构，实际使用中，还需要把这两部分数据组合成一个数据，使数据的管理和使用更方便。根据 RFC3161 规范建议，一般把时间戳数据作为 PKCS#7 的一个属性，即 PKCS#7 签名数据中包含时间戳。

在实际应用中，经常使用签名和时间戳对要发布的软件进行签名，这样用户安装软件时，就可以对软件的可信性进行验证。一般使用专用签名工具完成带时间戳的签名。

例如，打开一个带时间戳签名的软件文件的属性界面，如图 19-7 所示。

单击“详细信息按钮”，显示如图 19-8 所示的签名和时间戳。可以看出签名时间为 2013 年 12 月 26 日 14 时 50 分 28 秒。

图 19-7　带时间戳的软件数字签名信息

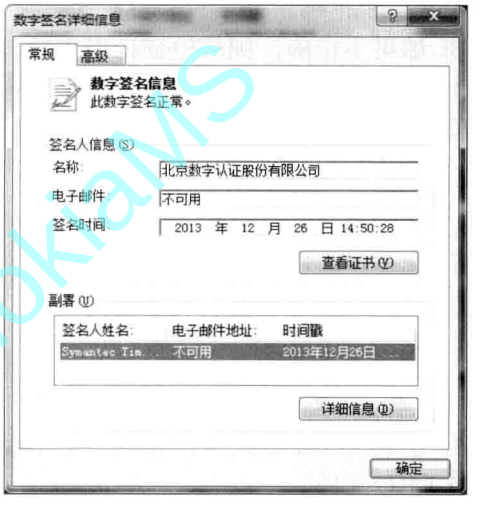

图 19-8　签名和时间戳

单击“详细信息”按钮，可以看到时间戳签发机构（Symantec Time Stamping Services Signer - G4），如图 19-9 所示。其证书信息如图 19-10 所示。

