**第10章**

**身份认证和PKI理论基础**

**10.1 身份认证概述**

随着计算机网络技术的发展以及网络应用在各行各业的迅速普及，网络安全越来越受到人 们的重视，作为网络安全的第一道门槛，网络身份认证技术已成为网络安全的一个重要课题， 它对网络应用的安全性起着至关重要的作用。世界各国经过多年的努力，已初步形成了一套比 较完整的身份认证系统解决方案，即公钥基础设施(PKI), 该方案为网络应用透明地提供了 通用的信息安全服务。当前，PKI 技术还处在不断发展与完善的阶段，通过对PKI的分析、设 计与系统应用等方面的研究，可以解决现有PKI 系统中存在的诸多问题与难题，增强PKI 系 统的安全性、可用性及易管理性，提高系统的效率，为网络安全系统应用提供更高效、更实用 的解决方案。

**10.1.1** **网络安全与身份认证**

21 世纪是网络信息的时代，随着微电子、光电子、计算机、通信和信息服务业的发展， Internet 已得到了广泛的应用。互联网络正以惊人的速度改变着人们的工作和生活方式，从机 构到个人都在越来越多地通过互联网或其他电子媒介发送电子邮件、互换资料及网上交易，这 无疑给社会、企业乃至个人带来了前所未有的便利。所有这一切正是得益于互联网的开放性和 匿名性的特征，然而开放性和匿名性也决定了单纯的互联网不可避免地存在信息安全隐患，表 现在互联网经常会受到各种各样的非法入侵和攻击，因而对互联网信息安全性的要求也愈来愈 高，特别是以 Internet为支撑平台的电子商务的出现和蓬勃发展，使人们对信息安全提出了更 高的要求，使得网络安全的重要性日益凸显。

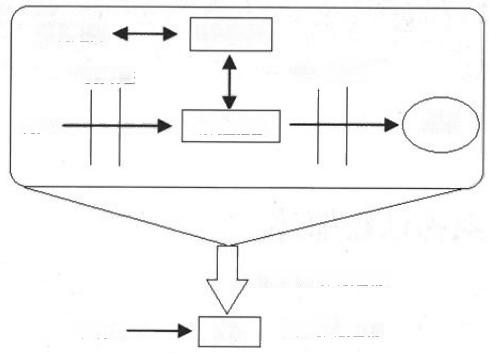
网络安全是一个很广泛的概念，广义的网络安全指网络中涉及的所有安全问题，范围涵盖 系统安全、信息安全和通信安全等内容。网络安全技术一般包括数据机密性(Data Confidentialily) 、 数据完整性 (Data Integrity)、身份认证 (Authentication) 、授权控制 (Authorization) 、 审计(Audit) 等多个方面，这些技术都是以密码学技术为基础的。其中， 身份认证技术用于实现网络通信双方身份可靠的验证，身份认证技术为其他安全技术提供基础， 例如基于身份的访问控制、计费等。对于大多数网络应用，尤其是电子商务这样的商业应用来



**Windows C/C++加密解密实战**

说，身份认证是其服务过程中的关键环节。

身份认证在网络安全中占据着十分重要的位置。身份认证是网络安全系统中的第一道关卡， 如图10-1所示。



**授权数据库**

**访问控制器**

**访问监控器**

**记录(实时入侵检测)**

**审计** **非实时入侵检测**

**安全管理员**

**审计员** —

**身份认证**

**用户**

**资源**

图10- 1

用户在访问网络系统之前，首先经过身份认证系统识别身份，然后访问监控器根据用户的 身份和授权数据库决定用户是否能够访问某个资源。授权数据库由安全管理员按照需要进行配 置。审计系统根据审计设置记录用户的请求和行为，同时入侵检测系统实时或非实时地判定是 否有入侵行为。访问控制和审计系统都要依赖于身份认证系统提供的“信息” 用户的身份。 可见身份认证是基本的安全服务，其他的安全服务都要依赖于它。一旦身价认证系统被攻破， 那么系统的所有安全措施将形同虚设。网络黑客攻击的目标往往就是身份认证系统。因此要加 快网络信息安全的建设，身份认证技术理论及其应用的研究是一个非常重要的课题。

**10.1.2** **网络环境下身份认证所面临的威胁**

网络中非法攻击者采用的攻击手段主要有：非法窃取合法用户的口令，因而可以访问对其 而言并未获得授权的系统资源；对合法用户的通信信息进行窃取分析并进行破译；截获合法的 用户信息，然后传送给接收者；阻止系统资源的合法管理和使用。目前身份认证所面临的威胁 主要有：

(1)中间人攻击。非法用户截获信息，替换或修改信息后再传送给接收者，或者非法用 户冒充合法用户发送信息，其目的在于盗取系统的可用性，阻止系统资源的合法管理和使用， 其原因主要是认证系统设计结构上的问题，比如一个典型的问题是很多身份认证协议只实现了 单向身份认证，其身份信息与认证信息可以相互分离。

(2)重放攻击。网络认证还需防止认证信息在网络传输过程中被第三方获取，并记载下 来，然后传送给接收者，这就是重放攻击。攻击的主要目的在于实现身份伪造，或者破坏合法 用户身份认证的同步性。

**第10章身份认证和PKI理论基础**

(3)密码分析攻击。攻击者通过密码分析破译用户口令/身份信息或猜测下次用户身份的 认证信息。系统实现上的简化可能为密码分析提供条件，系统设计原理上的缺陷可能为密码分 析创造条件。

(4)口令猜测攻击。侦听者在知道了认证算法后，可以对用户的口令字进行猜测，使用 计算机猜测口令字，利用得到的报文进行验证。这种攻击办法直接有效，特别是当用户的口令 有缺陷时，比如口令字短、使用名字做口令字、使用一个字(Word) 做口令字(可以使用字 典攻击)等。非法用户获得合法用户身份的口令，这样就可以访问对其而言并未获得授权的系 统资源。

(5)身份信息的暴露。认证时暴露身份信息是不可取的。某些信息尽管算不上秘密，但 大多数用户仍然不希望隐私资料任意扩散。例如在网上报案系统中，需要身份认证以确认信息 的来源是真实的。但如果认证过程中暴露了参与者的身份，则报案者完全可能受到打击报复， 从而影响公民举报犯罪的积极性。

(6)对认证服务器的攻击。认证服务器是身份认证系统的安全关键所在。在服务器中存 放了大量的用户认证信息和配套数据。如果攻破了身份认证服务器，那么后果将是灾难性的。

为了抵御网络环境下的身份认证面临的上述威胁，我们在进行网络身份认证技术研究、设 计和实现一个网络身份认证系统时，要满足信息来源的可信性、信息传输的完整性、信息传送 的不可抵赖性、控制非法用户对系统资源的访问等目标，同时身份认证系统还应考虑要达到抵 抗重放攻击、抵抗密码分析攻击、实现双向身份认证功能、提供双因子身份认证、实现良好的 认证同步机制、保护身份认证者的身份信息、提高身份认证协议的效率、减少认证服务器的敏 感信息等要求。身份认证技术研究及身份认证系统实现都是绕着上述目标和要求来进行的。

**10.1.3** **网络身份认证体系的发展现状**

身份认证是网络安全应用系统中的第一道防线，目的是验证通信双方的真实身份，防止非 法用户假冒合法用户窃取敏感数据。在安全的网络通信中，涉及的通信各方必须通过某种形式 的身份验证机制来证明他们的身份，验证用户的身份与其所宣称的是否一致，然后才能实现对 于不同用户的访问控制和记录。

一般来说，用户身份认证可以通过三种基本方式或其组合方式来实现：

(1)用户所知道的某个秘密信息，例如用户知道自己的口令。

(2)用户持有的某个秘密信息(硬件),用户必须持有合法的随身携带的物理介质，例 如智能卡中存储用户的个人特定私有信息，访问系统资源时必须有智能卡。

(3)用户所具有的生物特征，如指纹、声音、视网膜扫描等，但这种方案一般造价较高， 适用于保密程度很高的场合。

传统的认证技术采用简单的口令形式，系统事先保存每个用户的二元组信息(IDX,PwX), 进入系统时用户x 输入用户名IDx 和口令PwX, 系统对保存的用户信息与用户输入的信息进 行比较，从而判断用户身份的合法性。这种认证方法操作十分简单，但同时又很不安全，因为



**Windows C/C++加密解密实战**

其安全性仅仅基于用户口令的保密性，而用户口令一般较短且容易猜测，因此这种方案不能抵 御口令猜测攻击；另外，其最大的问题是用户名和口令都是以明文方式在网络中传输的，极易 遗受重放攻击和字典攻击，因此难以支持交换敏感的、重要的数据应用。目前一般都采用高强 度的密码技术进行身份认证。

当前互联网上典型的身份认证系统有基于共享密钥(或对称密钥)的集中式认证和以RSA 算法为代表的公钥认证两种。前者的代表是MIT(美国麻省理工大学)开发的Kerberos 协议， 后者是基于PKI 的系统。

Kerberos 提供了一种在开放式网络环境下进行身份认证的方法，是基于可信赖的第三方的 认证系统。Kerberos 基于对称密码学(采用的是 DES, 但也可以用其他算法代替),它与网 络上的每个实体分别共享一个不同的秘密密钥，是否知道该秘密密钥便是身份的证明。采用基 于PKI 的认证技术类似于 Kerberos 技术，它也依赖于共同信赖的第三方来实现认证，不同的 是它采用非对称密码体制(公钥体制),并利用“数字证书”这一静态的电子文件来实施公钥 认证。

与共享密钥认证相比，公钥认证的优势主要体现在两个方面：

(1)更高的安全强度。Kerberos 系统中的密钥分发中心(Key Distibution Center,KDC) 需要在线参与每一对通信双方的会话密钥协商过程，只要连入 Internet, 就有可能受到来自网 络的攻击，只要KDC 被攻破，整个 Kerberos 系统就完全崩溃。而且当通信对象很多时，KDC 就会成为网络瓶颈。与此不同，基于 PKI 的系统中通信方之间的相互认证并不需要证书权威 CA(Certificate Authority)的在线参与，管理证书的CA 可以离线操作，完全脱离外部 Internet 的骚扰，只要物理上是安全的，攻击者根本没法接触到CA, 更谈不上攻击了，所以CA 的安 全性比KDC 好。

(2)公钥系统便于提供严格意义上的数字签名服务，这在电子商务中是很重要的，而 Kerberos 协议最初是设计用来提供认证和密钥交换的，它不能用来进行数字签名，因而也不能 提供非否认机制。

在大规模的网络环境下，利用密码学技术进行通信方的身份认证，无论是共享密钥还是公 钥体制，理想的途径都是有一个权威的第三方来协助进行密钥分发及身份鉴别。在 PKI 体制 中，权威的第三方不需要在线参与认证过程，采用证书的形式使得整个安全体系有很好的扩展 性，它对数字签名的良好支持能为交易提供不可否认性的仲裁，这些都是共享密钥的认证系统 无法达到的，因而以PKI 体制为代表的公钥认证技术正逐渐取代共享密钥认证成为网络身份 认证和授权体系的主流。

公钥技术具有签名和加密的功能，可以分别构造基本挑战/响应协议。基于公钥加密的 双向挑战/响应协议的认证技术能够提供很可靠的认证服务。公钥认证需要双方事先已经拥 有对方的公开密钥，因此公钥的分发成为公钥认证协议的重要环节。公钥系统采用证书权威 机 构CA 签发证书的方式来分发公钥，X.509 协议定义了证书格式。公钥基础设施(PKI) 以 公钥技术为基础，它很好地解决了网络中用户的身份认证问题，并且保障了网络上信息传送 的准确性、完整性和不可否认性。也正是在它的支持下，在线支付得以实施，电子商务才真



**第10章身份认证和PKI 理论基础**

正得以开展起来。

目前公钥认证技术逐渐成为主流，基于X.509 证书和CA 的 PKI 认证系统将是 Internet 网络认证系统的主要发展方向。利用建立在PKI 基础上的X.509 数字证书，通过把要传输的 数字信息进行加密和签名，可以保证信息传输的机密性、真实性、完整性和不可否认性，从而 保证信息的安全传输。但是 PKI 也存在一些缺点，除了它的完整、庞大建设成本很高和实现 技术复杂外，目前重点还要考虑的是不同 PKI 系统之间如何实现相互兼容性和相互操作性， 如何建设沟通不同PKI 信任体系的管理机制和技术机制，实现CA 机构和 CA 机构的互联互通 问题。

**10.2 身份认证技术基础**

身份认证理论是一门新兴的理论，是现代密码学发展的重要分支。在一个身份认证系统设 计中，身份认证是第一道关卡，用户在访问所有系统之前，首先应该经过身份认证系统识别身 份，然后由身份认证系统根据用户的身份和授权数据库决定用户是否能够访问某个资源。

所谓身份认证，指的是证实被认证对象是否属实和是否有效的一个过程。其基本思想是通 过验证被认证对象的属性来达到确认被认证对象是否真实有效的目的。被认证对象的属性可以 是口令、数字签名或者像指纹、声音、视网膜这样的生理特征。身份认证常常被用于通信双方 相互确认身份，以保证通信的安全。

目前，身份认证技术已经在各个行业领域得到广泛的应用，根据实体间的关系可分为单向、 双向认证；根据认证信息的性质可分为秘密支持证明、物理介质证明、实体特征证明；根据认 证对象可分为实体对象身份认证、信息认证；根据双方的信任关系可分为无仲裁、有仲裁认证。

当前，网络上流行的身份认证技术主要有基于口令的认证、基于智能卡的认证、动态口令 认证、生物特性认证、USB Key 认证等，这些认证技术并非孤立，有很多认证过程同时使用 了多种认证机制，互相配置，以达到更加可靠安全的目的。

**10.2.1** **用户名/密码认证**

用户名/密码是很简单、很常用的身份认证方法，是基于“what you know”的验证手段。 每个用户的密码是由用户自己设定的，只有用户自己才知道。只要能够正确输入密码，计算机 就认为操作者是合法用户。实际上，由于许多用户为了防止忘记密码，经常采用诸如生日、电 话号码等容易被猜测的字符串作为密码，或者把密码抄在纸上放在一个自认为安全的地方，这 样很容易造成密码泄漏。即使能保证用户密码不被泄漏，由于密码是静态的数据，在验证过程 中需要在计算机内存和网络中传输，而每次验证使用的验证信息都是相同的，很容易被驻留在 计算机内存中的木马程序或网络中的监听设备截获。因此，从安全性上讲，用户名/密码方式 是一种极不安全的身份认证方式。



**Windows C/C++加密解密实战**

**10.2.2** **智能卡认证**

智能卡是一种内置集成电路的芯片，芯片中存有与用户身份相关的数据，智能卡由专门的 厂商通过专门的设备生产，是不可复制的硬件。智能卡由合法用户随身携带，登录时必须将智 能卡插入专用的读卡器读取其中的信息，以验证用户的身份。智能卡认证是基于“what you have” 的手段，通过智能卡硬件不可复制来保证用户身份不会被仿冒。然而由于每次从智能卡中读取

的数据是静态的，通过内存扫描或网络监听等技术，还是很容易截取到用户的身份验证信息， 因此还是存在安全隐患。

**10.2.3** **生物特征认证**

生物识别技术主要是指通过可测量的身体或行为等生物特征进行身份认证的一种技术。生 物特征是指唯一的、可以测量或可自动识别和验证的生理特征或行为方式。生物特征分为身体 特征和行为特征两类。身体特征包括：指纹、掌型、视网膜、虹膜、人体气味、脸型、手的血 管和DNA 等；行为特征包括：签名、语音、行走步态等。目前部分学者将视网膜识别、虹膜 识别和指纹识别等归为高级生物识别技术，将掌型识别、脸型识别、语音识别和签名识别等归 为次级生物识别技术，将血管纹理识别、人体气味识别、DNA 识别等归为“深奥的”生物识 别技术。

由于不同的人具有不同的生物特征，因此几乎不可能被仿冒。生物特征认证的安全性很高， 但各种相关识别技术还没有成熟，没有规模商品化，准确性和稳定性有待提高。生物特征认证 基于生物特征识别技术，受到现在的生物特征识别技术成熟度的影响，采用生物特征认证还具 有较大的局限性，特别是当生物特征缺失时，就可能没法利用。

