随着电子商务的迅速发展，信息安全已成为焦点问题之一，尤其是网上支付和网络银行对信息安全的要求显得更为突出。为了能在因特网上开展安全的电子商务活动，公开密钥基础设施（ PKI, Public Key Infrastructure ）逐步在国内外得到广泛应用。我们是否真的需要 PKI ， PKI 究竟有什么用？下面通过一个案例一步步地来剖析这个问题 : 甲想将一份合同文件通过 Internet 发给远在国外的乙，此合同文件对双方非常重要，不能有丝毫差错，而且此文件绝对不能被其他人得知其内容。如何才能实现这个合同的安全发送？   
问题 1: 最自然的想法是，甲必须对文件加密才能保证不被其他人查看其内容，那么 , 到底应该用什么加密技术，才能使合同传送既安全又快速呢 ?   
    可以采用一些成熟的对称加密算法 , 如 DES 、 3DES 、 RC5 等对文件加密。对称加密采用了对称密码编码技术，它的特点是文件加密和解密使用相同的密钥，即加密密钥也可以用做解密密钥，这种方法在密码学中叫做对称加密算法，   
问题 2: 如果黑客截获此文件，是否用同一算法就可以解密此文件呢 ?   
    不可以 , 因为加密和解密均需要两个组件 : 加密算法和对称密钥 , 加密算法需要用一个对称密钥来解密 , 黑客并不知道此密钥。   
问题 3: 既然黑客不知密钥，那么乙怎样才能安全地得到其密钥呢？用电话通知，若电话被窃听，通过 Internet 发此密钥给乙，可能被黑客截获，怎么办 ?   
    方法是用非对称密钥算法加密对称密钥后进行传送。与对称加密算法不同，非对称加密算法需要两个密钥：公开密钥（ Public Key ）和私有密钥（ Private Key ）。公开密钥与私有密钥是一对，如果用公开密钥对数据进行加密，只有用对应的私有密钥才能解密；如果用私有密钥对数据进行加密，只有用对应的公开密钥才能解密。因为加密和解密使用的是两个不同的密钥，所以这种算法叫做非对称加密算法 ( 公 / 私钥可由专门软件生成 ) 。甲乙双方各有一对公 / 私钥，公钥可在 Internet 上传送，私钥自己保存。这样甲就可以用乙的公钥加密问题 1 中提到的对称加密算法中的对称密钥。即使黑客截获到此密钥，也会因为黑客不知乙的私钥，而解不开对称密钥，因此也解不开密文，只有乙才能解开密文。   
问题 4 ：既然甲可以用乙的公钥加密其对称密钥，为什么不直接用乙的公钥加密其文件呢？这样不仅简单，而且省去了用对称加密算法加密文件的步骤？   
    不可以这么做。因为非对称密码算法有两个缺点 : 加密速度慢 , 比对称加密算法慢 10 ～ 100 倍 , 因此只可用其加密小数据 ( 如对称密钥 ) ，另外加密后会导致得到的密文变长。因此一般采用对称加密算法加密其文件 , 然后用非对称算法加密对称算法所用到的对称密钥。   
问题 5 ： 如果黑客截获到密文，同样也截获到用公钥加密的对称密钥，由于黑客无乙的私钥，因此他解不开对称密钥，但如果他用对称加密算法加密一份假文件 , 并用乙的公钥加密一份假文件的对称密钥，并发给乙，乙会以为收到的是甲发送的文件，会用其私钥解密假文件 , 并很高兴地阅读其内容，但却不知已经被替换。换句话说，乙并不知道这不是甲发给他的，怎么办 ?   
