yxwkaifa
知识改变命运

公告

昵称：yxwkaifa
园龄：2年10个月
粉丝：21
关注：0
+加关注

<	2017年4月					>
日	一	二	三	四	五	六
26	27	28	29	30	31	1
2	3	4	5	6	7	8
9	10	11	12	13	14	15
16	17	18	19	20	21	22
23	24	25	26	27	28	29
30	1	2	3	4	5	6

搜索

找找看
谷歌搜索

常用链接

我的随笔
我的评论
我的参与
最新评论
我的标签

随笔档案

2016年4月 (322)
2016年3月 (325)
2016年2月 (251)
2016年1月 (387)
2015年12月 (273)
2015年10月 (105)
2015年9月 (274)
2015年8月 (242)
2015年7月 (340)
2015年6月 (229)
2015年5月 (39)
2014年11月 (175)
2014年10月 (302)
2014年9月 (87)
2014年8月 (252)
2014年7月 (416)
2014年6月 (75)
2014年5月 (2)

最新评论

1. Re:PCB设计资料：看到最后才知道是福利
厉害了我的楼主大大
--_start

2. Re:做高通平台安卓驱动感言
你好 ，看到你的经验想向你多学习，看到后能否加我qq： 2856763420
--FengShen

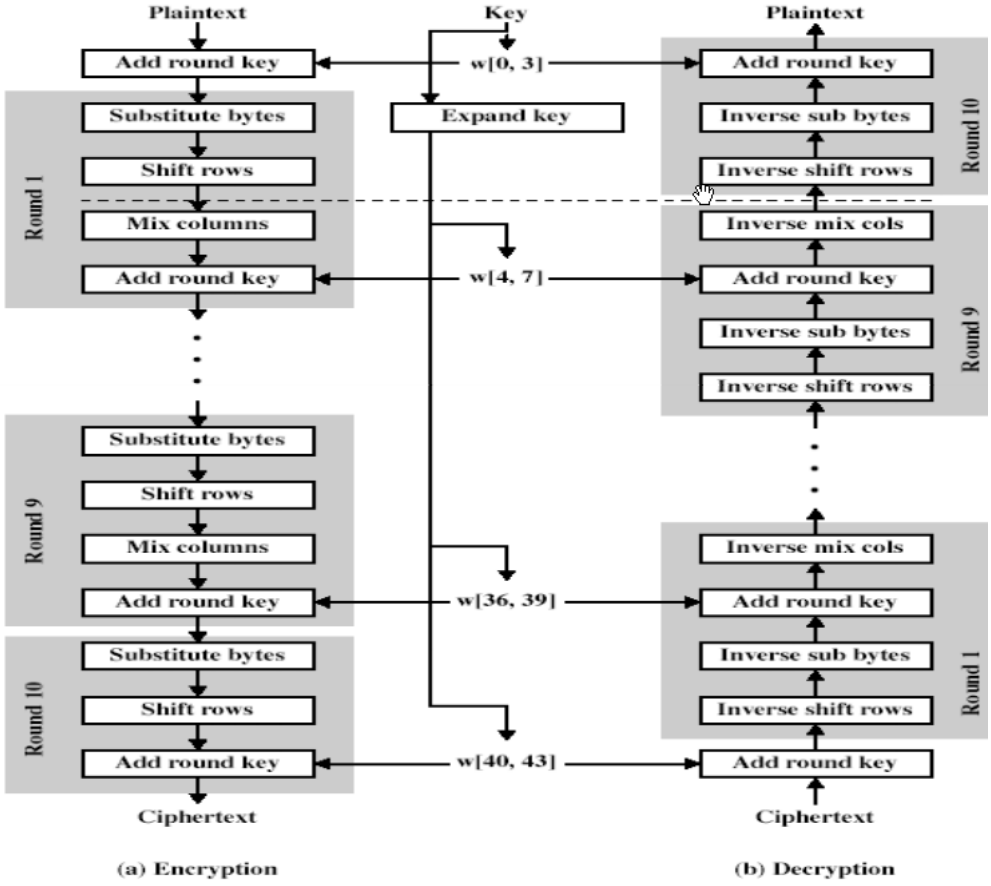
3. Re:.net web开发平台- 表单设计器 一(web版)
在吗，最近正在为这个自定义报表发愁，能分享一下吗，买也可以的？ 470706214@qq.com
--禅道

