中山大学移动信息工程学院移动网络安全实验报告
(2016学年春季学期)

刘爽
班级：13M2
ID:13354222
专业：移动互联网

2016年3月18日

1 实验题目

CBC/CTR解密。

1.1 实验要求

利用AES加/解密算法，实现CBC/CTR解密系统。

2 实验原理

AES（Advanced Encryption Standard）又叫Rijndael加密法，用来替代DES算法。常见AES加密模式有ECB、CBC、CFB、OFB和CTR等五种，CFB、OFB都带反馈，多用于流加密，CBC和CTR、ECB多用于独立block加密，由于ECB算法有缺陷（相同输入，相同输出，容易明文攻击），所以CBC和CTR加解密方式用的较多，也是很多标准规范要求的实现算法。
另外，AES跟Rijndael相比有点区别，AES使用的block size为16 bytes，为固定大小，而Rijndael大小更加灵活。AES的密钥长度支持128、192 或256 bits。

2.1 CBC原理

2.1.1 CBC的加密

总的来说，CBC的加密过程是明文(Plaintext)跟向量(Initialization Vector)异或，再用KEY进行加密，结果作为下个BLOCK的初始化向量。如Fig.1 所示。

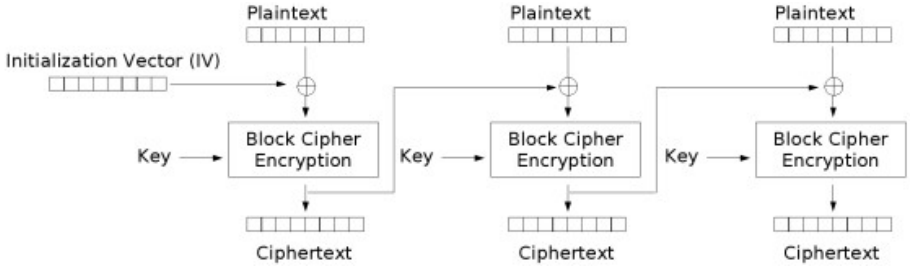

Figure 1: CBC Mode Encryption

2.1.2 CBC的解密

总的来说，CBC的解密过程是使用密钥先对密文解密，解密后再同初始向量异或得到明文。如Fig.2所示。

2.1.3 CBC的Padding

在这里使用PCKS7来进行Padding。即如果明文的长度不是16 Bytes的整数倍，则将长度取十六的整数倍的上界，缺少多少则补上的数字是多少。如果长

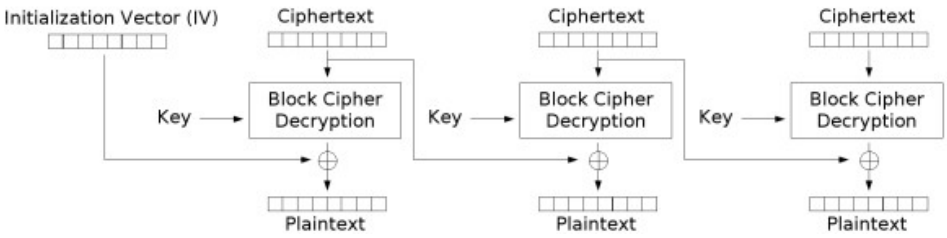

Figure 2: CBC Mode Decryption

度是16 Bytes的整数倍，则将长度增加16，并补上数字0x16。具体如Fig.3所示。

```
01
02 02
03 03 03
04 04 04 04
05 05 05 05 05
etc.
```

Figure 3: PCKS7 Padding

2.1.4 CBC的特点

CBC需要对明文块大小进行Padding（补位），由于前后加密的相关性，只能实施串行化动作，无法并行运算。另外，CBC需要参量：密钥（KEY）和初始化向量(Initialization Vector)。

2.2 CTR原理

2.2.1 CTR的加密

总的来说，CTR的解密过程是用密钥(KEY)对输入的计数器(Nounce Counter)加密，然后同明文(Plaintext)异或得到密文。具体如Fig.4 所示。

2.2.2 CTR的解密

总的来说，CBC的解密过程是用密钥(KEY)对输入计数器(Nounce Counter)加密，然后同密文(Ciphertext)异或得到明文。具体如Fig.5 所示。

2.2.3 CTR的Padding

在这里直接使用0x0来进行Padding，如果Plaintext的长度不为十六的整数倍的话，则将长度改为大于原有长度的16的整数倍的最小上界，并用0x0补充缺少的部分。