    答案是用数字签名证明其身份。数字签名是通过散列算法 , 如 MD5 、 SHA-1 等算法从大块的数据中提取一个摘要。而从这个摘要中不能通过散列算法恢复出任何一点原文，即得到的摘要不会透露出任何最初明文的信息，但如果原信息受到任何改动，得到的摘要却肯定会有所不同。因此甲可以对文件进行散列算法得到摘要，并用自己的私钥加密 ( 因为非对称算法可逆，即用私钥可解开公钥加密的文件，反之亦然 ) ，这样即使黑客截获也无用。因为黑客不会从摘要内获得任何信息，但乙却不一样，他可用甲的公钥解密，得到其摘要 ( 如果用甲的公钥能够解开此摘要，说明此摘要肯定是甲发的，因为只有甲的公钥才能解开用甲的私钥加密的信息 , 而甲的私钥只有甲自己知道 ) ，并对收到的文件 ( 解密后的合同文件 ) 也进行同样的散列算法，通过比较其摘要是否一样 , 就可得知此文件是否被篡改过  ( 因为若摘要相同，则肯定信息未被改动，这是散列算法的特点 ) 。这样不仅解决了证明发送人身份的问题，同时还解决了文件是否被篡改问题。   
问题 6 ： 通过对称加密算法加密其文件，再通过非对称算法加密其对称密钥 , 又通过散列算法证明其发送者身份和其信息的正确性，这样是否就万无一失了 ?   
    回答是否定的。问题在于乙并不能肯定他所用的所谓甲的公钥一定是甲的 , 解决办法是用数字证书来绑定公钥和公钥所属人。   
    数字证书是一个经证书授权中心数字签名的包含公开密钥拥有者信息以及公开密钥的文件 , 是网络通信中标识通信各方身份信息的一系列数据，它提供了一种在 Internet 上验证身份的方式，其作用类似于司机的驾驶执照或日常生活中的身份证，人们可以在交往中用它来识别对方的身份。   
    最简单的证书包含一个公开密钥、名称以及证书授权中心的数字签名。一般情况下证书中还包括密钥的有效时间、发证机关 ( 证书授权中心 ) 名称、该证书的序列号等信息。它是由一个权威机构—— CA 机构，又称为证书授权 (Certificate Authority) 中心发放的。 CA 机构作为电子商务交易中受信任的第三方，承担公钥体系中公钥的合法性检验的责任。 CA 中心为每个使用公开密钥的用户发放一个数字证书，数字证书的作用是证明证书中列出的用户合法拥有证书中列出的公开密钥。 CA 机构的数字签名使得攻击者不能伪造和篡改证书， CA 是 PKI 的核心，负责管理 PKI 结构下的所有用户（包括各种应用程序）的证书，把用户的公钥和用户的其他信息捆绑在一起，在网上验证用户的身份。   
    因为数字证书是公开的，就像公开的电话簿一样，在实践中，发送者（即甲）会将一份自己的数字证书的拷贝连同密文、摘要等放在一起发送给接收者（即乙），而乙则通过验证证书上权威机构的签名来检查此证书的有效性（只需用那个可信的权威机构的公钥来验证该证书上的签名就可以了），如果证书检查一切正常，那么就可以相信包含在该证书中的公钥的确属于列在证书中的那个人（即甲）。   
问题 7 ： 至此似乎很安全了。但仍存在安全漏洞，例如：甲虽将合同文件发给乙 , 但甲拒不承认在签名所显示的那一时刻签署过此文件 ( 数字签名就相当于书面合同的文字签名 ) ，并将此过错归咎于电脑，进而不履行合同，怎么办 ?   
    解决办法是采用可信的时钟服务 ( 由权威机构提供 ) ，即由可信的时间源和文件的签名者对文件进行联合签名。在书面合同中，文件签署的日期和签名一样均是十分重要的防止文件被伪造和篡改的关键性内容 ( 例如合同中一般规定在文件签署之日起生效 ) 。在电子文件中，由于用户桌面时间很容易改变 ( 不准确或可人为改变 ) ，由该时间产生的时间戳不可信赖，因此需要一个第三方来提供时间戳服务（数字时间戳服务（ DTS ）是网上安全服务项目，由专门的机构提供）。此服务能提供电子文件发表时间的安全保护。   