4. Re:Java SE学习之printf 日期转换符
我将计算机的语言切换成英语后，运行结果和上面的表一样了
--羽翼未丰之冲天槊

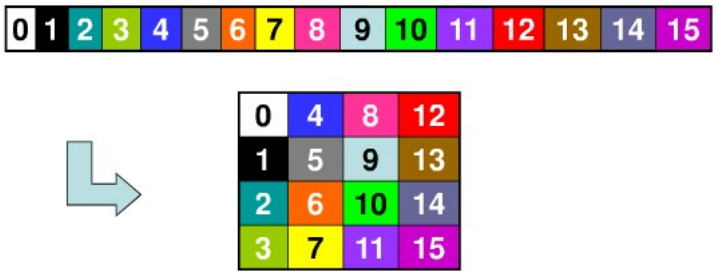
AES加密算法(C++实现,附源代码)

先搞定AES算法，基本变换包含SubBytes（字节替代）、ShiftRows（行移位）、MixColumns（列混淆）、AddRoundKey(轮密钥加)

其算法一般描写叙述为



明文及密钥的组织排列方式



ByteSubstitution （字节替代）

非线性的字节替代，单独处理每一个字节：

求该字节在有限域GF(2^8)上的乘法逆，"0"被映射为自身，即对于α∈GF(2^8)，求β∈GF(2^8)，

使得α·β=β·α=1mod($x^8+x^4+x^2+x+1$)。

对上一步求得的乘法逆作仿射变换

$y_i=x_i + x_{(i+4)mod8} + x_{(i+6)mod8} + x_{(i+7)mod8} + c_i$

(当中c_i是63_{10}即01100011_2的第i位），用矩阵表示为

$$\begin{bmatrix} y_0 \\ y_1 \\ y_2 \\ y_3 \\ y_4 \\ y_5 \\ y_6 \\ y_7 \end{bmatrix} = \begin{bmatrix} 1 & 1 & 1 & 1 & 1 & 0 & 0 & 0 \\ 0 & 1 & 1 & 1 & 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 & 1 & 1 & 1 & 0 \\ 0 & 0 & 0 & 1 & 1 & 1 & 1 & 1 \\ 1 & 0 & 0 & 0 & 1 & 1 & 1 & 1 \\ 1 & 1 & 0 & 0 & 0 & 1 & 1 & 1 \\ 1 & 1 & 1 & 0 & 0 & 0 & 1 & 1 \\ 1 & 1 & 1 & 1 & 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x_0 \\ x_1 \\ x_2 \\ x_3 \\ x_4 \\ x_5 \\ x_6 \\ x_7 \end{bmatrix} + \begin{bmatrix} 0 \\ 1 \\ 1 \\ 0 \\ 0 \\ 0 \\ 1 \\ 1 \end{bmatrix}$$

