计算机科学1999Vol. 26№. 7

PKI中的证书和发证机构

Certificate and Certification Authority in Public Key Infrastructure

金晓耿 郭巍 金亿平 张世永
(复旦大学计算机科学系 上海 200433)

Abstract Public Key Infrastructure is an effect way to solve security problems in the Internet. In a PKI, end users use certificates which are created by Certification Authority(CA) to protect the privacy of information and to do authentication. This paper discusses the two certificate generation methods, the CA arrangements, their advantages and disadvantages, and the trust issues in a PKI

Keywords Public Key Infrastructure(PKI), Certification Authority(CA), Certificate, Certificate Revocation List(CRL), Trust

1 引言

Internet从诞生之日起就是一个开放的系统，把许许多多的计算机系统互联起来形成一个庞大的通信网络，但是这种开放性使得Internet中的信息很容易被拦截、监视和窜改，导致人们不愿意使用Internet交流一些敏感或机密信息，如商业信息等。这在一定程度上阻碍了Internet的发展进程。

Internet用户所面临的问题可以归为两大类：信息保密和身份验证。信息保密要求传送的信息在途中不得被修改，而身份验证要求通信的双方确认对方的身份。解决的方法之一是加密，确保信息的保密和安全；二是认证(certification)，保证通信是在预期的两者之间进行。公钥加密机制非常适合于Internet这方面的需求。在一个公钥加密系统中，每个用户拥有一对密钥(私钥和公钥)，使用一个密钥加密的内容可以用也只能用另一个密钥解密，用户保留私钥而将公钥发布出去。于是其他人可以利用他的公钥发送只能由他阅读的信息；或者利用私钥签名，其他人用他的公钥去检验该信息是否来自于他[1]。一个实用、可靠的，用来发布公钥的系统，我们称之为PKI(Public Key Infrastructure)。

在PKI中，用户(在PKIX中称之为端实体EE：End Entity)使用证书来标识自己的身份。一张证书最简单可以只包含一个公钥，但在具体应用中这往往是不够的，必须还包含其他一些信息。证书有一定的有效期限，这可能是一段相对比较长的时间。在这段时间内，证书所包含的某些信息很可能变成无效的内容，譬如用户的Email地址换了。如果用户的私钥遭到破坏，那么相对应的证书就不能再使用。这样的证书应该及时撤消以保护用户的权益。证书撤消列表(CRL)就是用来解决这一问题的标准。最初的X.509 v1 CRL缺乏细致长远的考虑，存在一些问题，我们在第2节讨论证书的内容及其生成、分析CRL存在的一些问题并提出相应的解决办法。

证书由发证机构CA(Certification Authority)发行，带有CA的签名。显然由单个CA负责现实世界中所有证书的发放是不可能的，必定存在多个CA，这些CA通过一定的结构组织起来协同工作。由不同CA签发的证书要有一定的互操作性，否则诸侯割据的现象无助于问题的解决。在第3节中将详细讨论CA及其管理的问题。

基于PKI的应用能够达到什么程度的安全性，取决于用户赋予所使用证书的信任度。CA实际上是一个受信任的第三方，在签发证书前负责确认证书申请人的身份。因此CA本身的安全性在PKI中显得特别重要。怎样正确地取得CA的证书？CA在签发证书前如何核实申请人的身份？这些问题在第4

金晓耿 硕士生，研究方向为计算机网络和数据通信。郭巍、金亿平 硕士生，研究方向为计算机组织与系统结构。张世永 教授，研究方向为计算机网络、数据通信、OSI标准化、一致性测试。

