**《密码学》课程设计实验报告**

实验序号：02　　　　　　　　　　实验项目名称：分组密码AES

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 学　　号 |  | 姓　　名 |  | 专业、班 | | 18信安 |
| 实验地点 | B310 | 指导教师 | 王后珍 | 时间 | | 2020.11.30 |
| 1. 实验目的及要求   教学目的：   1. 掌握分组密码的基本概念； 2. 掌握AES密码算法； 3. 了解AES密码的安全性； 4. 掌握分组密码常用工作模式及其特点； 5. 熟悉分组密码的应用。   实验要求：   1. 熟悉AES算法的基本结构； 2. 掌握AES算法的基本运算； 3. 掌握AES算法的实现与优化方法； 4. 熟悉AES算法的安全性。   二、实验设备（环境）及要求  Windows操作系统，高级语言开发环境  三、实验内容与步骤  1. AES 算法的基本结构  输入：128位明文，128/192/256位密钥  子过程：  （1）S盒变换（教材 p86及p92表3-10）  （2）行移位（教材 p87表3-9）  （3）列混合（教材 p87及p93式3-32）  （4）轮密钥加（教材 p87）  输出：128位密文。    图：AES轮函数结构    图：AES轮密钥产生  2．AES算法的基本运算（重点）  方法：通过编程代码实现下列运算：  （1）上的加法（教材 p83定义3-2）  （为了描述方便，用花括号表示16进制，下同）  例：｛BC｝⊕｛6A｝=｛D6｝（下图中的A3,3⊕K3,3=B3,3）  计算或编程方法：按位异或（提示——C、Java等语言中的^运算符）  （2）上的多项式加法（教材 p83定义3-7）  例：a(x)={BC}x3+{42} x2+{9F} x+{4C}  K(x)= {6A}x3+{00} x2+{5C} x+{57}  a(x) ⊕K(x)= {D6}x3+{42} x2+{C3} x+{1B}  计算或编程方法：按位异或（提示——C、Java等语言中的^运算符）      对于AES中的轮密钥加运算，即可以表示为对应“字节”的加法，每格相加，即定义3-2；也可以表示为对应32位“字”的加法，每列相加，即定义3-7；甚至可以表示为整个128位“状态”的按位异或。  思考：在不同CPU架构下，哪种表示方法的执行速度最快？  （3）上的乘法（教材 p83定义3-8）  （a）借助xtime运算快速实现  原理：复习有限域的性质——分配率  对于｛02｝·｛？？｝（教材 p83定义3-5）定义为倍乘函数xtime，可以用移位运算和条件异或运算来快速实现。由于中的所有元素都可以表示为02的不同幂次的和，因此所有的乘法运算都能够通过重复调用倍乘函数xtime（定义3-5）和加法（定义3-2）快速实现。  例：y·｛15｝= y·｛01⊕04⊕10｝= y·｛01⊕022⊕024｝  = y⊕xtime(xtime(y))⊕xtime(xtime(xtime(xtime(y))))  = y⊕xtime(xtime(y⊕xtime(xtime(y))))  思考：该算法的效率分析？（最好情况、最坏情况）  改进：将xtime（y）的所有256种取值预计算，并造表。  （b）借助生成元快速实现  **GF**(2 8 )的全体非零元素对于乘法构成循环群。设a为生成元，则循环群  ***G*** ={a 0 ,a 1 ,…,a 254 }。  ***G***中的乘法运算  a p ·a q = a (p+q)mod 255 ，  于是可以把**GF**(2 8 )上的乘法简化为整数的加法运算。注意，零元素00与任何元素相乘都得00。  例：｛57｝·｛83｝=｛C1｝  计算或编程方法：  步骤1：（准备阶段）造表  预计算两个256字节的表：生成元为03的指数表（附表5）和生成元为03的对数表（附表6）  步骤2：查对数表  Log｛03｝｛57｝= 98 （注：指数表和对数表是16进制表述，高位-行号，低位-列号）  Log｛03｝｛83｝= 80  步骤3：｛57｝·｛83｝=｛03｝98·｛03｝80=｛03｝98+80 mod 255=｛03｝178  步骤4：查指数表｛03｝178=｛03｝｛B2｝=193=｛C1｝  思考：该算法的效率分析？（时间复杂度、空间复杂度）  （4）上的多项式乘法（教材 p83定义3-8、p93优化方案）  （a）AES中的列混合运算的实现  其中的运算按列（32位字）实现，当然也可表述为下面的4×4的字节矩阵相乘：    大家手工计算时，按列进行表述较为简单：    例如下面的列混合计算：    其中的第一列运算步骤为：    在GF(28)中，加法就是按位XOR操作，乘法是根据在上述方程所示的规则执行的。注意将某值乘上x(即{02})其结果就是将该值向左移一位， 如果该值的最高位为1，那么在移位后还要异或(0001 1011)。