**10.2.4** **动态口令**

动态口令技术是一种让用户密码按照时间或使用次数不断变化、每个密码只能使用一次的 技术。它采用一种叫作动态令牌的专用硬件，内置电源、密码生成芯片和显示屏，密码生成芯 片运行专门的密码算法，根据当前时间或使用次数生成当前密码并显示在显示屏上。认证服务 器采用相同的算法计算当前的有效密码，用户使用时只需要将动态令牌上显示的当前密码输入 客户端计算机，即可实现身份认证。由于每次使用的密码必须由动态令牌来产生，只有合法用 户才持有该硬件，因此只要通过密码验证就可以认为该用户的身份是可靠的。而用户每次使用 的密码都不相同，即使黑客截获了一次密码，也无法利用这个密码来仿冒合法用户的身份。

动态口令技术采用一次一密的方法，有效保证了用户身份的安全性。但是如果客户端与服 务端的时间或次数不能保持良好的同步，就可能发生合法用户无法登录的问题。并且用户每次 登录时需要通过键盘输入一长串无规律的密码，一旦输入错误就要重新操作，使用起来非常不 方便。国内目前较为典型的有VeriSign VIP动态口令技术和RSA 动态口令技术，而VeriSign 依托本土的数字认证厂商 iTrustChina, 在密码技术上针对国内进行了改良。



**第10章身份认证和PKI** **理论基础**

**10.2.5** **USB** **Key认证**

基 于USB Key的身份认证方式是近几年发展起来的一种方便、安全的身份认证技术。它 采用软硬件相结合、一次一密的强双因子认证模式，很好地解决了安全性与易用性之间的矛盾。 USB Key是一种USB 接口的硬件设备，它内置单片机或智能卡芯片，可以存储用户的密钥或 数字证书，利用 USB Key内置的密码算法实现对用户身份的认证。基于USB Key身份认证系 统主要有两种应用模式：一种是基于冲击响应的认证模式，另一种是基于PKI 体系的认证模 式。

**10.2.6** **基于冲击响应的认证模式**

USB Key内置单向散列算法(MD5), 预 先 在USB Key和服务器中存储一个证明用户身 份的密钥，当需要在网络上验证用户身份时，先由客户端向服务端发出一个验证请求。服务端 接到此请求后生成一个随机数回传给客户端PC 上插着的 USB Key,此为“冲击”。USB Key 使用该随机数与存储在 USB Key 中的密钥进行 MD5 运算，得到一个运算结果作为认证证据 传送给服务器，此为“响应”。与此同时，服务端使用该随机数与存储在服务端数据库中的该 客户密钥进行 MD5运算，如果服务器的运算结果与客户端传回的响应结果相同，就认为客户 端是一个合法用户。

可以用x 代表服务器提供的随机数，Key 代表密钥，y 代表随机数和密钥经过MD5 运算 后的结果，通过网络传输的只有随机数x 和运算结果y, 用户密钥身份认证技术基础密钥Key 既不在网络上传输又不在客户端计算机内存中出现，网络上的黑客和客户端计算机中的木马程 序都无法得到用户的密钥。由于每次认证过程使用的随机数“x” 和运算结果“y” 都不一样， 即使在网络传输的过程中认证数据被黑客截获，也无法逆推获得密钥。因此，从根本上保证了 用户身份无法被仿冒。

**10.2.7** **基于数字证书PKI** **的认证模式**

PKI(Public Key Infrastructure,公钥基础设施体系)利用一对互相匹配的密钥进行加密、 解密，一个公共密钥(公钥，Public Key)和一个私有密钥(私钥，Private Key)。其基本原 理是：由一个密钥进行加密的信息内容，只能由与之配对的另一个密钥才能进行解密。公钥可 以广泛地发给与自己有关的通信者，私钥则需要十分安全地存放起来。

每个用户拥有一个仅为本人所掌握的私钥，用它进行解密和签名；同时拥有一个公钥用于 文件发送时加密。当发送一份保密文件时，发送方使用接收方的公钥对数据加密，而接收方则 使用自己的私钥解密，这样信息就可以安全无误地到达目的地，即使被第三方截获，由于没有 相应的私钥，也无法进行解密。

冲击响应模式可以保证用户身份不被仿冒，但无法保证认证过程中数据在网络传输过程中 的安全。而基于PKI的“数字证书认证方式”可以有效保证用户的身份安全和数据传输安全。 数字证书是由可信任的第三方认证机构——数字证书认证中心(Certificate Authority,CA)颁 发的一组包含用户身份信息的数据结构，PKI 体系通过采用加密算法构建了一套完善的流程，



**Windows** **C/C++加密解密实战**

保证数字证书持有人的身份安全。而使用 USB Key可以保障数字证书无法被复制，所有密钥 运算在 USB Key中实现，用户密钥不在计算机内存出现，也不在网络中传播，只有 USB Key 的持有人才能够对数字证书进行操作，安全性有了保障。由于 USB Key具有安全可靠、便于 携带、使用方便、成本低廉的优点，加上PKI 体系完善的数据保护机制，因此使用USB Key 存储数字证书的认证方式已经成为目前主要的认证模式。

**10.3 PKI概述**

随着网络信息安全技术的发展，公钥基础设施——PKI 在国内外得到广泛的应用。我国目 前已经公布了国家PKI 的总体框架。它由国家电子政务PKI 体系和国家公共PKI 体系组成。 PKI 是目前解决电子商务安全的主要方案。

PKIX(Public-Key Infrastructure using X.509)工作组给PKI 的定义为：“是一组建立在公 开密钥算法基础上的硬件、软件、人员和应用程序的集合，它应具备产生、管理、存储、分发 和废止证书的能力”。PKI 是一种遵循一定标准的密钥管理平台，它能够为所有网络应用提供 加密和数字签名等密码服务及所必需的密钥和证书管理。

**10.3.1 PKI的国内外应用状态**

美国是最早提出PKI 概念的国家，并于1996年成立了美国联邦 PKI 筹委会。与PKI 相关 的绝大部分标准都由美国制定，其PKI 技术在世界上处于领先地位。2000年6月30日，美国 前总统克林顿正式签署美国《全球及全国商业电子签名法》给予电子签名、数字证书以法律上 的保护，这一决定使电子认证问题迅速成为各国政府关注的热点。加拿大在1993就已经开始 了政府PKI 体系雏形的研究工作，到2000年已在PKI 体系方面获得重要的进展，已建成的政 府 PKI 体系为联邦政府与公众机构、商业机构等进行电子数据交换时提供信息安全的保障， 推动了政府内部管理电子化的进程。加拿大与美国代表了发达国家PKI 发展的主流。

欧洲在PKI 基础建设方面也成绩显着，已颁布了93/1999EC 法规，强调技术中立、隐私 权保护、国内与国外相互认证以及无歧视等原则。为了解决各国 PKI 之间的协同工作问题， 它采取了一系统策略，如积极资助相关研究所、大学和企业研究PKI 相关技术，资助PKI 互 操作性相关技术研究，并建立CA 网络及其顶级CA。 并且于2000年10月成立了欧洲桥CA 指导委员会，于2001年3月23日成立了欧洲桥CA。

在亚洲，韩国是最早开发PKI 体系的困家。韩国的认证架构主要分为三个等级：最上一 层是信息通信部，中间是信息通信部设立的国家CA 中心，最下级是信息通信部指定的下级授 权认证机构 (LCA) 。 日本的 PKI 应用体系按公众和私人两大领域来划分，而且在公众领域 的市场还要进一步细分，主要分为商业、政府以及公众管理内务、电信、邮政三大块。此外， 还有很多国家在开展PKI 方面的研究，并且都成立了CA 认证机构。较有影响力的国外 PKI 公司有Baltimore和 Entrust, 其产品如 Entrust/PKI5.0,已经能较好地满足商业企业的实际需



**第10章身份认证和PKI** **理论基础**

求 。Verisign 公司也已经开始提供PKI 服务，Internet 上很多软件的签名认证都来自 Verisign 公司。

被誉为“PK 技术盛会”的亚洲PKI论坛第三届国际大会于2002年7月8日到10日在韩 国首尔举行。会议结果表明：目前PKI 技术在亚洲各国、各地区已经有了一定的发展与应用， 尤其在电子政务与电子商务领域，PKI 技术正在发挥着巨大的作用。但是，PKI 技术在整个亚 洲还处于“爬坡”阶段，还存在着许多亟待解决的问题。而在中国，PKI 技术在中国的商业银 行、政府采购以及网上购物中得到广泛应用，PKI 技术在中国有着广泛的应用前景。

我国的PKI 技术从1998年开始起步，由于政府和各有关部门近年来对PKI 产业的发展给 予了高度重视，2001年 PKI 技术被列为“十五”863计划信息安全主题重大项目，并于同年 10月成立了国家863计划信息安全基础设施研究中心。国家计委也在制定新的计划来支持PKI 产业的发展，在国家电子政务工程中明确提出了要构建PKI 体系。目前，我国已全面推动PKI 技术研究与应用。

1998年，国内第一家以实体形式运营的上海CA 中 心(SHECA) 成立。目前，国内的CA 机构分为区域型、行业型、商业型和企业型4类。截至2002年年底，前三种CA 机构已有60 余家，58%的省市建立了区域 CA, 部分部委建立了行业 CA。其中，全国性的行业CA 中心 有中国金融认证中心(CFCA) 、 中国电信认证中心(CTCA) 等。区域型CA有一定地区性， 也称为地区CA, 如上海CA 中心、广东电子商务认证中心。

我国正在拟订全面发展国内PKI 建设的规则，其中包括国家电子政务PKI 体系和国家公 共PKI 体系的建设。从2003年1月7日在京召开的中国PKI战略发展与应用研讨会可知，我 国将组建一个国家PKI 协调管理委员会来统管国内的PKI 建设，由其来负责制订国家PKI管 理政策、国家 PKI 体系发展规划，监督、指导国家电子政务PKI 体系和国家公共PKI 体系的 建设、运行和应用。据有关机构预测，有关电子政务的外网 PKI 体系建设即将展开，在电子 政务之后，将迎来电子商务这个PKI 建设的更大商机。中国的PKI 建设即将迎来大发展。

**10.3.2** **PKI的应用前景**

广泛的应用是普及一项技术的保障。PKI 支持SSL 、IP over VPN 、S/MIME等协议，这使 得它可以支持加密 Web、VPN、 安全邮件等应用。而且，PKI 支持不同CA 间的交叉认证，并 能实现证书、密钥对的自动更换，这扩展了它的应用范畴。

一个完整的PKI 产品除主要功能外，还包括交叉认证、支持LDAP 协议、支持用于认证 的智能卡等。此外，PKI 的特性融入各种应用(如防火墙、浏览器、电子邮件、群件、网络操 作系统)也正在成为趋势。基于PKI 技术的 IPSec协议现在已经成为架构VPN 的基础。它可 以为路由器之间、防火墙之间或者路由器和防火墙之间提供经过加密和认证的通信。目前，发 展很快的安全电子邮件协议是 S/MIME,S/MIME 是一个用于发送安全报文的IETF 标准。它 采用了 PKI 数字签名技术并支持消息和附件的加密，无须收发，双方共享相同密钥。目前该 标准包括密码报文语法、报文规范、证书处理以及证书申请语法等方面的内容。基于PKI 技 术的SSL/TLS 是互联网中访问Web 服务器很重要的安全协议。当然，它们也可以应用于基于



**Windows C/C++加密解密实战**

客户机/服务器模型的非 Web类型的应用系统。SSL/TLS都利用PKI的数字证书来认证客户和 服务器的身份。

从应用前景来看，随着Internet 应用的不断普及和深入，政府部门需要PKI 支持管理；商 业企业内部、企业与企业之间、区域性服务网络、电子商务网站都需要PKI 技术和解决方案； 大企业需要建立自己的PKI 平台；小企业需要社会提供商业PKI 服务。此外，作为PKI 的一 种应用，基于PKI 的虚拟专用网市场也随着B2B 电子商务的发展而迅速膨胀。

总的来说，PKI 的市场需求非常巨大，基于PKI 的应用包括许多内容，如wwW 安全、 电子邮件安全、电子数据交换、信用卡交易安全、VPN 等。从行业应用来看，电子商务、电 子政务、远程教育等方面都离不开PKI 技术。

**10.3.3 PKI 存在的问题及发展趋势**

尽管取得了很大的进展，在PKI 领域还存在以下问题亟待解决，今后PKI 技术将主要在 这些方面进行更深入的研究。

(1)X.509 属性证书

提起属性证书就不能不提起授权管理基础设施(Privilege Management Infrastructure,PMI)。 PMI 授权技术的核心思想是以资源管理为核心，将对资源的访问控制权统一交由授权机构进 行管理，即由资源的所有者来进行访问控制管理。与 PKI 信任技术相比，两者的主要区别在 于PKI 证明用户是谁，并将用户的身份信息保存在用户的公钥证书中；而PMI 证明这个用户 有什么权限、什么属性、能干什么,并将用户的属性信息保存在授权证书(又称管理证书)中。

例如，销售商为了决定一笔订货是否可信，是否应该发货给定货人，他就必须知道定货人的信 用情况，而不仅仅是其名字。为了使上述附加信息能够保存在证书中，X.509 v4中引入了公钥 证书扩展项，这种证书扩展项可以保存任何类型的附加数据。随后，各个证书系统纷纷引入了 自己的专有证书扩展项，以满足各自应用的需求。

(2)漫游证书

到目前为止，能提供证书和其对应私钥移动性的实际解决方案有两种：第一种是智能卡技 术，其缺点是易丢失和损坏，并且依赖读卡器；第二种是将证书和私钥复制到一张软盘备用， 但软盘不仅容易丢失和损坏，而且安全性也较差。一个更新的解决方案 漫游证书正逐步被 采用，它通过第三方软件提供，只需在任何系统中正确地配置，该软件(或者插件)就可以允 许用户访问自己的公钥/私钥对。它的基本原理很简单，即将用户的证书和私钥放在一个安全 的中央服务器上，当用户登录一个本地系统时，从服务器安全地检索出公钥/私钥对，并将其 放在本地系统的内存中以备后用，当用户完成工作并从本地系统注销后，该软件自动删除存放 在本地系统中的用户证书和私钥。这种解决方案的好处是可以明显提高易用性，降低证书的使 用成本，但它与已有的一些标准不一致，因而在应用中受到了一定限制。

(3)无线 PKI

随着无线通信技术的广泛应用，无线通信领域的安全问题也引起了广泛的重视。将PKI



**第10章身份认证和PKI** **理论基础**

技术直接应用于无线通信领域存在两方面的问题：其一是无线终端的资源有限(运算能力、 存储能力等);其二是通信模式不同。为了适应这些需求，目前已公布了WPKI 草案，其内 容涉及WPKI的运作方式、WPKI 如何与现行的PKI 服务相结合等。WPKI 中定义了三种不 同的通信安全模式：使用服务器证书的WTLS Class2模式、使用Client证书的 ITLS Class3 模式、使用Client 证书合并WMLScript的 Signet 模式。所谓的Class1 、Class2 及Class3 是 定义在WTLS 标准中的安全需求。在证书编码方面，WPKI 证书格式想尽量减少常规证书所 需的存储量。采用的机制有两种：其一是重新定义一种证书格式(WTLS 证书格式),以此 减小X.509 证书的尺寸；其二是采用ECC 算法减小证书的尺寸，因为ECC 密钥的长度比其 他算法的密钥要短得多。目前，对 PKI 技术的研究与应用正处于探索中，但它代表了 PKI 技术发展的一个重要趋势。

(4)信任模型

PKI 从根本上说致力于解决通过网络交互的实体之间的信任问题。信任模型的构建是PKI 系统在宏观角度上的核心问题。建立一个可以连接 Internet 上任意实体的全球化的信任体系是 PKI 研究的一个长期目标。PEW(Privacy Enhanced Mail)的失败证明了严格层次化的信任模 型不适用于Internet这样灵活的结构；以PGP(PrettyGood Privacy)为代表的以用户为中心的 信任模型无法扩展到大规模的应用；依赖于流行浏览器中预安装的信任CA 证书的Web 信任 模型在安全性上一直存在着很大的漏洞；通过交叉认证实现的分布式信任模型被广泛地应用， 但是路径长度与路径发现问题增加了PKI 系统使用时的复杂性。改进和结合各种已有模式的 新型信任模型也正在不断地涌现，但是建立真正的全球化信任体系仍然是 PKI 研究中的一个 难题。

