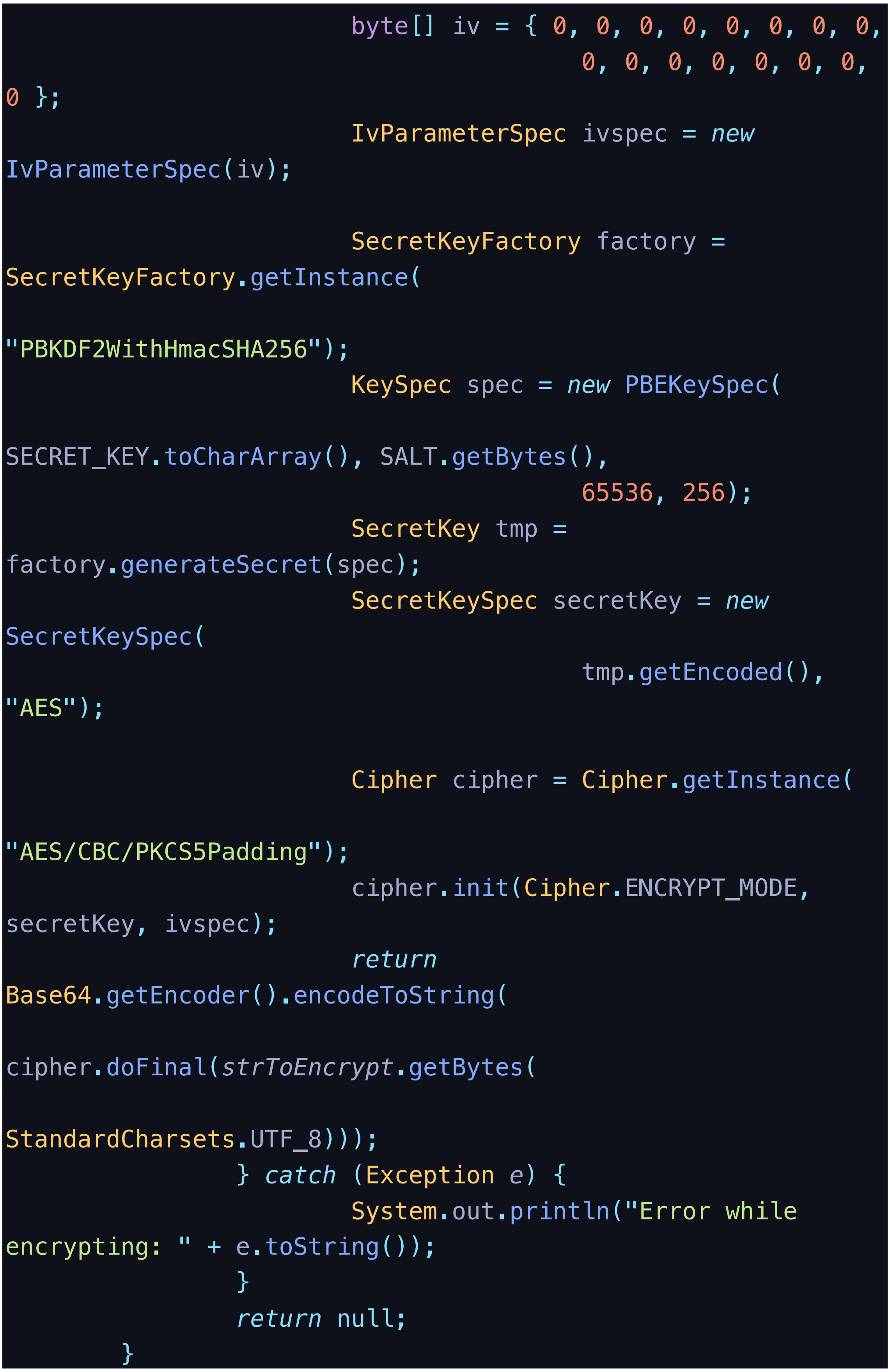
Related Theory: -

- *Symmetric key cryptography concepts : Fiestel and non-fiestel ciphers, confusion, diffusion*
- *Basic structure of assigned algorithm[AES/DES]*
- *Plaintext size, keysize, number of rounds, typical use cases*

