网络发布前言

《OpenSSL 与网络信息安全——基础、结构和指令》一书，本来规划是一系列的 OpenSSL 书籍中的一本（共规划四本），但既便《OpenSSL 与网络信息安全——基础、结构和指令》一书，历经两年多，依然不能完成手稿。究其原因，一则由于事情繁忙，实在难有闲暇，另则由于人的惰性。

如果说本书有一点点借鉴意义，那么其越早呈现给大家，对国内密码学的发展或者越能够起到一点点微薄之力。但是由于没有完稿，想通过正规的渠道交给大家，例如出版社，恐怕是难以实现，但幸好我们有了网络，这个畅通的渠道，我可以把这些零落的篇章，交给大家审阅。如果有后来者希望继续完成本书未完的篇幅，则是我最希望看到的。

本身完成之所以缓慢，是由于其中很多细节之处，都需要试验和编程来进行仔细的考究，因为技术文章，本是异常严谨的事情，不像写些无关痛痒的散文诗歌，可以即兴发挥。在写著本书过程中，虽然我努力做到仔细无误，但是错误肯定还是难以避免，希望同行们在阅读的过程中发现后能够及时在 www.OpenSSL.cn 网站上更正，假若能告知我，那就更不胜感谢！

王志海

2004.12.9

第1章 概述

王志海

1.1 信息安全

信息安全是本书要解决的主要问题，本节将介绍信息安全的基本概念和本书将要涉及的信息安全的内容。

1.1.1 信息安全概念

信息安全是自古以来就存在的概念，比如以前为了保证传递书信的保密性，使用腊封或其它方式将书信封装在信封内；还有使用暗号口令确认接受信息的人的身份等等方法。需要注意的是，信息安全技术是跟信息的载体形式和传送媒介密切相关的，信息载体的变化和信息传送媒介的变化必然会导致信息安全技术的变化发展。

在过去的二十多年中，信息技术取得令人惊异的发展，越来越多的有价值的信息和资料以数字信息存放在计算机等数字信息存储设备中。与此同时，信息共享技术也获得了巨大的突破，以 Internet 的发展为代表，短短的时间内，从美国军方的一个专用网络发展到联系着全世界千千万万人的膨大信息网络。这些客观的变化导致对信息安全的要求发生了重大的变化。

随着信息数字化以及计算机应用的发展，对存储在计算机中的文件和其它数字信息的保护需求成为了一种必然，尤其对一个能够通过公共网络进入的共享系统来说，这种需求显得尤为迫切。针对这种需求目前发展起来的技术有防病毒技术和防火墙技术等等。有些文献将这些保护数据、阻挡非法数据访问的技术统称为计算机安全或系统安全技术。

信息安全技术的另外一个重要变化是由于网络和通信设施的产生和应用引起的。这些网络和通信设施用来在用户各种终端以及计算机之间传输数据信息，这个传输过程很容易受到非法窃听等攻击，这就需要对在网络中传输的数据采取安全的保护措施。针对这种需求发展起来的技术有 VPN、SSL 等。有些文献将这种类型的技术统称为网络安全技术。

事实上，因为现在的绝大部分数据终端设备（包括计算机）基本上都是跟网络想联的，所以无论是计算机安全、系统安全还是网络安全，都不是完全相互独立的，而且，这些名词由于其笼统性，反而有概念不清和误导的可能。所以，我更愿意用具体一点的技术名词来说明不同的信息安全技术。

本书关注的是使用基于密码学原理来进行数据信息保护的技术，尤其偏向于利用该技术保护在网络中传输的数据。对于防火墙、防病毒以及入侵检测（IDS）等技术涉及的安全问题和解决方案，本书不作介绍。在本书的后续的章节，如果没有特别指明，信息安全的范围仅仅狭隘的包括使用基于密码学原理来进行数据信息保护的安全技术。

1.1.2 信息安全内容

正如 Bruce Schneier 所说，安全问题就如一条链子一样，必须保证每一个环节的安全才能达到使整个链子具有安全性。所以，在解决任何一个实际的或抽象 的系统的安全之前，

都应该首先分析其存在的可能的安全缺陷，进而采取相应的安全措施。为了使读者了解本书涉及的领域和需要解决的问题，下面介绍一下与本书内容相关的安全问题，并举一些相关的安全漏洞的例子，加深读者的理解。

机密性

机密性用于保护信息免受主动的非法窃取、阅读等攻击。在信息数字化之前，信息的机密性是依靠严格的管理制度和强大的物理手段来实现的，如戒备森严的房子和难以破坏的保险箱。对于独立的设备中（没有联网的计算机）的数字化信息，当然也可以依赖传统的保密手段，但是对于一般的共享系统或联网的系统来说，传统的方法就显得难以适应，必须采用新的针对数字信息的手段。

机密性涉及的范围是多方面的，主要包括内容的机密性和信息量的机密性。

内容的机密性是很容易理解的，就是确保数字信息的内容不被没有授权的人访问。内容的机密性既可以针对计算机中的一个重要文件，也可以是网络中传输的一些数据。对于计算机中的一个文件内容的保护情况显得可能简单一点，最简单的处理方法是我们只需要采用足够强大的加密算法将文件的内容加密即可。对于保护网络传输的信息，需要考虑的情况就会多一点，其中之一就是我们可能需要考虑对数据做不同层次的保护。比如，对一般的计算机之间的通信，我们可能只是选择其中重要的数据信息进行保护；而对于机密性要求非常高的办公室之间的两台计算机，我们可能会对它们之间传输的所有数据都进行保护。

信息量的机密性源于网络传输中通信量分析技术的产生，但我认为不仅仅限于网络通信量的分析，在本地的计算机系统中，一样存在类似的安全危险，我将它们统称为信息量的机密性。采用通信量分析进行攻击要求攻击者能够在通信设施上监听到通信的源和目的地址、通信频度、通信的数据长度、通信的时长等特征。对于本地计算机系统中的普通加密保护的文件，一样可以通过获取文件的长度信息、修改时间来获得有用的信息，更进一步，对一些结构化的文件，可以通过分析其各个结构的长度信息等来获得更多有用的信息，比如对加密数据库中各个字段长度的分析就可能得到大量的有用信息。信息量的分析还可以针对计算机系统中运行的程序，比如对加密程序的攻击就已经成功破解了 RSA 似钥。信息量的机密性在以前远远没有得到重视，但在今后的时间里，随着其相关攻击手段和事件的增加，必然会得到更大的发展。

完整性

完整性用来确保信息没有被修改，也可以防止假冒的信息。对于一个计算机中存储的信息来说，完整性的概念就是确保信息在存储的过程中没有被非法进行修改或替换。

对于网络信息来说，情况就复杂多了，主要分成面向连接和无连接两种情况。对于面向连接的情况来说，完整性是针对信息流的服务，它需要确保接受到的信息和发送的信息是一样的，没有篡改、插入、重排序、重复或者延迟，同时也要确保通信结束后数据在网络上的销毁。所以，面向连接的完整性服务不仅仅可以确保消息没有被非法篡改，还可以一定程度上防止拒绝服务攻击（DOS）。对于无连接的网络信息传输来说，完整性的内容跟计算机系统中文件对象是一样的，保护信息没有被篡改或替换。

鉴别

鉴别用来确认访问者的身份或消息的来源，防止冒充他人的行为发生。对于计算机中存储的信息来说，鉴别的功能就是确保访问者的身份，如最简单的是使用口令和密码来确保访问者的身份，安全一点的解决方案是通过 USB Key、IC 卡或其它信息的令牌进行身份鉴别。

对于网络传输中的信息来说，鉴别所需要确认的对象就有多种，可能是消息传输操作的用户，也可能是特定的应用程序，也可能是特定的 IP 地址，有时候可能是这些特征的综合，我们统称这些为实体。也就是说，在网络传输信息的过程中，鉴别的功能首先是确保通信双方的两个实体都是可信的，都是它们所宣称的实体；其次，鉴别还要确保在信息传输的过程

中不被第三方假冒两个合格方的任何一方来达到未授权接受信息或发送信息。

抗抵赖

抗抵赖的功能是保证消息制造或发出者不能在事后否认他制作或发出的消息。对于计算机中存储的信息，抗抵赖的功能就是确保文件在被合法授权用户修改后该用户否认自己做过这样的修改或创建这个文件的行为。这在传统的书面文件中，可以使用手写签名来解决这个问题，相应的，对于数字信息，可以使用数字签名来达到相同的目的。

对于网络信息传输的过程而言，抗抵赖的功能要求接受方能够验证消息的发送方，同时要求发送方能够验证消息的接受方，并能够在发生争议后向第三方（比如法庭）证明消息的发送方或接受方。

攻击举例

为了进一步说明上述四种信息安全措施的必要性，下面举一些针对性的攻击的例子，加深读者的理解。

Susn 使用 Mail 通过 Internet 向 Tom 传送一个带有机密信息的文件，但是由于没有使用安全措施保护该传送的信息，与 Susn 在同一公司局域网内工作的怀有不良目的 Jim 通过使用 Sniffiner 监听了 Susn 发送 Mail 的整个过程，因为 Internet 的信息是明文传送的，所以 Jim 成功获取了这个带有机密信息的文件。这是一个针对机密性攻击的例子。

Susn 是公司的总裁，他想给表现不错的 Eric 加 2000 元工资，起草了一份电子文件，叫秘书 Anna 发给财务部。Anna 是 Jim 的女朋友，而 Jim 正好跟 Eric 在同一个部门，于是 Anna 就将电子文件的名字 Eric 改成了 Jim，然后发给财务部，后果可想而知。这是针对完整性攻击的例子。

计算机 Server 上存有公司重要的客户资料，只有公司的客户部部门经理 Tom 有权限访问这些重要资料。Jim 是该公司一个员工，他准备跳槽，为了获得更多的客户资料，他一直想访问 Server 上的客户资料。有一天 Jim 无意中获得了 Tom 的帐号和密码（偷看的），然后就连接上 Server，告诉 Server 是 Tom 要访问客户资料，并按 Server 的要求输入了密码，顺利取得了想要的客户资料。这是针对鉴别攻击的例子。

Jim 在一个电子商务网站购买了一个笔记本电脑，并通过网络告诉银行从自己的帐户给电子商务网站转帐。Jim 在取得笔记本电脑后，发现自己其实并不需要笔记本电脑，但是退货已经不可能，于是就打算抵赖，他跟银行说自己其实并没有购买笔记本，是别人冒充了他，银行要负责任。如果银行没有一套针对这种情况的措施和方法，情况就会很麻烦。这是针对抗抵赖攻击的一个例子。

