# Bansilal Ramnath Agarwal Charitable Trust’s

Vishwakarma Institute of Technology, Pune-37

*(An autonomous Institute of Savitribai Phule Pune University)*

**

# Department of Computer Engineering

|  |  |
| --- | --- |
| **Name** | Thakare Prathamesh Prabhakar |
| **Batch** | Batch - 2 |
| **PRN** | 12220016 |
| **Roll No** | 58 |
| **Division** | CS-D |
| **Subject** | Cyber Security |

**Problem Statement:** Implement Simplified AES Algorithm.

**Code:**

Key Generation:

import java.util.\*;

public class SimplifiedAES{

static Scanner sc = new Scanner(System.in);

static HashMap<String, String> sBoxMap = new HashMap<>();

public static void main(String args[]){

sBoxMap.put("0000", "1001");

sBoxMap.put("0001", "0100");

sBoxMap.put("0010", "1010");

sBoxMap.put("0011", "1011");

sBoxMap.put("0100", "1101");

sBoxMap.put("0101", "0001");

sBoxMap.put("0110", "1000");

sBoxMap.put("0111", "0101");

sBoxMap.put("1000", "0110");

sBoxMap.put("1001", "0010");

sBoxMap.put("1010", "0000");

sBoxMap.put("1011", "0011");

sBoxMap.put("1100", "1100");

sBoxMap.put("1101", "1110");

sBoxMap.put("1110", "1111");

sBoxMap.put("1111", "0111");

String key = "";

System.out.println("Key Generation for Simplified AES");

System.out.print("Enter the key: ");

key = sc.nextLine();

String w0 = key.substring(0, 8);

String w1 = key.substring(8);

System.out.print("\nW0: "+ w0);

System.out.print("\nW1: "+ w0);

String w2Const = "10000000", w4Const = "00110000";

String w2\_part1 = xorOperation(w0, w2Const);

System.out.print("\nw2 part 1: "+ w2\_part1);

String w2\_part2 = subNib(rotNib(w1));

System.out.print("\nw2 part 2: "+ w2\_part2);

String w2 = xorOperation(w2\_part1, w2\_part2);

System.out.print("\nW2: "+ w2);

String w3 = xorOperation(w1, w2);

System.out.print("\nW3: "+ w3);

String w4\_part1 = xorOperation(w2, w4Const);

String w4\_part2 = subNib(rotNib(w3));

String w4 = xorOperation(w4\_part1, w4\_part2);

System.out.print("\nW4: " + w4);

String w5 = xorOperation(w4, w3);

System.out.print("\nW5: " + w5);

String key1 = w0 + w1;

String key2 = w2 + w3;

String key3 = w4 + w5;

System.out.print("\n\nKey1: " + key1 + " \nKey2: " +key2+ "\nKey3: " +key3);

/\*System.out.print("\n\nEnter you plain text: ");

String plainText = sc.nextLine();

String xorPTwithKey = xorOperation(plainText, key1);

System.out.print("XORed plain text: " + xorPTwithKey);

String xorPTSubNib = subNibForEncryption(xorPTwithKey);

System.out.print("\nXORed plain text after substitue nibble: " + xorPTSubNib);

String xorPTRotNib2\_4 = rotNibForEncryption(xorPTSubNib);

System.out.print("\nXORed plain text after rot nib: " + xorPTRotNib2\_4);

String me[][] = {{"0001", "0100"}, {"0100", "0001"}};

String sMatrix[][] = {{xorPTRotNib2\_4.substring(0, 4), xorPTRotNib2\_4.substring(8, 12)}, {xorPTRotNib2\_4.substring(4, 8) , xorPTRotNib2\_4.substring(12)}};

int mixColMultiplication[][] = new int[me.length][sMatrix.length];

String mixColMultiplicationStr = "";

for(int i = 0; i < me.length; i++){

for(int j = 0; j < sMatrix.length; j++) {

for(int k = 0; k < me.length; k++){

int prt1 = Integer.parseInt(me[i][k]) \* Integer.parseInt(sMatrix[k][j]);

String prt1Str = String.valueOf(prt1);

if(prt1Str.length() > 4)

}

}

}\*/

}

public static String xorOperation(String str1, String str2){

StringBuffer bfr = new StringBuffer();

int lenStr1 = str1.length();

int lenStr2 = str2.length();

for(int i = 0, j = 0; i < lenStr1 && j < lenStr2; i++, j++)

bfr.append(str1.charAt(i) ^ str2.charAt(j));

return bfr.toString();

}

public static String rotNib(String str){

StringBuffer bfr = new StringBuffer();

bfr.append(str.substring(4));

bfr.append(str.substring(0, 4));

return bfr.toString();

}

public static String rotNibForEncryption(String str){

StringBuffer bfr = new StringBuffer();

bfr.append(str.substring(0,4));

bfr.append(str.substring(12));

bfr.append(str.substring(8,12));

bfr.append(str.substring(4,8));

return bfr.toString();

}

public static String subNib(String str){

StringBuffer bfr = new StringBuffer();

bfr.append(sBoxMap.get(str.substring(0, 4)));

bfr.append(sBoxMap.get(str.substring(4)));

return bfr.toString();

}

public static String subNibForEncryption(String str){

StringBuffer bfr = new StringBuffer();

bfr.append(sBoxMap.get(str.substring(0,4)));

bfr.append(sBoxMap.get(str.substring(4,8)));

bfr.append(sBoxMap.get(str.substring(8,12)));

bfr.append(sBoxMap.get(str.substring(12)));

return bfr.toString();

