**2019-2020学年第二学期《网络安全实验》考查试题**

**（适用于17.1、17.2、17.7班）**

班级\_\_\_17.1\_\_\_\_\_\_ 学号\_\_17561140112\_\_\_\_ 姓名\_\_\_\_张峻溥\_\_\_\_\_

|  |  |  |  |
| --- | --- | --- | --- |
| 题号 | 一 | 二 | 总分 |
| 得分 |  |  |  |

**（注意：独立完成，不得相互抄袭，如有雷同，相关考生成绩均为0分）**

|  |  |
| --- | --- |
| 得分 |  |
| 评卷人 |  |

一、程序设计题（60分）。

设计实现AES算法，要求：（1）代码完整、正确、可运行（20分）；（2）给出两组输入输出数据（10分）；（3）画出算法流程图，流程图应与代码设计思路一致，流程图中各关键步骤需注明在代码中所使用函数或模块的名称（15分）；（4）代码中给出注释，详略得当，并与算法流程图相配合（15分）。

说明：本题代码及注释可输入；给出的两组数据可输入，并将运行情况截图（图片清晰、大小适中）；流程图手写拍照粘图（图片清晰整洁、大小适中）；在下面空白处答题，空位不够可继续增页。

1. 程序代码

/\*

    Zeolim - An AC a day keeps the bug away

\*/

//#pragma GCC optimize(2)

//#pragma GCC ("-W1,--stack=128000000")

#include <bits/stdc++.h>

using namespace std;

#define mp(x, y) make\_pair(x, y)

#define fr(x, y, z) for (int x = y; x < z; ++x)

#define pb(x) push\_back(x)

#define mem(x, y) memset(x, y, sizeof(x))

typedef long long ll;

typedef long long ull;

typedef long double ld;

typedef std::pair<int, int> pii;

typedef std::vector<int> vi;

// typedef \_\_int128 ill;

const ld PI = acos(-1.0);

const ld E = exp(1.0);

const ll INF = 0x3f3f3f3f3f3f3f3f;

const ll MOD = 1e9 + 7;

const ull P = 13331;

const int MAXN = 4e6 + 100;

uint8\_t\* encode(uint8\_t\* input);

uint8\_t\* decode(uint8\_t\* input);

void\* encode\_by\_group(void\* input, int length = 0);

void\* decode\_by\_group(void\* input, int length);

uint8\_t w[11][4][4];

void key\_expansion(uint8\_t\* key, uint8\_t w[][4][4]);

uint8\_t ff\_mul(uint8\_t a, uint8\_t b);

void sub\_bytes(uint8\_t temp[][4]);

void shift\_row(uint8\_t temp[][4]);

void mix\_columns(uint8\_t temp[][4]);

void add\_round\_key(uint8\_t temp[][4], uint8\_t k[][4]);

void inv\_sub\_bytes(uint8\_t temp[][4]);

void inv\_shift\_rows(uint8\_t temp[][4]);

void inv\_mix\_columns(uint8\_t temp[][4]);

uint8\_t Sbox[] = {

    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};