节阐述。

最后我们给出一个实例，介绍用户在PKI中相互交流信息的过程。

2 证书

证书由发证机构生成，包含用户的公钥以及其他一些信息，带有发证机构的签名，具有不可伪造性[2]。X.509是当前用得最广泛的一种证书格式，它除了包含用户的公钥，还有证书版本号、证书序列号、CA签名算法的ID、发行者名字、有效期限以及用户名字等等。在X.509 v2的证书中，加入了发行者UID(Unique Identifier)和用户UID，为了应用于更广泛的领域，在新版本的X.509证书(即X.509 v3)中，增加了证书扩展(extension)的机制，使得X.509 v3证书的扩展标准化和通用化。

证书是怎样生成的呢?有两种模式：集中式和分布式。集中式生成模式，就是公钥和私钥都由CA生成，公钥直接交给CA软件去生成证书，生成的证书通过适当的通道传送给用户，这个通道不需要是安全的，因为证书里有CA的签名，具有自保护的功能。分布式生成模式，就是密钥对由端实体EE生成，公钥被送往CA申请认证，如果申请有效(CA通过某些途径确证)，则CA返回相应的证书给申请人，并且同时可以将其发布到一些公共的证书库中，如X.500目录。

在集中式模式中，用户的申请和证书的传送在不同的实现中会有所不同。

手工分发[4]：在这种情况下，用户需要到CA由管理员登记在册，根据不同的安全策略，用户有可能被要求亲自去登记。然后CA生成一个已登记用户的令牌(在PKIX的术语中称为个人安全环境PSE)，该令牌包含用户的证书和相应的私钥，可以存储在磁盘或智能卡中交给用户。为提高令牌的安全性，可以使用一个PIN来保护它。这一技术不要求CA是在线的。

Web请求：用户使用浏览器访问CA有关申请证书的主页，输入一些必要的个人信息和口令之后，提交申请。该请求触发CA的相关程序为用户生成密钥对，然后CA通过Email形式通知用户去取回证书。通常Email里包含一个URL，当用户访问此URL时，会提示用户输入自己的口令，身份确认之后，证书通过HTTP消息(特殊的MIME类型)送给用户，并且保存于浏览器的证书数据库中。

在分布式模式中，密钥对由端实体生成，公钥送往CA签发。一个单独的公钥在传输过程中很容易被窜改，因为它没有任何ID与之相连。如何保护公钥在传送过程中免受窜改呢?有几种不同的方法。

·PKCS#10请求和PKCS#7响应：这是事实上的标准，使用最广泛的一种技术。一个PKCS#10的证书请求送往CA，而CA返回一个PKCS#7的证书响应。PKCS#10的消息带有一个数字签名，用来保护请求(特别是公钥)的完整性，并且提供请求者的身份验证。

·由PKCS#7保护的PKCS#10请求和PKCS#7响应：这是上述技术的一个变种，VeriSign使用的就是这种技术。PKCS#10的请求保护在一个PKCS#7的消息中。PKCS#7消息使用CA的公钥进行加密，因此只有CA才能将之解密，取出该证书请求。

·PKIMessage和Proprietary是在PKIX草稿中提出的两种新技术。

一张证书没有过期并不一定表明它是有效的。在有效期内，由于用户私钥受到破坏或证书的某些内容已改变，证书有可能被撤销。如何处理被撤销的证书呢?标准的方法是使用证书撤销列表CRL。一个CRL是一张被撤销证书的列表，带有CA的签名，并由CA负责定期更新。在验证证书的有效性时，用户必须检查最新的CRL以确保他将使用的证书未被撤销。

在CRL中我们关心的一个主要问题是它的大小。一个CA可能会签发上千甚至上万张证书，在如此多的证书中，撤销的比率通常很难预测，因此它的CRL可能变得非常大。一个CRL变得太大，分发CRL将占用太多的网络资源，用户要得到它会有困难，因为他们访问CA的带宽可能是有限的，并且，由于CRL是签过名的，在使用之前必须检验它的签名。检验一个巨大的CRL的签名，所花的时间将会很长。