(5)证书撤销

CA 如何发布证书撤销信息是影响其是否能被广泛应用的重要因素。1994年，美国的 MITRE 公司在一个报告中指出，撤销证书信息的发布将潜在地成为运营大规模PKI 系统成本 最昂贵的部分。同时，MITRE 公司提出了一种基本的证书撤销机制，即使用证书撤销列表机 制。基本证书撤销列表机制的缺陷在于CRL的长度可能很大，由此产生的网络带宽资源消耗 在大规模PKI 系统中是不可忽视的。同时，因为CRL 是周期性发布的，不适用于具有实时性 证书撤销信息需求的证书使用环境。针对这些问题，有很多改进的方案被提出，如增量CRL, 致力于减小发布CRL 的平均带宽和间隔时间；CRL 分布点，通过向不同的地点发布CRL 分 段减小每一个CRL 分段的长度；分时CRL, 减小发布CRL 的峰值带宽。此外，针对实时性 问题，目前广泛采取的方案是在线证书状态协议(Online Certificate Status Protocol,OCSP)。 基于该协议的证书撤销机制使验证者能够实时地对用于某特定交易的证书进行检查。上述各种 方案及其改进方案都致力于解决证书撤销中的一个或几个方面的问题，但是目前尚不存在一种 真正适用于大规模PK 系统的高性能的证书撤销方案。

(6)实体命名问题

证书绑定实体身份与实体公钥，而实体身份通过证书上的实体名表示。X.509 v3证书格式 定义中的实体名采用X.500 可识别名，即通常所说的DN。从理论上考虑，通过X.500DN区



**Windows C/C++加密解密实战**

分全球的不同实体是完全可以实现的，但是实际上DN 机制并不完全成功。首先，X.500 目录 概念并没有得到充分的推广和接受；其次，在很多场合中，X.500 命名机制中各个层次的命名 机构并不具有实际的权威性，它们对于名称分配可能是不必要的。证书中的扩展字段(Subject Alternative Name)正是基于这个原因产生的，但是仅仅通过这个字段增加实体命名方式并不 能从根本上解决问题。目前许多PKI 研究和标准化活动，比如SDSI(Simple Distributed Security Infrastructure,MIT 提出的一种试图解决分布式计算环境中安全问题的信任模型)、SPKI(Simple Public-Key Infrastructure,TETF SPKI工作组以简化证书格式为主要目的建立的PKI 信任模型) 等，都关注于解决PKI的实体命名问题。

**10.4 基于X.509** **证书的** **PKI** **认证体系**

目前，X.509 证书已得到广泛的应用，成为开放网络环境中公钥管理的重要手段。公开密 钥的管理是一个整体，除了数字证书外，还需要证书签发者(CA) 、 注册中心(RA) 、 存储 库 (Reposition)等多种实体的参与。各参与方都要维护自己的安全参数，如自己的密钥对、 所信任的CA 公钥以及所遵循的安全策略等。同时，为了保证公钥的有效性，还应该有合理的 证书撤销机制和证书发布策略。所有这些构成了公钥基础设施(PKI), 它建立在一套严格定 义的标准之上，这些标准用于控制证书生命周期的各个方面。

**10.4.1** **数字证书**

**1.基本定义**

数字证书就是互联网通信中标志通信各方身份信息的一系列数据，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。它是由一个权威的 证书授权机构发行的，人们可以在网上用它来识别对方的身份。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。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。

在数字签名过程中，人们用发送方的公钥对数字签名进行解密，从而来证实文件确实是发 送方发送的，但是没有证实发送方是否确实是其所声称的文件拥有者。在公钥体制中，公钥本 身的保密性并不重要，对公钥而言本来就是要公开的，没有防监听和泄漏的问题，但公钥的发 布仍然存在安全性问题，必须确信拿到的公钥确实是属于它申明的那个人，否则就无法保证系 统的安全性，攻击者就可能用伪造公钥制造伪签字行骗，防止这种情况出现的方法显然是通过 信任渠道得到公钥。目前的解决方法是通过签发数字证书把公钥与其真正的拥有者紧密结合起 来。

数字证书是一段包含用户身份信息、用户公钥信息以及身份验证机构数字签名的数据。身 份验证机构的数字签名可以确保证书信息的真实性。通常数字证书采用公钥体制，即利用一对 互相匹配的密钥进行加密、解密。每个用户自己设定特定的仅为本人所有的私有密钥(私钥),



**第10章身份认证和PKI** **理论基础**

用它进行解密和签名；同时设定公共密钥(公钥)并由本人公开，为一组用户所共享，用于加 密和验证签名。当发送一份保密文件时，发送方使用接收方的公钥对数据加密，而接收方则使 用自己的私钥解密，这样信息就可以安全无误地到达目的地了。通过数字的手段保证加密过程 是一个不可逆的过程，即只有用私钥才能解密。公开密钥技术解决了密钥发布的管理问题，用 户可以公开其公钥，而保留其私钥。

数字证书颁发过程一般为：用户首先产生自己的密钥对，并将公共密钥及部分个人身份信 息传送给认证中心。认证中心在核实身份后，将执行一些必要的步骤，以确信请求确实由用户 发送而来，然后认证中心将发给用户数字证书，该证书内包含用户的个人信息和其公钥信息， 同时还附有认证中心的签名信息。用户就可以使用自己的数字证书进行相关的各种活动了。数 字证书由独立的证书发行机构发布。数字证书各不相同，每种证书可提供不同级别的可信度。

**2.数字证书的特点**

数字证书在一个身份和该身份的持有者所拥有的公私钥对之间建立了一种联系，它具有以 下特点：

(1)数字证书是PKI 体系的核心元素

PKI的核心执行机构是CA 认证中心，认证中心所签发的数字证书是PKI 的核心组成部分， 而且是PKI基本的活动工具，是PKI的应用主体。它完成PKI 所提供的全部安全服务功能， 可以说PKI体系中的一切活动都是围绕数字证书进行的。

(2)数字证书是权威的电子文档

数字证书实际上是由可信的、公正的第三方权威认证机构所签发的。数字证书的内容必须 包含权威认证机构的数字签名，即对数字证书的内容进行散列杂凑值运算后，再用该CA 机构 的私钥对证书的杂凑值进行非对称加密运算，即CA 对证书的数字签名。CA 对其签发的数字 证书内容的签名具有法律效力，是符合国家电子签名法要求的，所以，它在网上交易、网上实 际相互认证的过程中是一个公认的、权威的电子文档。

(3)数字证书是网上身份的证明

互联网上的身份认证靠证书机制实现身份的识别与鉴别，因为数字证书的主要内容就有证 书持有者的真实姓名、身份唯一标识和该实体的公钥信息。电子认证机构CA 靠对实体签发的 这个数字证书来证实该实体在网上的真实身份。

(4)数字证书是PKI 体系公钥的载体

公钥基础设施是靠公私钥对的加/解密运算机制完成PKI服务的，私钥严格保密，公钥要 方便地公布。方便地传递和发布公钥是公钥基础设施的优势。公钥发布或传递的方式一是靠 LDAP目录服务器，即将CA 签发的证书发布在目录服务器上，供需进行通信的证书依赖方索 取；二是由通信双方的一方将公钥证书与加密(签名)后的数据一起发送给依赖方的证书用户。 这种公钥的传递载体就是数字证书。



**Windows C/C++加密解密实战**

**3.数字证书的格式**

数字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(证书投权中心)的名称、该证书的序列号等信息，数字证书 的格式遵循IUT-T X509国际标准。

X.509目前有三个版本：v1、v2和v3,X.509 v3证书标准是在v2 版的基础上对证书形式 形成能够附带额外信息的扩展项后形成的，如表10-1所示。

**表10-1** **X.509 v3证书标准的扩展项**

|  |  |  |
| --- | --- | --- |
| **序** **号** | **项名称** | **描** **述** |
| 1 | Version | 版本号 |
| 2 | seialNumber | 序列号 |
| 3 | Signature | 签名算法 |
| 4 | Issuer | 颁发者 |
| 5 | Validity | 有效日期 |
| 6 | Subject | 主体 |
| 7 | subjectPublicKeyInfo | 主体公钥信息 |
| 8 | issuserUniqueID | 颁发者唯一标识符 |
| 9 | subjectUniqueID | 主体唯一标识符 |
| 10 | Extensions | 扩展项 |

X.509 结 构 也 可 通 过ASN.1 标准编码，其基本数据结构描述为：

Certificate::=SEQUENCE{

tbsCertificate TBSCertificate,

signatureAlgorithm Algorithmldentifer, signatureValue BIT STRING

}

TBScertificate ::SEQUENCE{

Version [0]EXPLICIT Version DEFAUTV1, serivalNumber CertificateSerialNumber

signature Algorithmldentifier,

Issuer Name

validity Validity, subject Name,

subjectPublicKeylnfo SubjectPublicKeyInfo,

ssuerUniqueID[1]IMPLICIT Uniqueidentifier OPTIONAL,/如果出现该项，version 必须是v2 或 v3 subjectUniqueID[2]IMPLICIT Uniqueldentifier OPTIONAL,//如果出现该项，version 必须是v2 或 v3 externsion[3]EXPLICIT Extensions OPTIONAL,//如果出现该项version必须是v3

}

Version::=INTEGER(V1(0),V2(1),V3(2)} CerificationSerialNumber ::=INTEGER

Validity :=SEQUENCE{

notbefore Time, notafter Time} Time::={

utcTime UTCTime



**第10章身份认证和PKI 理论基础**

generalTime GeneralizedTime}

Uniqueldentifier::=BIT STRING

SubjectPublicKeyInfo ::=SEQUENCE{ algorithm Algorithmldentifie

subjectPublicKey BITSTRING}

Extension ::=SEQUENCE SIZE(1..MAX)OF Extension extnID OBJECT IDENTIFIER,

critical BOOLEAN DEFAULTFALSE, extn Value OCTET STRING}

上述的证书数据结构由 tbsCertificate 、signatureAlgorithm 和 signatureValue 三个域构成。 这些域的含义如下：

(1)tbsCertificate 域包含主体名称和签发者名称、主体的公钥、证书的有效期及其他的 相关信息。

(2)signatureAlgorithm 域包含证书签发机构签发该证书所使用的密码算法的标识符。一 个算法标识符的ASN.1 结构如下：

Algorithmldentifier::=SEQUENCE{ Algorithm OBJECT IDENTIFIER

parameters ANY DEFINED BY algorithm OPTIONAL}

算法标识符用来标识一个密码算法，其中的OBJECT IDENTIFIER 部分标识了具体的算法 ( 如DSAwith SHA-1),其可选参数的内容完全依赖于所标识的算法。该域的算法标识符必须 与 tbsCertificate 中的 signature 标识的签名算法相同。

SignatureValue 域包含对tbsCertificate 域进行数字签名的结果。采用ASN.IDER 编码的 tbsCertificate作为数字签名的输入，而签名的结果则按照ASN.1 编码成 BIT STRING类型并 保存在证书签名值域内。

**10.4.2** **数字信封**

数字信封是身份认证过程中常用的一种信息保护手段。

数字信封就是信息发送端利用接收端的公钥对一个通信密钥(对称密钥)进行加密，形成 一个数字信封并传送给对方。只有指定接收方才能用对应的私钥打开数字信封，获取该对称密 钥，用它来解读传送的信息。这就好比在实际生活中，将一把钥匙装在信封里，邮寄给对方， 对方收到信件后，将钥匙取出，再用它打开保密箱一样。

数字信封技术结合了对称密钥加密技术和公开密钥加密技术的优点，可克服对称密钥加密 中密钥分发困难和公开密钥加密中加密时间长的问题，使用两个层次的加密来获得公开密钥技 术的灵活性和对称密钥技术的高效性，保证信息的安全。

数字信封的具体实现步骤如下：

(1)信息发送方首先利用随机产生的对称密钥SK 加密待发送的信息E, 包括信息明文、 数字签名和发送者证书公钥。



**Windows C/C++加密解密实战**

(2)发送方利用接收方的公钥加密对称密钥，被公钥加密后的对称密钥被称为数字信封

DE。

(3)发送方将第一步和第二步的结果传给接收方。

(4)信息接收方用自己的私钥解密数字信封，得到对称密钥SK。

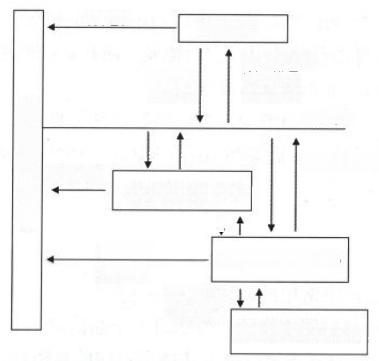
(5)利用对称密钥解密所得到的信息。

这样就保证了数据传输的真实性和完整性。信息发送方使用密码对信息进行加密，从而保 证只有规定的收信人才能阅读信的内容。采用数字信封技术后，即使加密文件被他人非法截获， 因为截获者无法得到发送方的通信密钥，故不可能对文件进行解密。

**10.4.3** **PKI体系结构**

**1.PKI 结构模型**

一个完整的PKI 产品通常应具备这些功能：根据X.509 标准发放证书，产生密钥对，管 理密钥和证书，为用户提供 PKI 服务，如用户安全登录、增加和删除用户、检验证书等。其 他相关功能还包括交叉认证、支持LDAP 协议、支持用于认证的智能卡等。图10-2所示是一 个典型的PKI 实体图。



发布证书 终端实体

初始注册/认证

证书更新 密钥对恢复

搬销请求 密钥对更新

PKI用户

PKI管理实体

发布证书

注册登记机关(RA)



发布证书

证书签证机关1(CA1)

发 布GPL

交叉认证 证书更新

证书签发机关2(CA2)

证书和证书列表仓库

图10-2

一个典型完整、有效的PKI应用系统至少应具有以下部分：

(1)认证中心 (CA)

CA 是 PKI 的核心，CA 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，CA 还要负责用户证书 废止列表 (CRL) 的管理。

(2)注册机构 (RA)

RA的主要功能为证实证书申请者的身份，批准证书申请者的证书申请，将申请者的身份



**第10章身份认证和PKI** **理论基础**

信息和公钥以数字签名的方式发送给CA, 验证其有效性，并向CA 发出该申请。在实际应用 中 ，PKI 的 RA 功能并不独立存在，而是合并在CA 之中。

(3)证书库

证书库存放了经CA 签发的证书和已撤销证书列表，用户可以使用应用程序从证书库中得 到对方的证书，验证其真伪，查询证书的状态。证书库通过目录技术实现网络服务。LDAP ( 轻 量级目录访问协议)定义了标准的协议来存取目录系统。支持 LDAP 协议的目录系统能够支 持大量用户同时访问，对检索请求也有很好的响应。

(4)证书的申请者和证书的信任方

证书的申请者也是证书的持有者。PKI可以为证书申请者提供包括证书请求、密钥对生成、 证书生成、密钥更新和证书撤销等功能。PKI 为证书信任方提供了检查证书申请者身份以及与 证书申请者进行安全数据交换的功能。证书信任方的功能包括接收证书、证书请求、核实证书、 数字加密、检查身份和数字签名等。

(5)客户端软件

为了方便用户操作，解决PKI 的应用问题，在客户端安装软件以实现数字签名、加密传 输数据等功能。此外，客户端软件还负责在认证过程中查询证书和相关证书的撤销消息以及进 行证书路径处理等。

除了上述基本的部分外，一个完备的PKI 还需要具备这些系统：密钥备份及恢复系统、 证书撤销处理系统、PKI 应用接口等。

**2.PKI** **的标准与协议**

从整个PKI体系建立与发展的历程来看，与PKI 相关的标准主要有：

(1)X.500(1993) 信息技术之开放系统互联：概念、模型及服务简述

X.500 是一套已经被国际标准化组织 (ISO) 接受的目录服务系统标准，它定义了一个机 构如何在全局范围内共享其名字和与之相关的对象。X.500是层次性的，其中的管理域(机构、 分支、部门和工作组)可以提供这些域内的用户和资源信息。在PKI 体系中，X.500 被用来唯 一标识一个实体，该实体可以是机构、组织、个人或一台服务器。X.500 被认为是实现目录服 务的较佳途径，但X.500 的实现需要较大的投资，并且比其他方式速度慢，而其优势是具有信 息模型、多功能和开放性。

(2)X.509(1993) 信息技术之开放系统互联：鉴别框架

X.509 是由国际电信联盟(ITU-T) 制定的数字证书标准。在X.500 确保用户名称唯一性 的基础上，X.509 为 X.500 用户名称提供了通信实体的鉴别机制，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 的最初版本公布于1988年。X.509 证书由用户公共密钥 和用户标识符组成，此外，还包括版本号、证书序列号、CA 标识符、签名算法标识、签发者 名称、证书有效期等信息。这一标准的新版本是X.509 V3,它定义了包含扩展信息的数字证 书。该版数字证书提供了一个扩展信息字段，用来提供更多的灵活性及特殊应用环境下所需的