本来打算把求乘法逆和仿射变换算法敲上去，最后还是放弃了...直接打置换表

```
unsigned
char

sBox[] =

{
/*
 0   1   2   3   4   5   6   7   8   9   a   b   c   d   e   f */

    0x63,0x7c,0x77,0x7b,0xf2,0x6b,0x6f,0xc5,0x30,0x01,0x67,0x2b,0xfe,0xd7,0xab,0x76,
/*0*/

    0xca,0x82,0xc9,0x7d,0xfa,0x59,0x47,0xf0,0xad,0xd4,0xa2,0xaf,0x9c,0xa4,0x72,0xc0,
/*1*/

    0xb7,0xfd,0x93,0x26,0x36,0x3f,0xf7,0xcc,0x34,0xa5,0xe5,0xf1,0x71,0xd8,0x31,0x15,
/*2*/

    0x04,0xc7,0x23,0xc3,0x18,0x96,0x05,0x9a,0x07,0x12,0x80,0xe2,0xeb,0x27,0xb2,0x75,
/*3*/

    0x09,0x83,0x2c,0x1a,0x1b,0x6e,0x5a,0xa0,0x52,0x3b,0xd6,0xb3,0x29,0xe3,0x2f,0x84,
/*4*/

    0x53,0xd1,0x00,0xed,0x20,0xfc,0xb1,0x5b,0x6a,0xcb,0xbe,0x39,0x4a,0x4c,0x58,0xcf,
/*5*/

    0xd0,0xef,0xaa,0xfb,0x43,0x4d,0x33,0x85,0x45,0xf9,0x02,0x7f,0x50,0x3c,0x9f,0xa8,
/*6*/

    0x51,0xa3,0x40,0x8f,0x92,0x9d,0x38,0xf5,0xbc,0xb6,0xda,0x21,0x10,0xff,0xf3,0xd2,
/*7*/

    0xcd,0x0c,0x13,0xec,0x5f,0x97,0x44,0x17,0xc4,0xa7,0x7e,0x3d,0x64,0x5d,0x19,0x73,
/*8*/

    0x60,0x81,0x4f,0xdc,0x22,0x2a,0x90,0x88,0x46,0xee,0xb8,0x14,0xde,0x5e,0x0b,0xdb,
/*9*/

    0xe0,0x32,0x3a,0x0a,0x49,0x06,0x24,0x5c,0xc2,0xd3,0xac,0x62,0x91,0x95,0xe4,0x79,
/*a*/

    0xe7,0xc8,0x37,0x6d,0x8d,0xd5,0x4e,0xa9,0x6c,0x56,0xf4,0xea,0x65,0x7a,0xae,0x08,
/*b*/

    0xba,0x78,0x25,0x2e,0x1c,0xa6,0xb4,0xc6,0xe8,0xdd,0x74,0x1f,0x4b,0xbd,0x8b,0x8a,
/*c*/

    0x70,0x3e,0xb5,0x66,0x48,0x03,0xf6,0x0e,0x61,0x35,0x57,0xb9,0x86,0xc1,0x1d,0x9e,
```

```
/*d*/

    0xe1,0xf8,0x98,0x11,0x69,0xd9,0x8e,0x94,0x9b,0x1e,0x87,0xe9,0xce,0x55,0x28,0xdf,
/*e*/

    0x8c,0xa1,0x89,0x0d,0xbf,0xe6,0x42,0x68,0x41,0x99,0x2d,0x0f,0xb0,0x54,0xbb,0x16
/*f*/

};
```