在X.509 v1的CRL中，必须包括所有由CA签发且最终被撤销的证书，它的大小正比于申请证书的请求数、证书的有效期以及被撤销可能的比率。被分发CRL所占用的网络资源将正比于分发对象的数目、CRL的大小和CRL更新的频率，因此X.509 v1的CRL在大系统中是不实用的。为了纠正这种情形，X.509 v2的CRL提出了CRL分发点(CRL Distribution Point)概念，它将CA签发的所有证书按照某种方式分段成子集，每一子集有它自己的更小的CRL。譬如，一个企业的CA可以给每个部门发

布不同的CRL，这样用户要验证某人的证书时，他只要检查那个部门的CRL就可以了，而不需要检查整个的CRL。

为了减少分发CRL所占用的网络资源，X.509 v1的CRL通常更新得不是很频繁(相对来说)，这样很难提供及时的最新的撤销信息。Delta CRL将解决这一问题。Delta CRL使用一个基准CRL或CRL分发点，它们可能不必频繁更新，而Delta CRL则频繁被发布，它只包含对基准CRL的更新部分。由于Delta CRL小，它可以提供及时的撤销信息而不会占用太多的网络资源。

3 发证机构CA

CA是一个受一个或多个用户信任的机构，负责生成和分配证书、也可以负责生成用户的密钥对[2]。在基于X.509的PKI中，CA是按层次结构来组织的。最简单的形式是只有一个CA，负责签发所辖区域内的用户证书，这只能适合于小规模的组织或企业(图1)。当一个大型企业或组织拥有多个部门，而各部门使用证书又有不同的策略要求时，上述模式就不适合了。我们可以在总公司建立一个高层CA，各部门分别建立各自的子CA，这些子CA的证书由高层CA签发，这就形成了如图2所示的结构。如果两个企业拥有各自独立的CA，而他们之间要

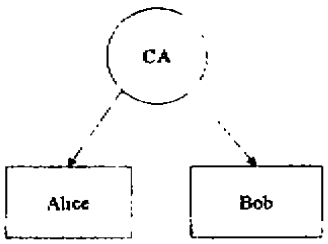

图1 最简单的层次结构

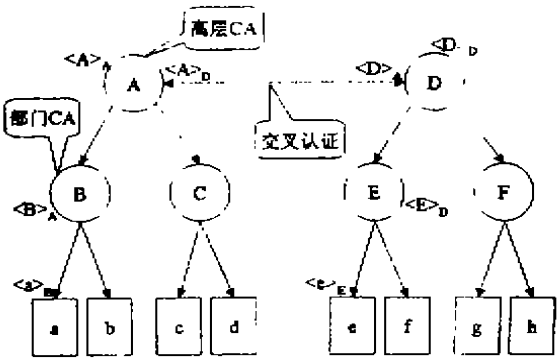

图2 带交叉认证的两层结构

利用证书进行通信，则两个CA必须进行交叉认证(图2)。

CA的通用层次结构如图3所示[3]，图中圆圈代表CA，方框代表端实体，箭头表示源给目的签发了证书。在这个结构中，每个CA不仅认证它的父亲，而且认证它的子女。虚线代表的是交叉证书。

有些PKI使用上述结构的一种变体，称为自顶向下的层次结构。这里每个CA只认证它的子女，最高层CA是所有认证路径的源，如图4所示[3]。

在自顶向下的层次结构中，所有用户必须使用最高层CA作为他们的根CA，这就要求所有用户在使用PKI前获得一份最高层CA公钥的拷贝。同时，所有用户必须完全信任最高层CA，一旦最高层CA的证书遭到破坏，就会破坏整个PKI的运作，因此在广域范围的PKI中不适用这种组织结构。

通用层次结构中，任何一个CA都可以是一条认证路径的根CA。但是整个结构依然依赖于上层的CA，特别是最高层的CA(图中的CA A)，很大一部分的认证路径要通过A，这迫使用户间接地信任CA A，于是CA A成为一个最易受到攻击的点。另