**Windows** **C/C++加密解密实战**

信息传送。

(3)PKCS 系列标准

由 RSA 实验室制订的PKCS 系列标准是一套针对PKI 体系的加解密、签名、密钥交换、 分发格式及行为的标准，该标准目前已经成为PKI体系中不可缺少的一部分。

(4)在线证书状态协议

OCSP(Online Certificate Status Protocol,在线证书状态协议)是IEIF 颁布的用于检查数 字证书在某一交易时刻是否仍然有效的标准。该标准提供给 PKI 用户一条方便快捷的数字证 书状态查询通道，使PKI 体系能够更有效、更安全地在各个领域中被广泛应用。

(5)轻量级目录访问协议

LDAP(Lightweight Directory Access Protocol,轻量级目录访问协议)规范(RFC1487) 简化了笨重的X.500目录访问协议，并且在功能性数据表示、编码和传输方面都进行了相应的 修改。1997年，LDAP 第3版成为互联网标准。目前，LDAPV3 已经在PKI 体系中被广泛地 应用于证书信息发布、CRL 信息发布、CA 政策以及与信息发布相关的各个方面。

除了以上协议外，还有一些构建在PKI 体系上的应用协议，包括SET 协议和SSL 协议。 目 前PKI 体系中已经包含众多的标准和标准协议，由于PKI技术的不断进步和完善，以及其 应用的不断普及，将来还会有更多的标准和协议加入。

**3.PKI 的功能**

PKI 提供了一整套安全机制，其主要包括以下功能：

(1)产生、验证和分发密钥

根据密钥生成模式不同，用户公私钥对的产生、验证及分发有两种方式：用户自己产生密 钥对，这种方式适用于分布式密钥生成模式；CA 为用户产生密钥对，这种方式适用于集中式 密钥生成模式。

(2)签名和验证

在 PKI 体系中，对信息和文件的签名以及对数字签名的验证都是很普遍的操作。其数字 签名和验证可采用多种方法，如RSA 、DES等。

(3)证书的获取

在验证信息的数字签名时，用户必须事先获取信息发送者的公钥证书，以对信息进行解密 验证，并验证发送者身份的有效性。

(4)验证证书

验证证书的过程是迭代寻找证书链中下一个证书和它相应的上级CA 证书。在检查每个证 书前必须检查相应的 CRL。用户检查证书的路径从最后一个证书的有效性开始，一旦验证通 过，就提取该证书的公钥，用于检查下一个证书，直到验证完发送者的签名证书，并用该证书 的公钥验证签名。这个过程是回溯的。



**第10章身份认证和** **PKI理论基础**

(5)保存证书

保存证书是指PKI 实体本地存储证书，以减少在PKI 体系中获得证书的时间，并提高数 字签名的效率。证书存储单元对证书进行定时维护，包括清除与新发布的CRL 文件比较作废 或过期的证书。

(6)证书废止的申请

当 PKI 中某实体的私钥被泄密时，被泄密的私钥对应的公钥证书应该作废。另一种情况 是证书持有者终止该证书的使用或与某组织的关系中止，该证书也应该作废。证书终止的方式 有两种，如果是密钥泄漏，证书持有者可以直接通知相应CA; 如果是因关系中止，就由原关 系中的组织方面通知 CA。

(7)密钥的恢复

在密钥泄密、证书作废后，为了恢复 PKI 实体的业务处理和产生数字签名，泄密实体将 获得一对新的密钥，并要求CA 产生新的证书。每一个实体产生新的密钥时，会获得CA 用新 私钥签发的新证书，而原来用泄密的密钥签发的旧证书将作废，并放入 CRL。

(8)CRL 的获取

每一个CA 均可以产生CRL 。CRL可以定期产生，也可以每次有证书作废请求后实时产 生 。CA 应将其产生的CRL 及时发送到目录服务器上去。CRL 的获取可以有多种方式：CA 产 生CRL 后，自动发送给下属各实体：大多数情况下，由使用证书的各PKI 实体从目录服务器 中获得相应的CRL。

(9)密钥更新

在密钥泄密的情况下，将产生新的密钥和证书。在密钥没有被泄密的情况下，密钥也应该 定时更换。更换的方式有多种，PKI 体系中的各实体可以在同一天，也可以在不同时间更换密 钥。无论哪种方式，PKI 中的实体都应该在密钥截止日期之前获得新的密钥对和新证书。

(10)交叉认证

交叉认证就是多个PKI 域之间实现互操作。交叉认证实现的方法有多种：一种方法是桥 接CA, 即用一个第三方CA 作为桥，将多个CA 连接起来，成为一个可信任的统一体；另一 种方法是多个CA 的根CA(RCA) 互相签发根证书，这样当不同PKI 域中的终端用户沿着不 同的认证链检验认证到根时，就能达到互相信任的目的。通常网络通信认证关系通过信任关系 树来实现，但是通过交叉认证机制会缩短信任关系路径，提高效率。

**10.4.4** **认证机构**

认证机构 (CA) 系统是PKI 的核心，主要负责产生、分配、管理所有参与的实体所需的 身份认证数字证书。每一份数字证书都与上一级的数字签名证书相关联，最终通过安全链追溯 到一个已知的并被广泛认为是安全的、权威的、足以信赖的机构 根认证中心(根CA)。

它对网上的数据加密、数字签名、防止抵赖、数据的完整性以及身份认证所需的密钥和证书进 行统一的集中管理，支持参与的各实体在网络环境中建立和维护信任关系，以保证网络的安全。



**Windows C/C++加密解密实战**

CA 系统在创建和发布证书时，首先获得用户的请求信息，其中包括公钥，根据用户信息 产生证书，并用自己的私钥对证书签名。其他实体将使用CA 的公钥对证书进行验证。若CA 可信，则验证证书的实体可信，证书的公钥属于该实体。

CA 还负责维护和发布证书、维护证书废止列表(Certifcate Revocation Lists,CRL,又 称 证书黑名单)。当一个证书的公钥因为其他原因(不是因为到期)无效时，CRL 提供一种通 知用户和其他应用的中心管理方式。CA 系统产生CRL 后，可放到LDAP服务器或 Web服务 器的合适位置，以浏览器的方式供用户查询和下载。

一个典型的CA 系统包括CA 服务器、RA 注册机构、LDAP 服务器、安全服务器和数据 库服务器。

(1)CA 服务器

CA 服务器是整个证书机构的核心，用于数字证书签发。首先产生自身的公私密钥，然后 生成数字证书，并将其传送给安全服务器。CA 还为操作员、安全服务器以及注册机构服务器 RA 生成数字证书，安全服务器之间也需要传递证书。CA 服务器作为整体的主要机构，出于 安全考虑，应将其与其他服务器相隔离。

(2)RA 注册机构

RA(Registration Authority)是数字证书注册审批机构。RA 系统是CA 的证书发放、管理 的延伸。它面向操作员，负责证书申请者的信息录入、审核以及证书发放等工作；同时，对发 放的证书完成相应的管理功能。RA 系统在整个CA 体系中起到中介的作用，一方面向CA 服 务器转发传过来的证书申请请求，另一方面向LDAP 服务器和安全服务器转发CA 颁发的证 书和证书撤销列表。

(3)安全服务器

安全服务器面向用户，用于提供证书的申请、证书的浏览、证书的撤销列表及证书下载等 安全服务。安全服务器与用户通信采用安全信道方式，该信道用安全服务器的数字证书(由 CA 颁发)加密，传送用户的申请信息，保证证书申请的安全性。

(4)LDAP 服务器

LDAP 服务器提供目录浏览服务，负责将RA传过来的用户信息及数字证书加入服务器中。 这样，用户通过访问LDAP服务器就能够得到其他用户的数字证书。

(5)数据库服务器

数据库服务器用于认证机构的数据(如密钥、用户信息等)、日志和统计信息的存储和管 理。数据库服务器可采用如磁盘阵列、双机备份和多处理器等方式提高可靠性、稳定性、可伸 缩性等。

**10.4.5** **基于X.509证书的身份认证**

X.509 是目前唯一已经实施的PKI 系统。X.509 V3是目前的新版本，在原有版本的基础 上扩充了许多功能。X.509是定义目录服务建设X.500 系列的一部分，其核心是建立存放每个



**第10章身份认证和** **PKI** **理论基础**

用户的公钥证书的目录(仓库)。用户公钥证书由可信赖的CA 创建，并由CA 或用户存放于 目录中。

目前以ITU-T X.509证书格式为基础的PKI 体制正逐渐取代对称密钥认证，成为网络身份 认证和授权体系的主流。PKI 体制的基本原理是利用“数字证书”这一静态的电子文件来实施 公钥认证。在PKI 体制下，通信双方首先交换证书，通过CA 公钥检验证书的正确性，可以 知道证书中的公钥对应的是一个特定的对象，然后用挑战应答(Challenge-Response) 协议就 可以判断对方是否持有证书中公钥相对应的私钥，从而完成身份认证过程。当然，这种认证的 有效性是基于用户的私钥不被泄漏的基础之上。

由于这种认证技术中采用了非对称密码体制，CA 和用户的私钥都不会在网络上传输。攻 击者即使截获了用户的证书，但由于无法获得用户的私钥，也就无法解读服务器传给用户的信 息，因此有效地保证了通信双方身份的真实性和不可抵赖性。

若用户A 想与用户B 通信，则A 首先查找数据库并得到一个从A 到 B 的证书路径和用 户B 的公开密钥，这时A可使用单向、双向和三向认证协议。

(1)单向认证(One-Way Authentication)协议是从A 到 B 的单向通信。它不但建立了A 和B 双方身份的证明以及从A 到 B 的任何通信信息的完整性，而且可以防止通信过程中的任

何重放攻击。单向认证单向鉴别涉及信息从一个用户 (A) 传送到另一个用户(B), 它建立

如下要素：

① A 的身份标识和由A产生的报文。

② 打算传递给B 的报文。

③ 报文的完整性和新鲜性(还没有发送过多次)。

在这个过程中仅验证发起实体的身份标识，而不验证对应的实体标识。

报文至少要包括一个时间戳Ta、一个现时Ra 和 B 的身份标识，它们均用A 的私用密钥 签名。时间戳由一个可选的产生时间和过期时间组成，这将防止报文的延迟传送。现时Ra 用 于检测重放攻击。现时值在报文的有效时间和过期时间内必须是唯一的，这样B 能存储这个 现时直到其过期并拒绝有相同的现时的报文。

(2)双向认证(Two-Way Authentication)协议与单向认证协议类似，但它增加了来自B 的应答，既保证是B 而不是冒名者发送来的应答，又保证双方通信的机密性并可防止重放攻 击。单向和双向认证协议都使用了时间标记。

双向认证除了单向认证列的三个要素外，还要建立如下要素：

① B 的身份标识和B 产生的回答报文。

② 打算传递给A 的报文。

③ 回答报文的完整性和新鲜性。

因此，双向鉴别允许通信双方验证对方的身份。

为了验证回答报文，回答报文包括A 的现时，还包括B 产生的一个时间戳和一个现时。



**Windows C/C++加密解密实战**

和前面一样，报文可能包括签名的附加信息和用A 的公开密钥加密的会话密钥。

(3)三向认证(Three-Way Authentication)协议另外增加了从A 到 B 的消息，并避免了 使用时间标记(用鉴别时间取代)。

X.509包括三个可选的认证过程以供不同的应用使用。所有的这些过程都使用公开密钥签 名。它假定双方都知道对方的公开密钥，通过从目录获取对方的证书，或者证书被包含在每方 的初始报文中。

三向认证在三向鉴别中包括一个从A到 B 的报文，它含有一个现时的签名备份。这样设 计的目的是无须检查时间戳，因为两个现时均需由另一端返回，每一端可以检查返回的现时来 探测重放攻击。当没有时钟同步时，需要采用这种方法。

用户的身份认证可以根据双方的约定选择采用 X.509 的三种强身份认证协议中的任何一 种。这三种协议都能够有效地防止中间人攻击和重发攻击等多种常用的攻击手段。以上三种强 度认证是一个逐步完善的过程，三向认证协议安全性最好。

**第11章** **实** **战PKI**

虽然本书是讲述Windows下的加解密编程，但考虑到CA 服务器都是部署在Linux下的， 因此这一章我们把CA 部署到Linux下，也为大家今后从事Linux下的加解密开发热热身，笔 者的下一本加解密图书将在Linux下进行。当然，其实本章搭建的CA 系统也可以在Windows 下进行，但为了综合Linux 和 Windows 的联合作战效果，我们特地在Linux 下签发证书，然 后在Windows 下解析证书。记住，一个密码行业的开发者，要同时会在 Linux 和Windows下 开发，这是基本功。

**11.1 只有密码算法是不够的**

前面介绍了非对称算法，是不是只有这些密码算法就可以进行安全通信并高枕无忧了呢? 答案是否定的。在实际应用中，简单地直接使用公钥密码算法存在较为严重的安全问题。先让 我们来看一下公钥密码算法的应用流程。

(1)李四独立地生成自己的密钥对(包括公钥和私钥),并且将公钥完全公开。

(2)当张三需要与李四进行秘密通信时，张三查找到李四的公钥，然后加密消息(实际 上一般用对称密钥加密消息，再用公钥加密对称密钥，因为公钥直接加密消息比较慢，这里为 了讲述方便、突出重点，假设公钥直接加密消息),将密文发送给李四。

(3)李四使用相应的私钥解密消息，得到明文。虽然张三可以通过公开的信道获取李四 的公钥，但是张三如何确定所得到的公钥就是属于李四的呢?如果攻击者王五生成一对公私密 钥对，谎称是李四的公钥，蒙在鼓里的张三就用假的李四的公钥去加密自己的消息，那么王五 就可以解密密文消息了，从而窃听本来张三发给李四的秘密信息，李四反而不能解密这些信息。 由此看出，如何保证张三能够正确地获取李四的公钥是非常重要的。

(4)当利用数字签名来判断数据发送者身份的时候，也需要确定公钥的归属。数字签名 就是对消息的摘要进行私钥加密，然后接收方用发送方的公钥进行解密，如果解密成功，就可 以确认发送方的身份(因为私钥只能是发送方所有)。但是，如果攻击者王五生成一对公私密 钥对，然后将公钥公开，并谎称是张三的公钥，王五就能以张三的名义对一份假消息进行加密， 然后接收方李四用“张三的公钥”(其实是王五的公钥)进行解密，一看解密成功，李四就认 为这份消息的确是张三发来的，以为发送方就是张三了。而实际上，张三的身份已经被攻击者 王五冒用了。



**Windows C/C++加密解密实战**

从上面的分析可以看出，要应用公钥密码算法，首先需要解决公钥归属问题，需要正确地 回答：公钥到底属于哪个人?或者说，正确回答：每一个用户的公钥是什么?值得强调的是， 我们所说的公钥归属或者说公钥属于谁，实际上是指谁拥有与该公钥配对的私钥，而不是简单 的公钥持有。

在 Diffie 和 Hellman 首次公开提出公钥密码算法的时候，也设想了相应的解决方案：每 个人的公钥都存储在专门的可信资料库上。当张三需要获取李四的公钥时，就向该可信资料库 查询。

Diffie和 Hellman所设想的可信资料库方式要求所有用户都能与其在线通信，每次使用公 钥都要向资料库查询。这种方式不方便离线用户使用，而且当用户大规模应用时，频繁地并发 查询也会对资料库带来很高的性能要求。查询过程中也可能存在一定的安全问题，如中间人攻 击等。为了更安全地提供公钥的拥有证明并减少在线的集中查询，Kohnfelder 在1978年提出 了数字证书的概念。由证书认证中心签发证书来解决公钥属于谁的问题。

在证书中包含持有者的公钥数据与其身份信息，并且由CA 对这些信息进行审查并进行数 字签名。数字签名保证了证书的不可篡改。这样就使得每个人可以有更多的途径来获得其他用 户的证书，通过验证证书上的数字签名就可以离线地判断公钥拥有的正确性。由于证书上带有 CA 的数字签名，用户可以在不可靠的介质上缓存证书而不必担心被篡改，可以离线验证和使 用，不必每一次使用都向资料库查询。

