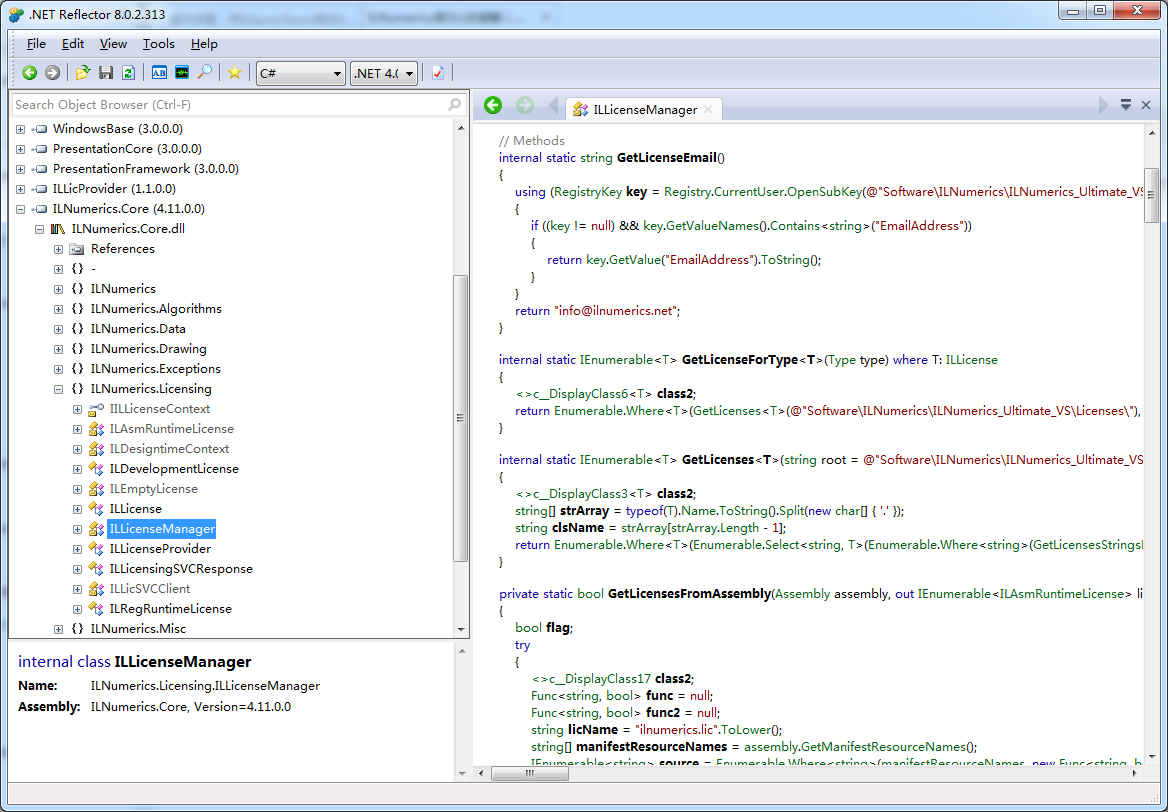
（注：这里的破解只以ILNumerics.Core.dll以及ILNumerics.Computing.dll为例进行，如果需要使用ILNumerics的其它DLL，也需要进行类似的操作破解）

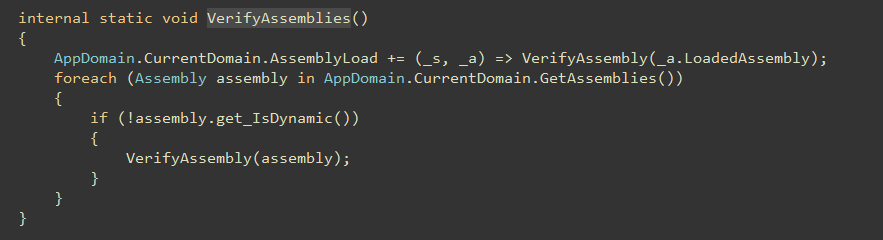
（注2：ILNumerics.Core.dll运行时，需要按照程序的平台（32/64位），将对应的DLL文件拷入程序目录，如果拷入后，报无法加载mkl\_custom.dll的错误时，可能是因为mkl\_custom.dll的某些依赖项缺失（正常的ILNumerics.Core.dll是需要安装的，安装时会装一些必须的库或DLL，但本文是直接引用DLL，所以没有），这时可以用Dependency Walker来查看mkl\_custom.dll的依赖项，看缺少了什么，补上就行，以本文的版本为例，缺失的是msvcr120.dll(Visual C++ Redistributable for Visual Studio 2013)，把相应的DLL拷到程序目录即可（见附件））

1.下载方式<http://ilnumerics.net/download.html> 本文下载版本为ILNumerics-Ultimate-VS-4.11.6011.1040

