下面的类实现了文件的加密和解密操作，试验了几种文件类型均没有问题，现在和大家共享一下。

namespace mycryptohelp  
{  
 /// <summary>  
 /// 异常处理类  
 /// </summary>  
 public class cryptohelpexception : applicationexception  
 {  
  public cryptohelpexception(string msg):base(msg){}  
 }

 /// <summary>  
 /// crypthelp  
 /// </summary>  
 public class cryptohelp  
 {  
  private const ulong fc\_tag = 0xfc010203040506cf;

  private const int buffer\_size = 128\*1024;  
    
  /// <summary>  
  /// 检验两个byte数组是否相同  
  /// </summary>  
  /// <param name="b1">byte数组</param>  
  /// <param name="b2">byte数组</param>  
  /// <returns>true－相等</returns>  
  private static bool checkbytearrays(byte[] b1, byte[] b2)  
  {  
   if(b1.length == b2.length)  
   {  
    for(int i = 0; i < b1.length; ++i)  
    {  
     if(b1[i] != b2[i])  
      return false;  
    }  
    return true;  
   }  
   return false;  
  }

  /// <summary>  
  /// 创建rijndael symmetricalgorithm  
  /// </summary>  
  /// <param name="password">密码</param>  
  /// <param name="salt"></param>  
  /// <returns>加密对象</returns>  
  private static symmetricalgorithm createrijndael(string password, byte[] salt)  
  {  
   passwordderivebytes pdb = new passwordderivebytes(password,salt,"sha256",1000);  
     
   symmetricalgorithm sma = rijndael.create();  
   sma.keysize = 256;  
   sma.key = pdb.getbytes(32);  
   sma.padding = paddingmode.pkcs7;  
   return sma;  
  }

  /// <summary>  
  /// 加密文件随机数生成  
  /// </summary>  
  private static randomnumbergenerator rand = new rngcryptoserviceprovider();

  /// <summary>  
  /// 生成指定长度的随机byte数组  
  /// </summary>  
  /// <param name="count">byte数组长度</param>  
  /// <returns>随机byte数组</returns>  
  private static byte[] generaterandombytes(int count)  
  {  
   byte[] bytes = new byte[count];  
   rand.getbytes(bytes);  
   return bytes;  
  }

  /// <summary>  
  /// 加密文件  
  /// </summary>  
  /// <param name="infile">待加密文件</param>  
  /// <param name="outfile">加密后输入文件</param>  
  /// <param name="password">加密密码</param>  
  public static void encryptfile(string infile, string outfile, string password)  
  {  
   using(filestream fin = file.openread(infile),  
       fout = file.openwrite(outfile))  
   {  
    long lsize = fin.length; // 输入文件长度  
    int size = (int)lsize;  
    byte[] bytes = new byte[buffer\_size]; // 缓存  
    int read = -1; // 输入文件读取数量  
    int value = 0;  
      
    // 获取iv和salt  
    byte[] iv = generaterandombytes(16);  
    byte[] salt = generaterandombytes(16);  
      
    // 创建加密对象  
    symmetricalgorithm sma = cryptohelp.createrijndael(password, salt);  
    sma.iv = iv;     
      
    // 在输出文件开始部分写入iv和salt  
    fout.write(iv,0,iv.length);  
    fout.write(salt,0,salt.length);  
      
    // 创建散列加密  
    hashalgorithm hasher = sha256.create();  
    using(cryptostream cout = new cryptostream(fout,sma.createencryptor(),cryptostreammode.write),  
        chash = new cryptostream(stream.null,hasher,cryptostreammode.write))  
    {  
     binarywriter bw = new binarywriter(cout);  
     bw.write(lsize);  
       
     bw.write(fc\_tag);

     // 读写字节块到加密流缓冲区  
     while( (read = fin.read(bytes,0,bytes.length)) != 0 )  
     {  
      cout.write(bytes,0,read);  
      chash.write(bytes,0,read);   
      value += read;  
     }  
     // 关闭加密流  
     chash.flush();  
     chash.close();

     // 读取散列  
     byte[] hash = hasher.hash;  
       
     // 输入文件写入散列  
     cout.write(hash,0,hash.length);

     // 关闭文件流  
     cout.flush();  
     cout.close();  
    }  
   }  
  }