目前SimpleFramework NGUI/UGUI版本的框架都自带了自动编码加密功能，在安卓+pc下面使用的是luajit，在MAC+iOS下面用的是Luac。  
  
使用步骤：  
（1）打开AppConst.cs里面的编码开关：public const bool LuaEncode = true;  
（2）然后单击Game/Build xxx Resource菜单，打包出来的lua文件已经是加密过得二进制了。  
  
插件包不提供这个功能，如果需要可以到uLua\Editor\Packager.cs里面复制出来。  
  
  
上述属于框架已经提供的现成功能，如果选择dostring，可以选择讲lua代码进行字符串加密，自定义加密算法，为了安全可以考虑加密算法存在于c代码的ulua底层库中，可以过滤一些小白。  
  
luac自定义加密办法：  
（1）有的人可以自己修改lua虚拟机代码，看到国外一篇帖子：  
if you are concerned that users will simply run  
"luac -l" on the compiled code and see what the program does, you can  
just change LUA\_SIGNATURE to something else (but keep a non-printable  
char as its first byte) and possibly also reorder the fields in the  
header. This will make off-the-shelf luac choke. Most users will just  
give up right there. Those that don't won't be stopped by anything more  
complicated.  
大体意思是，直接修改Lua.h头文件里面的头部标识（第一个字节不要改）：  
/\* mark for precompiled code (`Lua') \*/  
#define LUA\_SIGNATURE   "\033Lua"  
这种最容易被破解了，但是可以防住一些只会工具解密的小白。  
  
  
（2）reorder or alter the items in luaU\_header 修改lundump.c里面的 luaU\_header函数，使其插入自己的头字段，  
/\*  
\* make header  
\*/  
void luaU\_header (char\* h)  
{  
int x=1;  
memcpy(h,LUA\_SIGNATURE,sizeof(LUA\_SIGNATURE)-1);  
h+=sizeof(LUA\_SIGNATURE)-1;  
\*h++=(char)LUAC\_VERSION;  
\*h++=(char)LUAC\_FORMAT;  
\*h++=(char)\*(char\*)&x;                /\* endianness \*/  
\*h++=(char)sizeof(int);  
\*h++=(char)sizeof(int);  
\*h++=(char)sizeof(Instruction);  
\*h++=(char)sizeof(lua\_Number);  
\*h++=(char)(((lua\_Number)0.5)==0);        /\* is lua\_Number integral? \*/  
}  
  
（3）刀塔传奇使用的方式：相对较安全的方案是直接修改lopcodes里面的字节位，使其重写排序，来欺骗大多数的反编译程序。不过刀塔传奇的代码还是被逆向出来了。  
  
（4）最安全目前不可破解的是luajit的预编译binary。但是在iOS的编译操作步骤比较麻烦，需要将lua代码copy到iOS设备里面才能dump，然后再下载下来。也就是iPhone或者iPad，不能在Mac、PC编译iOS64的字节码。  
  
（5）自己在ulua的dll编码进去一个加解密函数，然后在loader里面，调用这个函数返回解密后的字节码，传输给loadbuffer函数。这个自解密的算法，应该也是比较安全的，但是问题是可能效率会比luavm自带的字节码慢些，因为多了一步解密操作。