uint8\_t InvSbox[256] = {

    0x52, 0x09, 0x6a, 0xd5, 0x30, 0x36, 0xa5, 0x38, 0xbf, 0x40, 0xa3, 0x9e,

    0x81, 0xf3, 0xd7, 0xfb, 0x7c, 0xe3, 0x39, 0x82, 0x9b, 0x2f, 0xff, 0x87,

    0x34, 0x8e, 0x43, 0x44, 0xc4, 0xde, 0xe9, 0xcb, 0x54, 0x7b, 0x94, 0x32,

    0xa6, 0xc2, 0x23, 0x3d, 0xee, 0x4c, 0x95, 0x0b, 0x42, 0xfa, 0xc3, 0x4e,

    0x08, 0x2e, 0xa1, 0x66, 0x28, 0xd9, 0x24, 0xb2, 0x76, 0x5b, 0xa2, 0x49,

    0x6d, 0x8b, 0xd1, 0x25, 0x72, 0xf8, 0xf6, 0x64, 0x86, 0x68, 0x98, 0x16,

    0xd4, 0xa4, 0x5c, 0xcc, 0x5d, 0x65, 0xb6, 0x92, 0x6c, 0x70, 0x48, 0x50,

    0xfd, 0xed, 0xb9, 0xda, 0x5e, 0x15, 0x46, 0x57, 0xa7, 0x8d, 0x9d, 0x84,

    0x90, 0xd8, 0xab, 0x00, 0x8c, 0xbc, 0xd3, 0x0a, 0xf7, 0xe4, 0x58, 0x05,

    0xb8, 0xb3, 0x45, 0x06, 0xd0, 0x2c, 0x1e, 0x8f, 0xca, 0x3f, 0x0f, 0x02,

    0xc1, 0xaf, 0xbd, 0x03, 0x01, 0x13, 0x8a, 0x6b, 0x3a, 0x91, 0x11, 0x41,

    0x4f, 0x67, 0xdc, 0xea, 0x97, 0xf2, 0xcf, 0xce, 0xf0, 0xb4, 0xe6, 0x73,

    0x96, 0xac, 0x74, 0x22, 0xe7, 0xad, 0x35, 0x85, 0xe2, 0xf9, 0x37, 0xe8,

    0x1c, 0x75, 0xdf, 0x6e, 0x47, 0xf1, 0x1a, 0x71, 0x1d, 0x29, 0xc5, 0x89,

    0x6f, 0xb7, 0x62, 0x0e, 0xaa, 0x18, 0xbe, 0x1b, 0xfc, 0x56, 0x3e, 0x4b,

    0xc6, 0xd2, 0x79, 0x20, 0x9a, 0xdb, 0xc0, 0xfe, 0x78, 0xcd, 0x5a, 0xf4,

    0x1f, 0xdd, 0xa8, 0x33, 0x88, 0x07, 0xc7, 0x31, 0xb1, 0x12, 0x10, 0x59,

    0x27, 0x80, 0xec, 0x5f, 0x60, 0x51, 0x7f, 0xa9, 0x19, 0xb5, 0x4a, 0x0d,

    0x2d, 0xe5, 0x7a, 0x9f, 0x93, 0xc9, 0x9c, 0xef, 0xa0, 0xe0, 0x3b, 0x4d,

    0xae, 0x2a, 0xf5, 0xb0, 0xc8, 0xeb, 0xbb, 0x3c, 0x83, 0x53, 0x99, 0x61,

    0x17, 0x2b, 0x04, 0x7e, 0xba, 0x77, 0xd6, 0x26, 0xe1, 0x69, 0x14, 0x63,

    0x55, 0x21, 0x0c, 0x7d};

// 初始化密钥

void key\_init(uint8\_t\* key) { key\_expansion(key, w); }

// 加密

uint8\_t\* encode(uint8\_t\* input) {

  uint8\_t temp[4][4];

  int i, r, c;

  //二维数组下标映射

  for (r = 0; r < 4; r++) {

    for (c = 0; c < 4; c++) {

      temp[r][c] = input[c \* 4 + r];

    }

  }

  add\_round\_key(temp, w[0]);  // 子密钥加密

  for (i = 1; i <= 10; i++) {

    sub\_bytes(temp);

    shift\_row(temp);

    if (i != 10) mix\_columns(temp);

    add\_round\_key(temp, w[i]);

  }

  for (r = 0; r < 4; r++) {

    for (c = 0; c < 4; c++) {

      input[c \* 4 + r] = temp[r][c];

    }

  }

  return input;

}

//反加密

uint8\_t\* decode(uint8\_t\* input) {

  uint8\_t temp[4][4];

  int i, r, c;

  for (r = 0; r < 4; r++) {

    for (c = 0; c < 4; c++) {

      temp[r][c] = input[c \* 4 + r];

    }

  }

  add\_round\_key(temp, w[10]);

  for (i = 9; i >= 0; i--) {

    inv\_shift\_rows(temp);

    inv\_sub\_bytes(temp);

    add\_round\_key(temp, w[i]);

    if (i) inv\_mix\_columns(temp);

  }

  for (r = 0; r < 4; r++) {

    for (c = 0; c < 4; c++) {

      input[c \* 4 + r] = temp[r][c];

    }

  }

  return input;