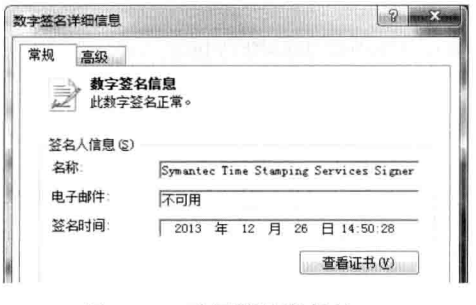

图 19-9　时间戳签发机构

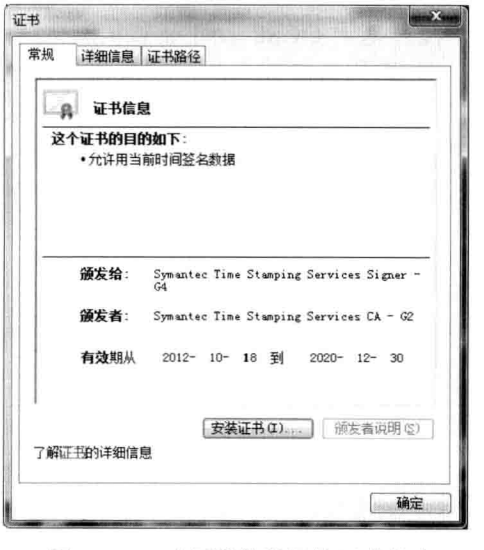

图 19-10　时间戳签发机构证书信息

19.5 证书有效性验证

用户在使用数字证书进行加密和验证数字签名时，必须首先验证证书的有效性。

验证证书有效性主要包括以下步骤。

1. 验证 CA 签名

① 将待验证证书设置为当前证书。

② 根据当前证书中签发者信息查询签发者证书，并使用签发者证书验证当前证书中的数字签名是否正确。

③ 如果不正确，则待验证证书无效。

④ 如果正确且签发者证书是预先确定的可信任 CA 证书，则说明待验证证书的 CA 签名正确；否则，将签发者证书设置为当前证书，重复步骤②～④。

2. 验证证书有效期

① 如果待验证证书中 notBefore 在当前日期之后，则该证书未生效。

② 如果待验证证书中 notAfter 在当前日期之前，则该证书已过期。

③ 否则，该证书处于有效期内。

3. 验证证书状态

（1）基于 CRL 验证证书状态

① 根据待验证证书中扩展项 CRLDistributionPoints 信息获得 CRL 地址，并根据该地址下载 CRL 文件。

② 检查 CRL 文件是否在有效期内，若不在则重新下载。

③ 验证 CRL 中数字签名是否正确，若不正确则重新下载。

④ 如果 CRL 中包括待验证证书的序列号，则说明该证书状态为无效；否则说明该证书状态为有效。

（2）基于 OCSP 验证证书状态

① 将待验证证书的序列号组织成 OCSP 请求包。

② 将 OCSP 请求包发送给 OCSP 服务器。

③ 从 OCSP 服务器获得 OCSP 响应包。

④ 解析 OCSP 响应包获得该证书的当前状态。

4. 验证证书其他属性

① 验证证书密钥用途：KeyUsage、ExtKeyUsage、NetscapeCertType。

② 验证证书策略：CertificatePolicies。

③ 其他专用属性。

第 20 章　通用应用技术

20.1　SSL/TLS（Secure Socket layer/Transport Layer Security）

20.1.1　概述

SSL/TLS 是 Secure Socket Layer/Transport Layer Security 的缩写。SSL 协议最初由 Netscape 公司设计开发，1994 年 11 月推出 SSL v2.0 版本。1996 年 3 月在对 SSL v2.0 进行重大改进的基础上，推出 SSL v3.0 版本，它解决了 SSL v2.0 中存在的许多问题，改进了它的许多局限性并且支持更多的加密算法。IETF 于 1999 年 1 月推出 TLS v1.0 协议。TLS v1.0 是一个 Internet 标准，完全建立在 SSL v3.0 协议的基础上，又称 SSL v3.1 协议。TLS v1.0 解决了 SSL v3.0 存在的一些问题，并在某些方面做了改进。其实，TLS v1.0 和 SSL v3.0 差异不是很大，但这些差异也足以令二者不具备互操作性。然而 TLS v1.0 协议提供了支持运行 SSL v3.0 协议的机制。

SSL/TLS 协议用于在两台机器之间提供端到端的数据保密性、数据完整性服务以及通信双方的身份认证服务。SSL 协议的优势在于它是与应用层协议独立无关的，高层的应用协议（如 HTTP、FTP、TELNET）能透明地建立于 SSL 协议之上。

SSL 协议位于 TCP/IP 协议模型的网络层和应用层之间，使用 TCP 来提供一种可靠的端到端的安全服务，它使客户/服务器应用之间的通信不被攻击窃听，并且始终对服务器进行认证，还可以选择对客户进行认证。SSL 协议在应用层通信之前就已经完成加密算法、通信密钥的协商以及服务器认证工作，在此之后，应用层协议所传送的数据都被加密。

SSL 实际上是由共同工作的两层协议组成的，如图 20-1 所示。从体系结构图可以看出 SSL 安全协议实际是握手协议（Handshake Protocol）、改变密码约定协议（Change Cipher Spec Protocol）、警告协议（Alter Protocol）、应用数据协议（Application Data Protocol）和记录协议（Record Protocol）组成的一个协议簇。

<table>
<tr><td>握手协议</td><td>警告协议</td><td>改变密码约定协议</td><td>应用数据协议</td></tr>
<tr><td colspan="4">记录协议</td></tr>
<tr><td colspan="4">TCP</td></tr>
<tr><td colspan="4">IP</td></tr>
</table>

图 20-1　SSL 协议体系结构

20.1.2　记录协议

SSL 记录协议为 SSL 连接提供了两种服务：

① 机密性服务：握手协议定义了共享的、可用于对 SSL 有效载荷进行常规加密的密钥。

② 消息完整性服务：握手协议还定义了共享的、可用来形成报文的鉴别码（MAC）的密钥。

在 SSL 协议中，所有的传输数据都被封装在记录中。记录是由记录头和长度不为 0 的记录数据组成的。所有的 SSL 通信都使用 SSL 记录协议，记录协议封装上层的握手协议、警告协议、改变密码约定协议和应用数据协议。SSL 记录协议规定了记录头和记录数据的具体格式。SSL 记录协议定义了要传输数据的格式，它位于可靠的传输协议之上（如 TCP），用于各种更高层协议的封装，记录协议主要完成分组和组合，压缩和解压缩，以及消息认证和加密等功能。

SSL 记录协议的数据处理过程描述如下：

① 记录层从上层接收应用数据。

② 记录层对数据进行分段，将其分割成可管理的明文块。每个记录的长度为 16KB 或者更小。记录层中不关心客户消息的边界，属于同一内容类型的消息可以放在同一记录层报文中传输。同样，一个较大的消息也可以被分段，通过多个记录层报文来传送。

③ 使用当前会话状态中定义的压缩算法对明文块记录进行压缩。这一步是可选的，这里的数据块压缩是无损压缩，并且增加的长度不能超过 1024 字节。

④ 对第 3 步的结果，使用当前会话状态中定义的消息验证码 MAC 算法和 MAC 密钥，计算 MAC 值，并添加到第 3 步的结果之后。消息验证码主要是为了检验消息的完整性。

⑤ 对第 4 步产生的结果利用 IDEA、DES、3DES 或其他加密算法进行加密。加密对内容长度的增长也不可以超过 1024 字节。

⑥ 对第 5 步产生的结果添加记录头。记录头的作用是为接收方提供对记录进行解释所必需的信息。记录头部分包含 3 部分信息：内容类型（8 位）、SSL 协议版本号和长度域（16 位）。内容类型字段标识消息类型，SSL 支持的 4 种类型：applcation_data、alert、handshake 和 change_cipher_spec。长度字段可以让接收方知道它要从线路上读取多少字节才能对消息进行处理。版本号只是一项确保每一方使用所磋商的版本的冗余性检查。