有了CA 的支持，张三和李四的通信可以按照下列步骤进行：

(1)李四生成自己的公私密钥对，将公钥和自己的身份资料信息提交给CA。

(2)CA 检查李四的身份证明后，确认无误后为李四签发数字证书，证书中包含李四的 身份信息和公钥，以及CA 对证书的签名结果。

(3)当张三需要与李四进行保密通信时，就可以查找李四的证书，然后使用CA 的公钥 来验证证书上的数字签名是否有效，确保证书不是攻击者伪造的。

(4)验证证书之后，张三就可以使用证书上所包含的公钥与李四进行保密通信和身份鉴 别等。

需要注意的是，张三可以从不可信的途径(如没有安全保护的WWW或 FP 服务器、匿名 的电子邮件等)获取证书，由CA 的数字签名来防止证书伪造或篡改。相比于 Diffie 和 Hellman 最初设想的在线安全资料库，张三并不需要与CA 在线通信，也不必考虑获取途径的安全问题， 如通信信道的安全问题。在上述过程中，主要包括三种执行不同功能的实体。

(1)证书认证中心

CA 具有自己的公私密钥对，负责为其他人签发证书，用自己的密钥来证实用户李四的公 钥信息。

(2)证书持有者 (CertificateHolder)

在上述通信过程中，李四拥有自己的证书和与证书中公钥匹配的私钥，被称为证书持有者。 证书持有者的身份信息和对应的公钥会出现在证书中。



**第** **1** **1** **章** **实** **战PKI**

(3)依赖方 (Relying Party)

在上述通信过程中，张三可以没有自己的公私密钥对和证书，与李四的安全通信依赖于 CA给李四签发的证书以及CA的公钥。我们一般将CA应用过程中使用其他人的证书来实现 安全功能(机密性、身份鉴别等)的通信实体称为依赖方，或者证书依赖方，如张三。

CA、证书持有者和依赖方共同组成了一个基本的安全系统，这个系统被称为 PKI 系统， 即公开密钥基础设施。PKI 系统中的基本功能组件(简称为基本组件)有三个：分别为 CA、 证书持有者和依赖方。

需要注意，证书持有者与依赖方的区分并不是绝对的，它们的区分只是相对的。只有对特 定的通信过程区分证书持有者与依赖方才有意义。同一个实体在不同的通信过程中，可能既是 证书持有者，又是依赖方。例如，当张三使用李四的证书进行数据加密时，我们将张三称为依 赖方，将李四称为证书持有者。如果张三也有自己的证书，李四利用张三的证书给张三发送机 密信息时，则张三是证书持有者，李四是依赖方。另一个更显着的例子是 SSL/TLS 的双向鉴 别握手过程。在该握手过程中，服务端和客户端都分别持有自己的证书，相互进行身份认证， 每一方都既是证书持有者，又是依赖方。虽然张三和李四都会拥有自己的公私密钥对，但是它 们只是利用证书来获取 PKI 的安全服务，并不为其他人提供证书签发服务。我们通常将使用 证书服务的实体统称为末端实体 (End Entity)。

**11.2 OpenSSL** **实现CA** **的搭建**

上面讲了一堆理论，相信大家已有困意。下面我们来实际操作和演示一遍，利用OpenSSL 实现CA的搭建。OpenSSL是一套开源软件，在Linux中可以很容易地安装。它能够很容易地 完成密钥生成以及证书管理。我们接下来就利用OpenSSL搭建CA 证书，并实现证书的申请 与分发。搭建过程中需要准备三台Linux 虚拟机，或者也可以准备三台Linux主机。这里我们 采用三台 Linux 虚拟机，这样投资最少，这三台虚拟机安装了CentOS 7,OpenSSL也是用其 自带的版本1.0.1e (当然其他版本用起来类似):



[root@localhost 桌面]#openssl version OpenSSL 1.0.1e-fips 11 Feb 2013

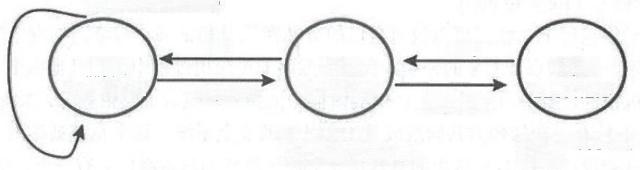
**11.2.1** **准备实验环境**

首先我们应该准备三台虚拟机，它们分别用来表示根CA 证书机构、子CA 证书机构和证 书申请用户。那么问题来了，用户向子CA 证书机构申请证书，子CA 机构向根CA 机构申请 授权，根CA 是如何取得证书的呢?答案是根CA自己给自己颁发证书。实验环境的拓扑结构 如图11-1所示。



Windows C/C++加密解密实战

根CA认证机构

自己给自己颁发证书

主机A

<120.4.2.6>

申请

颁发证书

主机B <120.4.2.7>

申请

颁发证书

主机C <120.4.2.8>

子CA 认证机构

向根CA 机构申请证书

普通用户向子CA申 请证书

图11-1

Linux虚拟机采用CentOS 7操作系统，虚拟机软件是VMware Workstation 12,通常可以 装完一台虚拟机，其他复制即可，网络连接模式都设置为桥接模式，如图11-2所示。



网络连接

桥接模式(B):直接连接物理网络 区复制物理网络连接状态(P)

图11-2

但要注意的是，复制后，可能会导致虚拟机Linux的网卡Mac地址相同，从而Ping不通 对方，此时可以在“虚拟机设置”中把现有网卡删除，再重新添加一块新的网卡。反正要做到 三台虚拟机要能相互Ping 通，因为下面要在线传送文件。另外，CentOS7 的防火墙默认是开 着的，这可能会影响我们Ping通，所以要把它关闭。首先查看防火墙的状态：

root@localhost ~]#firewall-cmd --state 如果是Running,就将其关闭：

root@localhost~]#systemctl stop firewalld

但这个关闭是临时性的，重启后又会打开。

**11.2.2** **熟** **悉CA** **环** **境**

我们的CA准备通过OpenSSL来实现，而CentOS7已经默认安装了OpenSSL,因此基本 的CA 基础环境也就有了。我们可以直接通过配置文件来熟悉这个CA 环境。

要手动创建 CA 证书，就必须首先了解OpenSSL 中关于CA 的配置，配置文件的位置在 /etc/pki/tls/openssl.cnf。我们通过命令cat 可以查看其内容，命令形式如下：

[root@localhost~]#cat /etc/pki/tls/openssl.cnf

然后就可以看到该配置文件的内容(因为内容较多，下面摘取部分，我们对其进行了解释):

#################################################################### [ca]

default\_ca=CA\_default #默认CA

#################################################################### [CA\_default ]

dir=/etc/pki/CA #CA的工作目录这里其实定义了一个变量，后面用美元符$可以引用该变量

certs=$dir/certs #证书存储路径



**第11章** **实战PKI**

|  |  |
| --- | --- |
| crl\_dir=Sdir/crl  database=Sdir/index.txt | #证书吊销列表  #证书数据库列表 |
| new\_certs\_dir=Sdir/newcerts certificate=Sdir/cacert.pem  serial=Sdir/seral  crlnumber=Sdir/crlnumber crl=Sdir/crl.pem  private\_key=Sdir/private/cakey.pem RANDFILE=Sdir/private/rand  x509\_extensions =usr\_cert | #新的证书路径  #CA自己的证书，.pem 是证书的二进制格式 #当前证书的编号，十六进制，默认为00  #当前要被吊销的证书编号，十六进制，默认为00 #当前CRL  #CA的私钥  #私有的随机数文件  #加入证书中的扩展部分 |

#Comment out the following two lines for the "traditional" #(and highly broken)format

name\_opt =ca\_default #命名方式

cert\_opt=ca\_default #CA的选项

#Extension copying option:use with caution #copy\_extensions=copy

#Extensions to add to a CRL.Note:Netscape communicator chokes on V2 CRLs #so this is commented out by default to leave a VI CRL.

#crlnumber must also be commented outto leave a V1 CRL.

#crl\_extensions=crl ext default\_days=365

default\_crl\_days=30 default\_md=default

preserve=no

#默认证书的有效期限

#CRL到下一个CRL 前的时间 zww #使用公钥默认MD

#保持传递的DN 排序

#指定请求的相似程度的几种不同方法 #对于类型CA, 列出的属性必须相同 policy=policy\_match #策略

#这里记录的是将来CA 在搭建的时候，以及客户端申请证书的时候，需要提交的信息的匹配程度

#For the CA policy [policy\_match]

countryName =match

stateOrProvinceName=match organizationName=match

organizationalUnitName =optional

commonName=supplied emailAddress=optional

#match意味着CA 以及子CA 必须一致

#国家

#州或者省 #组织公司

#为了“任何事”的政策，此时，必须列出所有可接受的“对象”类型

[policy\_anything] #可以对外提供证书申请，这时证书的匹配就可以不用那么严格

countryName=optional

stateOrProvinceName=optional localityName=optional



**Windows** **C/C++加密解密实战**

organizationName=optional

organizationalUnitName=optional commonName=supplied

emailAddress=optional

**11.2.3** **创建所需要的文件**

在CA 上有两个文件需要预先创建好，分别是/etc/pki/CAlindex.txt和/etc/pki/CA/serial。如 果不提前创建这两个文件，那么在生成证书的过程中会出现错误。

这里有一点需要注意，我们的实验环境中包含三个主机，其中两个的角色是作为CA 认证 机构存在的，即位于主机A 的根CA、位于主机B 的子CA, 所以创建所需要的文件的时候， 主机A 和主机B 都需要创建。

生成证书索引数据库文件：touch /etc/pki/CA/index.txt。

指定第一个颁发证书的序列号：echo 01>/etc/pki/CA/serial。

**11.2.4** **CA** **自签名证书(构造根CA)**

首先在主机A 上构造根CA 的证书。因为没有任何机构能够给根CA 颁发证书，所以只 能根CA 自己给自己颁发证书。首先要生成私钥文件，私钥文件是非常重要的文件，除了自己 以外，其他任何人都不能获取。所以在生成私钥文件的同时最好修改该文件的权限，并且采用 加密的形式进行生成。

我们可以通过执行OpenSSL中 的genrsa 命令生成私钥文件，并采用DES3 的方式对私钥 文件进行加密，同时临时指定umask,使得生成的私钥文件只对自己具有读写权限。过程如下：



[root@localhost ~]#(umask 066;openssl genrsa -out /etc/pki/CA/private/cakey.pem -des32048) Generating RSA private key,2048 bit long modulus

. . . + 十 十

..++十

|  |  |
| --- | --- |
| e is 65537(0x10001) |  |
| Enter pass phrase for /etc/pki/CA/private/cakey.pem | #这里需要输入口令 |
| Verifying-Enter pass phrase for /etc/pki/CA/private/cakey.pem | #这里确认口令 |
| [root@localhost~]# |  |



其 中 ，umask 用于设置所创建文件的权限掩码。openssl genrsa用于生成一个RSA 私钥， 后面指定了2048,因此生成的私钥是2048位的，私钥文件名是 cakey.pem, 是 .pem格式的， 并且已经被加密了，因为我们加了选项-des3。私钥通常用一个口令来保护，以防别人乱用， 这里我们输入的口令是“123456”,建议实际使用时使用更复杂的口令，以防猜测。

我们可以查看一下该私钥文件：

[root@localhost ~]#cat /etc/pki/CA/private/cakey.pem

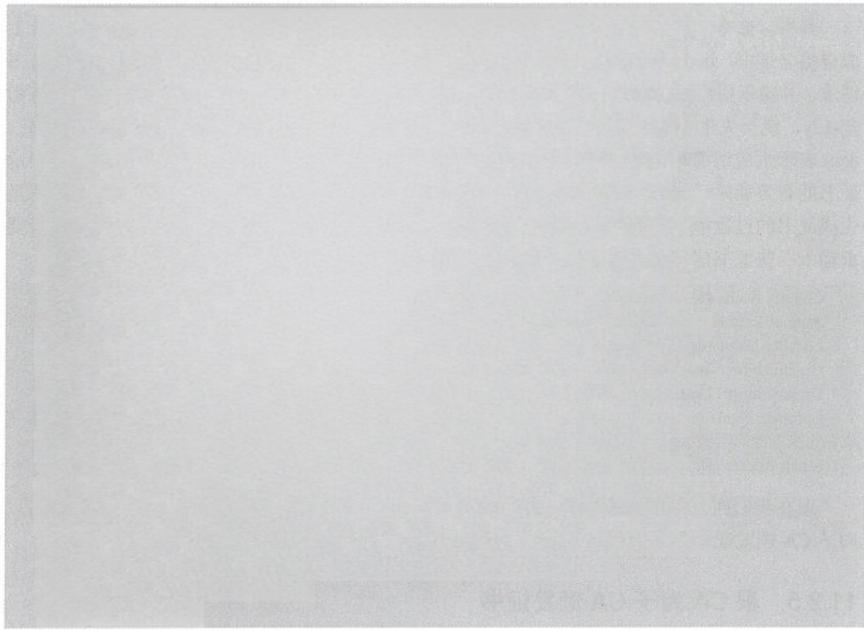
-----BEGIN RSA PRIVATE KEY----- Proc-Type:4,ENCRYPTED

DEK-Info:DES-EDE3-CBC,4F6722BEFIEA163C

5Ue9wR5izSC9N+UhU46FIFCvBV7EbNp2wYzbo2nqOVhyjySEIqfwPWzmfp3ztiMB



**第11章实战PKI**



LWgbRvfMQiTCpwqKw13k3C7yaWClfOkwsMaExPXuaIrdPuDHCXFG4VJhx97HUpv6 9JIRb2/OHAJVqw8zRdHhX38Da/6JqqZ1EnPAiXEqwnqjj9yCut6RNItEupKFmyE/

FKWOTGklbceaWZboq80mHznwsOQrzhGtzlGwsKc6bBnuSLoqd3w4jCYZI1CUStzW

6kKM9Qwcl6JwZtd61Yc72xsYkmxEWOGFVx9ZAW5t9XCpnuRjlKwS41hJdq1CZIbV zX++Yi/n4PNSOD/Go+7yvTgfJYNA2u+3wEVclleEHxut7ozd9vLDqYum5eR58x0P

RqAQ5nG3AdBN9LILR9Iw6z+ubLQozG+2xzDn7z/cVD+62gS+HO5H+Jiu7WX2o9LJ h/R8xzXB8EyySmw4loIXor6+xs9cilBnUkRfRZ5VBkUYw0b03xltyQYeqJQkfUzM

4hj7UVjXmy5qO2+tkPMRO//797uNFv8Ovi5pF2tkuh2xm4NnYcvrko5XcqUh3F7G

DvVQVjM+D2z9bloNbJsUSd+CchXgA2qf0qpeXjVRbYEO1CsOA5SopKMZ5qCdPLLQ uq6pfMbwNnyUg51/ekeIBCjrW7r/+EL3bpnWS+vGWfCGEEXIb2GN53k7hqg8TxGo

2jrH201vkwiWcopwxV8Bz3bq2ibMeg7xDctBQpSLO72MONPs+XKqG4sUXp9Fc2ft

yjY1Xaf8Qf7ypWa2pSd3j4ImtxTZEmAfV94dePyhZBLg3W4+e74Um7DUaNJ+Bsbv

ER6w29fPCZyFvF3aM0zlyrKNwOExEda53hMYyYU5z8qsQk5FNtQ3EoVjKBJ/JM/7 jd4QhQ9cFCXUqo/B+sdpn4yCQ50cYcUb416WZRAYFCLJxzQx4yUrVV5JTxVodLOq r/CN2z8Gjhs7kAClvs6V/UodrT2vV54/NmGPJypw+TZAHYPD9jVLD5JoIpz6FmVK

yaNxIbZiHYuLMOq7jZ8FP00PUiqQuCBUP/u7ns7XH75UbOco390HrOXxwsnBiXLK Hzi5gelyOREjW/65KTXJKOJx/Sy+me+tWIA2gWQ6qVEHV4/et78UVS00/2d//pUb

nSV68oHli87nCNN3xub3Q1kzknETVMN74sjgyhZiqUeIJ/3TTmZl+wG4MRSAP9Bp

jrYOvMfOAScn14BeitvwXuKWIM+TNxGFLIQzinijxX5339WArIVsvz92JEmSkid6

QPNBdtZ7GzlpgS8A57tcAZQkluBCPDo/6t2wKw+bG0n48RFgxRI1ZBS/r10dmm7z

//rgAfWISYLzWylh9njWMMGceBCNcwf9F4PcWv8Ov0G9dUIBORF401k8yCWo/dUt ZOdo830vqJp4Yhr3MAL9wO6VJEu+dO9heUIItLqzBH2SnungdmZemPTN25kAWXNi A9vliQGHpyOFB3CP+tL8rATSmSYThFh4WnJ8Do2evM6c9io+M0XzTOgp/DDISyLQ 392zZNAn/dvllqYdRirxU1hYq99bQRXKDzwdljmhH3E5xUG21MdyAg=

-----END RSA PRIVATE KEY-----

root@localhost~]#

私钥也是一个随机数，所以每次生成都不同。再次强调一下，私钥文件是非常重要的文件， 除了自己本身以外，其他任何人都不能获取。所以在生成私钥文件的同时最好修改该文件的权 限，并且采用加密的形式进行生成。

生成私钥后，我们就可以生成 一 张自签名证书了。

[root@localhost~]#openssl req -new-x509-key /etc/pki/CA/private/cakey.pem -days 7300-out /etc/pki/CA/cacert.pem

Enter pass phrase for /etc/pki/CA/private/cakey.pem:

You are about to be askedto enter information that will be incorporated into your certificate request.

