

END TERM PROJECT OF COMPUTER NETWORKING



Shubham Jindal
19223086
MCA(III sem)
NIT,Raipur

Symmetric Key Cryptography

Abstract

In this project ,I first explain the concept Cryptography and how it works. Cryptography is a method of storing and transmitting data in a particular form.It ensures that only the person for whom the message is intended can read the message.

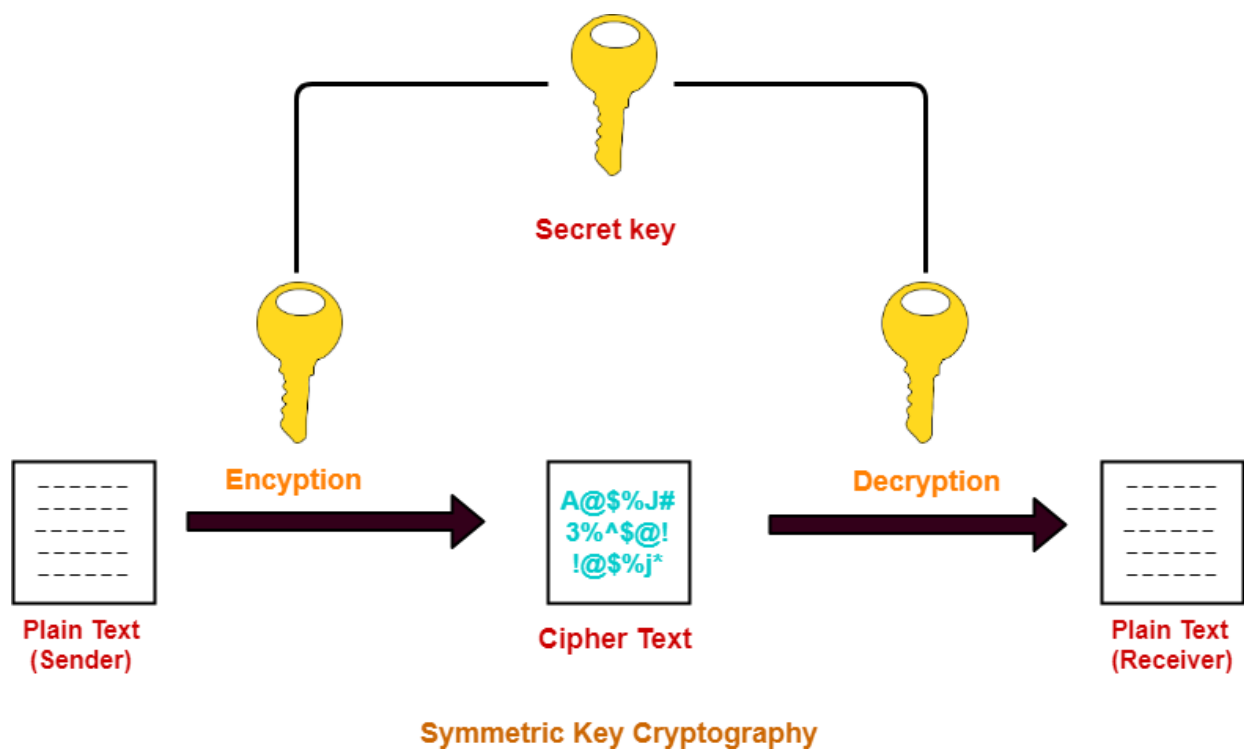
After that explain Cryptographic techniques i.e two techniques Symmetric and Asymmetric in brief .

After that explain the Symmetric key Cryptography and how it works to encrypt and decrypt the plain text to cipher text using AES algorithm (128 bit).

Also implementation it in java programming languages to how it works in real world .For Example Whatsapp
Used the cryptographic to convert plain text to cipher text.

Introduction

In Symmetric key Cryptographic ,Both sender and receiver use a common key to encrypt and decrypt the message.This secret key is known only to the sender and to the receiver.It is also called as **secret key cryptography**.



Problem Statement

We have to convert plain text that sends from sender to cipher text (i.e encryption) and sends the ciphertext through the Communication media .So,no Unauthorized person can read and understand our data and then decryption the ciphertext into plain text at receiver side
Whole process of encryption and decryption is done with the help of a single secret key.

I tried this problem Block algorithm and AES (Advanced Encryption Standard).