2. 尝试使用Reflector进行破解，发现里面混淆的程度不高，绝大多数代码都能看到，但是导出代码后无法编译



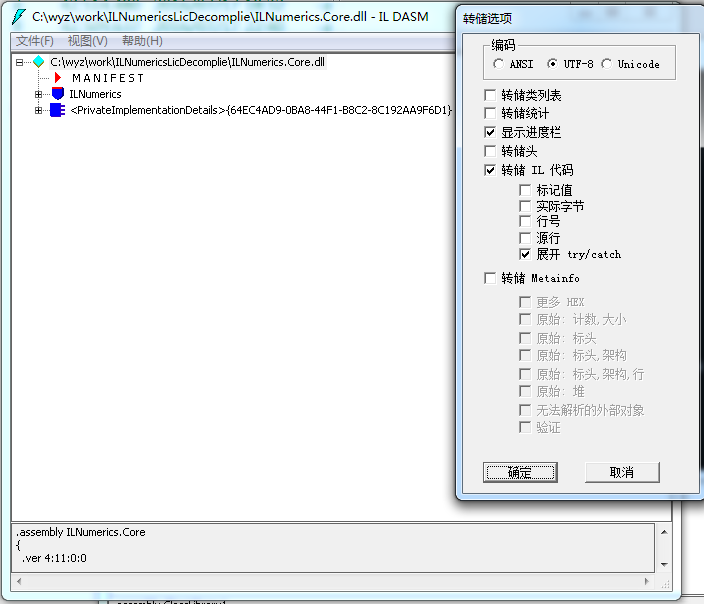
3.先分析一下代码，这个库的许可认证相关代码都在ILNumerics.Core.dll中，在ILicenseManager类里，使用Reflector查看，可以看到，最开始的入口VerifyAssemblies()



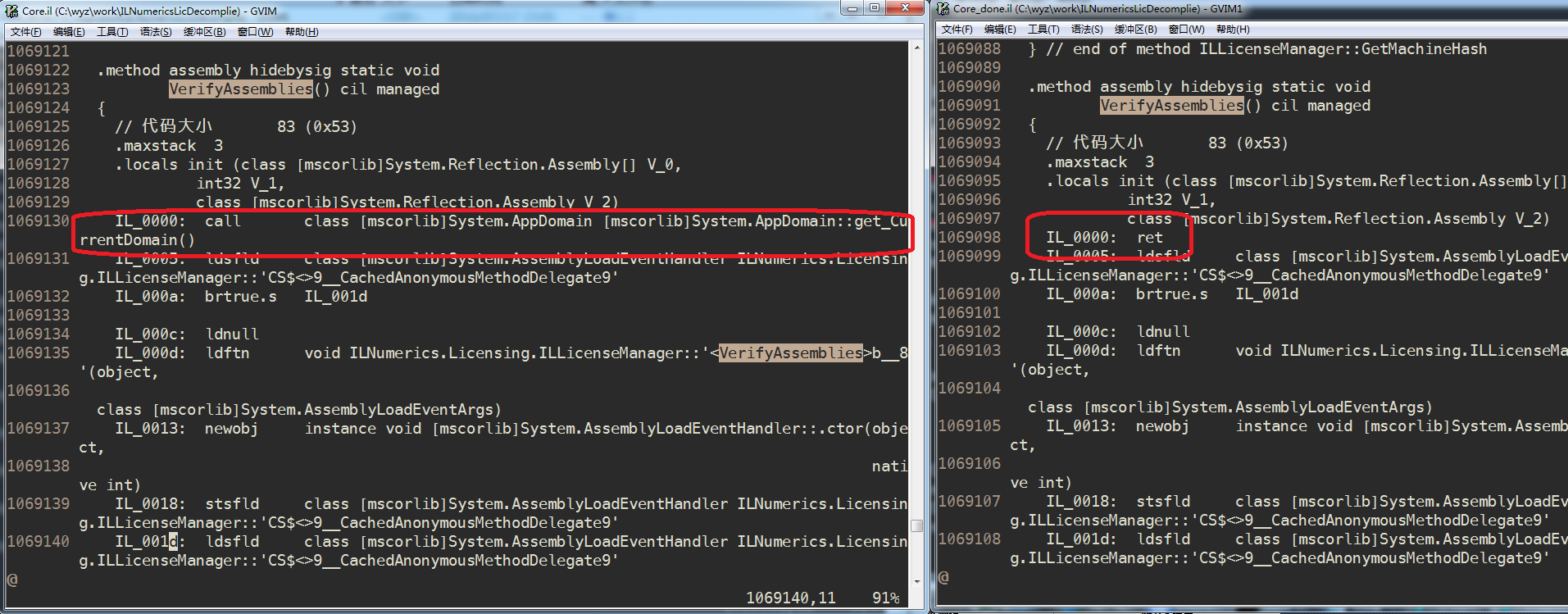
主要的机制猜测为，在VerifyAssemblies里检查许可，如果不符合，直接抛出异常，因此，猜测只需要将这几个检查许可的跳过就可以