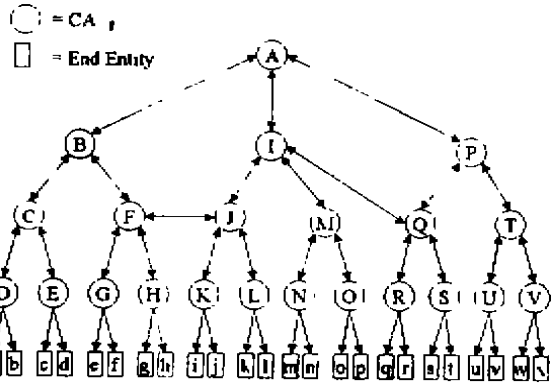

图3 CA的通用层次结构

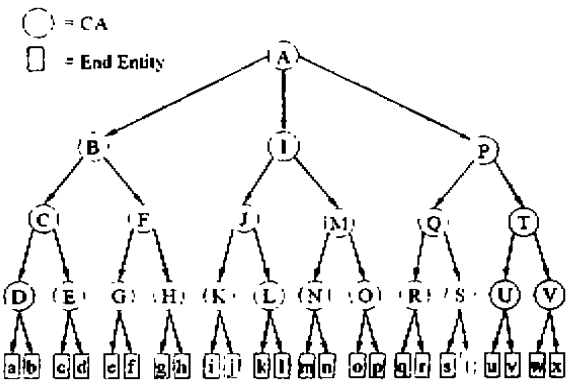

图4 CA的自顶向下层次结构

外认证路径有可能变得很长，交叉认证有助于缩短它的长度，但同时可能导致路径发现的复杂化。

4 信任

基于 PKI 的应用程序能够达到的安全程度，取决于它赋予所使用的证书的信任度，即取决于所使用证书中的公钥是不是真的由 Subject Name 定义的端实体（用户）所拥有。也就是说，这真的是 Alice 的证书吗？

如前所述，端实体的证书由 CA 签发，证书检验的过程有一步需要核实 CA 是否真的签发过该证书。这一过程依赖于端实体已获取的 CA 证书。通常 CA 的证书是自签名的，即 CA 的公钥用它自身相应的私钥来签名，自签名尽管可以保证证书的完整性，但它没有任何真实性的保护措施。这样，任何一个实体都可以产生一对密钥，生成一张自签名的证书，并声称是某某 CA 的证书（如 VeriSign 的证书）。因此用户必须确保所获得的 CA 证书的真实性。有几种方法来解决这一问题。

从已知网点下载 CA 的证书，这是最脆弱的一种方法。它的脆弱性在于该已知网点可能被伪造或破坏，返回一个错误的甚至是恶意的 CA 证书。在使用由这个假冒 CA 所签发的证书时，用户很容易受到攻击。嵌入证书主要应用于 Web 浏览器。很多浏览器出厂时都预装了某些 CA 的公钥或证书，如 Netscape Navigator 和 IE 都装有 VeriSign CA 的证书。这比前一种方法的安全程度要高，但是也意味着用户要依赖于浏览器产品所预装的 CA。如果某个 CA 的私钥受到破坏，用户怎么去重新装入新的 CA 证书呢？并且也限制了用户必须支持这些 CA 以及它们的拓扑。因此对于商业用户或企业来说，这种方法缺乏灵活性。安全发送是最安全的一种方法，有很大的灵活性。在这种情况下，CA 证书通过一个安全通道发送给用户，这一安全通道可以是将证书包在一个令牌中物理地发送给用户，如磁盘或智能卡，也可以是一个特殊的加密通信通道。这一方法允许 CA 处在企业的策略控制之下，并且可以支持各种 CA 拓扑。