}

}

Encryption and Decrytion:

import java.util.\*;

public class AES\_Encryption {

private static final int[] sBox = { 0x9, 0x4, 0xA, 0xB, 0xD, 0x1, 0x8, 0x5, 0x6, 0x2, 0x0, 0x3, 0xC, 0xE, 0xF, 0x7 };

private static final int[] sBoxI = { 0xA, 0x5, 0x9, 0xB, 0x1, 0x7, 0x8, 0xF, 0x6, 0x0, 0x2, 0x3, 0xC, 0x4, 0xD, 0xE };

private int[] preRoundKey;

private int[] round1Key;

private int[] round2Key;

private int subWord(int word) {

return (sBox[(word >> 4)] << 4) + sBox[word & 0x0F];

}

private int rotWord(int word) {

return ((word & 0x0F) << 4) + ((word & 0xF0) >> 4);

}

public void keygen(int key) {

int Rcon1 = 0x80;

int Rcon2 = 0x30;

int[] w = new int[6];

w[0] = (key & 0xFF00) >> 8;

w[1] = key & 0x00FF;

w[2] = w[0] ^ (subWord(rotWord(w[1])) ^ Rcon1);

w[3] = w[2] ^ w[1];

w[4] = w[2] ^ (subWord(rotWord(w[3])) ^ Rcon2);

w[5] = w[4] ^ w[3];

preRoundKey = intToState((w[0] << 8) + w[1]);

round1Key = intToState((w[2] << 8) + w[3]);

round2Key = intToState((w[4] << 8) + w[5]);

}

private int gfMult(int a, int b) {

int product = 0;

a = a & 0x0F;

b = b & 0x0F;

while (a != 0 && b != 0) {

if ((b & 1) == 1) {

product ^= a;

}

a <<= 1;

if ((a & (1 << 4)) != 0) {

a ^= 0b10011;

}

b >>= 1;

}

return product;

}

private int[] intToState(int n) {

int[] state = new int[4];

state[0] = (n >> 12) & 0xF;

state[1] = (n >> 4) & 0xF;

state[2] = (n >> 8) & 0xF;

state[3] = n & 0xF;

return state;

}

private int stateToInt(int[] m) {

return (m[0] << 12) + (m[1] << 4) + (m[2] << 8) + m[3];

}

private int[] addRoundKey(int[] s1, int[] s2) {

int[] result = new int[4];

for (int i = 0; i < 4; i++) {

result[i] = s1[i] ^ s2[i];

}

return result;

}

private int[] subNibbles(int[] sbox, int[] state) {

int[] result = new int[4];

for (int i = 0; i < 4; i++) {

result[i] = sbox[state[i]];

}

return result;

}

private int[] shiftRows(int[] state) {

return new int[] { state[0], state[1], state[3], state[2] };

}

private int[] mixColumns(int[] state) {

return new int[] {

state[0] ^ gfMult(4, state[2]),

state[1] ^ gfMult(4, state[3]),

state[2] ^ gfMult(4, state[0]),

state[3] ^ gfMult(4, state[1])

};

}

private int[] inverseMixColumns(int[] state) {

return new int[] {

gfMult(9, state[0]) ^ gfMult(2, state[2]),

gfMult(9, state[1]) ^ gfMult(2, state[3]),

gfMult(9, state[2]) ^ gfMult(2, state[0]),

gfMult(9, state[3]) ^ gfMult(2, state[1])

};

}

public static void printn(int num, int n) {

int[] res = new int[n];

for (int i = n - 1; i >= 0; i--) {

res[i] = num % 2;

num /= 2;

}

for (int i : res) {

System.out.print(i);

}

}

public int encrypt(int plaintext) {

int[] state = addRoundKey(preRoundKey, intToState(plaintext));

state = mixColumns(shiftRows(subNibbles(sBox, state)));

state = addRoundKey(round1Key, state);

state = shiftRows(subNibbles(sBox, state));

state = addRoundKey(round2Key, state);

return stateToInt(state);

}

public int decrypt(int ciphertext) {

int[] state = addRoundKey(round2Key, intToState(ciphertext));

state = subNibbles(sBoxI, shiftRows(state));

state = inverseMixColumns(addRoundKey(round1Key, state));

state = subNibbles(sBoxI, shiftRows(state));

state = addRoundKey(preRoundKey, state);

return stateToInt(state);

}

public static void main(String[] args) {

Scanner sc = new Scanner(System.in);

System.out.println("AES Encryption/Decryption Tool");

System.out.print("Enter the encryption/decryption key (as binary string): ");

String keyInput = sc.nextLine().trim();

if (keyInput.length() != 16) {

System.out.println("Key must be a binary string of length 16.");

return;

}

int key = Integer.parseInt(keyInput, 2);

AES\_Encryption aes = new AES\_Encryption();

aes.keygen(key);

System.out.print("Enter the plaintext/ciphertext (as binary string): ");

String textInput = sc.nextLine().trim();

if (textInput.length() != 16) {

System.out.println("Text must be a binary string of length 16.");

return;

}

int text = Integer.parseInt(textInput, 2);

System.out.print("Enter 1 to encrypt and 2 to decrypt: ");

int choice = sc.nextInt();

int result = 0;

switch (choice) {

case 1:

result = aes.encrypt(text);

System.out.print("Encrypted ciphertext (in binary): ");

break;

case 2:

result = aes.decrypt(text);

System.out.print("Decrypted plaintext (in binary): ");

break;

default:

System.out.println("Invalid choice.");

return;

}

printBinary(result, 16);

sc.close();

}

public static void printBinary(int num, int numBits) {

String binaryString = Integer.toBinaryString(num);

while (binaryString.length() < numBits) {

binaryString = "0" + binaryString;

}

System.out.println(binaryString);

}

}

**Output:**