4. 以ILNumerics.Core.dll以及ILNumerics.Computing.dll为例，进行破解，

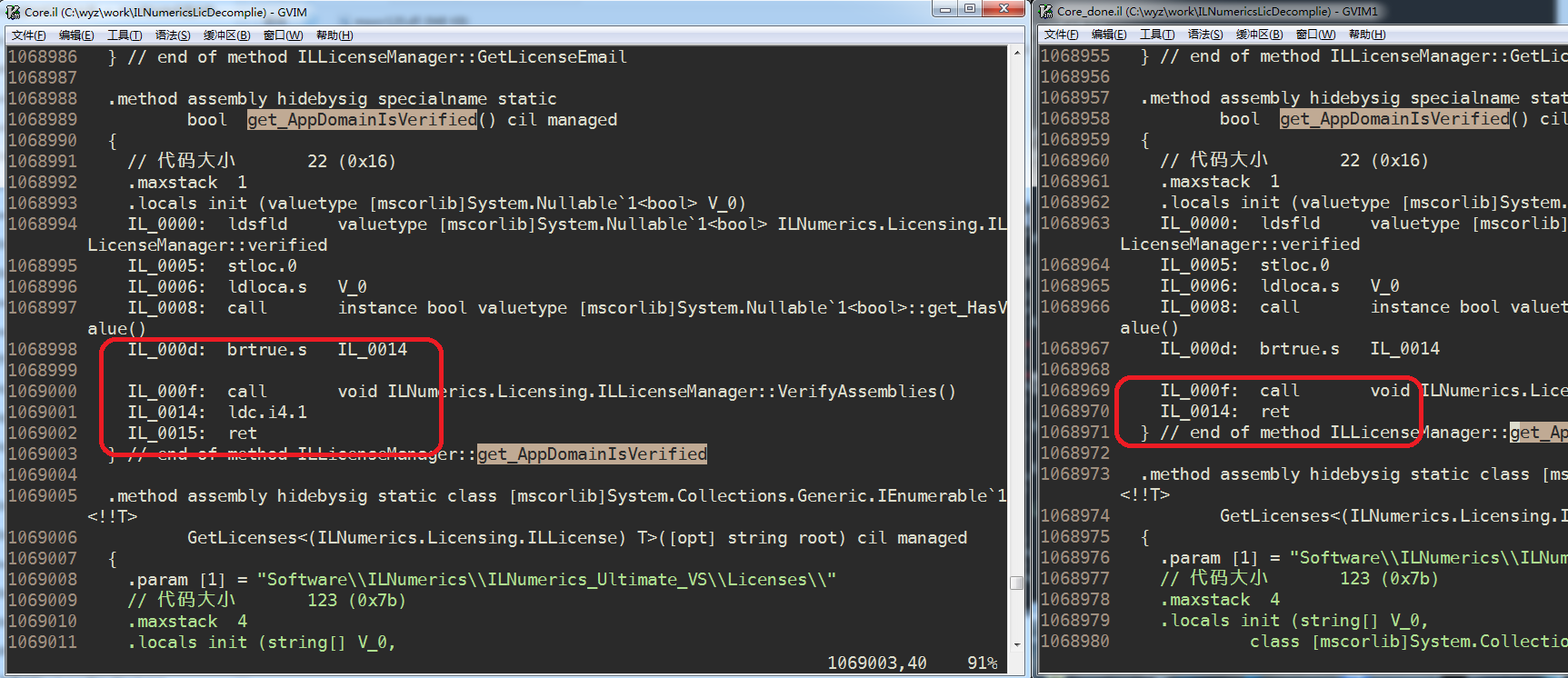
首先将这两个文件使用ildasm输出为il（分别输出为Core.il以及Computing.il，这里的名称可任意 ）



5.在Core.il中，查找刚才定位的两个关键函数，首先查找VerifyAssemblies，按下面进行修改



然后是AppDomainIsVerified，这里调用的具体函数是get\_AppDomainIsVerified，按下面进行修改（这个地方好像也可以不改）

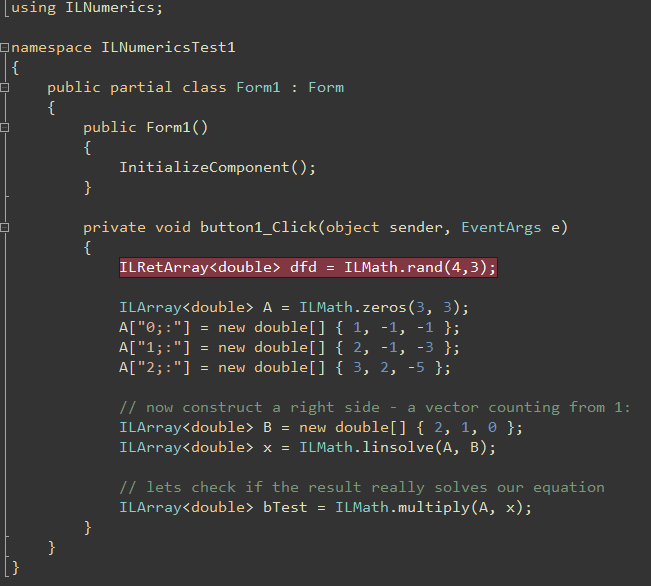


6. 使用ilasm重新编译成DLL（编译完成之后将Core.dll及Computin.dll改成原来的名称，即ILNumerics.Core.dll以及ILNumerics.Computing.dll）