What you are about to enter is what is called a Distinguished Name or a DN. There are quite a few fields but you can leave some blank

Forsome fields there willbe a default value If you enter!',the field will be left blank.

Country Name(2 letter code)[XX]:CN

State or Province Name(full name)[]:shandong

Locality Name(eg,city)[Default City]:qingdao

Organization Name(eg,company)[Default Company Ltd]:pojun.tech

Organizational Unit Name (eg,section)[]:opt

Common Name(eg,your name or your server's hostname)[]:ca.pojun.tech Email Address[]



**Windows C/C++加密解密实战**

其中，命令 openssl req的主要功能是生成证书请求文件、查看验证证书请求文件以及生 成自签名证书，这里调用该命令用来生成自签名证书cacert.pem;-new 表示生成一个新的证书 请求，并提示用户输入个人信息，比如后面让我们输入的CN 、Shandong 等信息，如果没有指 定-key, 就会先生成一个私钥文件，再生成证书请求； -x509 表示专用于CA 生成自签证书； -key 表示生成请求时用到的私钥文件，该选项只与生成证书请求选项-new 配合；-days n表 示 证书的有效期限； -out 指定生成的证书请求或者自签名证书名称。Enter pass phrase的意思是 生成证书的过程中需要输入之前设定的私钥的口令，这里是123456。在上面命令的末尾段要 求输入一些证书信息，解释如下：

Country Name(2 letter code)[XX]: //输入一个国家名字的缩写，可为空

State or Province Name(full name)[]: //州或省名称，全名，可为空

Locality Name(eg,city)[Default City]; //地区名称，如城市，可为空

Organization Name (eg,company)[Default Company Ltd]:1/组织名称，默认有限公司，可为空

Organizational Unit Name(eg,section)[]: //组织单元名称，可为空

Common Name (eg,your name or your server's hostname)[]:www.amber.com //常见的名字(例如你的名字 或你的服务器的主机名),输入该网址的域名，必填

Email Address []: //邮件地址，可为空

现在我们拥有一个CA 根证书了，证书路径为/etc/pki/CA/cacert.pem 。 有了根证书就可以 向 子CA 颁发证书，也就是签发一张证书给子CA。

**11.2.5 根CA为子CA颁发证书**

颁发证书将分成两个环节介绍，分别是子CA 证书机构向根CA 证书机构申请证书和普通 用户向子CA 证书机构申请证书。

申请并颁发证书的流程如下：

(1)在需要使用证书的主机(这里是子CA) 上生成证书请求。

(2)将证书的申请文件传递给根CA。

(3)根CA 签发证书。

(4)将根CA 生成的证书发送给子CA。

**1.子CA** **生成证书请求文件**

我们的环境中，B 充当子CA, 因此在主机B 上生成证书请求。首先在B 主机上生成私钥， 这个过程与前面根CA 机构生成私钥的过程是一致的。

这次，我们为子CA 生成一个1024位长度的私钥，并且没有采用加密的方式生成。在主 机B 的终端下输入如下命令：



[root@localhost 桌面]#(umask 066;openssl genrsa -out /etc/pki/CA/private/cakey.pem 1024) Generating RSA private key,1024 bit long modulus

… 十 十 士

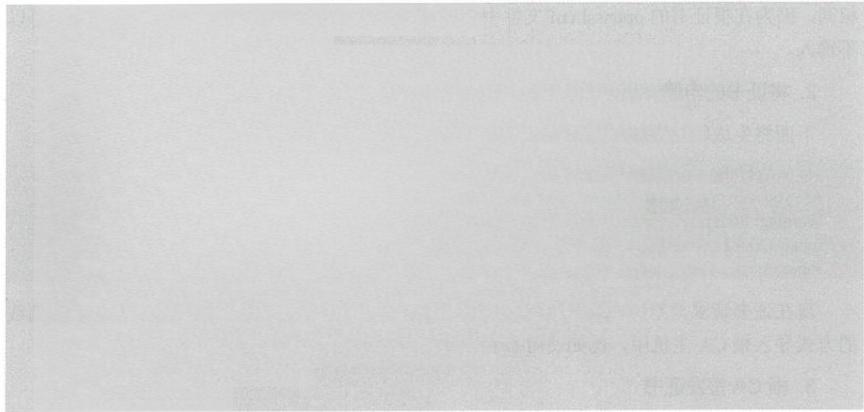
….....+t+tt

e is 65537(0x10001)

然后查看私钥文件 cakey.pem:



**第11章实战PKI**



[root@localhost 桌面]#cat/etc/pki/CA/private/cakey.pem -----BEGIN RSA PRIVATE KEY-----

MIICXAIBAAKBgQDSwWVSyLq4ZI/wZq75HPYfo6RtXOZAj+DNfjAyJmFe5ZN/EJe4 e913Gh7r/5sJkflizn1h2POaleoxTg7TUpdOG6e+xD8A00QZJuV3YEIdDwaT710d

CX3aJH+DTOUec3F/DgNw+hyncXpxOalafpYlicSoDzwTczdJ8HIHqLhljwIDAQAB

AoGBANIxU66Wx7KziOMIZiXJfqbbfFgeOP3XASuxWLwLjz0nlkz57XdvAdeRU5mn maaXyphEvMPjrkDg5kM6SIr2ajMK/0aXJFElmWiBcHqIN8t4cZucDOmrNmfPOZNV

Wbym0t8kq2KZDMZasQvDK5riaSnXeFQXtZQTAZRZmb+D+RnhAkEA64/99pVzI8Lg IrJ6A4yIB2S3XWwpgrnfyojSTEs3eFGewoaRIpSxSNf5qivliKrPplRy190maD5e

L1Z6904LdwJBAOUKbCe12b7qweRiYXORBZGYji4wKiNiWvoCFjBzWsdFG6JHVi+I

Zw3m5Z4agTs+aBgukQm+PgtxlmnsMWYCwakCQAEr+DFvOODOqVrCIISMAI4m3Bqk 3Rf/YLObNqCWhzIcBdQ14zbu0mrwWBeWnE+vudS1QNT+DqDaHpHRtk7dmEUCQDNK RjYOTxil0Y2nSIWLfkfAdfZ56rXJzL23wehPrMB7BVktyGsUjJ9cWYcyQEZYD096

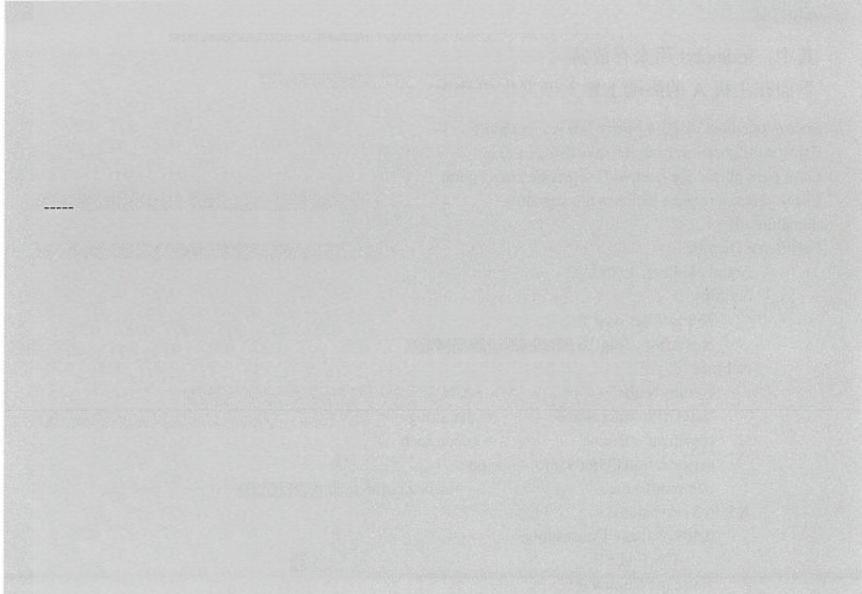
/hfEdnhxk1FdByLk8yECQGDtwaplTf7PcUOd6osgzi6iRN+2NCUwmZ4jlpWjOONg

yUSx870/9QyzYUErtOVYXNEoLJ+nOF/QnALeRUqNplI=

-----END RSA PRIVATE KEY----- [root@localhost 桌面]#

查看私钥文件可以发现，没有了加密的标识。同时，因为生成时指定了1024的长度，私 钥的长度明显变短了。

有了私钥就可以正式生成证书请求文件了。在主机B 终端下输入命令如下：



[root@localhost 桌面]#opensst req -new-key /etc/pki/CAprivate/cakey.pem -days 3650 -out /etc/pki/tls/subca.csr

You are about to be asked to enter information that will be incorporated into your certificate request.

What you are about to enter is what is called a Distinguished Name or a DN There are quite a few fields but you can leave some blank

For some fields there will be a default value, If you enter'!',the field will be left blank

Country Name(2 letter code)[XX]:CN

State or Province Name (full name)[]:shandong

Locality Name(eg,city)[Default City]:qingdao

Organization Name (eg,company)[Default Company Ltd]:pojun.tech Organizational Unit Name(eg,section)[]:opt

Common Name (eg,your name or your server's hostname)[]:subca.pojun.tech Email Address[]:

Please enter the following 'extra'attributes to be sent with your certificate request

A challenge password []:

Please enter the following 'extra'atributes to be sent with your certificaterequest

A challenge password []:123456

An optional company name []:magedu.com

其 中 ，subca.csr 就是生成的证书请求文件。其实这里的时间没有必要指定，因为证书的时 间是由颁发机构指定的，所以申请机构填写了时间也没用。其中有些信息必须与根证书的内容



**Windows C/C++加密解密实战**

相同，因为在根证书的 openssl.cnf文件中已经指定。另外，'extra'attributes 后面的信息也可以 不输入。

**2.将证书的申请文件传递给根CA**

下面将生成的证书的申请文件传递给根CA 机构，我们使用scp 命令进行网络复制。

scp /etc/pki/tls/subca.csr <120.4.2.6>:/ete/pki/CA

Are you sure you want tocontinue connecting(yes/no)?yes

Warning:Permanently added ' <120.4.2.6>'(ECDSA)to the list of known hosts. root@ <120.4.2.6>'s password:

subca.csr 100%729 0.7KB/s 00:00

现在证书请求文件subca.csr 在 主 机A 的/etc/pki/CA/ 下了。在实际操作中，也可以用离线 的方式导入根CA 主机中，比如使用U 盘载体等。

**3.根CA签发证书**

下面我们回到主机A ( 根CA) 上颁发证书。在第一次签发证书前，首先要在主机A 上 新

建两个文件：

touch /etc/pki/CA/index.txt

touch /etc/pki/CA/serial

echo "01">/etc/pki/CA/serial

其 中 ，index.txt 用来存放新签发证书的记录；serial 用来存放序列号，这里用了01。 下面在主机A 的终端上输入证书生成命令：

[root@localhost 桌面]#openssl ca -in /etc/pki/CA/subca.csr -out /etc/pki/CA/certs/subca.crt -days 3650

Using configurationfrom /etc/pki/tls/openssl.cnf

Enter passphrase for /etc/pki/CA/private/cakey.pem:

Check that the request matches the signature Signature ok

Certificate Details:

Serial Number:1(0x1) Validity

NotBefore:Aug 1905:10:432019 GMT Not After:Aug 1605:10:432029 GMT

Subject:

countryName

stateOrProvinceName

organizationName

organizationalUnitName commonName

X509v3 extensions

X509v3 Basic Constraints

CA:FALSE

Netscape Comment:

OpenSSL Generated Certificate X509v3 Subject Key Identifier:

A5:91:63:E6:85:BF:73:CB:CB:0B:B2:AE:CD:B5:B5:7D:6A:35:41:84

=CN

=shandong =pojun.tech

=subca.pojun.tech

=opt



**第11章实战** **PKI**

X509v3 Authority Key Identifier:

keyid:38:1D:62:19:59:D7:7B:31:12:CE:85:8E:43:E7:54:87:D6:D7:65:7C

Certificate is to be certified until Aug 1605:10:432029 GMT(3650 days) Sign the certificate?[y/n]:y

I out of I certificate requests certified,commit?[y/n]y

Write out database with I new entries Data Base Updated

在签发过程中，会用到根CA 的私钥，所以会询问根CA 私钥的口令。在后面还会有两次 询问，直接输入 y即可。生成成功后，查看 index.txt 文件，会看到增加了一条新的记录：

[root@localhost 桌面]#cat /etc/pki/CA/index.txt

V 290816051043Z 01 unknown /C=CN/ST=shandong/O=pojun.tech/OU=opt/CN=subca.pojun.tech

这说明我们的证书签发成功了。

**4** **.将根CA生成的证书传送给子** **CA**

传送方式依然可以采用离线或在线方式。这里采用在线方式。另外，主机B 是作为子CA 机构存在的，所以证书文件必须是cacert.pem(OpenSSL 命令需要.pem形式),否则子CA 将 不能够给其他用户颁发证书。

在主机A 的终端上输入命令如下：

[root@localhost桌面]#scp /etc/pki/CA/certs/subca.crt 120.4.2.7:/etc/pki/CA/cacert.pem

**11.2.6 普通用户向子CA 申请证书**

这个过程与子CA 向 根CA 申请证书的过程类似。基本步骤也是先生成用户私钥文件，再 生成证书请求文件，然后把证书请求文件发给子CA 让其签发出用户证书。

**1.生成用户私钥**

登录主机C 的终端，在命令行下输入私钥生成命令：

[root@localhost 桌面]#(umask 066;openssl genrsa-out /etc/pki/tls/private/app.key 1024) Generating RSA private key,1024 bit long modulus

.t 十 十tt 十

...+tt 士 e is 65537(0x10001)

现在/etc/pki/tls/private/路径下就有另一个用户私钥文件app.key 了。

**2.生成证书请求文件**

有了私钥文件，才可以生成证书请求文件。登录主机C 的终端，在命令行下输入生成证 书请求文件的命令：



[root@localhost 桌面]# openssl req -new-key /etc/pki/tls/private/app.key -out /etc/pki/tls/app.Csr You are about to be asked to enter information that will be incorporated



**Windows C/C++加密解密实战**

nto your certificate request.

What you are about to enter is what is called a Distinguished Name or a DN There are quite a few fields but you can leave some blank

For some fields there will be a default value If you enter !',the field will be left blank.

Country Name (2 letter code)[XX]:CN

State or Province Name(full name)[]:shangdong Locality Name (eg,city)[Default City]:qingdao

Organization Name(eg,company)[Default Company Ltd]:pojun.tech .Organizational Unit Name (eg,section)[]:dev

Common Name (eg,your name or your server's hostname)[]:user.pojun.tech Email Address[]:

Please enter the following 'extra'attributes

to be sent with your certificate request A challenge password []:123456

An optional company name []:

**3.将证书申请文件发送给子CA**

发送方式依然可以采用离线或在线方式。这里采用在线方式。

[root@localhost network-scripts]#scp /etc/pkitls/app.csr <120.4.2.7>:/etc/pki/CA The authenticity of host' <120.4.2.7>( <120.4.2.7>)'can't be established.