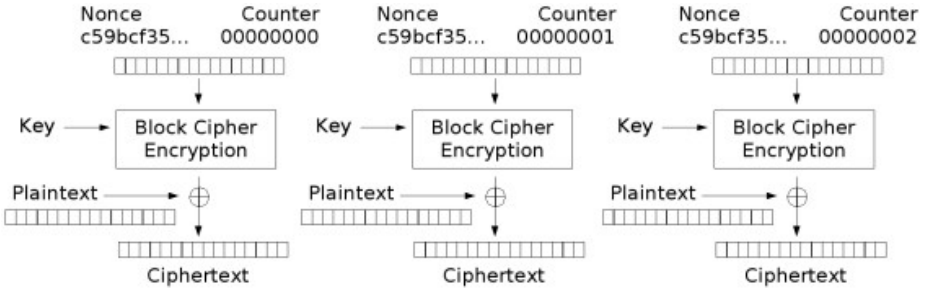

Figure 4: CTR Mode Encryption

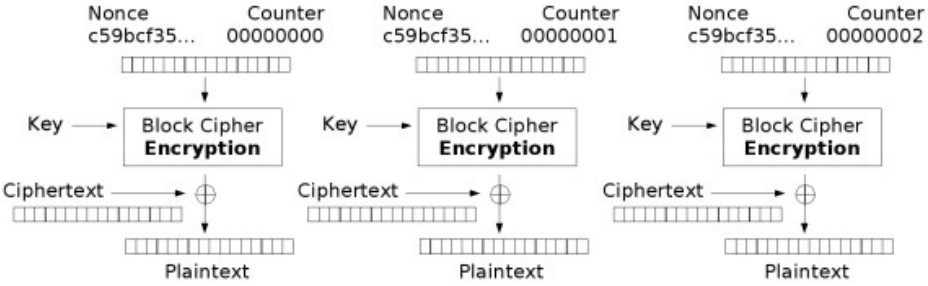

Figure 5: CTR Mode Decryption

2.2.4 CTR的特点

CTR不需要Padding，而且采用了流密钥方式加解密，适合于并行运算，C-TR涉及参量：Nounce随机数、Counter计数器和密钥。Nounce随机数和Counter计数器整体可看作计数器。只要算法约定好，就可以回避掉串行化运算。

3 实验内容

分析程序的整个实现过程，讲述每个函数的具体作用并附上关键代码部分。

3.1 变量初始化

所有的KEY, Ciphertext, Plaintext以及PT和CT的长度都在函数外进行初始化，当作全局变量使用。用Byte类型的数组储存每一个数，每个数都是由两个Byte组成，数组的每一个位置都储存这样一个数。因为在计算机语言中所有的值都是以二进制组成的，因而在之后的计算中直接异或就可以。

3.2 CBC_Decryption()函数

这个函数主要用于解密用CBC加密的密文。
具体过程包括：
-先进行补位操作。
-将初始化的KEY进行扩展。

-得到要处理的数据组数。
-在每个循环中，取16个需要解密的密文，对IV进行更新，用封装的AES_Decryption进行解密，将得到的结果与更新后的IV 进行异或操作。将解密后的结果存进专门储存明文的数据结构中,注意需要去除补位的值。
关键代码如Fig.6所示。

```
cout << "Now preparing to decrypt the CBC cipher texts..." << endl;
for(int i = 1;i < loop;i++)
{
    //get new 16 byte cipher texts
    Byte temp_CT[4][4] =
    {{CBCCT[16*i + 0], CBCCT[16*i + 4],  CBCCT[16*i + 8],  CBCCT[16*i + 12]},
    {CBCCT[16*i + 1],  CBCCT[16*i + 5],  CBCCT[16*i + 9],  CBCCT[16*i + 13]},
    {CBCCT[16*i + 2],  CBCCT[16*i + 6],  CBCCT[16*i + 10], CBCCT[16*i + 14]},
    {CBCCT[16*i + 3],  CBCCT[16*i + 7],  CBCCT[16*i + 11], CBCCT[16*i + 15]}};
    Byte temp_IV[16];
    //initilize the initial vector
    for(int j = 0;j < 16;j++){
        temp_IV[j] = CBCCT[16*(i-1)+j];
    }
    AES_Decryption(temp_CT, fullCBCKey);
    xor_Oper(temp_CT, temp_IV);
    //assign the result to the array which stores the plain text
    for(int j = 0;j < 4;j++){
        for(int k = 0;k < 4;k++){
            CBCPT[(i-1)*16+j+k*4] = temp_CT[j][k];
        }
    }
}
```

Figure 6: Partial codes of CBC_Decryption()