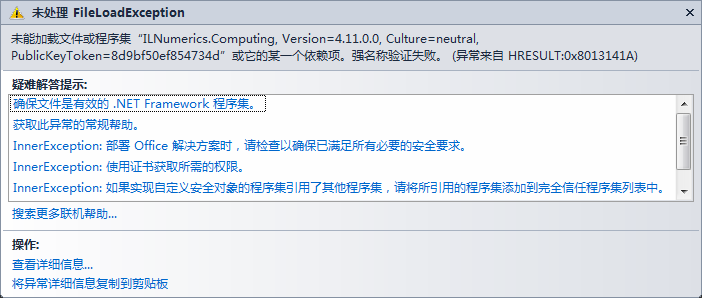
C:\Users\wyz\Documents\My Knowledge\temp\fb709785-a866-43df-9cda-b135a17c8650_4_files\b206b585-7d06-4825-a9b1-17d2c9bc0d76.png

C:\Users\wyz\Documents\My Knowledge\temp\fb709785-a866-43df-9cda-b135a17c8650_4_files\4a7042af-0eea-459a-b34f-6cada4250424.png

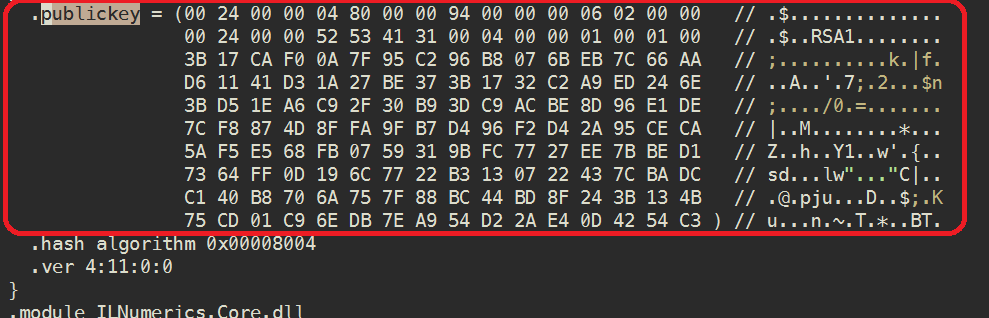
7. 建立一个代码测试用的工程，引用这两个DLL，测试DLL是否可用

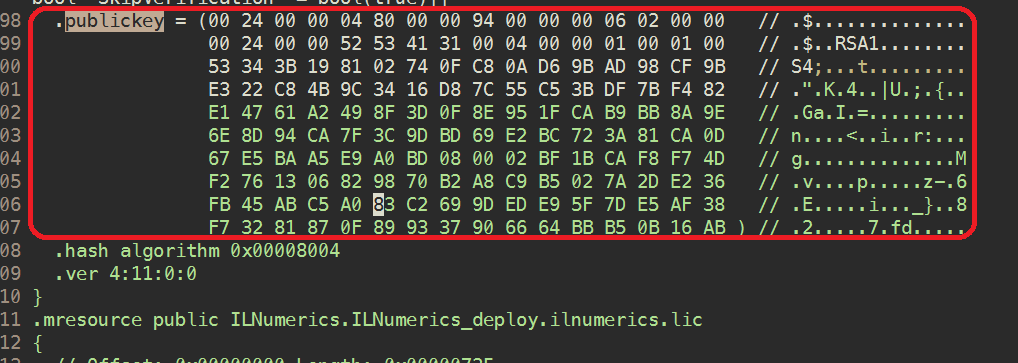


8.编译后，运行时出错，错误为“强名称验证失败”，这个错误可以参考另一篇文章“操作步骤：用ildasm/ilasm修改IL代码”中的相关内容



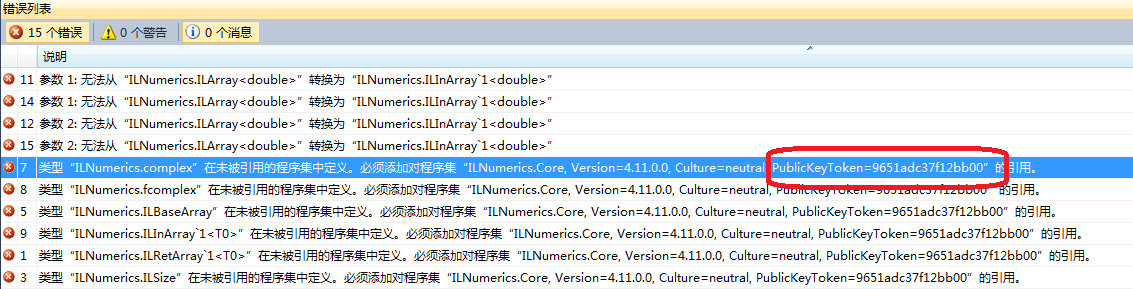
9.修改这个错误，打开Core.il及Computing.il，将里边的.publickey部分全部删除（注意不是.publickeytoken）