当用户提交了一个证书申请请求后，CA 怎样知道它的确是申请中所声称的那个用户呢？一种方法是利用用户的 Email 地址。在请求中用户提供他自身的 Email 地址，CA 验证该地址是否存在，或者往该地址发一封信，通知用户使用信中的口令去下载证书。这种方法有一个前提条件，即假设 Email 地址不能被欺骗。不幸的是，这很容易做到。第二种方法要求用户在申请证书前必须到 CA 登记。当 CA 收到申请请求后，检查用户的 ID 是否已经登记，若是则生成证书。在请求中用户可以提供一个口令用于下载证书。第三种方法是离线检查，即在签发证书之前，CA 首先进行。这不是一个自动的过程，会涉及很多人工操作。

PKI 中的信任问题非常复杂，不是可以单单由一个好的 PKI 设计可以解决的，没有绝对可信任的对象，更何况“在 Internet 上没人知道你是一条狗”（摘自 Bill Gates 的《未来之路》）。一个 PKI 仅仅是一种用来表述信任关系的工具，任何 PKI 若试图在此方面做得更多，则必会减少它的灵活性。

5 实例

如前所述，在 PKI 中，两个端实体使用证书来保证彼此之间通信的保密性和进行身份验证。证书使用之前必须验证其有效性，步骤一般是这样的：

1）确定用户和发行者的名字是否有效；

2）检查证书的有效期限，确定其是否过期；

3）确定证书是否已被撤销；

4）确定证书中的签名是否有效。

在第四步中，对于不同的 CA 拓扑结构，验证过程会有些不同。在简单的拓扑结构中（图1所示），只要取得 CA 的证书或公钥即可验证其有效性。但在图2所示的带有交叉认证的层次结构中，过程就复杂得多。下面以图2的结构为例说明如何验证证书中的 CA 签名的有效性。

假设两个端实体 a 和 e 进行通信，e 收到了 a 的证书，要验明他的身份，以 $(Alice)_A$ 表示 Alice 由 CA A 签发的证书。这里 e 收到的证书是 $(a)_B$。步骤如下所示：

1）获取发行者 B 的证书或公钥，这里是 $(B)_A$。

2）利用 B 的公钥验证 $(a)_B$ 是否真的是 B 签发的。

3）由于 B 不是根 CA，无法验证自己的身份，所以必须验证 $(B)_A$ 中的签名是否有效，于是需要取得证书 $(B)_A$ 的发行者 A 的证书。此例中 A 有两张证书：$(A)_A$、$(A)_D$，分别由 A 和 D 签发，而 $(A)_D$ 是我们所要的。

4）利用 A 的证书 $(A)_D$ 验证 $(B)_A$ 是由 A 签发。

5）同理，获取证书 $(A)_D$ 的发行者 D 的证书或公钥，$(D)_D$ 是所需的证书。

（下转第76页）

的 P-join 操作中的连接路径，对当前两个嵌入式关系从连接节点的顶层开始，应用算法1，可得到当前两个嵌入式关系的连接。以下是算法的描述。

算法 LCF (the Least Cost, the First join)

m 个嵌入式关系 $r_1, r_2, \cdots, r_m$ 按分解存储模型分别存储为：

$r_1^1, r_1^2, \cdots, r_1^{n1}$；

$r_2^1, r_2^2, \cdots, r_2^{n2}$；

……

$r_m^1, r_m^2, \cdots, r_m^{nm}$。

假定在每一个嵌入式关系 r_i 的($1 \leqslant i \leqslant m$)序列中，含有连接节点属性的分解关系位于关系序列的尾部。同时令任意两个关系 r_i 与 r_j($1 \leqslant i \leqslant m, 1 \leqslant j \leqslant m$)的第一对含有某些相同属性，并且这些属性的父节点在连接路径上的分解关系分别记做 $r_i(j)$ 和 $r_j(i)$。显然有，$r_i(j) \in \{r_i^1, r_i^2, \cdots, r_i^{ni}\}$，$r_j(i) \in \{r_j^1, r_j^2, \cdots, r_j^{nj}\}$。