    时间戳产生的过程为 : 用户首先将需要加时间戳的文件用哈希编码加密形成摘要，然后将该摘要发送到 DTS ， DTS 在加入了收到文件摘要的日期和时间信息后再对该文件加密（数字签名），然后送回用户。因此时间戳 (time-stamp) 是一个经加密后形成的凭证文档，它包括三个部分：需加时间戳的文件的摘要， DTS 收到文件的日期和时间， DTS 的数字签名。由于可信的时间源和文件的签名者对文件进行了联合签名 , 进而阻止了文档签名的那一方 ( 即甲方 ) 在时间上欺诈的可能性 , 因此具有不可否认性。   
问题 8: 有了数字证书将公 / 私钥和身份绑定 , 又有权威机构提供时钟服务使其具有不可否认性 , 是不是就万无一失了 ? 不 , 仍然有问题。乙还是不能证明对方就是甲，因为完全有可能是别人盗用了甲的私钥 ( 如别人趁甲不在使用甲的电脑 ), 然后以甲的身份来和乙传送信息 , 这怎么解决呢 ?   
    解决办法是使用强口令、认证令牌、智能卡和生物特征等技术对使用私钥的用户进行认证，以确定其是私钥的合法使用者。   
    解决这个问题之前我们先来看看目前实现的基于 PKI 的认证通常是如何工作的。以浏览器或者其他登记申请证书的应用程序为例说明，在第一次生成密钥的时候会创建一个密钥存储，浏览器用户会被提示输入一个口令，该口令将被用于构造保护该密钥存储所需的加密密钥。如果密钥存储只有脆弱的口令保护或根本没有口令保护，那么任何一个能够访问该电脑浏览器的用户都可以访问那些私钥和证书。在这种场景下 , 又怎么可能信任用 PKI 创建的身份呢 ? 正因为如此，一个强有力的 PKI 系统必须建立在对私钥拥有者进行强认证的基础之上，现在主要的认证技术有：强口令、认证令牌、智能卡和生物特征（如指纹，眼膜等认证）。   
    以认证令牌举例 : 假设用户的私钥被保存在后台服务器的加密容器里，要访问私钥，用户必须先使用认证令牌认证（如用户输入账户名、令牌上显示的通行码和 PIN 等），如果认证成功，该用户的加密容器就下载到用户系统并解密。   
通过以上问题的解决，就基本满足了安全发送文件的需求。下面总结一下这个过程 , 对甲而言整个发送过程如下 :   
1. 创建对称密钥 ( 相应软件生成，并且是一次性的 ) ，用其加密合同，并用乙的公钥打包对称密钥。   
2. 创建数字签名，对合同进行散列算法 ( 如 MD5 算法 ) 并产生原始摘要，甲用自己的私钥加密该摘要 ( 公 / 私钥既可自己创建也可由 CA 提供 ) 。   
3. 最后 , 甲将加密后的合同、打包后的密钥、加密后的摘要 , 以及甲的数字证书 ( 由权威机构 CA 签发 ) 一起发给乙。   
而乙接收加密文件后，需要完成以下动作 :   
1. 接收后，用乙的私钥解密得到对称密钥 , 并用对称密钥解开加密的合同 , 得到合同明文。   
2. 通过甲的数字证书获得属于甲的公钥 , 并用其解开摘要 ( 称做摘要 1) 。   
3. 对解密后的合同使用和发送者同样的散列算法来创建摘要 ( 称做摘要 2) 。   
4. 比较摘要 1 和摘要 2, 若相同 , 则表示信息未被篡改 , 且来自于甲。   
    甲乙传送信息过程看似并不复杂 , 但实际上它由许多基本成分组成 , 如 : 对称 / 非对称密钥密码技术、数字证书、数字签名、证书发放机构（ CA ）、公开密钥的安全策略等 , 这其中最重要、最复杂的是证书发放机构（ CA ）的构建。