ECDSA key fingerprint is 5a:29:ed:4e:08:31:64:84:36:72:c7:28:46:46:58:34. Are you sure you want to continue connecting(yes/no)?yes

Warning:Permanently added' <120.4.2.7>'(ECDSA)to the list of known hosts root@ <120.4.2.7>'s password:

app.Csr 100%692 0.7KB/s 00:01

**4** **.子CA** **签发用户证书**

子 CA 收到用户的证书申请文件后，如果觉得没问题，就可以为其签发证书。在第一次签 发证书前，首先要在主机B 上新建两个文件：

touch /etc/pki/CA/index.txt

touch /etc/pki/CA/serial

echo "01">/etc/pki/CA/serial

其 中 ，index.txt 用来存放新签发证书的记录；serial 用来存放序列号，这里用了01。 下面可以继续在主机B 的终端上输入证书生成命令：

[root@localhost 桌面]#openssl ca -in /ete/pki/CAapp.csr -out /etc/pki/CA/certslapp.crt -days 365

Using configuration from /etc/pki/tls/openssl.cnf Check that the request matches the signature

Signature ok

The stateOrProvinceName field needed to be the same in the CA certificate (shandong)and the request (shangdong

用户证书app.crt 签发成功了。下面我们将生成的证书传递给申请者(这里是用户)。



**第11章** **实战** **PKI**

**5.将生成的证书传送给用户**

传送方式依然可以采用离线或在线方式。这里采用在线方式。这里是把主机 B 上的文件 app.crt 发送给主机C。



[root@localhost桌面]#scp /etc/pki/CA/certs/app.crt <120.4.2.8>:/etc/pki/CA/certs/ The authenticity of host' <120.4.2.8>( <120.4.2.8>)'can't be established

ECDSA key fingerprint is 5a:29:ed:4e:08:31:64:84:36:72:c7:28:46:46:58:34. Are you sure you want to continue connecting(yes/no)?yes

Warning:Permanently added ' <120.4.2.8>'(ECDSA)to the list of known hosts root@ <120.4.2.8>'s password

app.crt 100% 0 0.0KB/s 00:00

此时在主机C 上可以看到有证书文件了。

[root@localhost~]#ls /etc/pki/CA/certs/app.crt etc/pki/CA/certs/app.crt

以上就是利用OpenSSL 实现一个小型CA 的操作过程，虽然很小型，但基本原理和基本 流程和专业CA 是一样的。建议大家学习时从小型系统入手，再慢慢地深入。

现在CA 操作基本流程完成了，证书也出来了。下面我们可以围绕证书进行编程操作。

**11.3 基于OpenSSL** **的证书编程**

身份认证、证书很重要，其重要性就像我们日常生活中的身份证一样，没有身份证寸步难 行。同样在网络世界中，没有证书，没人会承认你是王者还是小兵。

在Windows 平台下，假设要解析一个X509 证书文件，直接的办法是使用微软的CryptoAPI, 但是在非 Windows 平台下，只能使用强大的开源跨平台库 OpenSSL。一个 X509 证书通过 OpenSSL 解码之后，得到一个X509 类型的结构体指针。通过该结构体，我们就能够获取想要 的证书项和属性等。

X509 证书文件依据封装的不同，主要有下面三种类型：

(1)\*.cer: 单个X509证书文件，不含私钥，可以是二进制和Base64格式。该类型的证 书很常见。

(2)\*.p7b:PKCS#7 格式的证书链文件，包括一个或多个X509 证书，不含私钥。通常 从 CA 中心申请RSA 证书时，返回的签名证书就是.p7b 格式的证书文件。

(3)\*.pfx:PKCS#12 格式的证书文件，能够包括一个或者多个X509 证书，含有私钥， 一般有 Password 保护。通常从CA 中心申请RSA 证书时，加密证书和RSA 加密私钥就是一 个 .pfx 格式的文件。

证书如此重要，OpenSSL 当然对其提供了强大支持。现有的数字证书大都采用X509 规范， 主要由这些信息组成：版本号、证书序列号、有效期(证书生效和失效的时间)、拥有者信息



**Windows C/C++加密解密实战**

(姓名、单位、组织、城市、国家等)、颁发者信息、其他扩展信息(证书的扩展用法、CA 自定义的扩展项等)、拥有者的公钥、CA 对以上信息的签名。

OpenSSL实现了对X.509 数字证书的所有操作，包括签发数字证书、解析和验证证书等。 在实际应用开发中，针对证书应用，这里主要用到证书的验证(验证其证书链、有效期、吊销 列表以及其他限制规则等)、证书的解析(获得证书的版本、公钥、拥有者信息、颁发者信息、 有效期等)等操作。这些函数均定义在OpenSSL/x509.h中。涉及证书操作的主要函数有验证 证书(验证证书链、有效期、CRL) 、 解析证书(获得证书的版本、序列号、颁发者信息、主 题信息、公钥、有效期等)函数，首先我们来认识这些函数。

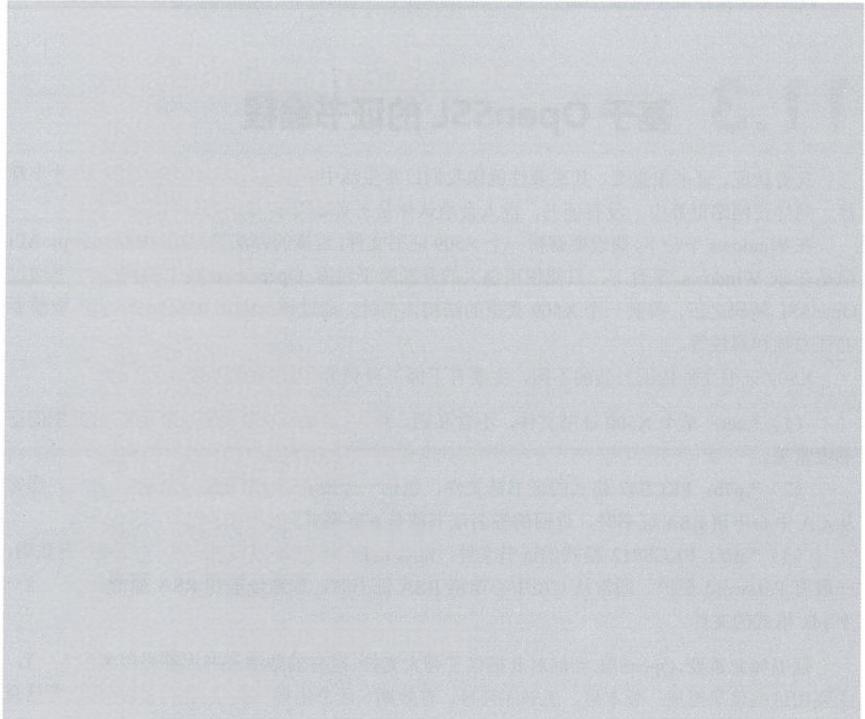
**11.3.1** **把** **DER** **编码转换为内部结构体函数d2i\_X509**

该函数将一个 DER 编码的证书转换为OpenSSL 内 部 结 构 体 (X509 类 型 ) , 该 函 数 声 明

如下：

X509\*d2i\_X509(X509\*\*cert,unsigned char\*\*d,int len);

其 中 ，cert[in] 是 X509 结构体的指针，表示要转码的证书，其中结构体X509 的定义如下：

struct x509\_st {

X509 CINF \*cert info X509 ALGOR \*sig alg

ASN1\_BIT\_STRING\*signature int valid;

int references char \*name

CRYPTO\_EX\_DATA ex\_data

/\*它们包含各种扩展值的副本\*/ long ex\_pathlen;

long ex\_pcpathlen;

unsigned long ex\_flags

unsigned long ex\_kusage; unsigned long ex\_xkusage;

unsigned long ex\_nscert

ASN1\_OCTET\_STRING\*skid; AUTHORITY\_KEYID \*akid;

//证书数据信息 //签名算法

//CA 对证书的签名值

X509\_POLICY\_CACHE \*policy\_cache; STACK\_OF(DIST\_POINT)\*crldp

STACK\_OF(GENERAL\_NAME)\*altname NAME\_CONSTRAINTS \*nc;

#ifndef OPENSSL NO\_RFC3779

STACK\_OF(IPAddressFamily)\*rfc3779\_addr; struct ASIdentifiers\_st \*rfc3779\_asid;

#endif

#ifndef OPENSSL\_NO\_SHA

unsigned charshal\_hash[SHA\_DIGEST\_LENGTH];

#endif

X509\_CERT\_AUX\*aux }/\*X509\*/;



第11章实战PKI

其中，X509\_CINF的定义如下：

|  |  |
| --- | --- |
| typedef struct x509\_cinf\_st{  ASN1\_INTEGER\*version;  ASN1 INTEGER \*serialNumber; X509\_ALGOR \*signature;  X509\_NAME\*issuer; X509\_VAL\*validity; X509\_NAME \*subject; | //证书版本，0表示 vl,1 表示 v2  //证书序列号 //签名算法  //颁发者信息  //有效期  //拥有者信息 |

X509\_PUBKEY\*key; //拥有者公钥

ASN1\_BIT\_STRING \*issuerUID;/\* 在v2 中是可选的\*/ ASN1\_BIT\_STRING\*subjectUID;/\* 在 v2中是可选的\*/ STACK\_OF(X509\_EXTENSION)\*extensions;

ASN1\_ENCODING enc; }X509\_CINF;

参数 d[in] 是 DER 编码的证书数据指针；len[in] 是证书数据长度。如果函数执行成功，就 返 回X509 结构体的证书数据。

**11.3.2** **获得证书版本函数** **X509\_get\_version**

该函数用于获取证书的版本。该函数是一个宏定义函数，定义如下： #define X509 get version(x)ASN¹ INTEGER get((x)->cert info->version)

其中，参数x[in] 指 向X509 结构体的指针。函数返回LONG 类型的证书版本号。

**11.3.3** **获得证书序列号函数** **X509\_get\_serialNumber**

该函数用于获得证书序列号，函数声明如下： ASN1 INTEGER\*X509 get serialNumber(X509\*x);

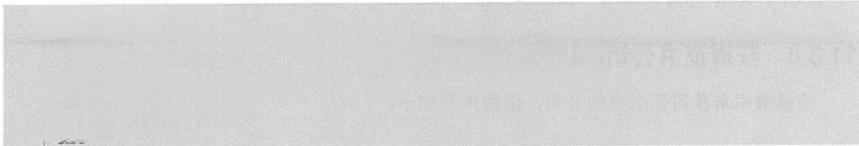
其 中 ，x[in] 是 X509 结构体的指针，表示要获取序列号的证书。函数返回ASN1\_INTEGER

\*类型的证书序列号。

**11.3.4 获得证书颁发者信息函数X509 get issuer\_name**

该函数用于获得证书颁发者的信息，函数声明如下： X509 NAME \*X509 get issuer name(X509 \*a);

其 中 ，a[in] 是 X509\* 类型的指针，表示证书。函数返回证书颁发者信息，X509\_NAME 的 定义如下：



struct X509\_name\_st{

STACK\_OF(X509\_NAME\_ENTRY)\*entries;

int modified; /\*如果需要生成bytes, 就为ture \*/

#ifndef OPENSSL\_NO\_BUFFER BUF\_MEM\*bytes;

#else



**Windows C/C++加密解密实战**

char \*bytes #endif

/\*无符号长散列\*/

unsigned char \*canon\_enc int canon\_enclen

}/\*X509\_NAME\*/;

**X509\_NAME\_ENTRY 的结构体定义如下：**

typedef struct X509\_name\_entry\_st { ASN1\_OBJECT\*object

ASN1\_STRING\*value int set;

int size /\*temp variable\*/

}X509\_NAME\_ENTRY;

X509\_NAME 结构体包括多个X509\_NAME\_ENTRY 结构体。X509\_NAME\_ENTRY 保 存 了颁发者的信息，这些信息包括对象和值 (Object 7和 Value) 。 对象的类型包括国家、通用 名、单位、组织、地区、邮件等。

**11.3.5** **获得证书拥有者信息函数X509\_get\_subject\_name**

该函数用于获得证书拥用者信息，函数声明如下： X509\_NAME\*X509\_get\_subject\_name(X509\*a);

其中，a[in] 是X509\* 类型的指针，表示证书。函数返回证书拥有者信息。

**11.3.6** **获得证书有效期的起始日期函数** **X509**  **get notBefore**

证书有效期从起始日期到结束日期，该函数用来获取证书有效期的起始日期。该函数是一 个宏定义函数，声明如下：

#define X509 get notBefore(x) ((x)->cert info->validity->notBefore)

其中，参数x[in]是 X509\* 类型的指针，表示证书。函数返回证书有效期的起始日期。

**11.3.7 获得证书有效期的终止日期函数X509\_get\_notAfter**

证书有效期从起始日期到结束日期，该函数用来获取证书有效期的结束日期。该函数是一 个宏定义函数，声明如下：

#define X509\_get\_notAfter(x) (x)->cert\_info->validity->notAfter)

其中，参数x[in]是X509\*类型的指针，表示证书。函数返回证书有效期的结束日期。

**11.3.8** **获得证书公钥函数X509\_get\_pubkey**

该函数用来获得证书中的公钥，函数声明如下：



第11章 实战 PKI

EVP\_PKEY\*X509\_get pubkey(X509\*x);

其中，参数x[in]是X509\* 类型的指针，表示证书。函数返回证书公钥。

**11.3.9** **创建证书存储区上下文环境函数X509\_STORE\_CTX**

该函数用于创建证书存储区上下文环境，函数声明如下： X509 STORE CTX\*X509 STORE CTX new();

如果函数操作成功，就返回证书存储区上下文环境指针，否则返回NULL。

**11.3.10** **释放证书存储区上下文环境函数X509\_STORE\_CTX\_free**

该函数用于释放证书存储区上下文环境，函数声明如下： void X509\_STORE\_CTX\_free(X509\_STORE\_CTX\*ctx)

其中，参数ctx[in] 表示证书存储区上下文环境的指针。

**11.3.11** **初始化证书存储区上下文环境函数X509\_STORE\_CTX\_init**

该函数用于初始化证书存储区上下文环境，主要功能是设置根证书、待验证的证书、CA 证书链等，函数声明如下：



intX509\_STORE\_CTX\_init(X509\_STORE\_CTX\*ctx,X509\_STORE \*store,X509\*x509,STACK\_OF(X509) \*chain);

其中，参数 ctx[in] 表示证书存储区上下文环境的指针，store[in] 表示根证书存储区，chain[in] 表示证书链。如果函数执行成功就返回1,否则返回0。

**11.3.12** **验证证书函数X509\_verify\_cert**

该函数用于验证证书，检查证书链，依次验证上级颁发者对证书的签名，一直到根证书。 该函数会检查证书是否过期，以及其他策略。如果设置了 CRL, 还会检查该证书是否在吊销 列表内。此函数必须在调用了STORE\_CTX\_init后才能使用。函数声明如下：

int X509 verify cert(X509 STORE CTX\*ctx);

其中，参数 ctx[in] 表示证书存储区上下文环境的指针。如果函数执行成功就返回1,否则 返回0。

**11.3.13** **创建证书存储区函数** **X509\_STORE\_new**

该函数用于创建一个证书存储区，函数声明如下： X509\_STORE\*X509\_STORE\_new(void);

函数返回X509\_STORE 结构体类型的指针。其中，X509\_STORE\_CTX 定义如下：



**Windows C/C++加密解密实战**

typedef struct x509\_store\_ctx\_st X509\_STORE\_CTX; struct x509\_store\_st {

/\*The following is a cache of trusted certs\*/

int cache /\*if true,stash any hits \*/

STACK\_OF(X509\_OBJECT)\*objs;/\*Cache of all objects\* /\*These are external lookup methods\*/

STACK\_OF(X509\_LOOKUP)\*get\_cert\_methods

X509\_VERIFY\_PARAM\*param

/\*Callbacks for various operations\*/ /\*called to verify a certificate \*/

int (\*verify)(X509\_STORE\_CTX\*ctx); /\*error callback \*/

int (\*verify\_cb)(int ok,X509\_STORE\_CTX\*ctx); /\*get issuers cert from ctx\*/

int(\*get\_issuer)(X509\*\*issuer,X509\_STORE\_CTX\*ctx,X509\*x); /\*check issued\*/

int(\*check\_issued)(X509\_STORE\_CTX\*ctx,X509\*x,X509 \*issuer) /\*Check revocation status of chain\*/

int(\*check\_revocation)(X509\_STORE\_CTX\*ctx); /\*retrieve CRL\*/

int(\*get\_crl)(X509\_STORE\_CTX\*ctx,X509\_CRL\*\*crl,X509\*x); /\*Check CRL validity \*/

int(\*check\_crl)(X509\_STORE\_CTX\*ctx,X509\_CRL\*crl); /\*Check certificate against CRL\*/

int(\*cert\_crl)(X509\_STORE\_CTX\*ctx,X509\_CRL\*crl,X509\*x);