}

// 分组加密

void\* encode\_by\_group(void\* input, int length) {

  uint8\_t\* in = (uint8\_t\*)input;

  int i;

  if (!length) {

    while (\*(in + length++))

      ;

    in = (uint8\_t\*)input;

  }

  for (i = 0; i < length; i += 16) {

    encode(in + i);

  }

  return input;

}

// 分组反加密

void\* decode\_by\_group(void\* input, int length) {

  uint8\_t\* in = (uint8\_t\*)input;

  int i;

  for (i = 0; i < length; i += 16) {

    decode(in + i);

  }

  return input;

}

// 密钥拓展

void key\_expansion(uint8\_t\* key, uint8\_t w[][4][4]) {

  int i, j, r, c;

  uint8\_t rc[] = {0x01, 0x02, 0x04, 0x08, 0x10, 0x20, 0x40, 0x80, 0x1b, 0x36};

  for (r = 0; r < 4; r++) {

    for (c = 0; c < 4; c++) {

      w[0][r][c] = key[r + c \* 4];

    }

  }

  for (i = 1; i <= 10; i++) {

    for (j = 0; j < 4; j++) {

      uint8\_t t[4];

      for (r = 0; r < 4; r++) {

        t[r] = j ? w[i][r][j - 1] : w[i - 1][r][3];

      }

      if (j == 0) {

        uint8\_t temp = t[0];

        for (r = 0; r < 3; r++) {

          t[r] = Sbox[t[(r + 1) % 4]];

        }

        t[3] = Sbox[temp];

        t[0] ^= rc[i - 1];

      }

      for (r = 0; r < 4; r++) {

        w[i][r][j] = w[i - 1][r][j] ^ t[r];

      }

    }

  }

}

// 字节乘法运算

uint8\_t ff\_mul(uint8\_t a, uint8\_t b) {

  uint8\_t bw[4];

  uint8\_t res = 0;

  int i;

  bw[0] = b;

  for (i = 1; i < 4; i++) {

    bw[i] = bw[i - 1] << 1;

    if (bw[i - 1] & 0x80) {

      bw[i] ^= 0x1b;

    }

  }

  for (i = 0; i < 4; i++) {

    if ((a >> i) & 0x01) {

      res ^= bw[i];

    }

  }

  return res;

}

// 字节替换

void sub\_bytes(uint8\_t temp[][4]) {

  int r, c;

  for (r = 0; r < 4; r++) {

    for (c = 0; c < 4; c++) {

      temp[r][c] = Sbox[temp[r][c]];

    }

  }

}

// 行位移

void shift\_row(uint8\_t temp[][4]) {

  uint8\_t t[4];

  int r, c;

  for (r = 1; r < 4; r++) {

    for (c = 0; c < 4; c++) {

      t[c] = temp[r][(c + r) % 4];

    }

    for (c = 0; c < 4; c++) {

      temp[r][c] = t[c];

    }

  }

}

// 列混合

void mix\_columns(uint8\_t temp[][4]) {

  uint8\_t t[4];

  int r, c;

  for (c = 0; c < 4; c++) {

    for (r = 0; r < 4; r++) {

      t[r] = temp[r][c];

    }

    for (r = 0; r < 4; r++) {

      temp[r][c] = ff\_mul(0x02, t[r]) ^ ff\_mul(0x03, t[(r + 1) % 4]) ^

                   ff\_mul(0x01, t[(r + 2) % 4]) ^ ff\_mul(0x01, t[(r + 3) % 4]);

    }

  }

}

// 子密钥加

void add\_round\_key(uint8\_t temp[][4], uint8\_t k[][4]) {

  int r, c;

  for (c = 0; c < 4; c++) {

    for (r = 0; r < 4; r++) {

      temp[r][c] ^= k[r][c];

    }

  }

}

// 逆子密钥加

void inv\_sub\_bytes(uint8\_t temp[][4]) {

  int r, c;

  for (r = 0; r < 4; r++) {

    for (c = 0; c < 4; c++) {

      temp[r][c] = InvSbox[temp[r][c]];