步1 建立连接序列 S，包含所有参与连接的关系 $r_1, r_2, \cdots, r_m$ 及各自的连接路径 $path_1, path_2, \cdots, path_m$。

步2 从序列 S 中选取两个关系 r_i 和 r_j，使得 $r_i(j)$ 或 $r_j(i)$ 的元组数在所有 m 个关系中最小。并且 $r_i(j)$ 和 $r_j(i)$ 在两个关系序列 $\{r_i^1, r_i^2, \cdots, r_i^{ni}\}$ 和 $\{r_j^1, r_j^2, \cdots, r_j^{nj}\}$ 中分别是 r_i^k 和 r_j^l。($1 \leqslant k \leqslant n_i, 1 \leqslant l \leqslant n_j$)

步3 对关系 r_i^k 和 r_j^l 沿着连接路径应用算法1，直到到达连接路径的终点(即路径决定节点)，然后，分别修改 r_i^k 和 r_j^l 的上一层分解关系每个元组的指针属性值。

步4 从当前两个嵌入式关系 r_i 和 r_j 的顶层开始，对每一对含有相同属性的分解关系 r_i^s 和 r_j^t($1 \leqslant s \leqslant k-1, 1 \leqslant t \leqslant l-1$)做自然连接。

步5 在 r_i 和 r_j 连接的最终模式树中，修改相关的指针属性值。

步6 将 r_i 和 r_j 的连接结果 r' 及连接路径 $path'$(由 r_i 和 r_j 的连接路径 $path_i$ 和 $path_j$ 确定)插入序列 S，同时从序列 S 中删除关系 r_i 和 r_j 及二者的连接路径。

步7 重复步2～6，直到序列 S 中仅剩余一个关系，即为连接结果。

讨论 从算法 LCF 的描述我们可以看出，两个嵌入式关系自然连接的复杂性主要产生于算法 LCF 的步3，随着连接路径长度(嵌入式深度)的增长，复杂性将成数量级地增长。但是，在连接路径已经确定的情况下，沿连接路径参与连接的第一对关系的连接代价直接影响到整个步3的代价。

算法 LCF 就是基于以上事实，每次从嵌入式关系序列中取出一对这样的关系：它们沿连接路径参与连接的第一对关系规模最小，这样就使每一次连接的代价始终是较小的，从而减少整个算法的代价。

参 考 文 献

1 Liu Hong-Cheu, et al. An Efficient Join for Nested Relational Databases. LNCS 1134. In: Wagner · Thoma, eds. DEXA' 96, Database and Expert Systems Application Springer Press

2 Roth M A, et al. Extended Algebra and Calculus for non-1NF Relational Databases. ACM Transactions on Database Systems, 1988, 13(4)

3 Liu H C, Ramamohanarao K. Algebraic Equivalences among Nested Relational Expressions. In: Proceedings of Third Int. Conf. on Information and Knowledge Management. Gaitherburg, Maryland, 1994

4 Garnett L, Tansel A. Equivalence of the Relational Algebra and Calculus for Nested Relations. Computers Math Applic., 1992, 23(10)

5 Hafez A, Ozsoyoglu G. Storage structures for nested relations. IEEE Database Engineering, 1988, 11(3)

(上接第86页)

6)验证 $(A)_D$ 是由 D 签发的，此时所有的验证结束。

于是我们得到 e 为验明 a 的身份所走的一条认证路径：$\langle a\rangle_B(B)_A\langle A\rangle_D(D)_D$。

参 考 文 献

1 Garfinkel S, Spafford G. Web Security & Commerce O' Reilly & Associates, June 1997

2 ITU-T. ITU-T Recommendation X. 509. ITU-T, June 1997

3 Ford W, Baum M. Secure Electronic Commerce: Building the Infrastructure for Digital Signatures and Encryption. Prentice Hall, 1997

4 Hughes J. Certificates Inter-operability. Entegrity Inc, July 1998