**1、cacert.pem 是https使用的证书吗？**

**答：是。如果服务器的证书本身就是权威CA颁发的，这个证书不配也没问题。一般在开发或测试环境调试https会有用。**

**2、客户端license密钥生成器通过什么算法利用cbframework\_license文件生成的3DES密钥的？**

**答：主要用的是cbframework\_license文件的Hash值，再根据一些简单的截取得到对称秘钥，具体计算看看代码实现就可以。没有什么原理，就是一些简单的变换算法，其实这里随便怎么生成秘钥都可以，仅仅是增加一些破解难度。**

**3、RSA 加密的过程:公钥加密\私钥解密**

**RSA 签名的过程:私钥签名\公钥验签.**

**疑问:目前我们的程序里只有一个公钥, 客户端对发送给服务器的网络请求的报文进行签名，是否应该使用私钥签名?如果使用私钥签名,所有用户的私钥如何下发\维护?**

**答：客户的私钥理论上应该是由客户自己生成的，服务器不应该有客户的私钥。服务器应该维护的是客户的公钥。**

**签名的作用是防抵赖，RSA算法效率较低，一般涉及关键交易才需要签名。**

**问题的关键在于客户端的秘钥保存困难，防抵赖这一功能的基本假设就是私钥是安全的，如果假设不成立，防抵赖也就不成立了。这就失去了签名的法律意义，只能作为一种增加破解难度的手段，相比协商对称秘钥并无明显优势，对服务器的压力也会增大，RSA比3DES慢的多，十几倍到几十倍的性能差距。**

**4、对比**

****

**查看实际代码的功能：**

**1）png：在代码中没有看到对图片进行公钥验签，只有跟清单列表的Hash对比的代码。**

**答：确实有这样的问题，其实Hash对比就够了，对清单列表做验签就可以了，还可以提升效率。最终的效果相当于也验证了每个文件的签名，因为清单列表做验签了。**

**2）html：没看到代码有压缩处理**

**答：确实html没有压缩，可以增加压缩的操作。**

**3）resource.json/allmenu.xml：**

**1）没看到代码有加密处理**

**这个设计是后来加的，但是可能没有实现，相关的API应该是有的，可以加上。**

**2）对resource.json 进行了 hash 对比，说明文件是完整的，**

**但是没有公钥验签处理？尤其是在无网情况下，resource.json不和服务器对比，**

**疑问:没有公钥验签处理，这个文件是否存在篡改的可能性，无法验证文件的来源是否合法。**

**答：应该是要验签的，可能是实现和设计不符。**

**4）menu.xml：**

**1）没看到代码有压缩处理**

**2）没看到代码有加密处理**

**答：这两点和resource.json一样也是后加的，可能没来及加上。**

**5) 目前针对JS、CSS、图片只是程序启动时进行验签的检查，**

**等检查结束后，这时如果篡改JS、css等文件，这个操作我们是无法确保安全的。**

**由于resource.json\menu.xml 是验证完毕后，直接加载到内存，所以不会存在篡改的时间段问题。由于JS、CSS、图片等是在使用的时候，才会加载，那么就为篡改提供了可能性。**

**答：要想更安全 就要自己计算文件依赖，在文件加载前检验，咱们的文件又是按需加载的，精确计算依赖也实现也比较复杂**