    }

  }

}

// 逆行位移

void inv\_shift\_rows(uint8\_t temp[][4]) {

  uint8\_t t[4];

  int r, c;

  for (r = 1; r < 4; r++) {

    for (c = 0; c < 4; c++) {

      t[c] = temp[r][(c - r + 4) % 4];

    }

    for (c = 0; c < 4; c++) {

      temp[r][c] = t[c];

    }

  }

}

// 逆行位移

void inv\_mix\_columns(uint8\_t temp[][4]) {

  uint8\_t t[4];

  int r, c;

  for (c = 0; c < 4; c++) {

    for (r = 0; r < 4; r++) {

      t[r] = temp[r][c];

    }

    for (r = 0; r < 4; r++) {

      temp[r][c] = ff\_mul(0x0e, t[r]) ^ ff\_mul(0x0b, t[(r + 1) % 4]) ^

                   ff\_mul(0x0d, t[(r + 2) % 4]) ^ ff\_mul(0x09, t[(r + 3) % 4]);

    }

  }

}

void out\_put(uint8\_t\* temp) {

  for (int i = 0; i < 16; i++) {

    printf("%02X ", temp[i]);

  }

  std::cout << '\n';

}

int main(int argc, char\* argv[]) {

  uint8\_t input[] = "woaimimaxuehahah";

  uint8\_t key[] = "woxiangqiaodaima";

  key\_init(key);

  std::cout << "input: " << reinterpret\_cast<char\*>(input) << "\n";

  std::cout << "key: " << reinterpret\_cast<char\*>(key) << "\n";

  encode(input);

  std::cout << "this is encoded input!\n";

  out\_put(input);

  std::cout << "\n";

  decode(input);

  std::cout << "this is decoded input!\n";

  out\_put(input);

  std::cout << reinterpret\_cast<char\*>(input) << '\n';

  return 0;

}

1. 两组输入输出数据
   1. 输入输出数据1

input: woaimimaxuehahah

key: woxiangqiaodaima

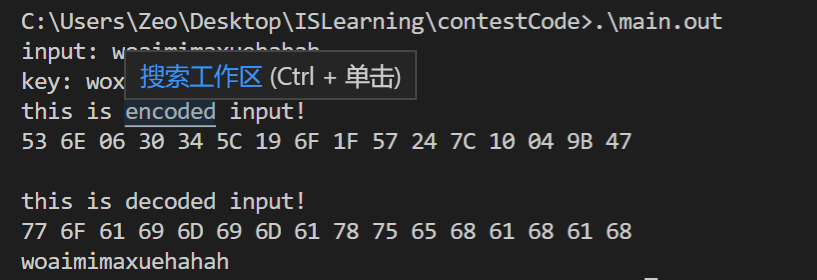
this is encoded input!

53 6E 06 30 34 5C 19 6F 1F 57 24 7C 10 04 9B 47

this is decoded input!

77 6F 61 69 6D 69 6D 61 78 75 65 68 61 68 61 68

Woaimimaxuehahah



* 1. 输入输出数据2

input: 1234567887654321

key: 1234567887654321

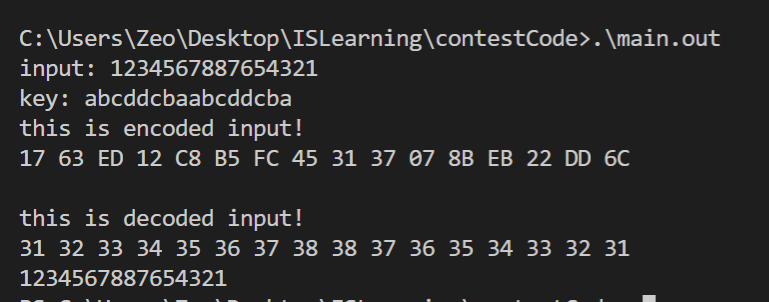
encode done!