在此强调下证书和签名：

    证书实际对于非对称加密算法（公钥加密）来说的，一般证书包括公钥、姓名、数字签名三个部分。证书好比身份证，证书机构（ca）就好比是公安局，职责就是负责管理用户的证书也就是身份证。比如我的公钥是FrankKey,姓名是Frank Xu Lei。公安局可以给我登记，但是怎么保证我和别的Frank XuLei区别开呢，于是公安局（证书机构）就使用我的名字和密钥做了个组合，再使用一种哈希算法，得出一串值，来标识我的唯一性，这个值就是我的身份证号码，也就是证书里的数字签名（消息摘要），同时为了不能让黑客仿造数字证书，数字证书的发行者用自己的私钥对数字签名进行加密，这样，使用该数字证书的网络交易实体就可以用证书发行者的公钥进行解密验证。（即：数字签名的目的是证明自己的身份的确是真实的自己而非其他人，所以需要用自己的私钥进行加密，让使用者用公钥进行解密验证，从而从技术上杜绝了伪造自己的黑客。）  
    假设一个朋友给我写信，他就可以到公安局（证书机构）来查找我的身份证（证书）。上面包括我的个人信息，可以保证这个公钥就是我的。然后他把新建进行加密，邮寄给我。别人即使拆开我的信件，因为没有密钥进行解密，所以无法阅读我的信件内容。这样就保证了信息安全。  
    所以说加密不一定要证书，取决于你数据安全具体的需求。一般大型的电子商务网站都有自己特定的证书。证书管理的机构比较有名的就是VeriSign（可以 说是互联网上的身份证管理局）。企业可以申请注册，它会给申请者生成特定的签名。我们自己的企业内部应用如果需要的话，可以在企业局域网内部建立企业私有的证书服务器，来产生和管理证书。其实X.509是由国际电信联盟（ITU-T）制定的一种定义证书格式和分布的国际标准(相当于制作身份证的规范)。为了提供公用网络用户目录信息服务，并规定了实体鉴别过程中广泛适用的证书语法和数据接口， X.509 称之为证书，或者说是身份证的一种形式，类似与我们现在的二代身份证，也是身份证的一种，根据特定的标准制作出来的。另外证书使用的时候还有有效期的限制，和我们的身份证的10年有效期一样。证书也可以设置有效期。

附：U盾的工作原理介绍

一、什么是U盾

     U盾，即工行2003年推出并获得国家专利的客户证书USBkey，是工行为您提供的办理网上银行业务的高级别安全工具。它外形酷似U盘，像一面盾牌，时刻保护着您的网上银行资金安全。

从技术角度看，U盾是用于网上银行电子签名和数字认证的工具，它内置微型智能卡处理器，采用1024位非对称密钥算法对网上数据进行加密、解密和数字签名，确保网上交易的保密性、真实性、完整性和不可否认性

二、工作原理

    U盾又作移动数字证书，它存放着你个人的数字证书，并不可读取。同样，银行也记录着你的数字证书。

当你尝试进行网上交易时，银行会向你发送由时间字串，地址字串，交易信息字串，防重放攻击字串组合在一起进行加密后得到的字串A，你的U盾将跟据你的个人证书对字串A进行不可逆运算得到字串B，并将字串B发送给银行，银行端也同时进行该不可逆运算，如果银行运算结果和你的运算结果一致便认为你合法，交易便可以完成，如果不一致便认为你不合法，交易便会失败。(理论上，不同的字串A不会得出相同的字串B，即一个字串A对应一个唯一的字串B;但是字串B和字串A无法得出你的数字证书，而且U盾具有不可读取性，所以任何人都无法获行你的数字证书。并且银行每次都会发不同的防重放字串(随机字串)和时间字串，所以当一次交易完成后，刚发出的B字串便不再有效。综上所述，理论上U盾是绝对安全的\*\*\*\*注意是理论上发生伪造概率大约为2的80次方分之一，但是如果有像变形金刚中的那种DNAbasecomputer的话。

参考文献：<http://blog.chinaunix.net/uid-23637692-id-3057988.html>（介绍了PKI体系和常见证书）