上述只是举了一些攻击的简单例子，事实上，实际的情况可能复杂的多，涉及的攻击可能也是多方面的。本书针对这些攻击采取的措施都是基于密码学原理的基础上的，下面我们就来了解一下密码学的基本背景。

密码学是本系列丛书基础中的基础，本节将概要地介绍密码学的功能、内容和应用，使读者对密码学有一个初步的了解。

1.2 密码学

密码学是本系列丛书基础中的基础，本节将概要地介绍密码学的功能、内容和应用，使读者对密码学有一个初步的了解。

1.2.1 密码学功能

最原始的密码学的作用是进行信息保密，即解决前面叙述的机密性问题。但现代发展起来的密码学功能已经远远超出了这个范围，它可以用来解决包括机密性、完整性、鉴别以及抗抵赖相关的各种难题，囊括了本书前面介绍的各种安全问题。

机密性问题是密码学最早关心的问题，也是核心的问题。目前针对机密性问题，密码学提出了各种算法，主要分为对称加密算法和公开密钥算法。不管是什么算法，都是为了在各种复杂和苛刻的条件下实现信息保密的功能，对称加密算法主要适应于通信双方已经共享了秘密的密钥的情况，而公开密钥算法则适用与通信双方没有共享的密钥的情况。目前常用的对称加密算法有 DES、3DES、AES 等等，常用的公开密钥算法有 RSA、DH 算法等等。

完整性问题在现代密码学中也得到了较好的解决，主要是采用了散列函数和数字签名相结合的算法。散列函数是一种单向映射函数，是密码学中确保数据完整性的核心算法，目前常用的有 MD5、SHA 和 SHA1 算法等等。

鉴别问题和抗抵赖问题在密码学中的解决不仅仅依赖密码算法本身，还依赖一套严格执行的密码协议或网络协议。这些协议以密码算法为基础，达到了鉴别和抗抵赖的功能。虽然设计一个好的密码协议并非如想象的那么容易，目前密码协议还是种类繁多，我们熟悉的有 Kerberos、SKID 等，网络协议有 SSL、SET 等。这些协议基本上都具备了鉴别和抗抵赖的功能。

1.2.2 密码学内容

现代密码学目前可以基本分为两大部分，密码算法和密码协议。密码协议是建立在密码算法的基础上实现的，但其完成的功能比单一的密码算法所能完成的功能丰富的多。

密码算法根据其完成的功能可以分为对称加密算法、公开密钥算法、数字签名算法以及信息摘要（散列函数）算法。对称加密算法主要完成了明文数据到密文数据的转换功能，其加密密钥和解密密钥是相同的。公开密钥算法完成的功能跟对称加密算法是相同的，但是其加密密钥和解密密钥不相同。数字签名算法的功能是实现鉴别和抗抵赖的功能，其具体的实现有时候可以采用对称加密算法或公开密钥算法实现，也有专门设计用于数字签名的算法。信息摘要算法主要是实现将大量的信息不可逆映射成一段定长或较短的信息而基本保持其独特性，所谓独特性，也就是说不同的信息经过相同的信息摘要算法映射后得到的结果应该也是不同的。

密码协议是使用密码学的协议，它包含了某种密码算法，但通常不仅仅是为了加密，而是为了更加复杂的特定目的设计的。参与协议的各方可能是各种各样的人，可能是相互信任的朋友，也可能是敌人，他们的目的可能是为了共享秘密、确定相互身份或者共同签署合同。在密码协议中使用密码算法一般来说是为了防止和发现窃听、攻击和欺骗等。

1.2.3 密码学应用

随着网络尤其是 Internet 的发展，密码学的应用已经越来越广泛。目前主要的应用领域有电子商务、电子政务、私人邮件、Web 安全访问以及虚拟专用网（VPN）等等网络应用。

电子商务是密码学迄今为止应用最成功的领域之一，主要涉及到了网络交易中的保密性、身份鉴别、完整性以及抗抵赖等功能。例如用户在某电子商务网站使用一个信用卡进行

购物时，必须保证整个过程时保密的而且时安全的，对于电子商务网站和银行来说，也必须确保用户的身份是可信的，并且能够向第三方证明用户确实执行了相关的操作。目前电子商务成功的协议之一是 SET 协议。

电子政务以及办公自动化系统的使用也越来越广泛，与之密切相关的安全性也出现了需求。比如公文的签发，需要数字签名，用户权限的控制需要进行身份确认等等。

电子邮件是普通网络用户使用最为广泛的 Internet 工具，因为邮件涉及到个人隐私等机密信息，所以保护电子邮件的安全早就成为了一个热点话题。目前，安全邮件系统 PGP 取得很大的成功，PGP 不仅仅实现了邮件信息在网络传输中的机密性，而且能够建立用户之间的信任关系，确认用户的身份，并保证邮件信息的完整性。

Web 已经成了信息发布的一个重要途径，一些企业可能希望一些重要的资料文件不被未经授权的用户访问，另一方面，用户为了安全的可能也需要验证服务器的身份。SSL 协议是解决这种需要的一个成功协议之一。

虚拟专用网技术（VPN）是解决大型的地理位置分布分散的公司或机构的低成本联网方案，该技术使用密码算法和密码协议通过 Internet 网建立起分支机构之间的虚拟专用网络。使得各分支机构之间能够安全的进行信息通信。

1.3 公钥基础设施

公钥基础设施（PKI）是近年来受到非常多关注的一项技术，本节介绍了公钥基础设施的基本概念以及其组件，并介绍了数字证书的基本作用。

1.3.1 公钥基础设施的必要性

公钥基础设施（PKI）是一种基于公开密钥算法的安全基础标准，它提供了一个框架，在这个框架内建立起了可以创建鉴定和认证过程需要的身份和相关信任关系，建立了可以管理的公开密钥加密系统。

虽然公钥基础设施是建立在密码学的公开密钥算法的基础上，但其解决了仅仅依靠密码学算法所无法解决的问题。公开密钥算法的密钥对能够实现对特定密钥对的鉴别和验证，但是无法建立将特定密钥对跟具体的个人身份联系起来的可信任关系。让我们来看一个公钥算法进行数字签名的例子。

创建一个数字签名，首先需要为文件的内容使用散列函数生成一个散列值以确保信息不被修改。然后就使用签名者的私钥对得到的散列值或摘要进行加密。数字签名的验证过程要求重新生成文档的散列值，再使用签名者的公钥对用签名者私钥加密的散列值进行解密，得到最初的散列值，然后比较这两个散列值。如果两个散列值一致，那么就认为签名通过。但是关键的问题是怎么确认那对先用于加密散列值，然后又用于解密散列值的那对密钥是确实属于所声称的签名者。对于一个聪明的入侵者来说，因为公钥是公开的，他可以通过某种方法用自己的公钥替换那个用于标识某个特定签名者的公钥，公钥被发布后，他就可以用自己的私钥生成一个能够通过上述验证过程的数字签名。所以，虽然拥有了公开密钥算法，但是算法本身并不足以确定一个可信人的身份。

公钥基础设施正是为了建立这种信任关系而产生的，它的主要目的就是建立可信任的数字身份，将特定密钥对和特定的人或实体联系起来，建立这种联系的主要形式就是颁发可信任的数字证书（或者叫电子证书）。

1.3.2 数字证书

对于初学者来说，数字证书的功能可能是难以理解的，事实上，你完全可以将数字证书跟你常用的身份证作对比。不过数字证书是你（或一个实体）在数字世界标识自己身份的证书罢了。下表显示了数字证书跟居民身份证书项目的一些对应关系，虽然这些对应不一定非常贴切，但显示了数字证书的真实作用。

表 1.1 数字证书跟居民身份证书项目的一些对应关系

身份证项目	数字证书项目
颁发机构是公安局	颁发机构是认证中心
公安局印章	认证中心电子签名
唯一的居民身份证号码	唯一的证书序列号
持有人姓名	证书拥有实体名称

事实上，在使用上，数字证书跟现实的证书也有很多相类似的地方，比如在银行开户时，银行需要你出示身份证并验证身份证的真假和是否确实是你自己的身份证；在数字世界也一样，当你在电子商务网站购物时，出示你的数字证书，电子商务网站会验证你的证书是否可信并确定是否确实是你自己的证书。

身份证的颁发需要公安局等权威机构执行一些列的材料核对和认证工作，并需要居委会和派出所等机构参与身份证的认证和管理工作。数字证书的颁发也一样，首先就需要一个权威的大家都信任的机构，此外，还需要其它一些设施或机构参与管理数字证书，这些数字世界设施的集合就是我们所说的公钥基础设施（PKI）。

1.3.3 公钥基础设施的组件

公钥基础设施并不是一个单一的设施，它由一系列的组件组成以完成其特定的功能。在一个 PKI 的作用域中，并非下面介绍的所有设施都是必须的，有些设施如 RA 可能并不需要。

认证机构（CA）

CA 可以说是 PKI 系统中的核心机构，负责确认身份和创建数字证书，建立一个身份和一对密钥之间的联系。作为一个程序员或技术人员，通常可能将 CA 跟证书签发服务器（或说证书签发应用程序）等同起来，事实上远远不是如此。CA 是一个软硬件和服务的集合，包括了人、操作流程、操作规程和认证策略、支持软硬件以及环境。一个成功的 CA 必须制定一些规则，使申请者和证书用户确信该 CA 所确认的身份适用于自己的目的并且是可信任的。一个 CA 是否能够获得成功，可能更重要的是在于其管理因素而不是技术因素。

注册机构（RA）

RA 负责证书申请人的资料登记和初始的身份鉴别，还可能需要接受证书用户提出的证书撤销等其它服务。事实上，RA 是一个可选的组件，在很多时候，它所负责的功能并不需要独立出来，而是可以成为证书服务器的一部分。一般来说，RA 最主要的职责就是接受申请人的申请请求，确认申请人的身份，然后将确认了身份的申请请求递交给 CA。

证书服务器

证书服务器是负责根据注册过程中提供的信息生成证书的计算机或服务程序。证书服务器将用户的公钥和其它一些信息形成证书结构并用 CA 的私钥进行签名，从而生成正式的数字证书。事实上，证书服务器还可能要完成其它一些功能，比如证书的存放、发布以及吊销等操作。技术人员在谈到 CA 时，通常就是指证书服务器。