10.再使用ilasm重新编译成DLL，替换掉测试工程里的DLL，重新编译执行

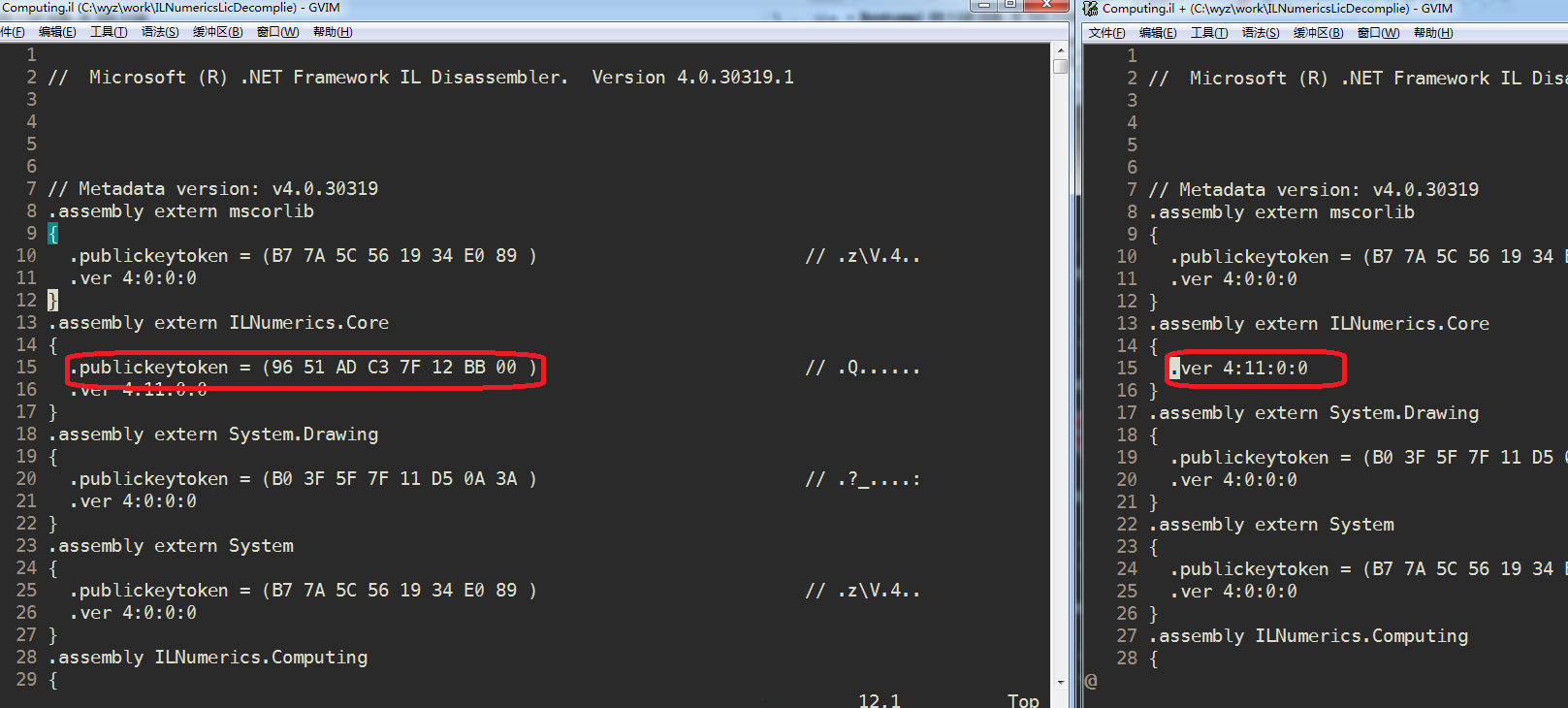
这时编译不通过，提示如下错误



这是因为ILNumerics.Computing.dll引用了ILNumerics.Core.dll，之前版本的dll中是有publickey的，引用时是注明了对应dll的publickey(上图红框)，刚才我们把强命名（publickey）去掉了，因此会导致在加载DLL产生错误，