（参考xtime的快速实现方法）  对第一个方程，我们有{02}•{87} = (0000 1110) ⊕ (0001 1011)=(0001 0101)；{03}•{6E} = {6E}⊕ ({02}•{6E}) = (0110 1110) ⊕(1101 1100) =(1011 0010)。 于是：  {02}•{87} = 0001 0101  {03}•{6E} = 1011 0010  {46} = 0100 0110  {A6} = 1010 0110  0100 0111 = {47}  其它的方程也可以通过类似的方式得以验证。  （b）列混合运算的优化方案1  加密过程：c(x) = {03}x3 +{01}x2+{01}x+{02}  解密过程：d(x) ＝ {0B} x3+ {0D} x2+{09}x+{0E}  仅牵涉到与固定系数02，03，01，01以及0E,0B,0D,09,所以在需要提高速度而存储空间较大的应用中可以预先计算所有256×6个乘法（01不用计算），这样需要**1.5K字节**空间，但可省去大量乘法运算，这样可使MixColumn运算和InvMixColumn运算的乘法速度更快。  （c）列混合运算的优化方案2  定义四个新表，T 0 到T 3 ：  S[a] 02 S[a] 03  S[a] S[a] 02  T 0 ＝ S[a] T 1 ＝ S[a]  S[a] 03 S[a]  (3-40)  S[a] S[a]  S[a] 03 S[a]  T 2 ＝ S[a] 02 T 3 ＝ S[a] 03  S[a] S[a] 02  T 0 到T 3 中的每一个都是一个256个4字节元素的表，它们共占**4KB**的存储空间。  利用T 0 到T 3 ，可通过查表实现圈变换，于是式（3-39）变为：  e j ＝T 0 [a 0 , j]⊕T 1 [a 1 , j+c1]⊕T 2 [a 2 , j+c2]⊕T 3 [a 3 , j+c3]⊕k j  (3-41)  这样，加密算法圈变换中的每一列变换，可通过式（3-56）作4次查表和4次异或运算得到。  注意，在最后一圈中，没有MixColumn变换。这说明我们不能按式（3-41）来计算，而只能按式（3-34）、（3-36）和（3-37）来计算。  （d）在单片机、手机、PDA等资源受限环境下的实现  在8位CPU上，行移位、轮密钥加、S盒变换都是对字节（8位）操作，容易实现。但对于32位字的列混合操作，实现过程（CPU位宽、存储受限）如下：  输入：4个字节a[0]、a[1]、a[2]、a[3]；  输出：4个字节a[0]、a[1]、a[2]、a[3]；  加密过程：t= a[0]⊕a[1]⊕a[2]⊕a[3] ；  u= a[0] ；  v= a[0]⊕a[1]；v=xtime(v)；a[0]=a[0] ⊕v⊕t；  v= a[1]⊕a[2]；v=xtime(v)；a[1]=a[1] ⊕v⊕t；  v= a[2]⊕a[3]；v=xtime(v)；a[2]=a[2] ⊕v⊕t；  v= a[3]⊕u；v=xtime(v)；a[3]=a[3] ⊕v⊕t；  思考：1、该算法的效率分析？（时间复杂度、空间复杂度）  2、该算法的正确性证明？  3、AES的安全性  （1）AES的S盒的实现  最简单、高效的实现方案：造表（教材p92表3-10）  思考：该算法的效率分析？（时间复杂度、空间复杂度）  （2）编程研究AES的S盒的以下特性：  ①明文输入改变1位，密文输出平均改变多少位？  ②S盒输入改变1位，S盒输出平均改变多少位？  ③L输入改变1位，L输出平均改变多少位？  ④对于一个输入，连续施加S盒变换，变换多少次时出现输出等于输入？  4.扩展思考（教材习题）   1. 比较AES和DES，说明它们各有什么特点？ 2. AES的解密算法与加密算法有什么不同？ 3. 在GF（28）中，01的逆元素是什么？ 4. 在AES中，对于字节“ 00”和“ 01”计算S盒的输出。 5. 证明：模x4+1，c(x)与d(x)互逆。 6. 证明：xi mod (x4+1)=xi mod 4 。 7. 利用AES的对数表或反对数表计算ByteSub(25)。 8. 求出AES的 S盒的逆矩阵。 9. 设S是状态，W是圈密钥：    1. 证明：InvShiftRow(InvByteSub(S))= InvByteSub(InvShiftRow(S))。    2. 证明：InvMixColunm(S⊕W)= InvMixColunm(S) ⊕InvMixColunm(W)。    3. 说明上述结论对AES解密算法的设计有何作用。 10. 了解AES采用的SP（代替-置换）结构的特点。   5. 扩展练习（附加题）  题目1：S盒的安全性测试：对于AES S盒，计算其差分分布表和非线性度；  注: 差分分布表的定义是对于一对任意输入x1和x2,满足Δx=x1+ x2,输出等于Δy=y1+ y2的统计计数中的最大次数  例如对于Δx=0,则Δy=0出现256次,其余Δy=1,2,3…,255出现0次,则Δx=0的差分次数是256;  题目2：S盒的设计：产生新的S盒使其达到题1中的性质最优；  例：AES的S盒是计算输入X的逆，然后做仿射变换得出输出Y=AX-1+B= AX254+B。  