以下是逆置换表，解密时使用

```
unsigned
char

invsBox[256] =

{
/*
 0    1    2    3    4    5    6    7    8    9    a    b    c    d    e    f */

    0x52,0x09,0x6a,0xd5,0x30,0x36,0xa5,0x38,0xbf,0x40,0xa3,0x9e,0x81,0xf3,0xd7,0xfb,
/*0*/

    0x7c,0xe3,0x39,0x82,0x9b,0x2f,0xff,0x87,0x34,0x8e,0x43,0x44,0xc4,0xde,0xe9,0xcb,
/*1*/

    0x54,0x7b,0x94,0x32,0xa6,0xc2,0x23,0x3d,0xee,0x4c,0x95,0x0b,0x42,0xfa,0xc3,0x4e,
/*2*/

    0x08,0x2e,0xa1,0x66,0x28,0xd9,0x24,0xb2,0x76,0x5b,0xa2,0x49,0x6d,0x8b,0xd1,0x25,
/*3*/

    0x72,0xf8,0xf6,0x64,0x86,0x68,0x98,0x16,0xd4,0xa4,0x5c,0xcc,0x5d,0x65,0xb6,0x92,
/*4*/

    0x6c,0x70,0x48,0x50,0xfd,0xed,0xb9,0xda,0x5e,0x15,0x46,0x57,0xa7,0x8d,0x9d,0x84,
/*5*/

    0x90,0xd8,0xab,0x00,0x8c,0xbc,0xd3,0x0a,0xf7,0xe4,0x58,0x05,0xb8,0xb3,0x45,0x06,
/*6*/

    0xd0,0x2c,0x1e,0x8f,0xca,0x3f,0x0f,0x02,0xc1,0xaf,0xbd,0x03,0x01,0x13,0x8a,0x6b,
/*7*/

    0x3a,0x91,0x11,0x41,0x4f,0x67,0xdc,0xea,0x97,0xf2,0xcf,0xce,0xf0,0xb4,0xe6,0x73,
/*8*/

    0x96,0xac,0x74,0x22,0xe7,0xad,0x35,0x85,0xe2,0xf9,0x37,0xe8,0x1c,0x75,0xdf,0x6e,
/*9*/

    0x47,0xf1,0x1a,0x71,0x1d,0x29,0xc5,0x89,0x6f,0xb7,0x62,0x0e,0xaa,0x18,0xbe,0x1b,
/*a*/

    0xfc,0x56,0x3e,0x4b,0xc6,0xd2,0x79,0x20,0x9a,0xdb,0xc0,0xfe,0x78,0xcd,0x5a,0xf4,
/*b*/

    0x1f,0xdd,0xa8,0x33,0x88,0x07,0xc7,0x31,0xb1,0x12,0x10,0x59,0x27,0x80,0xec,0x5f,
/*c*/

    0x60,0x51,0x7f,0xa9,0x19,0xb5,0x4a,0x0d,0x2d,0xe5,0x7a,0x9f,0x93,0xc9,0x9c,0xef,
/*d*/
```

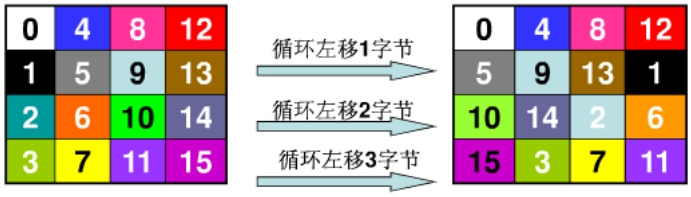
```
    0xa0,0xe0,0x3b,0x4d,0xae,0x2a,0xf5,0xb0,0xc8,0xeb,0xbb,0x3c,0x83,0x53,0x99,0x61,
/*e*/

    0x17,0x2b,0x04,0x7e,0xba,0x77,0xd6,0x26,0xe1,0x69,0x14,0x63,0x55,0x21,0x0c,0x7d
/*f*/

};
```

这里遇到问题了，本来用纯c初始化数组非常正常，封装成类以后发现不能初始化，无论是声明、构造函数都无法初始化，百歌谷度了一通后没有不论什么答案，无奈仅仅能在构造函数中声明一个局部变量数组并初始化，然后用memcpy，（成员变量名为Sbox/InvSbox，局部变量名sBox/invsBox）

```
void
AES::SubBytes(unsigned char
state[][4])
{
    int
r,c;
    for(r=0;
 r<4; r++)
    {
        for(c=0;
 c<4; c++)
        {
            state[r][c]
 = Sbox[state[r][c]];
        }
    }
}
```