Example And Solution

The most commonly used symmetric [algorithms](#) are AES-128, AES-192, and AES-256

Types of Symmetric Encryption

- **Block algorithms** are used to encrypt blocks of electronic data. Specified set lengths of bits are altered, while continuing to use the designated private key. This key is then used for each block. When network stream data is being encrypted, the encryption system retains the data in its memory components while waiting for the complete blocks. The time in which the system waits can lead to a certain security gap, and may undermine data security and integrity.

The solution to the problem includes a process where the block of data can be decreased and merged with preceding encrypted data block contents, until the rest of the blocks arrive. This process is known as feedback. When the entire block is received, only then is it encrypted.

- **Stream algorithms** are not retained in the encryption system's memory, but arrive in data stream algorithms. This type of procedure is considered somewhat safer, since a disk or system is not retaining the data without encryption in the memory components.

AES(Advanced Encryption Standard)

The features of AES are as follows –

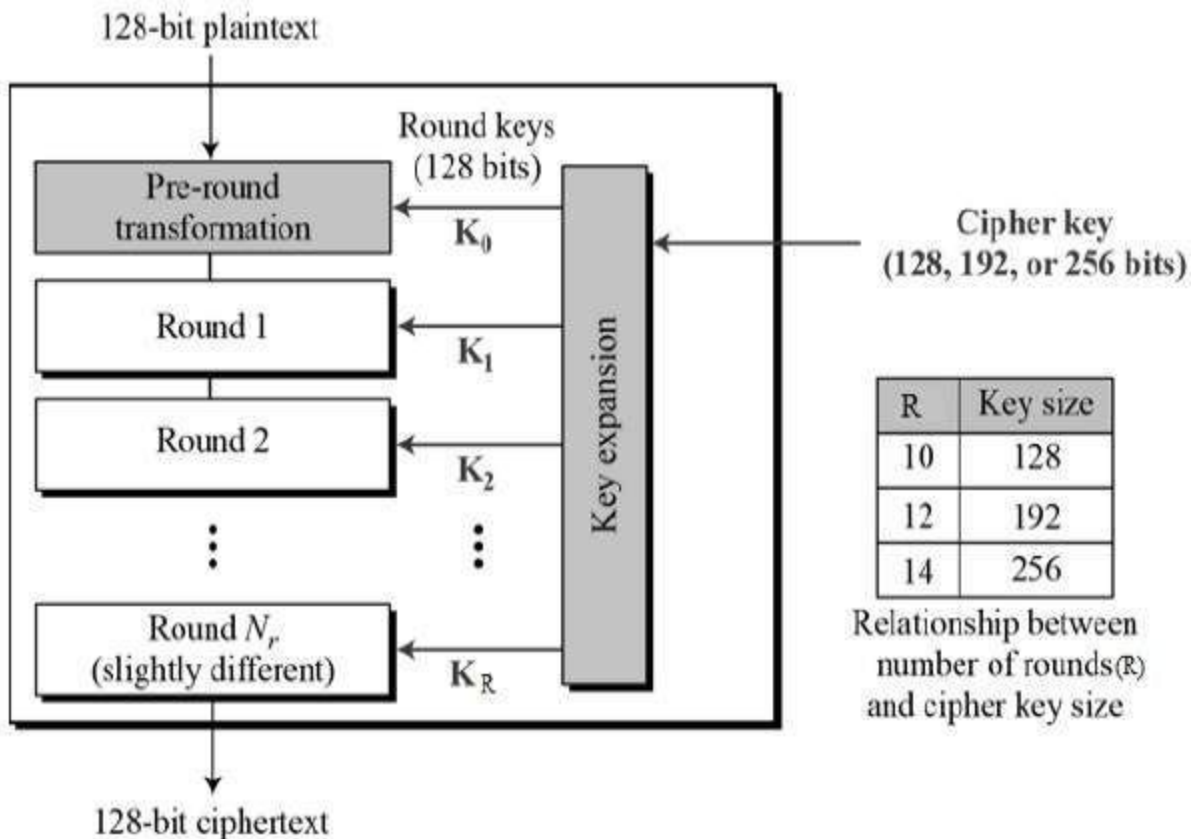
- Symmetric key symmetric block cipher
- 128-bit data, 128/192/256-bit keys
- Stronger and faster than Triple-DES
- Provide full specification and design details
- Software implementable in C and Java

Operation of AES

AES is an iterative rather than Feistel cipher. It is based on 'substitution–permutation network'. It comprises a series of linked operations, some of which involve replacing inputs by specific outputs (substitutions) and others involve shuffling bits around (permutations).