证书库

任何证书在使用之前，必须将证书以及其相应的公钥公布出去。证书库就是存储可以公开发布的证书的设施。通常以目录的形式组成 PKI 的证书库，如 X.500 目录或者 LDAP 目录。目前最为常见的是 LDAP 目录，它事实上是由一组对目录中定位信息的方法和协议描述组成。

证书验证

当证书用户收到一个证书的时候，需要对这个证书进行验证。证书验证的项目通常包括：

- 验证证书的签名者以确认是否信任该证书
- 检测证书有效期，确认证书是否有效
- 确认证书没有被签发它的 CA 撤销
- 检测证书预期用途跟 CA 在该证书中指定的策略是否相符合

证书的验证过程通常是对证书链的验证，这通常要执行多个上述项目的循环验证已得出最终验证结果。证书的验证可以由客户端的验证程序执行，也可以提供专门的验证服务，客户端可以通过使用这种服务来完成验证。

密钥恢复服务

密钥对是确保证书能够顺利签发和正常使用的基本前提，密钥对的产生形式是多样的，既可以是客户端的软件如 OpenSSL 的应用程序或 IE 浏览器的密钥存储器中产生，也可以是在 USB Key 或 Smart Card 这样的物理设备中产生，有时候也可能是在一个集中的密钥产生服务器中生成。无论那种情况，都需要密钥恢复服务，使得加密密钥可以存档，并且在它们丢失后能够恢复。设想如果 Susn 是公司的一个高层主管，她使用自己的密钥加密了众多的重要文件，如果有一天她在出差时不小心飞机失事，从而丢失了密钥，这时候如果不能恢复密钥，后果是可怕的。有时候，执法部门也会要求提供加密密钥。所以必须提供密钥恢复机制。

时间服务器

因为每个计算机的时间都是可以设置并且不尽相同的，但是证书的验证和签发名等都需要一个统一、可信和单调增加的时间。这就使得时间服务器的存在有了价值。时间服务器可以为 PKI 作用域中的各个应用程序和 PKI 组件提供数字式时间戳，从而确保 PKI 域能够可信正确地运行。

签名服务器

许多 PKI 的应用程序和服务需要执行数字签名的操作，比如对文件的签名和对交易信息的签名等等。可能很多应用程序本身不具备签名的功能，那么这时候可以使用签名服务器提供的服务，集中为各个应用程序生成签名。通常，签名服务器还提供验证签名的服务。

1.4 安全协议

安全协议是密码学在计算机网络应用中的具体形式，几乎绝大部分密码算法最后都要以安全协议的形式得到应用。本节介绍通用的网络模型以及建立于其上的各种安全协议，重点介绍了 SSL 协议。

1.4.1 网络模型和安全协议类型

虽然 ISO 网络模型是最为标准化的网络模型，但是，目前得到普遍应用的是 TCP/IP 网络，在本书我们讨论采用的是从 TCP/IP 网络模型抽象出来的 5 层网络模型，其结构和对应

的协议层如图 1-1 所示。

目前比较成功的安全协议，基本上是在上面三层即应用层、传输层和网络层实现。下面我们介绍几个具有代表性的协议。

应用层安全协议 PGP

PGP（Pretty Good Privacy）协议是由 Philip Zimmerann 设计的免费保密电子邮件程序。严格来说，PGP 程序不能说是一个网络协议，但它确实是主要用于在网络上安全传递电子邮件。该程序能够实现对电子邮件网上传输的安全性，具备了邮件加密、对邮件发送人的身份鉴别、对邮件的数字签名和完整性等强大的功能。最新版本的 PGP 采用了多种加密算法以供用户选择，比如对称加密算法采用 IDEA 和 3DES 等，公开密钥算法采用了 RSA 和 DH 等，信息摘要算法采用了 MD5 和 SHA1 等。

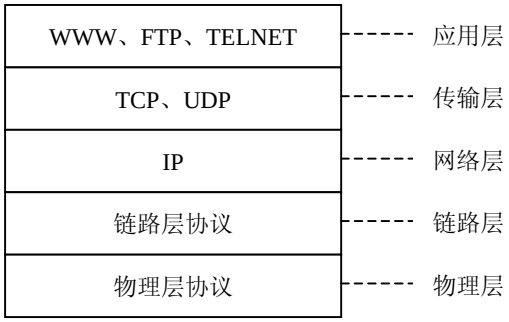

图 1-1 网络模型

在使用 PGP 的时候，用户之间首先要建立信任关系，这种信任关系的建立主要由用户自己确定。用户可以通过对相互公开密钥的签名建立一个 PGP 用户的互联组。比如 Tom 认识 Jim，Tom 将自己的公钥传给 Jim 之后，Jim 对 Tom 的公钥上签名并传回给 Tom，Tom 保存这份签名的公钥。当 Tom 想和 Eric 通信时，就将有 Tom 签名的自己的公钥拷贝给 Eric，Eric 可能以前已经有 Jim 的公钥拷贝并信任 Jim 签名的其它用户的公钥，所以当它用 Jim 验证该签名有效时，就接受 Tom，这样 Jim 就将 Tom 介绍给了 Eric。

传输层安全协议 SSL

SSL(Security Socket Layer)是由 Netscape 公司提出的一种安全协议，目前其版本已经发展到 SSL v3，标准化版本为 TLS(Transport Layer Security)。SSL 协议目前广泛地使用在 WWW 协议中，主流的 Web 服务器和浏览器都支持 SSL 协议，但是 SSL 协议绝不仅仅针对于 WWW 协议，它可以应用于传输层之上的所有服务，在下面一节我们会更加详细的介绍一下 SSL 协议。

网络层安全协议 VPN

虚拟专用网（VPN）技术确切的说不能说是一个协议，目前实现 VPN 的技术有多种多样，这里仅仅指使用 IPSec 隧道方式建立起来的 VPN。PGP 保护的是特定 Mail 应用的数据，SSL 协议如果有必要，可以建立两个特定主机之间的部分应用程序的安全通信信道，而 VPN 做的工作是保护任何两个子网或主机之间传输的数据的安全。VPN 能够加密和鉴别在网络层的所有通信量，VPN 提供了在公用网和 Internet 上建立安全通信信道的能力。

VPN 使用的典型例子是加入一个机构有多个地理上分散的局域网，这些局域网内部是运行着各种不安全的协议。为了管理上的需要，机构希望将所有这些局域网连接起来，显然，如果采用租用专业的方式花费将会非常昂贵，但是普通的 Internet 连接显然是不安全的。这时候 VPN 就是一个很好的解决方案，实施 VPN 方案的各个局域网物理上通过 Internet 相互连接在一起，但是所有从局域网进入 Internet 的信息包都要经过 VPN 设备的鉴别和加密，所有从 Internet 就如局域网的数据也都有 VPN 设备进行鉴别和解密。这样，对于 Internet 上局域网外的其它用户来说，这些局域网就是难以进入的。对于局域网的用户和应用程序来说，

VPN 的这些操作是透明的，跟另一个局域网的用户建立连接跟与本地局域网某个用户建立连接是一样方便的。

1.4.2 SSL 协议

SSL 协议是本书以及本系列丛书重点介绍的协议，所以有必要在进入本书的正题之前对该 SSL 协议做更多的一些了解。SSL 协议最初是由 Netscape 公司提出的，并且已经在其第三版的基础上形成了 Internet 标准化版本 TLS。SSL 的提出最初是为了 Web 的安全性，但是 SSL 协议不仅仅能够 Web 的安全性问题，这对一般的技术人员和程序员来说是一个好消息，因为 Web 的安全性很大程度上掌握在象 Microsoft 这些巨型公司中。

SSL 协议被设计成使用 TCP 协议提供端到端的安全服务，实际上，SSL 有一组协议组成，而不是单个协议。SSL 的协议栈如图 1-2 所示。

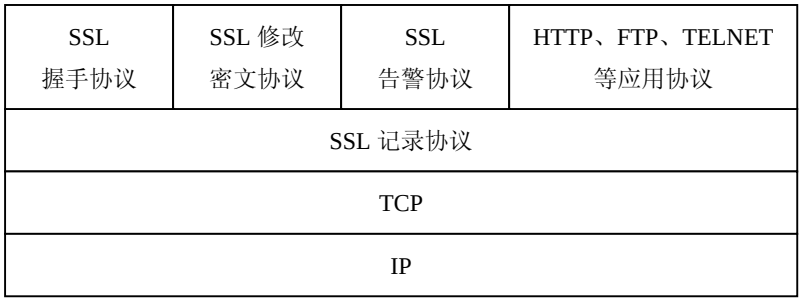

图 1-2 SSL 协议栈模型

SSL 记录协议为不同的高层协议提供安全服务，HTTP、FTP 等高层应用协议都可以在 SSL 协议上运行。SSL 握手协议、SSL 修改密文协议和 SSL 告警协议也是 SSL 协议的一部分，它们的作用是用来管理与 SSL 有关的交换。

SSL 协议中有两个重要的概念，即连接和会话。连接是指两台主机之间提供特定类型服务的传输，是点对点的关系。一般来说，连接是短暂的，每一个连接都与一个会话想关联。会话是客户和服务器之间的关联，会话是通过握手协议进行创建的。会话是加密安全参数的一个集合，包含了比如加密算法、临时加密密钥和初始向量等。会话可以被多个连接所共享，这样可以避免为每个连接重新进行安全参数的协商而花费昂贵的时间代价。任何一对服务器和客户之间可以存在多个安全 SSL 连接，这些连接可以共享一个会话，也可以共享不同的会话。理论上说，一对服务器和客户之间也可以存在多个会话，但是由于这样付出相当高的代价，所以一般来说不支持这种做法。

SSL 记录协议为 SSL 连接提供了机密性和报文完整性两种服务。机密性和报文完整性所需要的密钥都是在握手协议中协商提供的。记录协议接受到传输的应用报文后，将数据分片成可管理的块，可选地压缩数据，应用 MAC，加密和增加首部，然后使用 TCP 报文传输。记录层接受到底层发来的数据后，进行解密、验证、解压和重新排序组合，然后交给上层的应用协议。