ShiftRows（行移位变换）

行移位变换完毕基于行的循环位移操作，变换方法：



即行移位变换作用于行上，第0行不变，第1行循环左移1个字节，第2行循环左移2个字节，第3行循环左移3个字节。

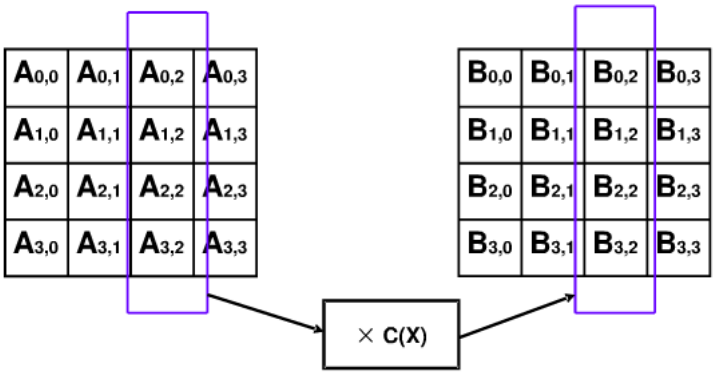
```
void
AES::ShiftRows(unsigned char
```

```
state[][4])
{
    unsigned char t[4];
    int r,c;
    for(r=1; r<4; r++)
    {
        for(c=0; c<4; c++)
        {
            t[c] = state[r][(c+r)%4];
        }
        for(c=0; c<4; c++)
        {
            state[r][c] = t[c];
        }
    }
}
```

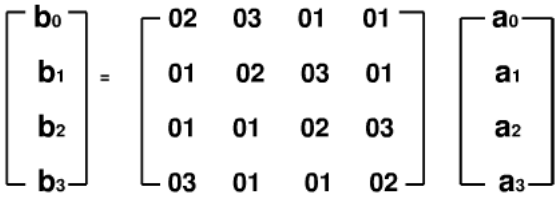
MixColumns（列混淆变换）

逐列混合，方法：

$b(x) = (03 \cdot x^3 + 01 \cdot x^2 + 01 \cdot x + 02) \cdot a(x) \bmod (x^4 + 1)$



矩阵表示形式：




```
void
AES::MixColumns(unsigned char
state[][4])
{
    unsigned
char
t[4];
    int
r,c;
    for(c=0;
 c< 4; c++)
    {
        for(r=0;
 r<4; r++)
        {
            t[r]
 = state[r][c];
        }
        for(r=0;
 r<4; r++)
        {
            state[r][c]
 = FFmul(0x02, t[r])
                        ^
 FFmul(0x03, t[(r+1)%4])
                        ^
 FFmul(0x01, t[(r+2)%4])
                        ^
 FFmul(0x01, t[(r+3)%4]);
        }
    }
}
```

```
29
30  unsigned char
31  AES::FFmul(unsigned char
32  a, unsigned char
33  b)
34  {
35      unsigned char
36  bw[4];
37      unsigned char
38  res=0;
39      int
40  i;
41      bw[0] = b;
42      for(i=1; i<4; i++)
43      {
            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;

}
```

当中FFmul为有限域GF(2^8)上的乘法，标准算法应该是循环8次（b与a的每一位相乘，结果相加），但这里仅仅用到最低2位，解密时用到的逆列混淆也仅仅用了低4位，所以在这里高4位的运算是多余的，仅仅计算低4位。

AddRoundKey （轮密钥加变换）

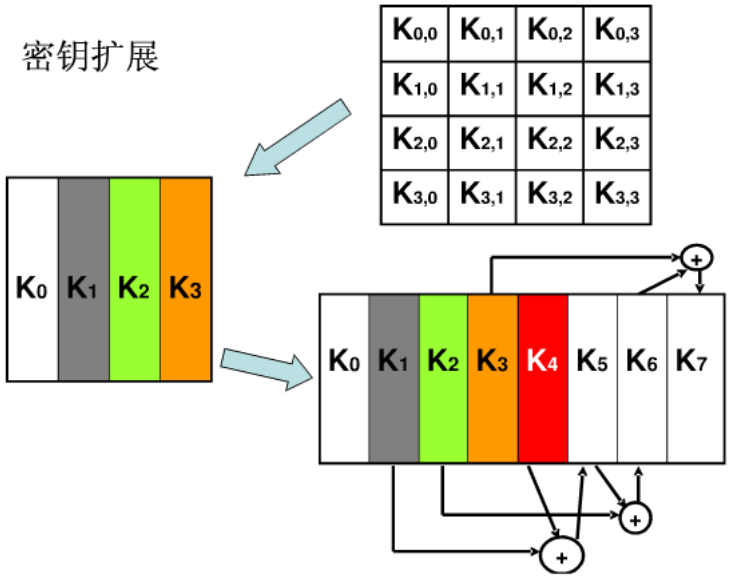
简单来说就是逐字节相加，有限域GF(2^8)上的加法是模2加法，即异或

```
1  void
2  AES::AddRoundKey(unsigned char
3  state[][4], unsigned char
4  k[][4])
5  {
6      int
7  r,c;
8      for(c=0;
    c<4; c++)
9      {
10         for(r=0;
11 r<4; r++)
           {
               state[r][c]
    ^= k[r][c];
           }
       }
   }
```