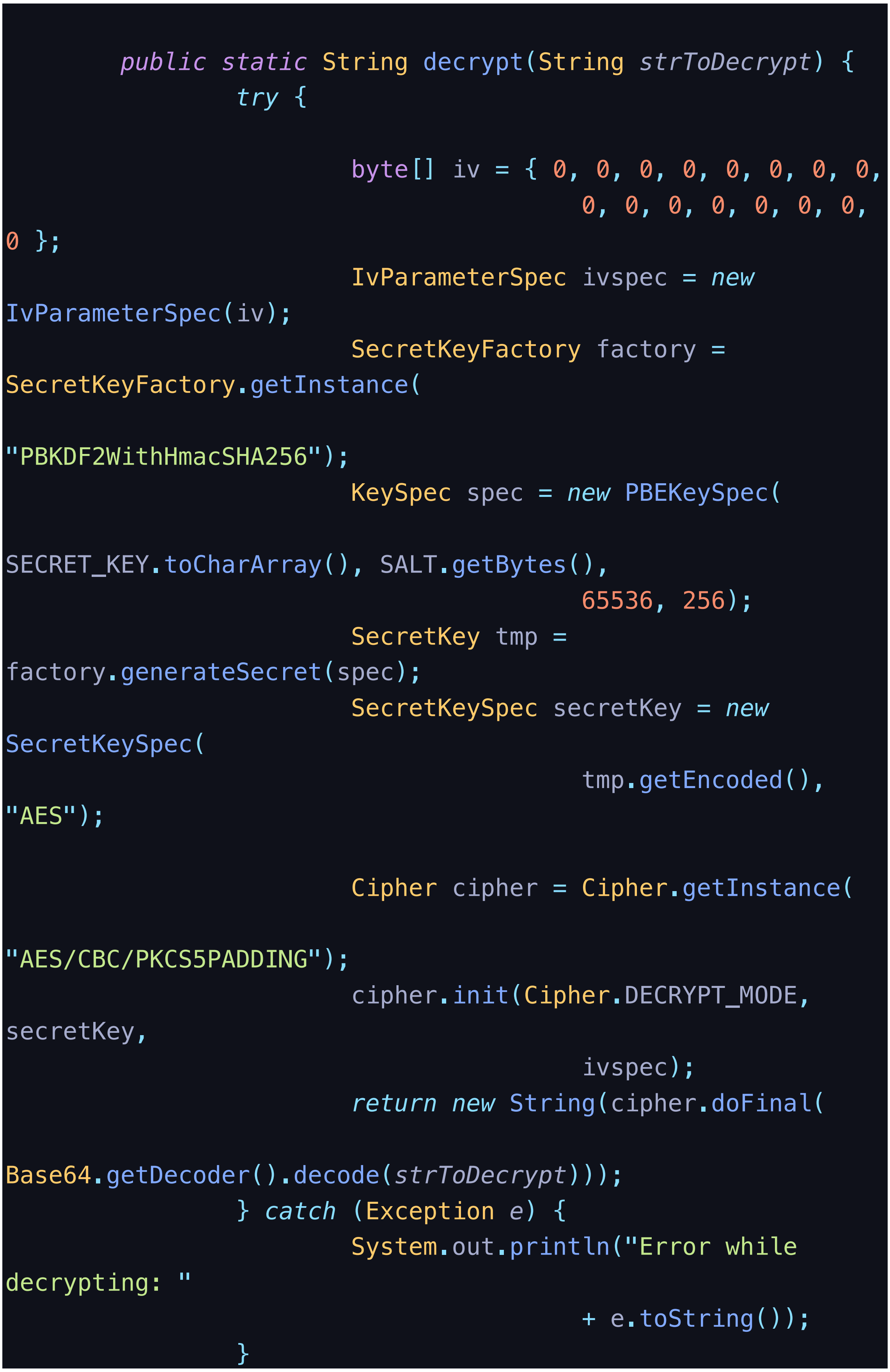
Program:



```
/**
 * It uses AES 256 bit encryption in CBC mode, with a random
IV and a random 256 bit key, which is
 * derived from a password using PBKDF2WithHmacSHA256
 */
import java.nio.charset.StandardCharsets;
import java.security.spec.KeySpec;
import java.util.Base64;
import java.util.Scanner;
import javax.crypto.Cipher;
import javax.crypto.SecretKey;
import javax.crypto.SecretKeyFactory;
import javax.crypto.spec.IvParameterSpec;
import javax.crypto.spec.PBEKeySpec;
import javax.crypto.spec.SecretKeySpec;

class AES {
        private static final String SECRET_KEY =
"secret_key_pargat_vatsal";
        private static final String SALT = "saltsalt";

        public static String encrypt(String strToEncrypt) {
                try {
```