SSL 修改密文协议是一个最简单的 SSL 相关协议，它只有一个报文，报文由值为 1 的单个字节组成。这个协议的唯一作用就是将挂起状态被复制到当前状态，改变连接将要使用的密文族。

SSL 告警协议是将 SSL 有关的告警信息传送给通信的对方实体。SSL 告警协议跟其它使用 SSL 的应用协议（如 HTTP 协议）一样，报文安装当前状态的被压缩和加密。

SSL 握手协议是 SSL 协议中最复杂的协议。服务器和客户端使用这个协议相互鉴别对

方的身份、协商加密算法和 MAC 算法以及在 SSL 记录协议中加密数据的加密密钥和初始向量。握手协议是建立 SSL 连接首先应该执行的协议，必须在传输任何数据之前完成。SSL 的握手协议由一系列报文组成，根据功能基本上可以分成四个阶段。

第一阶段是建立安全能力。所谓安全能力是指将要在通信中使用的加密算法、签名算法、密钥交换算法、MAC 算法以及其它一些记录协议需要使用的必要参数如初始向量等等。这个阶段由两个参数相同的报文组成，一个 Client_hello 报文，一个是 Server_hello 报文，协议的发起由客户端执行。这个阶段完成后，就完成了安全能力的建立。

第二个阶段是服务器鉴别和密钥交换。如果服务器需要被鉴别，这个阶段将以服务器给客户端发送自己的证书开始执行。这个阶段，服务器可能向客户端发送的信息包括证书或证书链报文、密钥交换报文、客户证书请求报文以及证书完成报文。除了最后一个报文，并非所有报文都是必须的，很多情况，可能只要发送其中的一个两个报文即可完成这个握手阶段。

第三个阶段是客户鉴别和密钥交换。受到服务器证书完成报文后，客户端首先验证服务器是否提供合法的证书，检测服务器的参数是否可以接受，如果这些都满足条件，客户端就向服务器发送客户证书报文（或者无证书告警信息）、客户密钥交换报文和证书验证报文中的一个或多个。除了客户密钥交换报文，其它两个报文在某些情况下不是必需的。

第四阶段是完成握手阶段。这个阶段完成安全连接的建立。首先是客户通过修改密文协议发送改变算法定义报文将挂起的算法族定义复制到当前的算法族定义。然后客户立刻接着发送在新的算法、密钥和密码下的完成报文。服务器对这两个报文的响应是发送自己的改变算法定义报文将挂起状态复制到当前状态，并发送完成报文。到此为止，握手协议完成，客户端和服务器建立了安全连接，应用层协议可以使用 SSL 连接进行安全的数据通信了。

1.5 OpenSSL

介绍 OpenSSL 是本书的重要目的，本节将对 OpenSSL 做一个总的概述，介绍 OpenSSL 的简史和 OpenSSL 的组成，并讨论其优缺点。

1.5.1 OpenSSL 简史

上面我们简单介绍了 SSL 协议，与所有的协议一样，都只是一些规则而已，真正要应用，必须将所有的协议规则转换成代码，对于任何个人和组织来说，这都是一个艰巨的任务，尤其是 SSL 协议这样一种涉及到诸多专业知识的协议。目前，主流的 Web 服务器和浏览器都支持 SSL 协议，但由于这些商业产品的源代码不是开放的，使得对这些商业产品的信任和研究都大大落后。

OpenSSL 是一个开放源代码的 SSL 协议的产品实现，它采用 C 语言作为开发语言，具备了跨系统的性能。OpenSSL 项目最早由加拿大人 Eric A. Yang 和 Tim J. Hudson 开发，现在由 OpenSSL 项目小组负责改进和开发，这个小组是由全球的一些技术精湛的志愿技术人员组成，它们的劳动都是无偿的，在此我们应该向他们表示崇高的敬意。

OpenSSL 最早的版本在 1995 年发布，1998 年后开始由 OpenSSL 项目组维护和开发。当前最新的版本是 0.9.7b 版本，完全实现了对 SSLv1、SSLv2、SSLv3 和 TLS 的支持。OpenSSL 的源代码库可以从 OpenSSL 的官方网站 www.openssl.org 自由下载，并可以免费用于任何商业或非商业的目的。由于采用 C 语言开发，OpenSSL 的源代码库具有良好的跨平台性能，支持 Linux、Unix、Windows、Mac 和 VMS 等多种平台。目前，OpenSSL 已经得到了广泛的应用，

许多类型的软件中的安全部分都使用了 OpenSSL 的库，如 VOIP 的 OpenH323 协议、Apache 服务器、Linux 安全模块等等。我们有理由预期，OpenSSL 和其所倡导的开放源码的思想必将被众多的支持者更加发扬光大。

1.5.2 OpenSSL 的组成

虽然 OpenSSL 使用 SSL 作为其名字的重要组成部分，但其实现的功能确远远超出了 SSL 协议本身。OpenSSL 事实上包括了三部分：SSL 协议、密码算法库和应用程序。

SSL 协议部分完全实现和封装了 SSL 协议的三个版本和 TLS 协议，SSL 协议库的实现是在密码算法库的基础上实现的。使用该库，你完全可以建立一个 SSL 服务器和 SSL 客户端。该部分在 Linux 下编译会形成一个明文 libssl.a 的库，在 Windows 下则是名为 ssleay32.lib 的库。

密码算法库是一个强大完整的密码算法库，它是 OpenSSL 的基础部分，也是很值得一般密码安全技术人员研究的部分，它实现了目前大部分主流的密码算法和标准。主要包括公开密钥算法、对称加密算法、散列函数算法、X509 数字证书标准、PKCS12、PKCS7 等标准。事实上，OpenSSL 的 SSL 协议部分和应用程序部分都是基于这个库开发的。目前，这个库除了可以使用本身的缺省算法外，在 0.9.6 版本之后，还提供了 Engine 机制，用于将如加密卡这样外部的加密算法实现集成到 OpenSSL 中。密码算法库在 Linux 编译后其库文件名称为 libcrypto.a，在 Windows 下编译后其库文件为 libeay32.lib。

应用程序部分是 OpenSSL 最生动的部分，也是 OpenSSL 使用入门部分。该部分基于上述的密码算法库和 SSL 协议库实现了很多实用和范例性的应用程序，覆盖了众多的密码学应用。主要包括了各种算法的加密程序和各种类型密钥的产生程序(如 RSA、Md5、Enc 等等)、证书签发和验证程序（如 Ca、X509、Crl 等）、SSL 连接测试程序（如 S_client 和 S_server 等）以及其它的标准应用程序（如 Pkcs12 和 Smime 等）。在某些时候，不需要做二次开发，仅仅使用这些应用程序便能得到我们的应用要求，比如采用 Ca 程序就能基本上实现一个小型的 CA 功能。这些应用程序同时也是很好的使用 OpenSSL 加密算法库和 SSL 协议库的优秀例子，比如 Ca、Req 和 X509 程序就是使用 OpenSSL 的库开发一个 CA 中心服务器的优秀例子，又如 S_client 和 S_server 程序就是利用 SSL 协议库建立 SSL 安全连接的优秀例子。对于初学者来说，研读这些应用程序的源码通常是最好的入门途径。

1.5.3 OpenSSL 优缺点

除了 OpenSSL 之外，技术开发人员还有其它的一些密码算法库或 SSL 函数库可选。Wei Dai 写的 Crypto++就是著名的开放源代码算法库之一，该库使用 C++语言作为开发语言，由于采用面向对象的 C++语言，对于初学者来说可能更容易理解和接受。但是该库仅仅实现了一些常用的密码算法，而没有实现诸如 X509 标准和 SSL 协议，其功能仅仅是 OpenSSL 的一个子集。此外，Microsoft 也提供了一个密码库 CryptoAPI，跟几乎大部分的 Microsoft 产品一样，CryptoAPI 不是开放源代码的，并且，它只能在 Windows 平台使用，对于开发者来说是一个黑匣子，很技术人员不喜欢这点，我也不喜欢。

与其它的一些同类型密码库相比，OpenSSL 具有以下优点：

- 采用 C 语言开发，支持多种操作系统，可移植性好

- 功能全面，支持大部分主流密码算法、相关标准协议和 SSL 协议
- 开放源代码，可信任，能够根据自己的需要进行修改，对技术人员有借鉴和研究的价值
- 具备应用程序，既能直接使用，也可以方便进行二次开发
- 免费使用，能够用于商业和非商业目的

当然，OpenSSL 也有如下一些缺点：

- 采用非面向对象的 C 语言开发，对于初学者来说有一定的困难，也不利于代码的剥离
- 文档不全面，增加了使用的困难性

总的来说，OpenSSL 是一个非常优秀的软件包，很值得密码安全技术人员研究和使用，这也是写本书的目的所在。

1.6 本书概要

本书是“OpenSSL 与网络信息安全”系列丛书的第一本，偏重于基础，是 OpenSSL 的入门书籍，其章节安排基本是安照基本理论、应用理论到具体应用的顺序进行安排。

第一部分是密码学基础，包括第 2 章到第 6 章。该部分主要介绍了密码学的基本概念、密码技术的基本实现、对称加密算法、公开密钥算法和单向散列函数算法等密码学知识。如果读者对密码学已经有相当的了解，可以跳过这部分。

第二部分是公钥基础设施基础，包括第 7 章到第 9 章。该部分主要介绍公钥基础设施的基本概念和内容、数字证书和 CA 以及 PKI 的应用等。同样，如果读者对公钥基础设施已经了解，那么这部分也可以省略不看。

第三部分是 SSL 协议基础，包括第 10 章到第 12 章。该部分主要详细介绍了 SSL 协议。对于 SSL 协议熟悉的读者，也可以跳过这一部分。

第四部分是 OpenSSL 结构，包括第 13 章到第 15 章。该部分介绍了 OpenSSL 的结构、编译和安装方法以及其使用的基本概念。

第五部分是 OpenSSL 指令，包括第 16 到第 22 章。该部分详细介绍了 OpenSSL 的应用程序指令的使用方法和各项参数的意义，是本书的主题之一。

1.7 推荐资料

由于 OpenSSL 是基于密码学和网络的基础上，所以需要比较多的基础知识，本书对这些基础知识的介绍并不完备，所以向读者推荐一些参考资料，以便读者做更深入的学习和理解。