尝试Y=AXC+B的形式,   1. C要求汉明重量为7(例如AES中254=11111110), 2. 新盒可以改变仿射变换使用的（满秩）矩阵A或向量B   给出结果,并计算其差分分布表和非线性度.  四、实验结果与数据处理  ***代码整体说明***  相关文件(rijndael.cpp/rijndael.hpp/utils.hpp/utils.cpp)  接口设计如下图，一共包含三个类，分别为   * ***RijndaelPredefinedArrays***：预先定义的表 * ***KeyExpansionHelper***：密钥扩展类 * ***RijnDael***：AES加解密类   其中每个类的接口定义分别如下(**蓝色框标出为公开方法**)    ***RijndaelPredefinedArrays***类接口    ***KeyExpansionHelper***类接口    ***Rijndael***类接口  程序的执行可以用如下的流程图来表示    AES流程图  ***核心代码说明***  首先是为了加速AES的速度**预先定义的一些表**    预先定义的表    接下来介绍AES中的S盒变换**ByteSub**。其原理如下图所示    ByteSub运算  注意到在上述运算中随着变化，因此我们可以提前造表来提高运算效率，它对应着***RijndaelPredefinedArrays::s\_box***，其内容如下所示    S盒表    同样的道理，由于轮常数***Rcon***与密钥长度无关，因此我们可以提前造表。它对应着***RijndaelPredefinedArrays::rcon***，其内容如下所示    rcon表    有了这两张表，我们的**密钥扩展**就能够轻松实现了，其核心代码如下    密钥扩展代码  在介绍加密算法之前，我们首先需要知道在轮函数中使用的四种变换   * ***ByteSub***：S盒变换 * ***ShiftRow***：行移位变换 * ***MixColumn***：列混合变换 * ***AddRoundKey***：轮密钥加变换   其中***ByteSub***在前面已经介绍过了，下面就不再介绍，而***AddRoundKey***只是简单的异或运算，这里也不再单独介绍，接下来我们主要详细介绍一下行移位变换***ShiftRow***以及列混合变换***MixColumn***的实现  **行移位变换**是对状态***state***的行进行循环移位变换。在数据块长度为128比特的AES中，第0行不移位，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，与其相关的代码如下所示    ShiftRow代码  **列混合变换**是AES算法中不好理解的一点，它把状态中的每一列看作上的多项式，并与一个固定多项式相乘然后模多项式，其中满足  由于与是互素的，因此我们在解密的时候只需要用其逆多项式做相应的逆变换即可进行解密  注意到，多项式的相乘可以写成矩阵形式    列混合变换矩阵形式  因此我们在写代码的时候模仿了矩阵的乘法    列混合变换代码    结合上述操作，可以得到AES的轮函数组成如下图所示    轮函数组成图    其中最后一轮的轮函数稍有不同，即没有列混合操作    final轮函数    由于AES算法不是**对合运算**，因此其解密算法与加密算法稍有不同，在具体的实现中，我们只需要对密文做加密算法中的**逆运算**  由此可以得到在解密运算中的轮函数由如下部件组成   * ***AddRoundKey***：轮密钥异或 * ***InvMixColumn***：逆列混合运算 * ***InvShiftRow***：逆行移位变换 * ***InvSubByte***：逆S盒变换   对应代码如下图所示    逆轮函数  由于逆变换和原变换的原理几乎一样，因此我们不再依次介绍逆变换的过程，只是给出其组成图    逆轮函数组成    同加密一样，最后一个逆轮函数没有逆列混合变换    最后逆轮函数组成  ***实验结果演示***  这里我们采用该[例子](https://kavaliro.com/wp-content/uploads/2014/03/AES.pdf)  其中密钥扩展后的结果为    密钥扩展结果  加解密的结果为    加解密结果    可以发现与其结果完全一致  五、分析与讨论   1. 和DES实验相比，由于AES不具有对合性，因此在实现上稍微复杂 2. 在AES的S盒运算中，通过提前造表的方法来提高运算的效率 3. 由于必须要能够解密，因此AES在加密中的每一个操作都在解密中有一个逆操作与其对应 4. AES在加密算法的首尾都使用了初始轮密钥加函数***AddRoundKey***，克服了DES中初始置换和逆置换都没有密钥参与的缺点 5. AES不存在DES中的若密钥和半若密钥 6. AES支持不同的密钥长度，提供了不同的安全性 | | | | | | |
| 六、教师评语  签名：  日期： | | | | | 成绩 | |