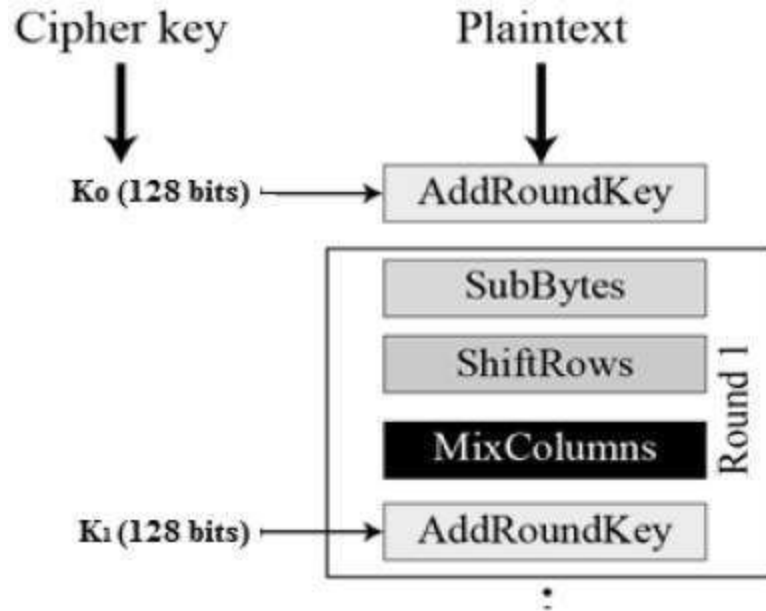
Interestingly, AES performs all its computations on bytes rather than bits. Hence, AES treats the 128 bits of a plaintext block as 16 bytes. These 16 bytes are arranged in four columns and four rows for processing as a matrix –

Unlike DES, the number of rounds in AES is variable and depends on the length of the key. AES uses 10 rounds for 128-bit keys, 12 rounds for 192-bit keys and 14 rounds for 256-bit keys. Each of these rounds uses a different 128-bit round key, which is calculated from the original AES key.



Encryption Process

Here, we restrict to description of a typical round of AES encryption. Each round comprises four sub-processes. The first round process is depicted below -



Byte Substitution (SubBytes)

The 16 input bytes are substituted by looking up a fixed table (S-box) given in design. The result is in a matrix of four rows and four columns.

Shiftrows

Each of the four rows of the matrix is shifted to the left. Any entries that 'fall off' are re-inserted on the right side of the row. Shift is carried out as follows –

- First row is not shifted.
- Second row is shifted one (byte) position to the left.
- Third row is shifted two positions to the left.
- Fourth row is shifted three positions to the left.
- The result is a new matrix consisting of the same 16 bytes but shifted with respect to each other.

MixColumns

Each column of four bytes is now transformed using a special mathematical function. This function takes as input the four bytes of one column and outputs four completely new bytes, which replace the original column. The result is another new matrix

consisting of 16 new bytes. It should be noted that this step is not performed in the last round.

Addroundkey

The 16 bytes of the matrix are now considered as 128 bits and are XORed to the 128 bits of the round key. If this is the last round then the output is the ciphertext. Otherwise, the resulting 128 bits are interpreted as 16 bytes and we begin another similar round.

Decryption Process

The process of decryption of an AES ciphertext is similar to the encryption process in the reverse order. Each round consists of the four processes conducted in the reverse order –

- Add round key
- Mix columns
- Shift rows
- Byte substitution

Since sub-processes in each round are in reverse manner, unlike for a Feistel Cipher, the encryption and decryption algorithms need to be separately implemented, although they are very closely related.