1.7.1 推荐书籍

- 《计算机网络（第三版）》，清华大学出版社，Andrew S. Tanenbaum 著。该书是计算机网络经典和集大成的教材，系统的阐述了计算机网络的原理、结构以及相关的问题。
- 《应用密码学—协议、算法和 C 源程序》,机械工业出版社，Bruece Schneier 著。该书是密码学的经典著作，对今年来密码学的研究成果做了全景式的概括。该书力图从概念上解释密码学，抛开了繁琐的数学公式。
- 《密码编码学与网络安全:原理与实践(第二版)》,电子工业出版社,William Stallings 著。该书对密码学的基本理论做了系统的介绍，并详细介绍了密码学的应用。
- 《公钥基础设施(PKI)—实现和管理电子安全》，清华大学出版社，Andrew Nash 等著。该书全面介绍了公钥基础设施的概念，并对其实现、应用和发展前景做了介绍。

1.7.2 推荐网络资源

- http://www.openssl.org，OpenSSL 官方网站，可以自由下载 OpenSSL 的最新版本和

以前的各个版本，并有 FAQ 和其它 OpenSSL 相关网站的连接。

- http://www.openssl.cn，国内唯一的OpenSSL中文专业站点，也是本书的支持站点，除了丰富的中文资料和例程下载之外，还有众多OpenSSL技术人员参与交流。
- http://www.google.com，强大的搜索引擎，你可以输入关键字如OpenSSL和PKI等查找你需要的资料。

第 13 章 OpenSSL 概述

王志海

13.1 OpenSSL 背景

本节将对 OpenSSL 的背景作一个简要的介绍，并介绍一些其它类似的软件开发包，从而让读者在对比中得出 OpenSSL 的特点。

13.1.1 OpenSSL 简介

前面我们介绍过众多的密码算法、公钥基础设施标准以及 SSL 协议，或许这些有趣的功能会让你产生实现所有这些算法和标准的想法。果真如此，在对你表示敬佩的同时，我还是忍不住提醒你：这是一个令人望而生畏的过程。这个工作不再是简单的读懂几本密码学专著和协议文档那么简单，而是要理解所有这些算法、标准和协议文档的每一个细节，并用你可能很熟悉的 C 语言字符一个一个去实现这些定义和过程。我不知道你将需要多少时间来完成这项有趣而可怕的工作，但肯定不是一年两年的问题。我相信，上面的话肯定不会让你绝望，事实上，自从你拿到本书开始，就不会对这样的话绝望，因为至少你知道有 OpenSSL 这个工具可以使你逃避这个可怕的工作。

我们首先应该感谢 Eric A. Young 和 Tim J. Hudson，他们自 1995 年开始编写后来具有巨大影响的 OpenSSL 软件包，更令我们高兴的是，这是一个没有太多限制的开放源代码的软件包，这使得我们可以利用这个软件包做很多事情。Eric A. Young 和 Tim J. Hudson 是加拿大人，后来由于写 OpenSSL 功成名就之后就到大公司里赚大钱去了。1998 年，OpenSSL 项目组接管了 OpenSSL 的开发工作，并推出了 OpenSSL 的 0.9.1 版，到目前为止，OpenSSL 的算法已经非常完善，对 SSL2.0、SSL3.0 以及 TLS1.0 都支持。OpenSSL 目前最新的版本是 0.9.7b 版。

OpenSSL 采用 C 语言作为开发语言，这使得 OpenSSL 具有优秀的跨平台性能，这对于广大技术人员来说是一件非常美妙的事情，可以在不同的平台使用同样熟悉的东西。OpenSSL 支持 Linux、Windows、BSD、Mac、VMS 等平台，这使得 OpenSSL 具有广泛的适用性。不过，对于目前新成长起来的 C++程序员，可能对于 C 语言的代码不是很习惯，但习惯 C 语言总比使用 C++重新写一个跟 OpenSSL 相同功能的软件包轻松不少。

13.1.2 其它密码算法开发包

事实上，任何一个普通的密码学应用开发人员可能都不需要亲自写繁琐难懂的密码算法和实现那些标准制定者们炮制出来的可怕标准，因为这个计算机世界里除了 OpenSSL，还有很多可选择密码算法开发包。这些开发包有些是诸如 Microsoft 这种超级商家提供的，有些则是由象 Eirc A. Young 这样的无私贡献者提供的。这些开发包功能和有缺点不尽相同，下面我们对几个常见的做简单介绍。

我们首先应该想到的就是著名的 Microsoft 提供的密码算法库 CryptoAPI。CryptoAPI 当

前最新的版本是 CryptAPI2.0，包含了各种密码算法、密钥管理以及证书管理功能，功能相当完善和强大。事实上，Windows 平台基于密码学的安全结构基本上是以 CryptoAPI 为基础的，如支持 SSL、TLS 以及 PCT 协议的 SChannel 就是基于 CryptoAPI 之上的，而 IE 也是与 CryptoAPI 间接相关。Windows 的 CryptoAPI 事实上是定义了一个 Windows 平台通用的接口标准，它使用密码支持组件（CSP）提供底层真正的加密操作。CSP 机制使得 CryptoAPI 能够非常灵活，可以方便地使用第三方提供的加密库或硬件设备，事实上，为了跟在 Microsoft 的后面混口饭吃，很多智能卡（Smart Card）厂商都提供了 CSP 接口。CryptoAPI 虽然有这么多的方便之处，但 Microsoft 的产品缺点却是人所共知的。首先 CryptoAPI 是基于 Windows 平台的，不能象 OpenSSL 那样可以有良好的跨平台性能，如果你想在 Linux 下进行密码学应用的开发，这段话你算是白看了；其次 CryptoAPI 不提供源代码，对于你来说，CryptoAPI 是一个黑箱子，根本没有办法知道它里面执行了什么动作，对于一个充满好奇的程序员来说，实在是一件很郁闷的事情，而对于密码实现技术的研究人员来说，CrytoAPI 的意义也极其有限；最后，CrytoAPI 是美国公司的产品，对于象中国这样的国家来说，密码出口产品的算法受到很大限制，对于许多领域的安全来说，这是一个不能接受的事实。

David 的力作 Crypto++也是一个不错的开放源代码的密码算法包，它使用 C++写成，对于 C++程序员来说，这是一个好消息，甚至可能马上兴高采烈地丢掉本书。但是不要高兴的太早，因为 Crypto++仅仅提供了密码算法的实现，而对需要密钥管理封装和证书管理等相关功能并没有提供很好的支持，如果你需要这些功能，建议你最好还是将本书后面的章节看下去。

如果你是 Java 程序员，还可以选择 JSSE 的密码算法开发包，这是一个功能和 OpenSSL 几乎相当的开发包，区别的最大不同之处在于它是用 Java 写成的。关于安全产品使用 Java 与 C/C++开发那一个比较好的问题一直是一个争论的焦点，事实上作为一个技术人员，你可能最后根本不会受这些争论的影响，而是取决于自己对这两种语言的偏爱。

13.2 OpenSSL 结构

我第一次从 OpenSSL 的官方网站下载了 OpenSSL 开发包并解压到我的硬盘上的时候，面对 OpenSSL 目录里面的众多目录和文件，感觉是满头雾水，就象老虎抓着刺猬，不知道该如何下手。相信你或多或少有这种不愉快的感觉，本节将对 OpenSSL 的这些目录结构作一个大概的介绍，这对于开始 OpenSSL 的了解是必须的。

13.2.1 OpenSSL 总体结构

OpenSSL 整个软件包大概可以分成三个主要的功能部分：密码算法库、SSL 协议库以及应用程序。OpenSSL 的目录结构自然也是围绕这三个功能部分进行规划的。

首先，我们注意到 OpenSSL 的根目录下有不少文件，这些文件包含 OpenSSL 各个平台下编译安装的说明文档、编译安装的配置文件以及 OpenSSL 本身版本变化的一些说明文档。诸如 INSTALL.*这样名称的文件，都是安装编译说明文件，后缀名是平台的名称。比如 INSTALL.w32，就是 Windows 平台的 OpenSSL 安装编译说明文件。只有 Linux 的安装编译说明文件是不带后缀的，就是 INSTALL，由此可见 OpenSSL 肯定是出身 Linux 家族了。表 13-1 列出了系统平台和对应安装文件的关系。其它一些文件的作用根据其文件名就可以知道个大概，如果你刚刚接触 OpenSSL，这些文件还是值得一读的。

接下来该逐一看看那些目录了，这是 OpenSSL 真正的精华所在，也就是令我们心动的

源程序了。并非所有这些子目录都是很重要的，我们在后面对那些重要的子目录将作一个详细的介绍，这里先对一些不是很重要的目录作一个简述。MacOS、ms、os2 以及 VMS 这几个目录，包含了在不同的平台编译时候的环境变量配置文件，在安装编译完成之后，这几个目录就没有作用了。Bugs、certs、perl、shlib、times、tools 以及 utils 目录都是一些辅助的目录，里面包含的文件对于我们使用 OpenSSL 进行工作并没有很多的帮助，所以可以不作深究。当然，这些目录中的文件在编译的时候起的作用可能是不可或缺的，但是这并非我们关注的焦点。

表 13-1 OpenSSL 安装说明文件名和相应的平台

文件名	系统平台简单描述
INSTALL	Linux 等 Unix 平台
INSTALL.W32	Windows 平台，包括 Windows98、Windows2000、WindowsNT 和 WindowsXP 等
INSTALL.WCE	WinCE 平台
INSTALL.MacOS	苹果电脑的操作平台 MacOS
INSTALL.OS2	OS2 操作平台
INSTALL.VMS	VMS 平台，一种在 Windows 系统下虚拟操作系统
INSTALL.DJGPP	DJGPP 平台，一种在 Windows 系统下的虚拟操作系统

Crypto 目录是 OpenSSL 所有密码算法和一些 PKI 相关标准源码存放的目录，也是 OpenSSL 最重要的一个目录。SSL 目录是 SSL 协议各个版本的实现源码存放的目录。Doc 目录是 OpenSSL 使用的说明文档存放的目录，这个目录对于 OpenSSL 使用者来说具有“芝麻开门”的作用。Apps 目录存放了 OpenSSL 所用应用程序的源代码文件，也是研究 OpenSSL 的 API 很好的例子。Demos 目录就是一些乐意奉献的人写的 OpenSSL 应用的例子了，在你开始使用 OpenSSL 进行工作之前，可以看看这个目录，或许会有所帮助。Include 目录是使用 OpenSSL 的库进行编程的时候可能需要使用到的一些头文件。Test 目录测试 OpenSSL 一些自身测试程序源文件所在的地方。表 13-2 是一些重要目录的名称和其对应的功能描述。

