**Submitted by – Paras Jain (2018kucp1006) ISS Lab**

Assignment: Implement AES encryption Algorithm

Code:-

//Main class

class AESEncryptionMain {

    // converting from hexadecimal to text form

    static String chr(String fin) {

        StringBuilder ans = new StringBuilder("");

        for (int i = 1; i < fin.length(); i += 3) {

            String tmp = fin.substring(i, i + 2);

            int x = Integer.parseInt(tmp, 16);

            char c = (char) x;

            ans.insert(ans.length(), c);

        }

        return ans.toString();

    }

    static void pnt(Word[] k, Helper h) {

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

            if (i % 4 == 0) {

                System.out.println();

            }

            System.out.print(h.tohex(k[i].word));

        }

    }

    public static void main(String[] args) {

        Helper h = new Helper();

        String txt = "I\_am\_Paras\_Jain\_", key = "IIIT2018KUCP1006";

        Aes aes = new Aes(txt, key);// I\_am\_Paras\_Jain\_

        System.out.println("Plain text is: "+txt+"\nKey is: "+key);

        String fin = h.tohex(aes.encrypt());

        System.out.println("Encrypted Message: " + chr(fin));

    }

}

// class inclusing key expansion

class Aes {

    String key;

    String text;

    private Helper h;

    Word keys[];

    // constructor

    Aes(String txt, String k) {

        key = k;

        text = txt;

    }

    // encrypt function

    String encrypt() {

        String ret = "";

        h = new Helper();

        text = h.tobin(text);

        key = h.tobin(key);

        keyExp();

        // putting text into matrix

        Word[] state = new Word[4];

        for (int i = 0; i < 4; i++) {

            state[i] = new Word(text.substring(i \* 32, (i + 1) \* 32));

        }

        state = transpose(state);

        // Initial transformation

        state = ark(state, 0);

        // my ID is 1006 hence 5 normal rounds

        for (int i = 1; i <= 5; i++){

            applyS(state);

            lss(state);

            state = ark(state, i);

            String ans="";

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

                ans += state[j].word;

            }

            System.out.println("Round "+ i+":"+h.tohex(ans));

        }

        // final round

        applyS(state);

        lss(state);

        state = ark(state, 10);

        state = transpose(state);

        for (int i = 0; i < 4; i++){

            ret += state[i].word;

        }

        System.out.println("Round "+ "6"+":"+h.tohex(ret));

        return ret;

    }

    // taking transpose

    Word[] transpose(Word[] txt) {

        Word[] ret = new Word[4];

        for (int i = 0; i < 4; i++) {

            ret[i] = new Word(txt[0].getByt(i) + txt[1].getByt(i) + txt[2].getByt(i) + txt[3].getByt(i));

        }

        return ret;

    }

    // key expansion

    void keyExp() {

        keys = new Word[44];

        for (int i = 0; i < 4; i++) {

            keys[i] = new Word(key.substring(i \* 32, (i + 1) \* 32));

        }

        for (int i = 4; i < keys.length; i += 4) {

            keys[i] = keys[i - 4].xorW(g(keys[i - 1], i / 4));

            keys[i + 1] = keys[i - 3].xorW(keys[i]);

            keys[i + 2] = keys[i - 2].xorW(keys[i + 1]);

            keys[i + 3] = keys[i - 1].xorW(keys[i + 2]);

        }

    }

    void applyS(Word[] txt) {

        for (int i = 0; i < 4; i++) {

            txt[i].appSb();

        }

    }

    void lss(Word[] txt) {

        for (int i = 0; i < 4; i++) {

            txt[i].lSw(i);

        }

    }

    Word[] ark(Word[] txt, int n) {

        Word[] temp;

        temp = transpose(txt);

        for (int i = 0; i < 4; i++) {

            temp[i] = temp[i].xorW(keys[(n \* 4) + i]);

        }

        return transpose(temp);

    }

    Word g(Word wrd, int rn) {

        Word ret = new Word(wrd.word);

        ret.lSw(1);

        ret.appSb();

        ret.appCnst(rn - 1);

        return ret;

    }

}

class Helper {

    // initialising s\_box array taken from the book

    String sBox[] = { "63", "7c", "77", "7b", "f2", "6b", "6f", "c5", "30", "01", "67", "2b", "fe", "d7", "ab", "76",

            "ca", "82", "c9", "7d", "fa", "59", "47", "f0", "ad", "d4", "a2", "af", "9c", "a4", "72", "c0", "b7", "fd",

            "93", "26", "36", "3f", "f7", "cc", "34", "a5", "e5", "f1", "71", "d8", "31", "15", "04", "c7", "23", "c3",

            "18", "96", "05", "9a", "07", "12", "80", "e2", "eb", "27", "b2", "75", "09", "83", "2c", "1a", "1b", "6e",