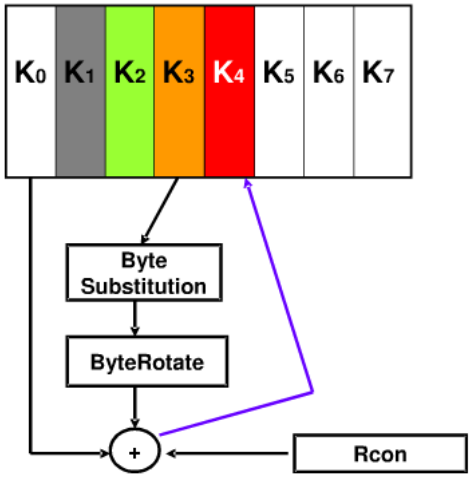
KeyExpansion （密钥扩展）

将输入的密钥扩展为11组128位密钥组，当中第0组为输入密钥本身

其后第n组第i列 为 第n-1组第i列 与 第n组第i-1列之和（模2加法，1<= i <=3）



对于每一组 第一列即i=0，有特殊的处理



将前一列即第n-1组第3列的4个字节循环左移1个字节，
并对每一个字节进行字节替代变换SubBytes
将第一行（即第一个字节）与轮常量rc[n]相加
最后再与前一组该列相加

```
1  void
2  AES::KeyExpansion(unsigned char*
    key, unsigned char
3  w[][4][4])
4  {
5      int
6  i,j,r,c;
7      unsigned
8  char
9  rc[] = {0x01, 0x02, 0x04, 0x08, 0x10, 0x20, 0x40, 0x80, 0x1b, 0x36};
10     for(r=0;
    r<4; r++)
```

```
11      {
12          for(c=0;
 c<4; c++)
13          {
14              w[0][r][c]
15  = key[r+c*4];
16          }
17      }
18      for(i=1;
 i<=10; i++)
19      {
20          for(j=0;
21  j<4; j++)
22          {
23              unsigned
 char
24  t[4];
25              for(r=0;
26  r<4; r++)
27              {
28                  t[r]
 = j ? w[i][r][j-1] : w[i-1][r][3];
29              }
30              if(j
31  == 0)
32              {
33                  unsigned
 char
34  temp = t[0];
35                  for(r=0;
36  r<3; r++)
37                  {
                        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];

            }

        }

    }

}
```