Implementation

We use java programming language ,AES (256 -bit)
Algorithm to implementation the problem.

```
// Java program to generate a symmetric key
import java.security.SecureRandom;
import javax.crypto.KeyGenerator;
import javax.crypto.SecretKey;
import javax.xml.bind.DatatypeConverter;

public class Symmetric
{
```



```
public static final String AES = "AES";

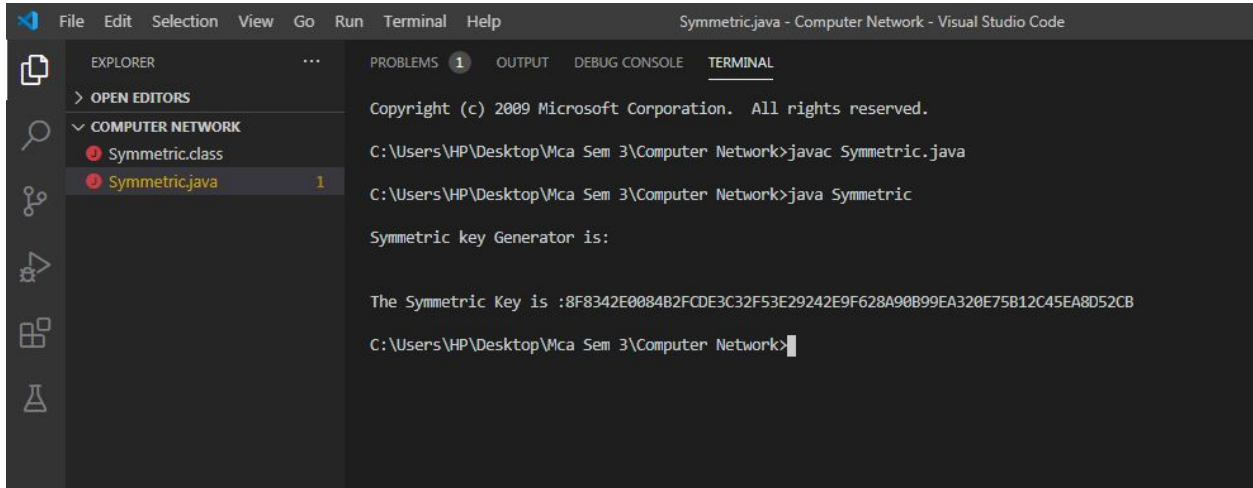
// Function to create a secret key
public static SecretKey createAESKey() throws Exception
{
    // Creating a new instance of SecureRandom class.
    SecureRandom securerandom = new SecureRandom();

    // Passing the string to KeyGenerator
    KeyGenerator keygenerator = KeyGenerator.getInstance(AES);

    // Initializing the KeyGenerator with 256 bits.
    keygenerator.init(256, securerandom);
    SecretKey key = keygenerator.generateKey();
    return key;
}

// Driver code
public static void main(String args[]) throws Exception
{
    SecretKey Symmetrickey = createAESKey();
    System.out.println("\nSymmetric key Generator is:\n");
    System.out.print("The Symmetric Key is :"+
DatatypeConverter.printHexBinary(Symmetrickey.getEncoded()));
}
}
```