            "5a", "a0", "52", "3b", "d6", "b3", "29", "e3", "2f", "84", "53", "d1", "00", "ed", "20", "fc", "b1", "5b",

            "6a", "cb", "be", "39", "4a", "4c", "58", "cf", "d0", "ef", "aa", "fb", "43", "4d", "33", "85", "45", "f9",

            "02", "7f", "50", "3c", "9f", "a8", "51", "a3", "40", "8f", "92", "9d", "38", "f5", "bc", "b6", "da", "21",

            "10", "ff", "f3", "d2", "cd", "0c", "13", "ec", "5f", "97", "44", "17", "c4", "a7", "7e", "3d", "64", "5d",

            "19", "73", "60", "81", "4f", "dc", "22", "2a", "90", "88", "46", "ee", "b8", "14", "de", "5e", "0b", "db",

            "e0", "32", "3a", "0a", "49", "06", "24", "5c", "c2", "d3", "ac", "62", "91", "95", "e4", "79", "e7", "c8",

            "37", "6d", "8d", "d5", "4e", "a9", "6c", "56", "f4", "ea", "65", "7a", "ae", "08", "ba", "78", "25", "2e",

            "1c", "a6", "b4", "c6", "e8", "dd", "74", "1f", "4b", "bd", "8b", "8a", "70", "3e", "b5", "66", "48", "03",

            "f6", "0e", "61", "35", "57", "b9", "86", "c1", "1d", "9e", "e1", "f8", "98", "11", "69", "d9", "8e", "94",

            "9b", "1e", "87", "e9", "ce", "55", "28", "df", "8c", "a1", "89", "0d", "bf", "e6", "42", "68", "41", "99",

            "2d", "0f", "b0", "54", "bb", "16" };

    String cnst4k[] = { "01", "02", "04", "08", "10", "20", "40", "80", "1b", "36" };

    String tobin(String str) {

        // function to convert ascii to 8bit-binary string

        int l = str.length();

        char temp;

        int ascii;

        String bin, ret = "";

        for (int i = 0; i < l; i++){

            // for every char in string

            temp = str.charAt(i);

            ascii = (int) temp;// change into corresponding int value

            // gives ascii value

            bin = Integer.toBinaryString(ascii);

            // convert into binary using inbuild java function

            while (bin.length() < 8) {

                // if generated binary string length is less than 8

                // add 0 at starting

                bin = '0' + bin;

            }

            ret += bin;

        }

        return ret;

    }

    String appSbox(String txt){

        int x = Integer.parseInt(txt.substring(0, 4), 2);

        int y = Integer.parseInt(txt.substring(4), 2);

        int ind = x \* 16 + y;

        // converting from hexadecimal to binary

        return hex2bin(sBox[ind]);

    }

    // converting from hexadecimal to binary

    String hex2bin(String hex) {

        String ret = Integer.toBinaryString(Integer.parseInt(hex, 16));

        int l = hex.length();

        while (ret.length() < l \* 4) {

            ret = "0" + ret;

        }

        return ret;

    }

    // xoring 2 strings

    String xorS(String a, String b) {

        String temp = "";

        int len = a.length();

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

            temp += a.charAt(i) ^ b.charAt(i);

        }

        return temp;

    }

    // function to convert binary string to hex

    String tohex(String bitsS) {

        String ret = "", temp;

        int l = bitsS.length();

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

            if (i % 8 == 0)

                ret += ' ';

            temp = bitsS.substring(i, i + 4);

            ret += Integer.toString(Integer.parseInt(temp, 2), 16);

        }

        return ret;

    }

}

class Word {

    String word;

    int l = 32;

    private Helper h;

    Word(String w) {

        word = w.substring(0, l);

        h = new Helper();

    }

    Word xorW(Word other) {

        String temp = "";

        for (int i = 0; i < l; i++) {

            temp += this.word.charAt(i) ^ other.word.charAt(i);

        }

        return new Word(temp);

    }

    String getByt(int n) {

        return word.substring(8 \* n, (n + 1) \* 8);

    }

    void xorW(String other) {

        String temp = "";

        for (int i = 0; i < l; i++) {

            temp += this.word.charAt(i) ^ other.charAt(i);

        }

        word = temp;

    }

    void appCnst(int rn) {

        String c = h.hex2bin(h.cnst4k[rn]) + h.hex2bin("000000");

        xorW(c);

    }

    void lSw(int n) {

        String temp = word.substring(8 \* n);

        temp += word.substring(0, 8 \* n);

        word = temp;

    }

    void appSb() {

        String temp = "";

        for (int i = 0; i < 4; i++) {

            temp += h.appSbox(word.substring(i \* 8, (i + 1) \* 8));

        }

        word = temp;

    }

}

**Output:**