3.3 CTR_Decryption()函数

这个函数主要用于解密用CTR加密的密文。
具体过程包括：
-先进行补位操作。
-将初始化的KEY进行扩展。
-初始化Nounce Counter。
-得到要处理的数据组数。
-在每个循环中，取16个需要解密的密文，对IV进行更新，用封装的AES_Encryption进行解密，将得到的结果与Nounce Counter进行异或操作。将解密后的结果存进专门储存明文的数据结构中。注意去除补位的值。如Fig.9所示。
-对Nounce Counter进行加一操作，注意需要进位的情况。

3.4 xor_Oper()函数

这个函数定义了一个二维矩阵和一个数组之间的异或操作。因为在计算机中所有的进制都是以二进制存储的，因而直接进行异或就可以。具体如Fig.7所示。

```
//Define the exclusive operation between two byte matrices
//the final result is assigned to the matrix a
void xor_Oper(Byte a[][4], Byte b[]){
    Byte temp[4][4];
    for(int i = 0;i < 4;i++){
        for(int j = 0;j <4;j++){
            temp[i][j] = a[i][j] ^ b[i+j*4];
        }
    }
    for(int i = 0;i < 4;i++){
        for(int j = 0;j < 4;j++){
            a[i][j] = temp[i][j];
        }
    }
}
```

Figure 7: xor_Oper()

4 实验结果与分析

4.1 实验结果

实验结果如Fig.8所示。

```
Phase1:CBC Decryption
Now preparing to do the padding...
Now preparing to expand the key...
Now preparing to decrypt the CBC cipher texts...
CBC Decryption done.
The plain text by CBC Decryption is as followed:

Action speaks louder than words☺

Phase2:CTR Decryption
Now preparing to do the padding...
Now preparing to expand the key...
Now preparing to initialize the nounce counter...
Now preparing to decrypt the CBC cipher texts...
CTR Decryption done.
The plain text by CTR Decryption is as followed:

Keep on going never give up.

Press any key to continue . . .
```

Figure 8: Experimental Result

4.2 结果分析

由Fig.8可知，由CBC加密的明文是：Action speaks louder than words.由CTR加密的明文是Keep on going never give up.
这两段密文都被完美的解密出来了。可见，代码的步骤都是正确的。

只要知道了KEY的值和Nounce Counter的值，就可以完整的解密出来。
另外，第一段密文解密出来的最后一位有个笑脸，原本以为解密过程有错误。但是前面的密文解密都没出错，因而推测那个笑脸是属于明文的一部分的。

5 实验感想

此次实验因为无需实现AES的过程，难度适中，有以下感想与收获。

1. 理解AES原理很重要，即使它不需要自己实现。在执行AES的加密函数时，输出的结果总是与预期结果不对，曾经一度以为是AES封装的函数出了错误。但是在查阅书籍之后，发现矩阵中储存的值不是从左到右来依次储存的，而是从上到下依次储存的。因为矩阵的格式出错，因而AES的加密结果一直出错，修正之后，问题解决。

2. 写代码的时候顺便输出每个过程的提示词对于之后的Debug很重要。一方面，对使用这个程序的人来说，他可以知道程序执行的步骤，从而对整个过程有更详细的了解；另一方面，在进行程序崩溃的Debug时，原先代码中的过程提示词可以帮助我们更快速地定位程序是在哪个部分出现问题的，从而省去了自己要从头一个一个地方Debug的麻烦。

3. 学会充分利用各种资源。有一段时间，程序输出的值总是乱码，而自己从代码本身的逻辑中找不出问题。后来通过群上上传的Tips得知需要解密的部分是从第17位开始的，如果将前16位算进去，则会影响后面的解密过程，而使得基本上所有的结果都是乱码。在修正这个错误之后，问题得到解决。

4. 写代码时，需要充分考虑各种情况。在程序的最后输出结果时，需要将原本补位的值去掉，否则那些值会变成乱码输出。另外，在进行Nounce Counter的加1操作时，需要考虑进位的情况，否则最终输出的结果可能是错的。

```
for(int i = 1;i < loop;i++)
{
    Byte temp_IV[4][4];
    Byte temp_CT[16];
    //get the new 16 bytes cipher texts
    for(int j = 0;j < 16;j++){
        temp_CT[j] = CTRCT[i*16+j];
    }
    //initialize the intitial vector
    for(int j = 0;j < 4;j++){
        for(int k = 0;k < 4;k++){
            temp_IV[j][k] = nounceCounter[j][k];
        }
    }
    AES_Encryption(temp_IV, fullCTRKey);
    xor_Oper(temp_IV, temp_CT);
    //assign the final result to the array which stores the plain text
    for(int j = 0;j < 4;j++){
        for(int k = 0;k < 4;k++){
            CTRPT[(i-1)*16+j+k*4] = temp_IV[j][k];
        }
    }
```

Figure 9: Partial codes of CTR_Decryption()

参考文献

[1] 天缘， http://www.metsky.com/archives/585.html