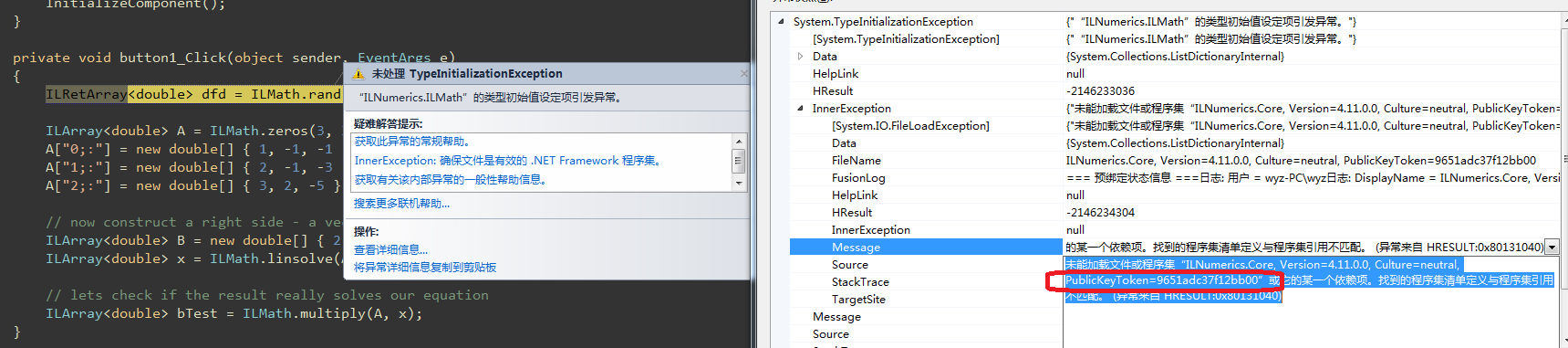
这里需要在引用时将注明的publickey去掉

打开Computing.il，修改如下代码：

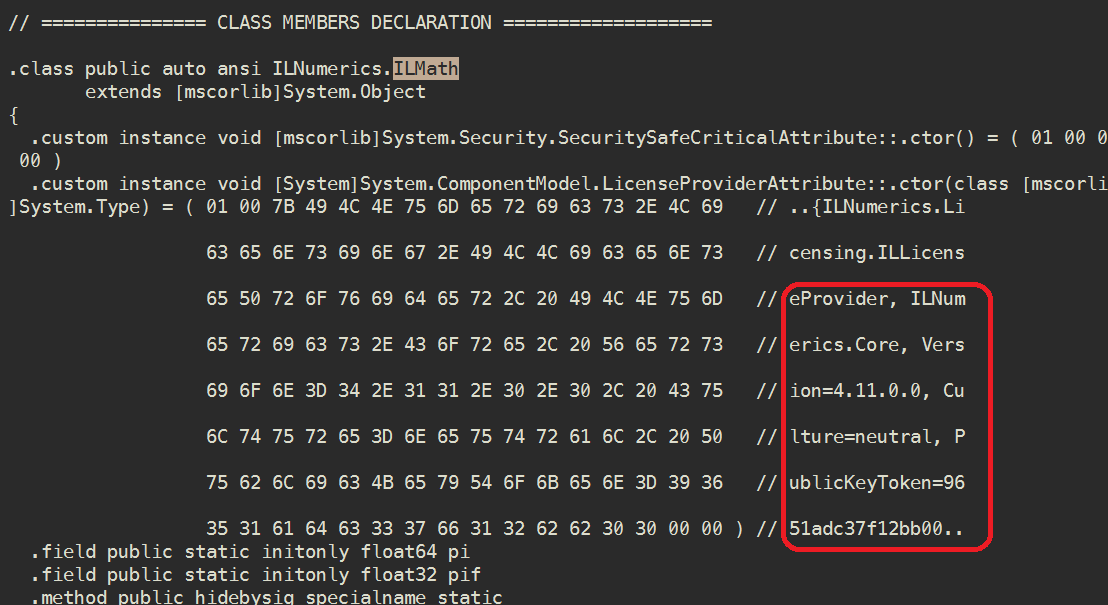


11.使用ilasm重新编译成DLL，替换掉测试工程里的DLL，重新编译执行

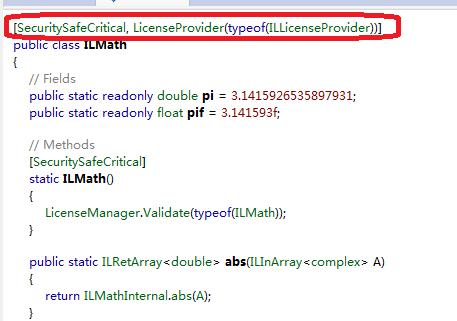
此时能够编译成功，运行时报错



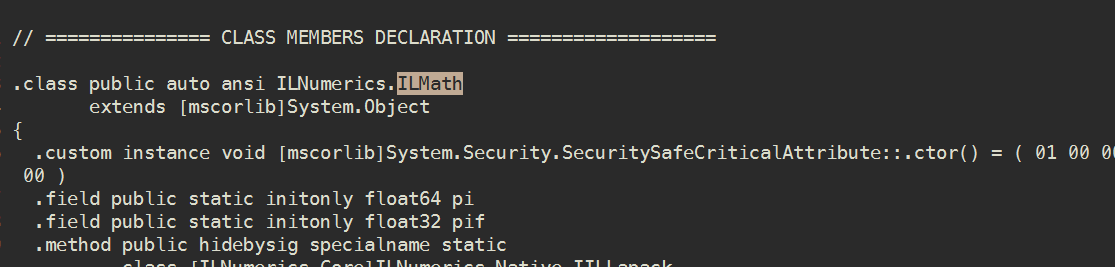
可以看到，很奇怪，刚才已经去掉了对publickey的指定，为什么这里还会有publickey呢？再在Compute.il中查找，