表 13-2 OpenSSL 部分目录的功能说明

目录名	功能描述
Crypto	存放 OpenSSL 所有加密算法源码文件和相关标注如 X.509 源码文件，是 OpenSSL 中最重要的目录，包含了 OpenSSL 密码算法库的所有内容。
SSL	存放 OpenSSL 中 SSL 协议各个版本和 TLS 1.0 协议源码文件，包含了 OpenSSL 协议库的所有内容。
Apps	存放 OpenSSL 中所有应用程序源码文件，如 CA、X509 等应用程序的源文件就存放在这里。
Doc	存放了 OpenSSL 中所有的使用说明文档，包含三个部分：应用程序说明文档、加密算法库 API 说明文档以及 SSL 协议 API 说明文档。
Demos	存放了一些基于 OpenSSL 的应用程序例子，这些例子一般都很简单，演示怎么使用 OpenSSL 其中的一个功能。
Include	存放了使用 OpenSSL 的库时需要的头文件。
Test	存放了 OpenSSL 自身功能测试程序的源码文件。

如果你在 Windows 平台下将 OpenSSL 编译成功后，还会增加三个新的目录：inc32、out32dll、tmp32dll。Inc32 目录根 Include 目录相似，存放的是 Windows 平台下使用 OpenSSL 进行编程需要包含的头文件。Out32dll 则存放了 OpenSSL 编译成功后的可执行应用程序、链接库 LIB 文件和动态 DLL 文件。Tmp32dll 则是在编译过程中存放 OBJ 等临时文件的目录。

13.2.2 OpenSSL 算法目录

OpenSSL 的算法目录 Crypto 目录包含了 OpenSSL 密码算法库的所有源代码文件，是 OpenSSL 中最重要的目录之一。OpenSSL 的密码算法库包含了 OpenSSL 中所有密码算法、密钥管理和证书管理相关标准的实现，在 Windows 下编程后的库文件名为 libeay32.lib，在 Linux 下编译后生产的库文件名为 libcrypto.a。Crypto 目录下包含了众多的子目录，这些目录大多数以相关的算法或标准名称的简写命名，阅读过前面章节的读者对这些名称应该至少有那么一点点印象，如果你真的没有印象，建议首先阅读前面的 12 章内容。当然,并非所有这些目录存放的源文件都是密码算法和标准，有些是 OpenSSL 本身的一些相关功能文件，如 BIO、DSO 和 EVP 等。

为了让读者对这些目录有一个总体的了解，表 13-3 列出了 Crypto 目录主要的子目录的功能和简单说明。

表 13-3 Crypto 子目录列表

目录名称	目录类型	内容或功能描述
Aes	对称算法	美国新的对称加密算法标准 AES 算法源码。
Bf	对称算法	Blowfish 对称加密算法源码。
Cast	对称算法	CAST 对称加密算法源码。
Des	对称算法	包括了 DES 和 3DES 对称加密算法源码。
Idea	对称算法	IDEA 对称加密算法源码。
Rc2	对称算法	RC2 对称加密算法源码。
Rc4	对称算法	RC4 对称加密算法源码。
Rc5	对称算法	RC5 对称加密算法源码。
Dh	非对称算法	DH 非对称密钥交换算法源码。
Dsa	非对称算法	DSA 非对称算法源码，用于数字签名。
Ec	非对称算法	EC 椭圆曲线算法源码。
Rsa	非对称算法	RSA 非对称加密算法源码，既可以用于密钥交换，也可以用于数字签名。
Md2	信息摘要算法	MD2 信息摘要算法源码。
Md5	信息摘要算法	MD5 信息摘要算法源码。
Mdc2	信息摘要算法	MDC2 信息摘要算法源码。
Sha	信息摘要算法	SHA 信息摘要算法源码，包括了 SHA1 算法。
Ripemd	信息摘要算法	RIPEMD-160 信息摘要算法源码。
Comp	数据压缩算法	数据压缩算法的函数接口，目前没有压缩算法，只是定义了一些空的接口函数。
Asn1	PKI 相关标准	ASN.1 标准实现源码，只实现了 PKI 相关的部分，不是完全实现。包括 DER 编解码等功能。
Ocsp	PKI 相关标准	OCSP（在线证书服务协议）实现源码。
Pem	PKI 相关标准	PEM 标准实现源码，包括了 PEM 的编解码功能。
Pkcs7	PKI 相关标准	PKCS#7 标准实现源码。PKCS#7 是实现加密信息封装的标准，包括了证书封装的标准和加密数据的封装标准。
Pkcs12	PKI 相关标准	PKCS#12 标准实现源码。包括了 PKCS#12 文件的编解码功能。PKCS#12 是一种常用的证书和密钥封装格式。
X509	PKI 相关标准	X.509 标准的实现源码。包括了 X.509 的编解码功能，证书管理功能等。
X509v3	PKI 相关标准	X.509 第三版扩展功能的实现源码。

Krb5	其它标准支持	支持 Kerberos 协议的一些接口函数和结构定义。
Hmac	其它标准支持	HMAC 标准的支持结构和函数源源代码。
Lhash	其它标准支持	动态 HASH 表结构和函数源代码。
Bio	自定义	OpenSSL 自身定义的一种抽象 IO 接口，封装了各种平台的几乎所有 IO 接口，如文件、内存、缓存、标准输入输出以及 Socket 等等。
Bn	自定义	OpenSSL 实现大数管理的结构及其函数。
Buffer	自定义	OpenSSL 自定义的缓冲区结构体。
Conf	自定义	OpenSSL 自定义的管理配置结构和函数。
Dso	自定义	OpenSSL 自定义的加载动态库的管理函数接口。如使用 Engine 机制就用到了这些函数提供的功能。
Engine	自定义	OpenSSL 自定义的 Engine 机制源代码。Engine 机制运行 OpenSSL 使用第三方提供的软件密码算法库或者硬件加密设备进行数据加密等运算。相当于 Windows 平台的 CSP 机制。
Err	自定义	OpenSSL 自定义的错误信息处理机制。
Evp	自定义	OpenSSL 定义的一组高层算法封装函数，包括了对称加密算法封装、非对称加密算法封装、签名验证算法封装以及信息摘要算法封装，类似 PKCS#11 提供的接口标准。
Objects	自定义	OpenSSL 管理各种数据对象的定义和函数。事实上，Objects 的 OID 是根据 ASN.1 的标准进行命名的，不完全是 OpenSSL 自定义的结构。
Rand	自定义	OpenSSL 的安全随机数产生函数和管理函数。
Stack	自定义	定义了 OpenSSL 中 STACK 结构和相关管理函数。
Threads	自定义	OpenSSL 处理线程的一些机制。
Txt_db	自定义	OpenSSL 提供的文本证书库的管理机制。
Ui	自定义	OpenSSL 定义的一下用户接口交换函数。
Perlasm	自定义	编译的时候需要用到的一些 Perl 辅助配置文件。

通过表 13-3，对 OpenSSL 密码算法库提供的功能也就有了一个大概的了解。事实上，在 Crypto 根目录下还有一些文件，也是密码算法库的组成部分之一，但是由于不是很主要的部分，这里就不再一一介绍。对 OpenSSL 提供的密码算法大概总结一下：对称加密算法 8 种，非对称加密算法 4 种，信息摘要算法 5 种。

13.2.3 OpenSSL 文档目录

OpenSSL 的文档目录 Doc 对于刚刚接触 OpenSSL 的人来说应该是一个最值得留恋的地方，从这里开始漫长的 OpenSSL 之旅是一个很好的选择。由于非商业软件开发的共同缺点，OpenSSL 提供的文档并不全面，甚至可以说是非常有限，而且不能及时跟着版本更新。但是对于初学者来说，这些文档能解决不少问题，事实上这也不能责怪 OpenSSL 的开发者，因为对于一个酷爱编写代码的优秀技术人员来说，写文档实在是一件枯燥无比的工作。当然，这些文档都是使用英文撰写的，如果你在读本书，你可能就是因为不太喜欢英文。

OpenSSL 的文档使用 Perl 文档格式保存，为.pod 文件，对于一些用户，比如 Windows 平台的用户，阅读这样的文档可能存在一些麻烦，因为直接用写字板或者其它阅读器打开这些文档会使格式显得凌乱。可以使用 Perl 工具 pod2text 或者 pod2html 命令将文档转换成 txt 文本格式或者 html 格式以方便阅读。

OpenSSL 的文档主要分为三部分：应用程序说明文档、密码算法库 API 文档以及 SSL

协议库 API 文档，分别对应 Doc 根目录下的三个子目录 Apps、Crypto 和 SSL。应用程序说明文档目录（Apps）包含了大部分 OpenSSL 应用程序的使用和参数说明，并有部分例子。密码算法库 API 文档（Crypto）则包含了部分 OpenSSL 密码算法库的 API 的使用说明，可惜不是很全面。SSL 协议库 API 文档包含了 OpenSSL 实现的 SSL 协议和 TLS 协议的大部分 API 使用说明，该部分由于变动比较缓慢，所以文档相对全面一些。此外，目前版本的 OpenSSL 文档目录下还有一个 Howto 子目录，现在只有一个文件，内容是关于证书的一些问题，希望后续的版本能够丰富这个 Howto 目录。OpenSSL 文档目录的根目录下还有两个综述性质的文档 ssleay.txt 和 openssl.txt，也是值得一看的。

13.3 OpenSSL 功能

作为一个基于密码学的安全开发包，OpenSSL 提供的功能相当强大和全面，囊括了主要的密码算法、常用的密钥和证书封装管理功能以及 SSL 协议，并提供了丰富的应用程序供测试或其它目的使用。

13.3.1 对称加密算法

OpenSSL 一共提供了 8 种对称加密算法，其中 7 种是分组加密算法，仅有的一种流加密算法是 RC4。这 7 种分组加密算法分别是 AES、DES、Blowfish、CAST、IDEA、RC2、RC5，都支持电子密码本模式（ECB）、加密分组链接模式（CBC）、加密反馈模式（CFB）和输出反馈模式（OFB）四种常用的分组密码加密模式。其中，AES 使用的加密反馈模式（CFB）和输出反馈模式（OFB）分组长度是 128 位，其它算法使用的则是 64 位。事实上，DES 算法里面不仅仅是常用的 DES 算法，还支持三个密钥和两个密钥 3DES 算法。