STACK\_OF(X509)\*(\*lookup\_certs)(X509\_STORE\_CTX\*ctx,X509\_NAME\*nm)

STACK\_OF(X509\_CRL)\*(\*lookup\_crls)(X509\_STORE\_CTX\*ctx,X509\_NAME\*nm) int(\*cleanup)(X509\_STORE\_CTX\*ctx);

CRYPTO\_EX\_DATA ex\_data;

int references

}/\*X509\_STORE\*/;

**11.3.14 释放证书存储区函数X509\_STORE\_free**

该函数用于释放证书存储区，函数声明如下： void X509\_STORE\_free(X509\_STORE\*v);

其中，参数v 表示一个要释放的证书存储区。

**11.3.15 向证书存储区添加证书函数X509\_STORE\_add\_cert**

该函数将信任的根证书存储到证书存储区，函数声明如下： int X509\_STORE\_add\_cert(X509\_STORE\*ctx,X509\*x);

其中，参数ctx [in]表示证书存储区，x[in] 是受信任的根证书。如果函数执行成功就返回1, 否则返回0。



第11章实战PKI

**11.3.16** **向证书存储区添加证书吊销列表函数** **X509\_STORE\_add\_crl**

该函数用于向证书存储区添加证书吊销列表，函数声明如下： intX509\_STORE\_add\_crl(X509\_STORE\*ctx,X509\_CRL\*x);

其中，参数ctx [in]表示证书存储区，x[in]表示证书吊销列表。如果函数执行成功就返回1, 否则返回0。

**11.3.17** **释放X509** **结构体函数** **X509\_free**

该函数用于释放X509 结构体，函数声明如下： void X509\_free(X509 \*a);

其 中 ，a[in]是X509 结构体的指针，表示要释放的证书。

**11.4 证书编程实战**

前面我们介绍了OpenSSL 库中一些常用的证书函数。现在我们就利用这些函数小试牛刀。 功能很简单，就是解析一个DER 编码的RSA 证书。

首先要准备好证书，前面我们搭建 CA 的时候产生了一个 subca.crt 证书，这个证书是PEM 编码的，现在我们将其转为DER 编码的证书。转换很简单，因为OpenSSL 提供了相应的转换 命令。先进入subca.crt所在的目录，然后在终端下输入如下命令：

openssl x509-in subca.crt -outform der -out subca.der

此时，在同目录下生成一个DER 编码的证书文件subca.der,下面我们可以编程对其进行 解析。为了方便大家使用，我们把 subca.der放到了下例的工程目录下。

【例11.1】解析 DER 编码的证书

( 1 ) 打 开VC 2017,新建一个对话框工程，工程名是test。

( 2 ) 在VC 2017中，切换到对话框设计界面，然后从控件工具箱中拖曳一个编辑框和按 钮到对话框上，并设置编辑框的Read only属性为True,这样编辑框就只读了，再设置 Mutiline 属性为True,这样可以支持多行文本，再设置 Auto VScroll和Vertical Scroll属性为True,这 样就会出现垂直滚动条。

最后设置按钮的标题为“选择证书”。该按钮的功能是选择一个证书文件并解析，解析后 的结果显示在只读的编辑框中。

(3)为按钮添加事件处理函数，代码如下：



void CtestDlg::OnBnClickedSelCert()



**Windows C/C++加密解密实战**

//TODO: 在此添加控件通知处理程序代码 unsigned char buf[4096]="";

CFileDialog dlg(TRUE,//TRUE是创建打开文件对话框，FALSE 则创建的是保存文件对话框 ".der" //默认打开文件的类型

NULL //默认打开的文件名

OFN\_HIDEREADONLY|OFN\_OVERWRITEPROMPT,// 打开只读文件

"文本文件(\*.der)\*.der|所有文件(\*)\*\*|"); //所有可以打开的文件类型 if(dlg.DoModal()=IDOK)

{

CString strPath =dlg.GetPathName();

FILE\*fp=fopen(LPSTR)(LPCSTR)stPath,"rb"); if(!fp)

{

AfxMessageBox("文件打开失败"); return;

}

int nSize =fread(buf,1,4096,fp); AnsX509(buf,nSize)

m\_strCert =gstr;

UpdateData(FALSE); }

在代码中，首先调用文件选择对话框让用户选择一个DER 编码的证书文件。然后读取文 件数据，并存于缓冲区 buf 中，接着调用自定义的 AnsX509 函数进行解析，代码如下：

void AnsX509(unsigned char \*usrCertificate,unsigned long usrCertificateLen) {

X509\*x509Cert=NULL; 1/X509证书结构体

unsigned char \*pTmp=NULL;

X509\_NAME \*issuer=NULL; //X509\_NAME结构体，保存证书颁发者信息

X509\_NAME \*subject=NULL; //X509\_NAME 结构体，保存证书拥有者信息

int i;

int entriesNum;

X509\_NAME\_ENTRY \*name\_entry;

ASN1\_INTEGER\*Serial=NULL;// 保存证书序列号

long Nid;

|  |  |
| --- | --- |
| ASN1 TIME \*time  EVP\_PKEY\*pubKey long Version  unsigned char derpubkey[1024]; int derpubkeyLen  unsigned char msginfo[1024]; int msginfoLen  unsigned short \*pUtf8=NULL; int nUtf8;  int rv;  char szSign[256];  ULONG ulen=256 | 1/保存证书有效期时间 //保存证书公钥  //保存证书版本 |



**第11章实战** **PKI**

char szTmp[256]="";

//把DER 证书转化为X509 结构体 pTmp=usrCertificate;

x509Cert =d2i\_X509(NULL,(const unsigned char\*\*)&pTmp,usrCertificateLen); if(x509Cert==NULL)

{

AfxMessageBox("解析失败：非DER 证书"); return;

}

//获取证书版本

Version=X509\_get\_version(x509Cert);

myprintf("X509 Version:V%ld\rln",Version+1); //获取证书序列号

Serial=X509\_get\_serialNumber(x509Cert);

//打印证书序列号

myprintf(" 序列号：");

for(i=0;i<Serial->length;i++) {

myprintf("%02x",Serial->data[i]);



myprintf("r\n")

if(-1=get\_SignatureAlgOid(x509Cert,szSign,&ulen) return;

myprintf("签名算法：%s\r\n",szSign);

1/获取证书颁发者信息，X509\_NAME 结构体保存了多项信息，包括国家、组织、部门、通用名、Mail

等

issuer=X509\_get\_issuer\_name(x509Cert); //获取X509\_NAME 条目个数

entriesNum=sk\_X509\_NAME\_ENTRY\_num(issuer->entries);

//循环读取各条目信息

for(i=0;i<entriesNum;i++)



//获取第i 个条目值

name\_entry=sk\_X509\_NAME\_ENTRY\_value(issuer->entries,i); //获取对象ID

Nid=OBJ\_obj2nid(name\_entry->object) //判断条目编码的类型

if(name\_entry->value->type ==V\_ASN1\_UTF8STRING)//把UTF8 编码数据转化成可见字符

{

nUtf8=2\*name\_entry->value->length

pUtf8=(unsigned short\*)malloc(nUtf8); memset(pUtf8,0,nUtf8);

rv=MultiByteToWideChar( CP\_UTF8,

0,

(char\*)name\_entry->value->data



**Windows C/C++加密解密实战**

name\_entry->value->length

(LPWSTR)pUtf8, nUtf8);

rv=WideCharToMultiByte(

CP\_ACP, 0,

(LPCWSTR)pUtf8 rv,

(char\*)msginfo, nUtf8,

NULL,

NULL)

free(pUtf8);

pUtf8=NULL;

msginfoLen =rv;

msginfo[msginfoLen]='\0';

}

else {

msginfoLen =name\_entry->value->length;

memcpy(msginfo,name\_entry->value->data,msginfoLen); msginfo[msginfoLen]="\0';

}

//根据NID 打印出信息 switch(Nid)

{

case NID\_countryName: //国家

myprintf(" 签发者国家：%strn",msginfo); break;

case NID\_stateOrProvinceName: */省*

myprintf("签发者省份：%strln",msginfo); break;

case NID\_localityName: //地区

myprintf("签发者 localityName: %slr\n",msginfo)

break;

case NID\_organizationName: //组织

myprintf("签发者 organizationName: %s\r\n",msginfo);

break;

case NID\_organizationalUnitName://单位

myprintf("签发者 organizationalUnitName: %s\r\n",msginfo);

break;

case NID\_commonName //通用名

myprintf("签发者 commonName:%sr\n",msginfo) break;

case NID\_pkcs9\_emailAddress: //Mail

myprintf("签发者emailAddress:%slrìn",msginfo) break;

}//end switch }

//获取证书主题信息

subject=X509\_get\_subject\_name(x509Cert);



**第11章实战** **PKI**

//获得证书主题信息条目个数

entriesNum=sk\_X509\_NAME\_ENTRY\_num(subject->entries); //循环读取条目信息

for(i=0;i<entriesNum;i++) {

/获取第i个条目值

name\_entry=sk\_X509\_NAME\_ENTRY\_value(subject->entries,i); Nid=OBJ\_obj2nid(name\_entry->object);

//判断条目编码的类型

if (name\_entry->value->type==V\_ASN1\_UTF8STRING)//把UTF8编码数据转化成可见字符 {

nUtf8=2\*name\_entry->value->length pUtf8=(unsigned short\*)malloc(nUtf8); memset(pUtf8,0,nUtf8);

rv=MultiByteToWideChar( CP\_UTF8,

0,

(char\*)name\_entry->value->data name\_entry->value->length

(LPWSTR)pUt8,

nUt8);

rv=WideCharToMultiByte(

CP ACP. 0,

(LPCWSTR)pUtf8, rv,

(char\*)msginfo,

nUtf8, NULL, NULL)

free(pUtf8);

pUt8=NULL;

msginfoLen=rv;

msginfo[msginfoLen]=10';

}

else {

msginfoLen=name\_entry->value->length;

memcpy(msginfo,name\_entry->value->data,msginfoLen) msginfo[msginfoLen]="0';

}

switch (Nid)

一

case NID\_countryName: //国家

myprintf("持有者 countryName: %strln",msginfo);

break;

case NID\_stateOrProvinceName: *//省*

myprintf("持有者 ProvinceName:%strn",msginfo) break;

**Windows** **C/C++加密解密实战**

case NID\_localityName //地区

myprintf("持有者 localityName: %s\r\n",msginfo)

break;

case NID\_organizationName: //组织

myprintf("持有者 organizationName: %slr\n",msginfo);

break

case NID\_organizationalUnitName://单位

myprintf("持有者 organizationalUnitName: %slr\n",msginfo)

break;

case NID\_commonName //通用名

myprintf(”持有者commonName:%slrìn",msginfo) break;

case NID\_pkcs9\_emailAddress: //Mail

myprintf("持有者 emailAddress: %s\r\n",msginfo);

break; }//end switch

}

//获取证书生效日期

time=X509\_get\_notBefore(x509Cert);

myprintf("Cert notBefore: %s\r\n",time->data)

//获取证书过期日期

time=X509\_get\_notAfter(x509Cert);

myprintf("Cert notAfter:%slrln",time->data)

//获取证书公钥

if(szSign[4]='8)// 判断是不是RSA 公钥

{

myprintf("RSA 公钥：\r\n");

pubKey=X509\_get\_pubkey(x509Cert); if(lpubKey)

goto end;

pTmp=derpubkey;

//把证书公钥专为DER 编码的数据

derpubkeyLen=i2d\_PublicKey(pubKey,&pTmp); for(i=0;i<derpubkeyLen;i++)

{

if(i>0&&i%16=0) myprintf("\r\n");

myprint("%02x",derpubkey[i]);

} }

end

myprintf("\r\n")

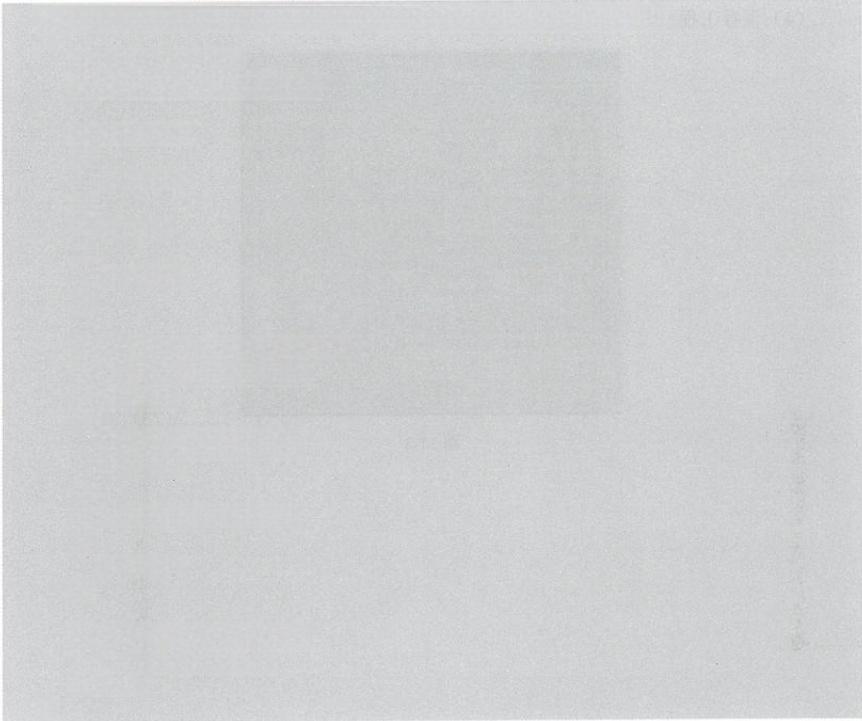
X509\_free(x509Cert) }

代码很简单，主要是调用OpenSSL 提供的库函数。我们对代码做了详尽的注释。值得注



**第11章实战PKI**

意的是，证书所包含的签名算法可能不同，因此我们需要判断证书中的签名算法是哪种算法， 因此定义函数get\_SignatureAlgOid,该函数定义如下：

ULONG get\_SignatureAlgOid(X509\*x509Cert,LPSTR IpscOid,ULONG \*pulLen)



char oid[128]={0};

ASN1\_OBJECT\*salg=NULL;

if(!x509Cert) {

return -1;



if(lpulLen) {

retun -1; }

salg=x509Cert->sig\_alg->algorithm; OBJ\_obj2txt(oid,128,salg,1);

if (IlpscOid) {

\*pulLen=strlen(oid)+1; returm -1;

}

if(\*pulLen<strlen(oid)+1) {

return -1; }

strncpy(lpscOid,oid,\*pulLen); \*pulLen=strlen(oid)+1;

return 0; }

结构体x509Cert内的成员字段sig\_alg->algorithm包含算法类型。得到的签名算法名称存 于输出参数lpscOid中。

另外，我们把解析后的信息都归总保存到一个全局的字符串gst 中，定义如下： CString gstr;

而向 gstr 保存信息是通过自定义函数myprintf 进行的，该函数定义如下：

void myprintf(const char \*format,..) {

va\_list vl

char Buffer[2\*MAX\_PATH]={0}; //根据实际情况定大小

LONG nRes;

va\_start(vl,format);

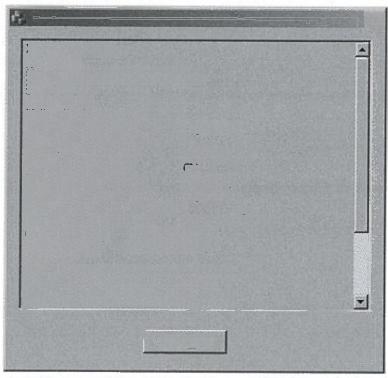
vsprintf(Buffer,format,vl) CString str;



**Windows** **C/C++加密解密实战**

str.Format("%s",Buffer); gstr +=str;

(4)保存工程并运行，运行结果如图11-3所示。

×

X509 Version:V3 序列号：01

签 名其法：1.2.840.113549.1.1.11 签发者国家： QN

签发者省份： shandong 签发者localtyName:qngdao

签发者 organiationName:pojun.tech

签发 者organtationaUntName: opt

签发者 commonName: a.pofun.tech

持有者countryName: CN

持有者ProvnceName: shandong 持有者 organzationName: pojun.tech

持有者organtationaUntName: opt 持有者commonName: subca.pofun.tech Cert notBefore: 190819051043Z

Cert notAfter: 290816051043Z

RSA公钥：

30818902818100d2c16552c8bab8648f f066aef91cf61fa3a46d5ce6408feOcd

7e303226615ee5937f1097b87bdd771a

选择证书

图11-