发现这里还引用了之前的publickey，这段代码不知道是干什么用的，使用Reflector查看这个类

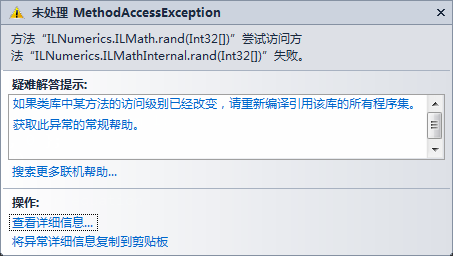


发现这是类的一个属性，试试在il里把这个属性直接干掉

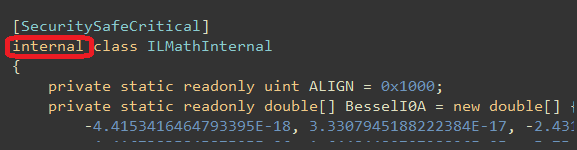


12.使用ilasm重新编译成DLL，替换掉测试工程里的DLL，重新编译执行

报下面的错误

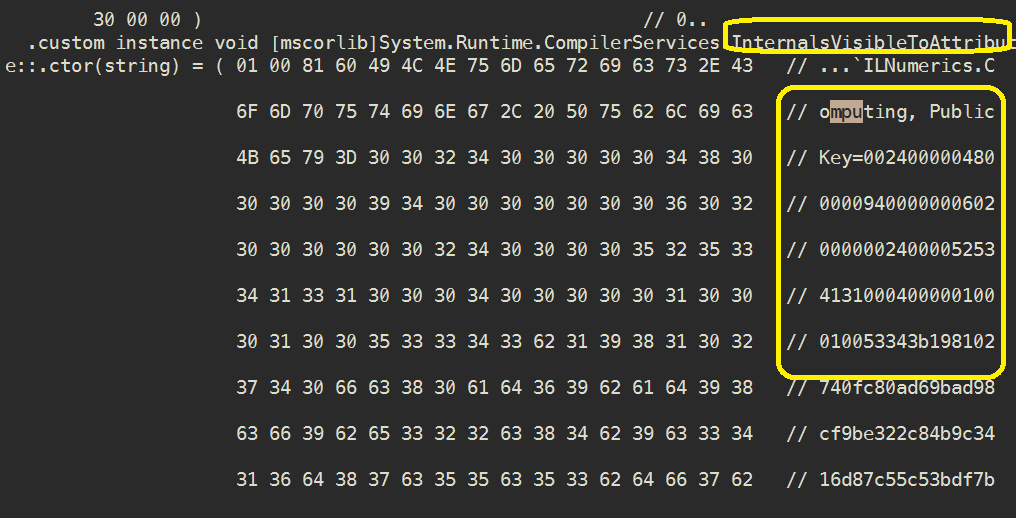


这是个访问权限的错误，ILMathInternal是ILNumerics.Core.dll中的一个类，声明为Internal



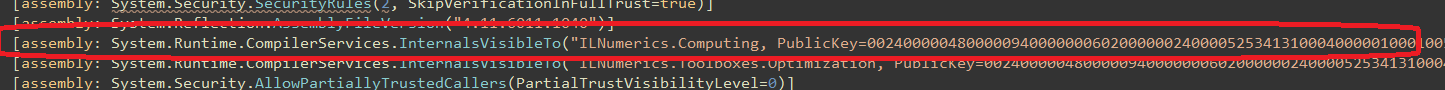
正常在外部的DLL（例如本文中的ILNumerics.Computing.dll）中是无法访问这个类的，但可以在DLL的AssemblyInfo.cs中添加assembly: InternalsVisibleTo属性来允许外部对象访问（具体方法和原理可以参考另外两篇文章《DevExpress源代码的编译》和《使用InternalsVisibleTo给assembly添加“友元assembly”》）

