MSCS 630 – SECURITY ALGORITHMS AND PROTOCOLS

Lab 4

APRIL 4, 2018

THOMPSON RAJAN
CWID: 20082947

Generating AES Round Keys

1. AESCipher.java

```java
/**
 * File: AESCipher.java
 * Author: Thompson Rajan
 * Course: MSCS 630 Security Algorithms and Protocols
 * Assignment: Lab 4
 * Due Date: Wednesday, April 4, 2018
 * Version: 1.0
 *
 * This file contains the implementation to generate round keys for AES Cipher.
 */

/**
 * This class contains the static methods to generate AES keys.
 */
public class AESCipher {

  /**
   * This static method generates the 11 round keys for the AES cipher.
   * @param keyHex - Input Key
   * @return - String array of 11 round keys
   */
  static String[] aesRoundKeys(String keyHex){
    // Array to store the 11 round keys.
    String[] roundKeysHex = new String[11];

    // Initialize string array.
    for(int i = 0; i < roundKeysHex.length;i++){
      roundKeysHex[i] = "";
    }

    // This array is used extensively to operate on each round key values.
    int[][] roundKeys = new int[4][44];
    int k = 0;

    // Get key value by individual bytes and store in the first four columns.
    for(int i = 0; i < 4 ; i++){
      for(int j = 0; j < 4; j++){
        if(i < 4){
          roundKeys[j][i] = Integer.parseInt(keyHex.substring(k, k+2)
                  ,16);
          k += 2;
        }
      }
    }

    // Vectors operated with integer decimal values.
    for(int i = 4; i < 44; i++ ){

      // XOR last 4th column and the previous column and then store the
      // result in the current column leaving the first column of the round key.
      if (i % 4 != 0) {
        for (int j = 0; j < 4; j++) {
          roundKeys[j][i] = roundKeys[j][i - 4] ^ roundKeys[j][i - 1];
        }
```

```
      }
      else {
        int round = i / 4;
        int rCon = Integer.parseInt(aesRCon(round), 16);

        // Temporary vector to evaluate the new vector.
        int[] tempKey = new int[4];

        // Left Shift by 1 and S-Box transformation.
        for(int j = 0; j < 4; j++){
          tempKey[(j + 3) % 4] = Integer.parseInt(aesSBox(
                  Integer.toHexString(roundKeys[j][i - 1])), 16);
        }

        // XOR first element with corresponding Rijndael constant from the
        // rCon table
        tempKey[0] = tempKey[0] ^ rCon;

        // XOR new vector with the last 4th column of the key vector.
        for (int j = 0; j < 4; j++) {
          roundKeys[j][i] = roundKeys[j][i - 4] ^ tempKey[j];
        }
      }
    }

    // Implementation to convert results to hexadecimal string array.
    for(int i = 0; i < 44; i++){
      int round = i / 4;
      for(int j = 0; j < 4; j++){

        // Padding 0 before single digit strings wherever possible.
        if(Integer.toHexString(roundKeys[j][i]).length() == 1){
          if(Integer.toHexString(roundKeys[j][i]).equals("0")){
            roundKeysHex[round] += "00";
          }
          else
          roundKeysHex[round]+= "0" + Integer.toHexString(roundKeys[j][i])
                  .toUpperCase();
        }
        else {
          roundKeysHex[round] += Integer.toHexString(roundKeys[j][i])
                  .toUpperCase();
        }
      }
    }
    return roundKeysHex;
  }

  /**
   * This method transforms does the S-Box transform based on the sBox table.
   * @param inHex - Input hex string value
   * @return - Transformed hex string value
   */
  static String aesSBox(String inHex) {

    int it = Integer.parseInt(inHex,16);
    char x= s[it];

    return Integer.toHexString((int) x).toUpperCase();
  }
```

```java
/**