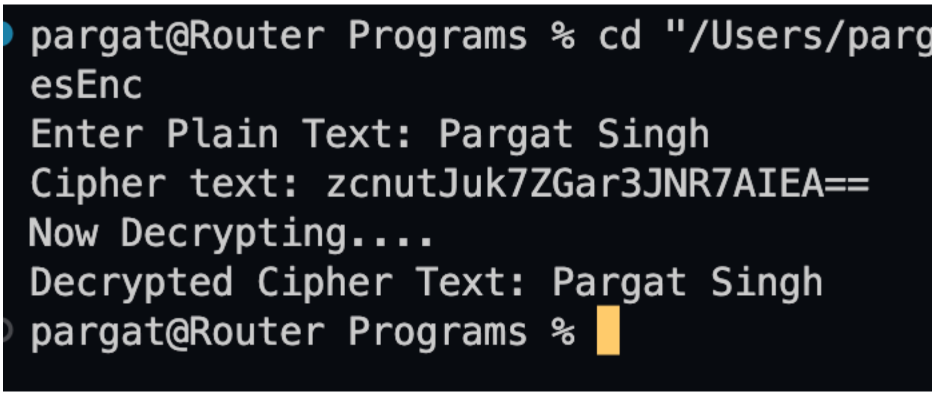


```
                        byte[] iv = { 0, 0, 0, 0, 0, 0, 0, 0,
                                        0, 0, 0, 0, 0, 0, 0,
0 };
                        IvParameterSpec ivspec = new
IvParameterSpec(iv);

                        SecretKeyFactory factory =
SecretKeyFactory.getInstance(

"PBKDF2WithHmacSHA256");
                        KeySpec spec = new PBEKeySpec(

SECRET_KEY.toCharArray(), SALT.getBytes(),
                                        65536, 256);
                        SecretKey tmp =
factory.generateSecret(spec);
                        SecretKeySpec secretKey = new
SecretKeySpec(
                                        tmp.getEncoded(),
"AES");

                        Cipher cipher = Cipher.getInstance(

"AES/CBC/PKCS5Padding");
                        cipher.init(Cipher.ENCRYPT_MODE,
secretKey, ivspec);
                        return
Base64.getEncoder().encodeToString(

cipher.doFinal(strToEncrypt.getBytes(

StandardCharsets.UTF_8)));
                } catch (Exception e) {
                        System.out.println("Error while
encrypting: " + e.toString());
                }
                return null;
        }
```



```
        public static String decrypt(String strToDecrypt) {
                try {

                        byte[] iv = { 0, 0, 0, 0, 0, 0, 0, 0,
                                        0, 0, 0, 0, 0, 0, 0,
0 };
                        IvParameterSpec ivspec = new
IvParameterSpec(iv);
                        SecretKeyFactory factory =
SecretKeyFactory.getInstance(

"PBKDF2WithHmacSHA256");
                        KeySpec spec = new PBEKeySpec(

SECRET_KEY.toCharArray(), SALT.getBytes(),
                                        65536, 256);
                        SecretKey tmp =
factory.generateSecret(spec);
                        SecretKeySpec secretKey = new
SecretKeySpec(
                                        tmp.getEncoded(),
"AES");

                        Cipher cipher = Cipher.getInstance(

"AES/CBC/PKCS5PADDING");
                        cipher.init(Cipher.DECRYPT_MODE,
secretKey,
                                        ivspec);
                        return new String(cipher.doFinal(

Base64.getDecoder().decode(strToDecrypt)));
                } catch (Exception e) {
                        System.out.println("Error while
decrypting: "
                                        + e.toString());
                }
                return null;
        }
}

public class aesEnc {
        public static void main(String[] args) {
                Scanner sc = new Scanner(System.in);
                System.out.print("Enter Plain Text: ");
                String plaintext = sc.nextLine();
                String ciphertext = AES.encrypt(plaintext);
                String decryptedText =
AES.decrypt(ciphertext);
                System.out.println("Cipher text: " +
ciphertext);
                System.out.println("Now Decrypting....");
                System.out.println("Decrypted Cipher Text: "
+ decryptedText);
        }
}
```

Output Screenshots:



```
pargat@Router Programs % cd "/Users/parg
esEnc
Enter Plain Text: Pargat Singh
Cipher text: zcnutJuk7ZGar3JNR7AIEA==
Now Decrypting....
Decrypted Cipher Text: Pargat Singh
pargat@Router Programs %
```

Conclusion:-

Successfully implemented the given program.

Postlab:

1. **Compare and contrast AES/DES**

AES	DES
AES stands for Advanced Encryption Standard.	DES stands for Data Encryption Standard.
Key length varies from 128 bits, 192 bits to 256 bits.	Key length is of 56 bits.
Rounds per key length: • 128 bits - 10 • 192 bits - 12 • 256 bits - 14	16 rounds of identical operations.
AES structure is based on substitution-permutation network.	DES structure is based on Feistal network.
The operation rounds involved in AES encryption are Byte Substitution, Shift Row, Mix Column, and Key Addition.	Expansion, XOR operation with round key, Substitution, and Permutation are the rounds used in DES encryption
AES can encrypt 128 bits of plain text.	DES can encrypt 64 bits of plain text.
AES is derived from Square cipher.	DES is derived from Lucifer cipher.
AES was designed by Vincent Rijmen and Joan Daemen.	DES was designed by IBM.
No known attacks.	Brute-force, Linear crypt-analysis and Differential crypt-analysis.
AES can encrypt plain text up to 128 bits.	DES can encrypt 64 bits of plain text.

2. Comment on strengths and weaknesses of symmetric key cryptosystem.

Strengths

- A symmetric cryptosystem is faster.
- In Symmetric Cryptosystems, encrypted data can be transferred on the link even if there is a possibility that the data will be intercepted. Since there is no key transmitted with the data, the chances of data being decrypted are null.
- A symmetric cryptosystem uses password authentication to prove the receiver's identity.
- A system only which possesses the secret key can decrypt a message.

Weaknesses

- Symmetric cryptosystems have a problem of key transportation. The secret key is to be transmitted to the receiving system before the actual message is to be transmitted. Every means of electronic communication is insecure as it is impossible to guarantee that no one will be able to tap communication channels. So the only secure way of exchanging keys would be exchanging them personally.
- Cannot provide digital signatures that cannot be repudiated