本来的DLL是加了的，相应的IL代码（IL代码与AssemblyInfo.cs中的格式不一样）如下：

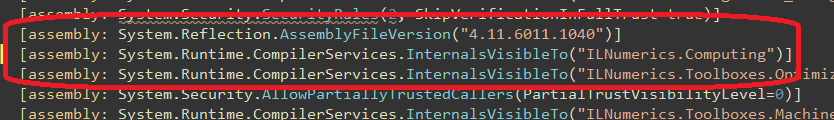


但是可以看到，这个InternalsVisibleTo的对象也是限定了PublicKey的，现在需要将这个PublicKey的限定去掉

本来在AssemblyInfo.cs里很好修改，只要将

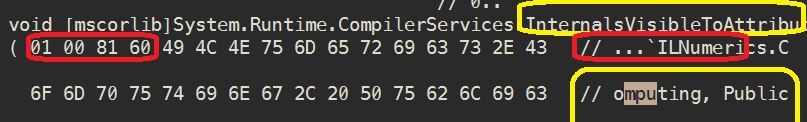


修改为



即可

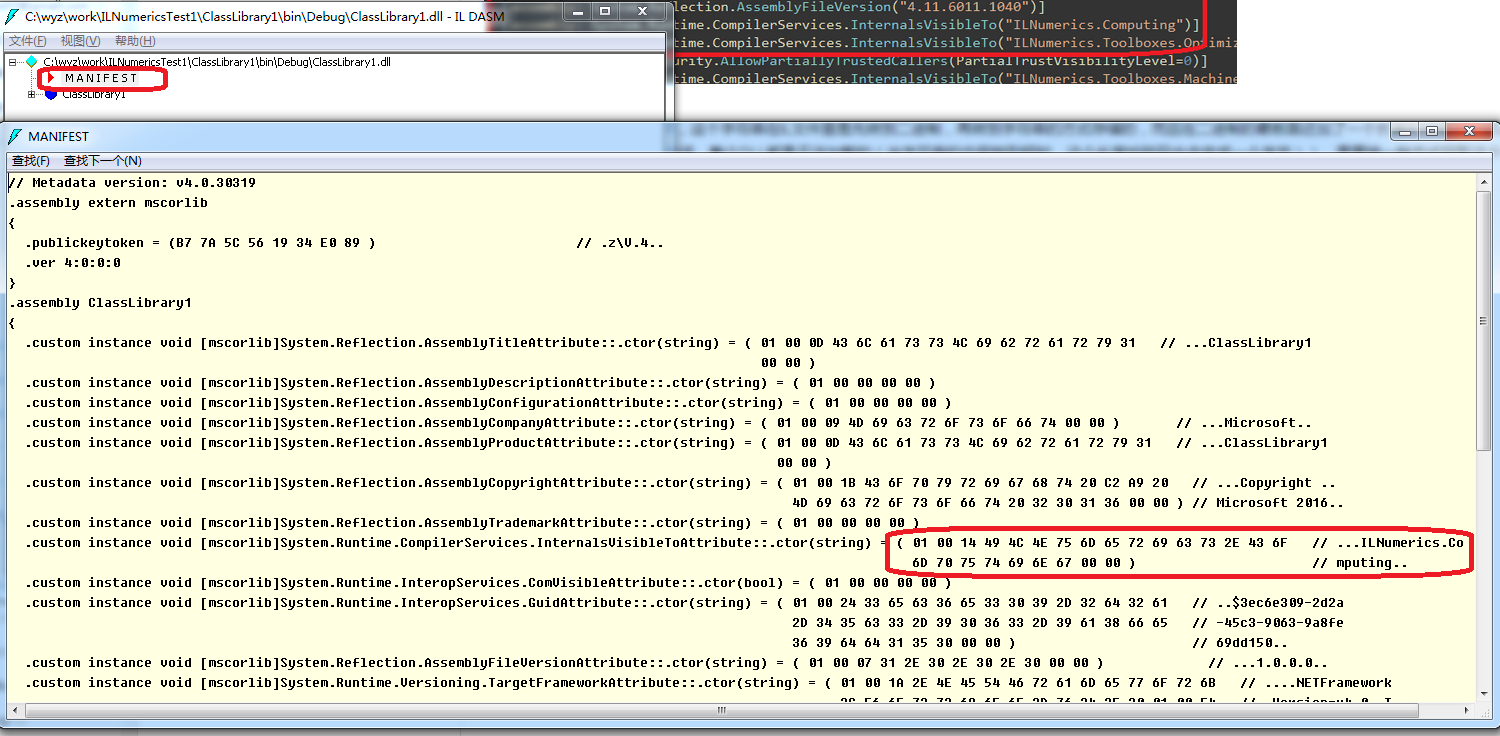
但是在IL里修改起来比较麻烦，这个字符串在IL文件里是先转到二进制，再转到字符串的方式存储的，而且在二进制的最前面还加了一个长度的校验码（可以看最前面有4个额外的字节（下图红框），后两个字节（81 60）就是长度检验码，如果不正确的话，整个DLL都是无法加载的（当字符串的内容特别短时，这个长度检验码也会变成一个字节）），需要换一种方式获取这个字符串



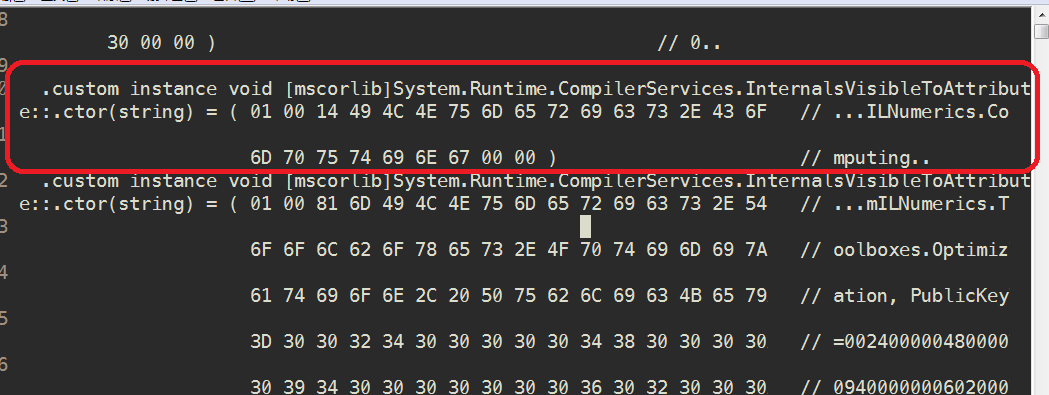
具体方法为：新建一个空的类库工程，然后在这个类库工程的AssemblyInfo.cs文件中，加入要添加内容，比如：



编译这个空类库，得到对应的DLL文件，然后用ildasm打开，查看其MANIFEST



这就是我们需要的值了，把这个值拷贝到Core.il中，替换掉原来的内容



13.使用ilasm重新编译成DLL，替换掉测试工程里的DLL，重新编译执行

顺利执行