Output-1



The screenshot shows the Visual Studio Code interface with the 'TERMINAL' tab active. The terminal output displays the execution of a Java program named 'Symmetric.java'. The program generates a symmetric key and prints it to the console.

```
Copyright (c) 2009 Microsoft Corporation. All rights reserved.

C:\Users\HP\Desktop\Mca Sem 3\Computer Network>javac Symmetric.java

C:\Users\HP\Desktop\Mca Sem 3\Computer Network>java Symmetric

Symmetric key Generator is:

The Symmetric Key is :8F8342E0084B2FCDE3C32F53E29242E9F628A90B99EA320E75B12C45EA8D52CB

C:\Users\HP\Desktop\Mca Sem 3\Computer Network>
```

```
// Java program to implement the encryption and decryption

import java.security.SecureRandom;
import java.util.Scanner;
import javax.crypto.Cipher;
import javax.crypto.KeyGenerator;
import javax.crypto.SecretKey;
import javax.crypto.spec.IvParameterSpec;
import javax.xml.bind.DatatypeConverter;

// Creating the symmetric class which implements the
Encryption_and_decryption
public class Encryption_and_decryption {

    private static final String AES = "AES";

    // We are using a Block cipher(CBC mode)
    private static final String AES_CIPHER_ALGORITHM =
"AES/CBC/PKCS5PADDING";

    private static Scanner message;
```

```
// Function to create a secret key
public static SecretKey createAESKey()
    throws Exception
{
    SecureRandom securerandom = new SecureRandom();
    KeyGenerator keygenerator = KeyGenerator.getInstance(AES);

    keygenerator.init(256, securerandom);
    SecretKey key = keygenerator.generateKey();
    return key;
}

// Function to initialize a vector with an arbitrary value
public static byte[] createInitializationVector()
{
    // Used with encryption
    byte[] initializationVector = new byte[16];
    SecureRandom secureRandom = new SecureRandom();
    secureRandom.nextBytes(initializationVector);
    return initializationVector;
}

// This function takes plaintext, the key with an initialization
vector to convert plainText into CipherText.
public static byte[] do_AESEncryption(
    String plainText,
    SecretKey secretKey,
    byte[] initializationVector)
    throws Exception
{
    Cipher cipher = Cipher.getInstance(AES_CIPHER_ALGORITHM);

    IvParameterSpec ivParameterSpec = new
IvParameterSpec(initializationVector);

    cipher.init(Cipher.ENCRYPT_MODE, secretKey, ivParameterSpec);

    return cipher.doFinal(plainText.getBytes());
}
```

```
// This function performs the
// reverse operation of the do_AESEncryption function. It converts
ciphertext to the plaintext using the key.
public static String do_AESDecryption( byte[] cipherText, SecretKey
secretKey, byte[] initializationVector) throws Exception
{
    Cipher cipher = Cipher.getInstance(AES_CIPHER_ALGORITHM);

    IvParameterSpec ivParameterSpec = new
IvParameterSpec(initializationVector);

    cipher.init(Cipher.DECRYPT_MODE, secretKey, ivParameterSpec);

    byte[] result = cipher.doFinal(cipherText);

    return new String(result);
}

// Driver code
public static void main(String args[]) throws Exception
{
    SecretKey Symmetrickey = createAESKey();
    System.out.println("\n\t Encryption and Deacrypting Program\n");
    //System.out.println("The Symmetric Key is :"+
DatatypeConverter.printHexBinary(Symmetrickey.getEncoded()));

    byte[] initializationVector = createInitializationVector();

    String plainText = "shubham,This is the message "+ "I want To
Encrypt.";

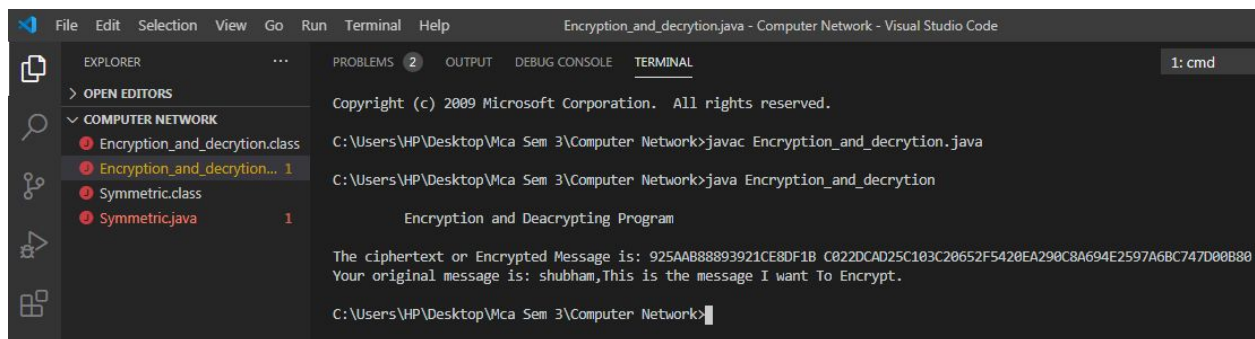
    // Encrypting the message using the symmetric key
    byte[] cipherText = do_AESEncryption(plainText, Symmetrickey,
initializationVector);

    System.out.println("The ciphertext or "+ "Encrypted Message is: "+
DatatypeConverter.printHexBinary(cipherText));
```

```
// Decrypting the encrypted message
String decryptedText = do_AESDecryption(cipherText, SymmetricKey,
initializationVector);

System.out.println( "Your original message is: "+ decryptedText);
}
}
```

Output-2



The screenshot shows the Visual Studio Code interface with the 'TERMINAL' tab active. The terminal output is as follows:

```
Copyright (c) 2009 Microsoft Corporation. All rights reserved.

C:\Users\HP\Desktop\Vca Sem 3\Computer Network>javac Encryption_and_decryption.java
C:\Users\HP\Desktop\Vca Sem 3\Computer Network>java Encryption_and_decryption

Encryption and Decrypting Program

The ciphertext or Encrypted Message is: 925AAB88893921CE8DF1B C022DCAD25C103C20652F5420EA290C8A694E2597A6BC747D00B80
Your original message is: shubham,This is the message I want To Encrypt.

C:\Users\HP\Desktop\Vca Sem 3\Computer Network>
```

Conclusion

In present day cryptography, AES is widely adopted and supported in both hardware and software. Till date, no practical cryptanalytic attacks against AES have been discovered. Additionally, AES has built-in flexibility of key length, which allows a degree of 'future-proofing' against progress in the ability to perform exhaustive key searches.

However, just as for DES, the AES security is assured only if it is correctly implemented and good key management is employed.