20.1.3 握手协议

SSL 中最复杂、最重要的部分是握手协议。这个协议用于建立会话，协商加密方法、鉴别方法、压缩方法和初始化操作，使服务器和客户能够相互鉴别对方的身份、协商加密和 MAC 算法以及用来保护在 SSL 记录中发送数据的加密密钥。在传输任何应用数据之前，都要使用握手协议。

SSL 握手协议流程如图 20-2 所示，具体的数据处理过程描述如下：

① 客户端将它所支持的算法列表连同一个密钥产生过程用作输入的随机数发送给服务器。

② 服务器根据列表内容选择一种加密算法，并将其连同一份包含服务器公用密钥的证书发送给客户端。该证书还包含了用于认证目的的服务器标识，服务器同时还提供了一个作为密钥产生过程部分输入的随机数。

③ 客户端对服务器的证书进行验证，并抽取服务器的公钥。然后，再产生一个称作 pre_master_secret 的随机密码串，并使用服务器的公钥对其进行加密。最后，客户端将加密后的信息发送给服务器。

④ 客户端与服务器端根据 pre_master_secret 以及客户端与服务器端交换的随机数值，独立计算出加密和 MAC 密钥。

⑤ 客户端将所有握手消息的 MAC 值发送给服务器。

⑥ 服务器将所有握手消息的 MAC 值发送给客户端。

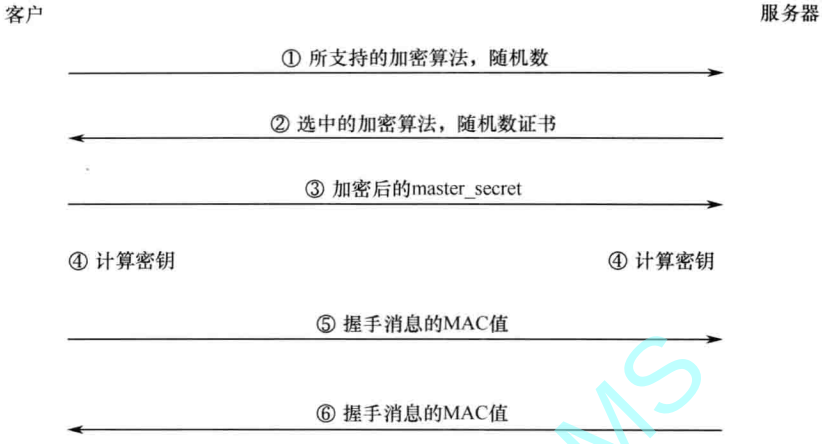

图 20-2　SSL 握手协议流程

第①步和第②步实现的目标是就一组算法达成一致。客户端告诉服务器它所支持的算法，而服务器选择其中的一种算法。当客户端收到服务器在第②步所发的消息时，它也会知道这种算法，所以双方在以后的通信中知道要使用的协议版本和算法。

第②步和第③步的目标是确立一组加密算法。第②步服务器向客户端提供其证书，这样就可以允许客户端给服务器传送密码。经过第③步后，客户端与服务器端都知道了 pre_master_secret。客户端知道 pre_master_secret 是因为这是它产生的，而服务器则是通过解密而得到 pre_master_secret 的。第③步是握手过程的关键步骤，所要保护的数据都依赖于 pre_master_secret 的安全。客户端使用服务器的公钥（从服务器证书中抽取）来加密共享密钥，而服务器使用其私钥对共享密钥进行解密。握手的剩余步骤主要用于确保这种交换过程的安全进行。

第④步中，客户端与服务器分别使用相同的密钥导出函数（Key Derivation Function，KDF）来产生 master_secret，最后再次通过 KDF 使用 master_secret 来产生加密密钥。

第⑤步与第⑥步用以防止握手本身遭受篡改。客户端提供多种算法的情况相当常见，某些强度弱而某些强度强。攻击者可以删除客户端在第①步所提供的所有高强度算法，迫使服务器选择一种弱强度的算法。第⑤步与第⑥步的 MAC 交换就能阻止这种攻击，因为客户端的 MAC 是根据原始消息计算得出的，而服务器的 MAC 是根据攻击者修改过的消息得出的，这样经过检查就会发现不匹配。由于客户端与服务器所提供的随机数为密钥产生过程的输入，所以握手不会受到重放攻击的影响。这些消息是首个在新的加密算法与密钥下加密的消息。

因此，在此过程结束时，客户端与服务器已就使用的加密算法达成一致，并拥有了一组与那些算法一起使用的密钥。更重要的是，它们可以确信攻击者没有干扰握手过程，所以磋商过程反映了双方的真实意图。

20.1.4　警告协议

SSL 警告协议是用来为对方传递 SSL 的相关警告。如果在通信过程中某一方发现任何异常，就需要给对方发送一条警示消息通告。