虽然每种加密算法都定义了自己的接口函数，但是 OpenSSL 还使用 EVP 封装了所有的对称加密算法，使得各种对成加密算法能够使用统一的 API 接口 EVP_Encrypt 和 EVP_Decrypt 进行数据的加密和解密，大大提供了代码的可重用性能。

13.3.2 非对称加密算法

OpenSSL 一共实现了 4 种非对称加密算法，包括 DH 算法、RSA 算法、DSA 算法和椭圆曲线算法（EC）。DH 算法一般用户密钥交换。RSA 算法既可以用于密钥交换，也可以用于数字签名，当然，如果你能够忍受其缓慢的速度，那么也可以用于数据加密。DSA 算法则一般只用于数字签名。

跟对称加密算法相似，OpenSSL 也使用 EVP 技术对不同功能的非对称加密算法进行封装，提供了统一的 API 接口。如果使用非对称加密算法进行密钥交换或者密钥加密，则使用 EVP_Seal 和 EVP_Open 进行加密和解密；如果使用非对称加密算法进行数字签名，则使用 EVP_Sign 和 EVP_Verify 进行签名和验证。

13.3.3 信息摘要算法

OpenSSL 实现了 5 种信息摘要算法，分别是 MD2、MD5、MDC2、SHA（SHA1）和 RIPEMD。SHA 算法事实上包括了 SHA 和 SHA1 两种信息摘要算法，此外，OpenSSL 还实

现了 DSS 标准中规定的两种信息摘要算法 DSS 和 DSS1。

OpenSSL 采用 EVP_Digest 接口作为信息摘要算法统一的 EVP 接口，对所有信息摘要算法进行了封装，提供了代码的重用性。当然，跟对称加密算法和非对称加密算法不一样，信息摘要算法是不可逆的，不需要一个解密的逆函数。

13.3.4 密钥和证书管理

密钥和证书管理是 PKI 的一个重要组成部分，OpenSSL 为之提供了丰富的功能，支持多种标准。

首先，OpenSSL 实现了 ASN.1 的证书和密钥相关标准，提供了对证书、公钥、私钥、证书请求以及 CRL 等数据对象的 DER、PEM 和 BASE64 的编解码功能。OpenSSL 提供了产生各种公开密钥对和对称密钥的方法、函数和应用程序，同时提供了对公钥和私钥的 DER 编解码功能。并实现了私钥的 PKCS#12 和 PKCS#8 的编解码功能。OpenSSL 在标准中提供了对私钥的加密保护功能，使得密钥可以安全地进行存储和分发。

在此基础上，OpenSSL 实现了对证书的 X.509 标准编解码、PKCS#12 格式的编解码以及 PKCS#7 的编解码功能。并提供了一种文本数据库，支持证书的管理功能，包括证书密钥产生、请求产生、证书签发、吊销和验证等功能。

事实上，OpenSSL 提供的 CA 应用程序就是一个小型的证书管理中心（CA），实现了证书签发的整个流程和证书管理的大部分机制。

13.3.5 SSL 和 TLS 协议

虽然已经有众多的软件实现了 OpenSSL 的功能，但是 OpenSSL 里面实现的 SSL 协议能够让我们对 SSL 协议有一个更加清楚的认识，因为至少存在两点：一是 OpenSSL 实现的 SSL 协议是开放源代码的，我们可以追究 SSL 协议实现的每一个细节；二是 OpenSSL 实现的 SSL 协议是纯粹的 SSL 协议，没有跟其它协议（如 HTTP）协议结合在一起，澄清了 SSL 协议的本来面目。由于 SSL 协议现在经常跟 HTTP 协议在一起应用形成 HTTPS 协议，所以给很多人误认为 SSL 协议就是为了保护 Web 安全性的，这实在是一个很大的误解！

OpenSSL 实现了 SSL 协议的 SSLv2 和 SSLv3，支持了其中绝大部分算法协议。OpenSSL 也实现了 TLSv1.0，TLS 是 SSLv3 的标准化版，虽然区别不大，但毕竟有很多细节不尽相同。OpenSSL 除了提供了使用 SSL 协议和 TLS 协议的 API 接口函数之外，还提供了两个不错的应用程序 S_Client 和 S_Server。S_Client 用来模拟 SSL 客户端，可以用来测试 SSL 服务器，比如 IIS 和带 mod_ssl 的 Apache 等；而 S_Server 模拟了一个 SSL 服务器，可以用来测试 SSL 客户端，比如 IE 和 Netscape 等。事实上，由于是开放源代码的，S_Client 和 S_Server 程序的源代码还是很好的 OpenSSL 的 SSL 接口 API 使用例子。

13.3.6 应用程序

OpenSSL 的应用程序已经成为了 OpenSSL 重要的一个组成部分，其重要性恐怕是 OpenSSL 的开发者开始没有想到的。现在 OpenSSL 的应用中，很多都是基于 OpenSSL 的应用程序而不是其 API 的，如 OpenCA，就是完全使用 OpenSSL 的应用程序实现的。OpenSSL 的应用程序是基于 OpenSSL 的密码算法库和 SSL 协议库写成的，所以也是一些非常好的

OpenSSL 的 API 使用范例，读懂所有这些范例，你对 OpenSSL 的 API 使用了解就比较全面了，当然，这也是一项锻炼你的意志力的工作。

OpenSSL 的应用程序提供了相对全面的功能，在相当多的人看来，OpenSSL 已经为自己做好了一切，不需要再做更多的开发工作了，所以，他们也把这些应用程序成为 OpenSSL 的指令。OpenSSL 的应用程序主要包括密钥生成、证书管理、格式转换、数据加密和签名、SSL 测试以及其它辅助配置功能。表 13-4 是 OpenSSL-0.9.7 版本的指令列表。在表 13-4 中，根据指令的性质，对指令进行了归类，这些归类不一定科学，但是对于本书的描述来说，会有帮助。这些类型包括：对称密钥指令、非对称密钥指令、信息摘要和签名指令、证书签发和管理指令、标准转换指令、SSL 测试指令以及其它指令。

指令	类型	功能说明
Asn1parse	其它	对 ASN.1 编码的文件或字符串进行解析，比如对证书文件，可以使用该命令对其进行解释，它会将其中每一个数据对象打印处理。
Ca	证书签发和管理	该命令是一个功能强大的命令，模拟了一个小型 CA 的功能，并跟 OpenSSL 提供的文本数据库联系起来作为证书数据库。该命令具有证书签发、验证、吊销等功能。
Ciphers	其它	该命令可以列出不同的协议支持的算法体系。
Crl	证书签发和管理	该命令可以对吊销证书列表（CRL）文件进行文本解析和验证。
Crl2pkcs7	格式转换	该命令可以将 CRL 和多个证书封装成一个 PKCS#7 格式的证书文件。
Dgst	信息摘要和签名	该命令可以使用不同的信息摘要算法对输入的信息进行信息摘要操作。并且可以对摘要的信息进行签名或者验证。
Dh	非对称密钥	生成和处理 DH 密钥参数文件，Dhparam 已经集成了该命令的功能，一般不使用该命令了。
Dhparam	非对称密钥	用于生成 DH 密钥参数文件、解析 DH 密钥参数文件以及格式转换等。在新版本中，这个命令集成了 dh 和 gendh 命令的功能，而 dh 和 gendh 命令在未来的版本中可能会消失或者用作其它用途。
Dsa	非对称密钥	该命令用于对 DSA 密钥的格式转换以及信息输出处理，并且可以对 DSA 密钥进行加密。
Dsaparam	非对称密钥	该命令用于生成和处理 DSA 参数文件，并且可以用于生成 DSA 密钥。
Enc	对称密钥	该命令可以使用 OpenSSL 支持的各种对称加密算法对给定的数据或者文件进行加密或者解密。
Engine	其它	该程序显示 OpenSSL 支持的 Engine 接口列表，并可以测试 OpenSSL 支持的 engine 接口是否有效。
Errstr	其它	根据给定错误代码显示相应的错误信息。
Gendh	非对称密钥	生成 DH 密钥，该功能已经集成到 dhparam 命令中。
Gendsa	非对称密钥	根据 DSA 参数文件生成一个 DSA 私钥，可以采用不同的算法对 DSA 私钥加密保护。
Genrsa	非对称密钥	生成 RSA 私钥，可以同时 RSA 私钥进行加密保护。
Nseq	格式转换	将普通 X.509 证书转换成 Netscape 格式的证书，也可以将 Netscape 证书转换成 X.509 证书。
Ocsp	其它	是一个实现了在线证书状态协议（OCSP）的命令工具，可以对

		证书的有效性进行验证等 OCSP 任务操作。
Passwd	其它	根据给定的口令通过 HASH 算法生成密钥。
Pkcs12	格式转换	该命令可以将 X.509 证书和 PEM 编码的私钥封装成 PKCS#12 格式的证书，也可以将 PKCS#12 格式的证书转换成 X.509 证书和私钥。
Pkcs7	格式转换	该命令将 PKCS#7 格式的文件转换成普通的 X.509 格式的证书或 CRL。
Pkcs8	格式转换	该命令可以将私钥加密转换成 PKCS#8 格式或者将 PKCS#8 格式私钥转换成普通的 PEM 或 DER 编码私钥。
Rand	其它	该命令可以产生一系列的伪随机数比特，并保存在文件中。
Req	证书签发和管理	该命令生成证书标准的请求文件，而且可以生成自签名的根证书。
Rsa	非对称密钥	该命令对 RSA 密钥进行格式转换和文本解析输出等处理，格式转换的时候可以对密钥进行加密。
Rsautl	非对称密钥	该命令采用 RSA 算法对输入数据进行签名、验证、加密和解密等操作。
S_client	SSL 测试	该命令模拟一个 SSL 客户端，可以对支持 SSL 的服务器进行测试和调试操作。
S_server	SSL 测试	该命令模拟一个 SSL 服务器，可以对支持 SSL 的服务器（如 IE）进行测试和链接操作。
S_time	SSL 测试	该命令可以用来测试建立一个 SSL 链接的时间。
Sess_id	SSL 测试	该命令可以处理经编码保存下来的 SSL Session 结构并可以根据选项打印出其中的信息。
Smime	其它	该命令可以用来对 S/MIME 邮件进行加密、解密、签名和验证等操作。
Speed	其它	该命令测试算法的速度，如果有硬件加密设备，也可以测试硬件加密设备的速度。
Spkac	其它	该命令用来处理 Nestcape 的签名公钥和挑战文件（SPKAC），可以验证 SPKAC 的签名，打印信息，也可以用来生成 SPKAC 文件。
Verify	证书签发和管理	该命令用来验证证书或者证书链的合法性。
Version	其它	该命令用来输出 OpenSSL 的版本信息。
X509	证书签发和管理	该命令用来显示证书内容以及签发新的证书。