解密的基本运算

AES解密算法与加密不同，基本运算中除了AddRoundKey（轮密钥加）不变外，其余的都须要进行逆变换，即

InvSubBytes（逆字节替代）、InvShiftRows（逆行移位）、InvMixColumns（逆列混淆）

```
1  void
2  AES::InvSubBytes(unsigned char
3  state[][4])
4  {
5      int
6  r,c;
7      for(r=0;
    r<4; r++)
8      {
9          for(c=0;
10     c<4; c++)
11         {
12             state[r][c]
    = InvSbox[state[r][c]];
13         }
14     }
15 }
16
17 void
18 AES::InvShiftRows(unsigned char
19 state[][4])
```

```
{
    unsigned char t[4];
    int r,c;
    for(r=1; r<4; r++)
    {
        for(c=0; c<4; c++)
        {
            t[c] = state[r][(c-r+4)%4];
        }
        for(c=0; c<4; c++)
        {
            state[r][c] = t[c];
        }
    }
}

void AES::InvMixColumns(unsigned char state[][4])
{
    unsigned char t[4];
    int r,c;
    for(c=0; c< 4; c++)
    {
```

```
        for(r=0;
 r<4; r++)

        {

            t[r]
 = state[r][c];

        }

        for(r=0;
 r<4; r++)

        {

            state[r][c]
 = FFmul(0x0e, t[r])

                          ^
 FFmul(0x0b, t[(r+1)%4])

                          ^
 FFmul(0x0d, t[(r+2)%4])

                          ^
 FFmul(0x09, t[(r+3)%4]);

        }

    }

}
```

加密过程

先将输入的明文按列序组合成4*4的矩阵，直接与第0组密钥（即输入的密钥）相加（异或），作为轮加密的输入

然后循环10次进行SubBytes、ShiftRows、MixColumns、AddRoundKey运算，最后恢复原序列

须要注意的是最后一轮并不进行MixColumns（列混淆变换）

```
unsigned
char*
 AES::Cipher(unsigned char*
 input)

{

    unsigned
char

state[4][4];

    int

i,r,c;



    for(r=0;
 r<4; r++)

    {
```

```
12          for(c=0;
    c<4 ;c++)
13          {
14              state[r][c]
15      = input[c*4+r];
16          }
17      }
18
19      AddRoundKey(state,w[0]);
20
21      for(i=1;
    i<=10; i++)
22      {
23          SubBytes(state);
24          ShiftRows(state);
25          if(i!=10)MixColumns(state);
26          AddRoundKey(state,w[i]);
27      }
28
29      for(r=0;
    r<4; r++)
31      {
32          for(c=0;
    c<4 ;c++)
33          {
                input[c*4+r]
    = state[r][c];
            }
        }

        return
    input;
    }
```

解密过程

```
unsigned char* AES::InvCipher(unsigned char* input)
{
    unsigned char state[4][4];
    int i,r,c;

    for(r=0; r<4; r++)
    {
        for(c=0; c<4 ;c++)
        {
            state[r][c] = input[c*4+r];
        }
    }

    AddRoundKey(state, w[10]);
    for(i=9; i>=0; i--)
    {
        InvShiftRows(state);
        InvSubBytes(state);
        AddRoundKey(state, w[i]);
        if(i)InvMixColumns(state);
    }

    for(r=0; r<4; r++)
    {
        for(c=0; c<4 ;c++)
```

```
            {

                input[c*4+r]
 = state[r][c];

            }

        }



        return

    input;

    }
```

对外部数据的加密/解密

至此已经实现了AES加密与解密的原型，在使用的时候一般处理的是字符串等，而不是直接传入128位的数据，所以要封装一下对外部数据的加解密处理

```
1  void*
    AES::Cipher(void*
2   input, int

3  length)

4  {

5      unsigned
   char*
6   in = (unsigned char*)
    input;
7
       int

8
   i;

9
       if(!length)

10
       {

11
           while(*(in+length++));

12
           in
13  = (unsigned char*)
    input;

14
       }

15
       for(i=0;
16  i<length; i+=16)

17     {

18         Cipher(in+i);

19     }

20     return

21 input;
```

```
}


void*
 AES::InvCipher(void*
 input, int

length)

{

    unsigned
char*
 in = (unsigned char*)
 input;

    int

i;

    for(i=0;
 i<length; i+=16)

    {

        InvCipher(in+i);

    }

    return

input;

}
```


yxwkaifa
关注 - 0
粉丝 - 21
+加关注

« 上一篇：HDU 4293 Groups (线性dp)
» 下一篇：【Bootstrap3.0建站笔记三】AspNetPager分页，每一列都可排序

posted @ 2014-07-08 18:49yxwkaifa 阅读(269) 评论(0) 编辑 收藏