17 63 ED 12 C8 B5 FC 45 31 37 07 8B EB 22 DD 6C

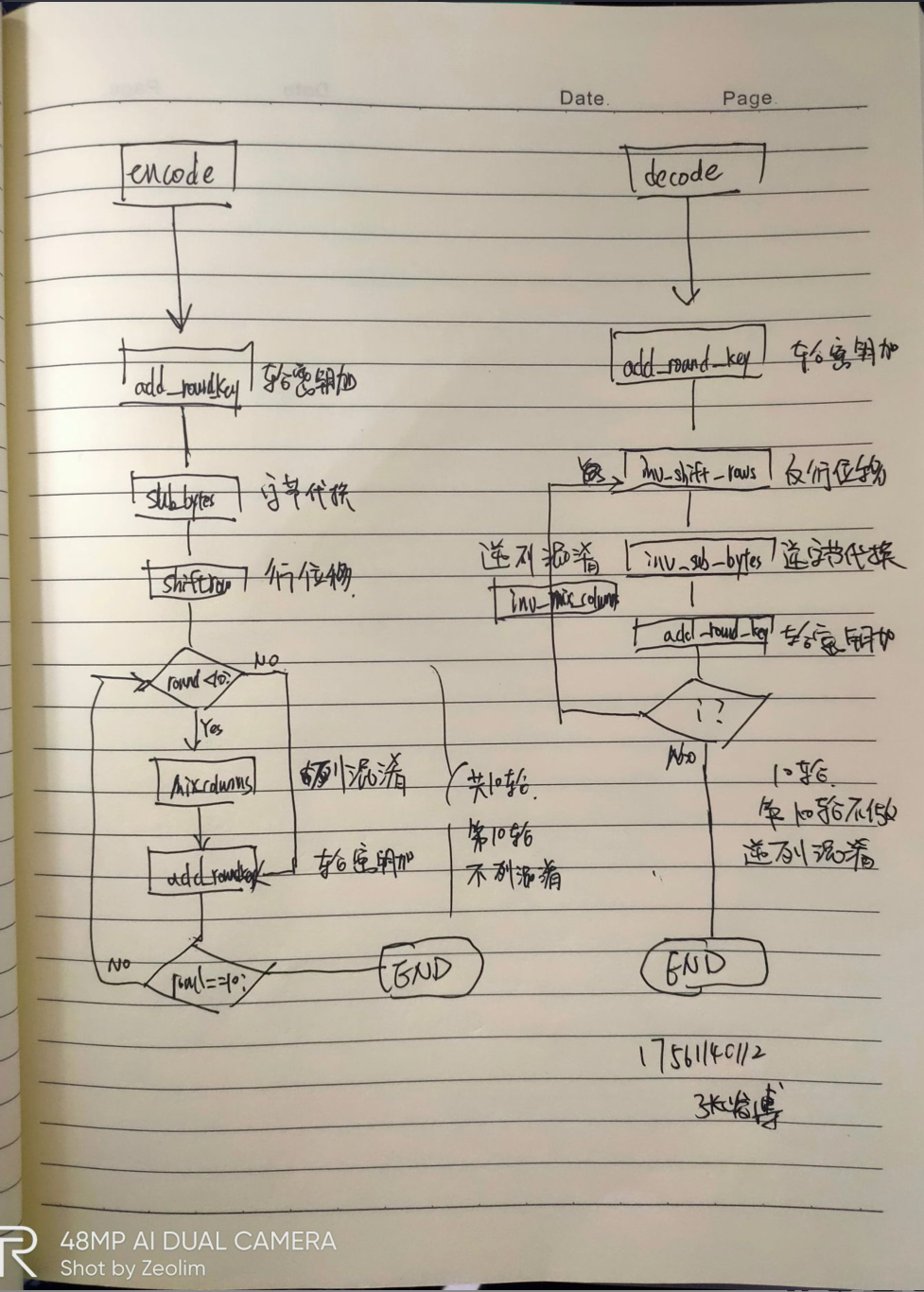
decode done!

31 32 33 34 35 36 37 38 38 37 36 35 34 33 32 31

1234567887654321



1. 流程图



|  |  |
| --- | --- |
| 得分 |  |
| 评卷人 |  |

二、编码/解码题（40分）。（请保持本题干放在页头）

请人工解码以下Base64码：“U2VDdVIxdFk=”（不包括双引号），求解码得到数据的ascii字符。要求写出解码结果及计算过程，手写拍照粘图（图片清晰整洁、大小适中），只有结果没有过程不得分。