 * This method gets the Rijndael key based on the round number.
 * @param round - Current round number
 * @return - Rijndael key hex string value.
 */
static String aesRCon(int round){
  char x = r[round];

  return Integer.toHexString((int) x).toUpperCase();
}

// S-Box Transformation Table
private static final char[] s = {
  0x63, 0x7C, 0x77, 0x7B, 0xF2, 0x6B, 0x6F, 0xC5,
  0x30, 0x01, 0x67, 0x2B, 0xFE, 0xD7, 0xAB, 0x76,
  0xCA, 0x82, 0xC9, 0x7D, 0xFA, 0x59, 0x47, 0xF0,
  0xAD, 0xD4, 0xA2, 0xAF, 0x9C, 0xA4, 0x72, 0xC0,
  0xB7, 0xFD, 0x93, 0x26, 0x36, 0x3F, 0xF7, 0xCC,
  0x34, 0xA5, 0xE5, 0xF1, 0x71, 0xD8, 0x31, 0x15,
  0x04, 0xC7, 0x23, 0xC3, 0x18, 0x96, 0x05, 0x9A,
  0x07, 0x12, 0x80, 0xE2, 0xEB, 0x27, 0xB2, 0x75,
  0x09, 0x83, 0x2C, 0x1A, 0x1B, 0x6E, 0x5A, 0xA0,
  0x52, 0x3B, 0xD6, 0xB3, 0x29, 0xE3, 0x2F, 0x84,
  0x53, 0xD1, 0x00, 0xED, 0x20, 0xFC, 0xB1, 0x5B,
  0x6A, 0xCB, 0xBE, 0x39, 0x4A, 0x4C, 0x58, 0xCF,
  0xD0, 0xEF, 0xAA, 0xFB, 0x43, 0x4D, 0x33, 0x85,
  0x45, 0xF9, 0x02, 0x7F, 0x50, 0x3C, 0x9F, 0xA8,
  0x51, 0xA3, 0x40, 0x8F, 0x92, 0x9D, 0x38, 0xF5,
  0xBC, 0xB6, 0xDA, 0x21, 0x10, 0xFF, 0xF3, 0xD2,
  0xCD, 0x0C, 0x13, 0xEC, 0x5F, 0x97, 0x44, 0x17,
  0xC4, 0xA7, 0x7E, 0x3D, 0x64, 0x5D, 0x19, 0x73,
  0x60, 0x81, 0x4F, 0xDC, 0x22, 0x2A, 0x90, 0x88,
  0x46, 0xEE, 0xB8, 0x14, 0xDE, 0x5E, 0x0B, 0xDB,
  0xE0, 0x32, 0x3A, 0x0A, 0x49, 0x06, 0x24, 0x5C,
  0xC2, 0xD3, 0xAC, 0x62, 0x91, 0x95, 0xE4, 0x79,
  0xE7, 0xC8, 0x37, 0x6D, 0x8D, 0xD5, 0x4E, 0xA9,
  0x6C, 0x56, 0xF4, 0xEA, 0x65, 0x7A, 0xAE, 0x08,
  0xBA, 0x78, 0x25, 0x2E, 0x1C, 0xA6, 0xB4, 0xC6,
  0xE8, 0xDD, 0x74, 0x1F, 0x4B, 0xBD, 0x8B, 0x8A,
  0x70, 0x3E, 0xB5, 0x66, 0x48, 0x03, 0xF6, 0x0E,
  0x61, 0x35, 0x57, 0xB9, 0x86, 0xC1, 0x1D, 0x9E,
  0xE1, 0xF8, 0x98, 0x11, 0x69, 0xD9, 0x8E, 0x94,
  0x9B, 0x1E, 0x87, 0xE9, 0xCE, 0x55, 0x28, 0xDF,
  0x8C, 0xA1, 0x89, 0x0D, 0xBF, 0xE6, 0x42, 0x68,
  0x41, 0x99, 0x2D, 0x0F, 0xB0, 0x54, 0xBB, 0x16
};

// Rijndael Key Schedule
private static final char[] r = {
  0x8d, 0x01, 0x02, 0x04, 0x08, 0x10, 0x20, 0x40,
  0x80, 0x1b, 0x36, 0x6c, 0xd8, 0xab, 0x4d, 0x9a,
  0x2f, 0x5e, 0xbc, 0x63, 0xc6, 0x97, 0x35, 0x6a,
  0xd4, 0xb3, 0x7d, 0xfa, 0xef, 0xc5, 0x91, 0x39,
  0x72, 0xe4, 0xd3, 0xbd, 0x61, 0xc2, 0x9f, 0x25,
  0x4a, 0x94, 0x33, 0x66, 0xcc, 0x83, 0x1d, 0x3a,
  0x74, 0xe8, 0xcb, 0x8d, 0x01, 0x02, 0x04, 0x08,
  0x10, 0x20, 0x40, 0x80, 0x1b, 0x36, 0x6c, 0xd8,
  0xab, 0x4d, 0x9a, 0x2f, 0x5e, 0xbc, 0x63, 0xc6,
  0x97, 0x35, 0x6a, 0xd4, 0xb3, 0x7d, 0xfa, 0xef,
  0xc5, 0x91, 0x39, 0x72, 0xe4, 0xd3, 0xbd, 0x61,
```

```
    0xc2, 0x9f, 0x25, 0x4a, 0x94, 0x33, 0x66, 0xcc,
    0x83, 0x1d, 0x3a, 0x74, 0xe8, 0xcb, 0x8d, 0x01,
    0x02, 0x04, 0x08, 0x10, 0x20, 0x40, 0x80, 0x1b,
    0x36, 0x6c, 0xd8, 0xab, 0x4d, 0x9a, 0x2f, 0x5e,
    0xbc, 0x63, 0xc6, 0x97, 0x35, 0x6a, 0xd4, 0xb3,
    0x7d, 0xfa, 0xef, 0xc5, 0x91, 0x39, 0x72, 0xe4,
    0xd3, 0xbd, 0x61, 0xc2, 0x9f, 0x25, 0x4a, 0x94,
    0x33, 0x66, 0xcc, 0x83, 0x1d, 0x3a, 0x74, 0xe8,
    0xcb, 0x8d, 0x01, 0x02, 0x04, 0x08, 0x10, 0x20,
    0x40, 0x80, 0x1b, 0x36, 0x6c, 0xd8, 0xab, 0x4d,
    0x9a, 0x2f, 0x5e, 0xbc, 0x63, 0xc6, 0x97, 0x35,
    0x6a, 0xd4, 0xb3, 0x7d, 0xfa, 0xef, 0xc5, 0x91,
    0x39, 0x72, 0xe4, 0xd3, 0xbd, 0x61, 0xc2, 0x9f,
    0x25, 0x4a, 0x94, 0x33, 0x66, 0xcc, 0x83, 0x1d,
    0x3a, 0x74, 0xe8, 0xcb, 0x8d, 0x01, 0x02, 0x04,
    0x08, 0x10, 0x20, 0x40, 0x80, 0x1b, 0x36, 0x6c,
    0xd8, 0xab, 0x4d, 0x9a, 0x2f, 0x5e, 0xbc, 0x63,
    0xc6, 0x97, 0x35, 0x6a, 0xd4, 0xb3, 0x7d, 0xfa,
    0xef, 0xc5, 0x91, 0x39, 0x72, 0xe4, 0xd3, 0xbd,
    0x61, 0xc2, 0x9f, 0x25, 0x4a, 0x94, 0x33, 0x66,
    0xcc, 0x83, 0x1d, 0x3a, 0x74, 0xe8, 0xcb, 0x8d
  };
}
```