在上述列出的指令中，并非是唯一的指令用法，有些指令，如 dgst，事实上包括了很多指令，如 Sha、md5 等等，直接使用这些算法名字作为指令也是一样的效果的。如下面两个命令执行结果是一样的：

- 采用 dgst 命令进行信息摘要操作

```
OpenSSL> dgst -md5 x509.o
MD5(x509.o)= 079a8fe71c51fe606d58d0c8117ac1f6
```

- 采用md5命令进行信息摘要操作

```
OpenSSL> md5 x509.o
MD5(x509.o)= 079a8fe71c51fe606d58d0c8117ac1f6
```

这样的命令还有 Enc，它也是可以采用不同的命令形式执行相同的操作，如下面两个指

令执行结果是相同的：

- 采用 enc 命令执行加密操作

```
OpenSSL> enc -des -in x509.o -out x5091.txt
enter des-cbc encryption password:
Verifying - enter des-cbc encryption password:
```

- 直接使用des命令执行加密操作

```
OpenSSL> des -in x509.o -out x5092.txt
enter des-cbc encryption password:
Verifying - enter des-cbc encryption password:
```

从上面的命令运行可以看到，它们事实上都是采用 DES-CBC 方式对文件进行加密。

一般来说，执行 OpenSSL 的应用程序是从 OpenSSL 指令开始，该指令事实上提供了一个进入所有其它指令的接口。开始进入该平台，可以在 openssl 执行程序下输入：

```
[dragonking@ann425 apps]$ ./openssl
OpenSSL>
```

在出现的提示符下输入相应命令即可。

如果在 Windows 平台下，你使用“MS Dev Studio workspace for OpenSSL”对 OpenSSL 源码进行编译，你可以得到独立的上述介绍的 OpenSSL 大部分应用程序，所以可以直接使用其名称直接运行，而不用先运行 OpenSSL.exe 命令，这跟在 OpenSSL 命令提供的平台下执行是一样的。事实上，OpenSSL 指令本身就是调用其它命令实现的。如 Windows 平台下面命令执行结果是相同的：

- 启动 OpenSSL 命令后再执行命令程序

```
G:\openssl\openssl-0.9.7\out32dll>openssl
OpenSSL> dgst -md5 .rnd
MD5(.rnd)= a53cf1c25f6acde6c55b2eaf4a9e3a9f
```

- 直接执行命令程序

```
G:\openssl\openssl-0.9.7\out32dll>dgst -md5 .rnd
MD5(.rnd)= a53cf1c25f6acde6c55b2eaf4a9e3a9f
```

13.3.7 Engine 机制

Engine 机制的出现是在 OpenSSL 的 0.9.6 版的事情，开始的时候是将普通版本跟支持 Engine 的版本分开的，到了 OpenSSL 的 0.9.7 版，Engine 机制集成到了 OpenSSL 的内核中，成为了 OpenSSL 不可缺少的一部分。

Engine 机制目的是为了使 OpenSSL 能够透明地使用第三方提供的软件加密库或者硬件加密设备进行加密。OpenSSL 的 Engine 机制成功地达到了这个目的，这使得 OpenSSL 已经不仅仅使一个加密库，而是提供了一个通用地加密接口，能够与绝大部分加密库或者加密设备协调工作。当然，要使特定加密库或加密设备更 OpenSSL 协调工作，需要写少量的接口代码，但是这样的工作量并不大，虽然还是需要一点密码学的知识。Engine 机制的功能跟 Windows 提供的 CSP 功能目标是基本相同的。

目前，OpenSSL 的 0.9.7 版本支持的内嵌第三方加密设备有 8 种，包括：CryptoSwift、nCipher、Atalla、Nuron、UBSEC、Aep、SureWare 以及 IBM 4758 CCA 的硬件加密设备。现在还出现了支持 PKCS#11 接口的 Engine 接口，支持微软 CryptoAPI 的接口也有人进行开发。当然，所有上述 Engine 接口支持不一定很全面，比如，可能支持其中一两种公开密钥

算法。表 13-5 是 OpenSSL-0.9.7 版本支持的硬件及其对应的简要描述名称，这个简要描述名称在很多时候是要使用的，如编程或执行命令的时候，简要密钥名称是大小写敏感的，目前一般都是采用小写字符。

表 13-5 OpenSSL 支持的 Engine 接口

简要名称	Engine 接口描述
dynamic	动态加载 Engine 设备的接口
cswift	CryptoSwift 的硬件加密设备 Engine 支持
chil	nChipher 硬件加密设备 Engine 支持
atalla	Atalla 硬件加密设备 Engine 支持
nuron	Nuron 硬件加密设备 Engine 支持
ubsec	UBSEC 硬件加密设备 Engine 支持
aep	Aep 硬件加密设备 Engine 支持
sureware	SureWare 硬件加密设备 Engine 支持
4758cca	IBM 4758 CCA 硬件加密设备 Engine 支持

13.3.8 辅助功能

前面介绍了 OpenSSL 的大部分功能，如果你第一次接触 OpenSSL 而又对密码学技术有所了解，这些功能相信会让你激动不已，如果你愿意，OpenSSL 其实还有一些能让你高兴的地方。

首先最值的一提的就是 OpenSSL 的 BIO 机制。BIO 机制是 OpenSSL 提供的一种高层 IO 接口，该接口封装了几乎所有类型的 IO 接口，如内存访问、文件访问以及 Socket 等。这使得代码的重用性大幅度提高，OpenSSL 提供 API 的复杂性也降低了很多。前面介绍的 EVP 封装也提高了 OpenSSL 代码的可重用性。

从前面应用程序的介绍就可以得知，OpenSSL 对于随机数的生成和管理也提供了一整套的解决方法和支持 API 函数。随机数的好坏是决定一个密钥是否安全的重要前提，OpenSSL 给我提供了这么一个解决方案，虽然不一定在所有的应用中都能令人满意。

OpenSSL 还提供了其它的一些辅助功能，如从口令生成密钥的 API，证书签发和管理中的配置文件机制等等。如果你有足够的耐心，将会在深入使用 OpenSSL 的过程慢慢发现很多这样的小功能，让你不断有新的惊喜。

13.4 OpenSSL 应用

OpenSSL 的应用一般可以分为两种不同的方式：基于 OpenSSL 指令的应用和基于 OpenSSL 加密库和协议库的应用。前者更容易一些，而后者需要做的工作更多一些。当然，这些应用不一定是截然分开的，你当然可以两种都用一点，比如使用 SSL 协议的 API，但是证书可以使用 OpenSSL 的指令签发。

13.4.1 基于 OpenSSL 指令的应用

基于 OpenSSL 指令的应用很容易，只要安装好了 OpenSSL，就可以开始使用了。 最简单的应用就是使用 Req、CA 以及 X509 指令来签发一个证书了，该证书可以用于各种目的，比如很多 Apache 服务器就是使用 OpenSSL 的指令生成的证书作为服务器证书。此外，作为测试目的，使用 OpenSSL 指令生成一个证书也是不错的选择。

对 OpenSSL 指令的集大成者应该是 OpenCA 了，它是一个完全基于 OpenSSL 的指令开发的一个证书认证中心（CA）程序，没有使用任何 OpenSSL 的 API。当然，它不仅仅使用了 OpenSSL 一个软件包，它还使用了诸如 OpenLDAP 等软件包，OpenCA 从一个侧面上证明了 OpenSSL 指令的作用是很全面的。

13.4.2 基于 OpenSSL 函数的应用

基于 OpenSSL 函数库的应用相比与基于 OpenSSL 指令的应用开发的工作量会大很多，但这并不意味着其应用会更少。事实上，基于 OpenSSL 的应用大部分是这种方式的，这种方式使得开发者能够根据自己的需求灵活地进行选择，而不必受 OpenSSL 有限的指令的限制。

Mod_ssl 是一个基于 OpenSSL 指令应用的范例，Apache 和 mod_ssl 结合起来实现 HTTPS 服务成功地在 Linux 和 Windows 等平台运行。Stunnel 是另外一个成功应用 OpenSSL 函数库的程序，该程序提供一种包含客户端和服务器的安全代理服务器功能，事实上就是在客户端和服务器使用 OpenSSL 建立一条安全的 SSL 通道。

除了这些广为人知的项目外，有很多公司的商业密码产品都是基于 OpenSSL 的函数库开发的，在 OpenSSL 提供了 Engine 机制后，这种势头会更加明显。当然，出于各种目的，相当一部分公司在自己的产品中不会声明这一点。

13.5 OpenSSL 授权

OpenSSL 的授权问题很多人并不重视，或者，你也不愿意浪费宝贵的时间在这一节上面。但是，阅读本书的人基本上都是以通过自己知识在产品中的实现来体现自己的价值的，所以努力维护知识产权，从根本上还是维护了我们自己的利益。

OpenSSL 的授权由两部分组成：OpenSSL 授权和原始的 SSLeay 授权。当然，这两种授权大同小异，基本上是没有限制，只要在产品中加入或者保留 OpenSSL 和 SSLeay 的版权声明就行。详细的授权内容请参考 OpenSSL 根目录下的 LICENSE。

13.6 本章小结

本章对 OpenSSL 做了一个初步的介绍，给读者全面展示了 OpenSSL 的历史、结构、功能和应用。

本章首先介绍了 OpenSSL 的背景，并将 OpenSSL 和其它同类的加密算法库做了比较。

本章还介绍了 OpenSSL 的结构，对 OpenSSL 的目录进行了总体的阐述，让读者对 OpenSSL 的结构有了一个全面的认识。

接着重点介绍了 OpenSSL 的功能，对 OpenSSL 的功能做了一个快速的浏览，展现了 OpenSSL 的魅力所在。

最后，本章还对 OpenSSL 的应用现状和实例做了简单的介绍，并对 OpenSSL 的授权做了强调。