2. DriverAES.java

```java
/**
 * File: DriverAES.java
 * Author: Thompson Rajan
 * Course: MSCS 630 Security Algorithms and Protocols
 * Assignment: Lab 4
 * Due Date: Wednesday, April 4, 2018
 * Version: 1.0
 *
 * This file contains a class that calls the AESCipher to generate secure keys.
 */

import java.util.Scanner;

/**
 * This class calls the AESCipher class to generate round keys for a given key.
 */
public class DriverAES {
  /**
   * This method makes a static call to aesRoundKeys to get the round keys.
   * @param args - null
   */
  public static void main(String[] args) {

    Scanner in = new Scanner(System.in);

    String keyHex = in.next();

    // Get round keys
    String[] roundKeys = AESCipher.aesRoundKeys(keyHex);
    for(int i = 0; i < roundKeys.length; i++){
      System.out.println(roundKeys[i]);
    }
  }
}
```

Output:

```
4 — -bash — 80×24
148-100-144-110:4 tom$ cat input.1
5468617473206D79204B756E67204675
148-100-144-110:4 tom$ java DriverAES < input.1
5468617473206D79204B756E67204675
E232FCF191129188B159E4E6D679A293
56082007C71AB18F76435569A03AF7FA
D2600DE7157ABC686339E901C3031EFB
A11202C9B468BEA1D75157A01452495B
B1293B3305418592D210D232C6429B69
BD3DC287B87C47156A6C9527AC2E0E4E
CC96ED1674EAAA031E863F24B2A8316A
8E51EF21FABB4522E43D7A0656954B6C
BFE2BF904559FAB2A16480B4F7F1CBD8
28FDDEF86DA4244ACCC0A4FE3B316F26
148-100-144-110:4 tom$
```

Date: 04-04-18

Thompson Rajan
CWID: 20082947