第33卷 第12期 计 算 机 工 程 2007年6月
Vol.33 No.12 Computer Engineering June 2007
·安全技术· 文章编号：1000—3428(2007)12—0188—03 文献标识码：A 中图分类号：TP393.08

一种改进的PMI属性证书撤销方案

赵 明，刘 佳
(解放军信息工程大学信息工程学院，郑州 450002)

摘 要：在PMI授权管理体系中，有关属性证书的维护是其中重要的组成部分，尤其在用户角色相对固定的大规模应用环境中不合理的证书撤销管理将会带来巨大的运算或网络传输负担。并且证书撤销列表的发布是由属性权威(AA)生成，然后交由一个公开目录对外发布。由于公开的目录不能够保证是安全可靠的，在撤销列表的发布中不能假设发布机构是可以信赖的，因此证书撤销列表也需要AA的签名来保证其真实性。该文提出了一种属性证书撤销列表的维护方案，解决了上述的问题，对方案的有效性、安全性和性能进行了分析。
关键词：PMI；属性权威；属性证书；Hash表

An Improved Scheme of PMI Attribute Certificate Revocation

ZHAO Ming, LIU Jia
(Information Engineer Institute, Information Engineer University of PLA, Zhengzhou 450002)

【Abstract】 In the system of privilege management infrastructure (PMI), the maintenance of attribute certificate is very important. Especially in the application environment where users' roles are relatively fixed, unreasonable certificate revocation management will bring enormous operations or burden onto network transmission. And the release of certificate revocation list is founded by attribute authority (AA) and handed to a public directory to release. Because public directory cannot be ensured safe or credible, it cannot be supposed that the institution is credible in the release of revocation list. So the sign of AA is needed to ensure the authenticity of certificate revocation list. This paper puts forward a maintenance scheme of certificate revocation list, which solves the above problems. The validity, security and capability of this scheme are analyzed.
【Key words】 PMI; Attribute authority(AA); Attribute certificate; Hash table

在公钥基础设施的使用过程中，超越当前PKI提供的身份验证和机密性、步入授权验证的领域、提供PKI信息环境的权限管理已成为当务之急。PMI是在PKI发展的过程中，为了将用户权限的管理与其公钥(身份的标志)的管理分离，由IETF提出的一种标准。PKI以公钥证书为基础，实现用户身份的统一管理，而PMI以X.509v4(2000)标准中提出的属性证书为基础，实现用户权限的统一管理。在开放的网络环境中，PMI的核心设施属性权威(attribute authority, AA)为不同的用户和机构进行属性证书的创建、存储、颁发和撤销。随着人们对网络安全的重视和对PKI/PMI 逐步深入的认识，越来越多的企业、机构已经开始采纳这一技术。

1 已有的证书撤销方案

属性证书的管理是授权管理系统中极为重要的组成部分。如果用户角色变动或证书终止使用就需要提前撤销用户的属性证书。在验证一个用户的证书时要检验该证书是否已经被撤销。目前证书撤销信息的组织方式主要有CRL(Certificate Revocation List)[1]、CRS(Certificate Revocation System)[2]和CRT(Certificate Revocation Tree)[3]。

以上的证书撤销方式同样适用于属性证书的撤销管理上。其中使用最广泛的是ACRL[4]。一个ACRL是由AA签发的列表，其中包含了所有被AA撤销的属性证书的序列号。这个列表包含了更新生效的时间以及有效期。在有效期结束之前即使属性证书撤销列表没有任何变化，AA也必须签发新的列表来代替旧的。这个列表将被发送至发布机构进行发布，如果一个用户需要检查一个属性证书是否已经被撤销，他需要到发布机构的目录下载整个列表来确定证书是否已经被撤销。这个方案存在的主要问题是用户与发布目录之间繁重的通信量。随着AA中心的规模逐渐增大，撤销的属性证书也逐渐增多，用户为了检查一个属性证书的有效性，必须下载整个撤销列表，势必造成通信的浪费。

CRS是由Micali提出的证书撤销方案，其主要思想是CA周期性地利用杂凑函数对系统中所有证书的状态给出证明。CRS能够显著减小需要提供给用户数据的大小，减少了证书撤销机构与用户之间的通信流量。然而在这种情况下，CA与发布机构之间的通信量、CA的计算量、发布机构的计算量都需要大大增加。

CRT采用了Merkle二叉杂凑树来表示某个域中的所有撤销信息。为了产生一个杂凑树，每个CA需要产生一系列表达式。CRT的缺点是更新成本高，任何更新都会引起整个树的变化，从而导致整个树的重新计算。

由于CRL具有简单和易于实现的特点，成为了目前应用最为广泛的撤销机制。同时为了提高证书撤销列表管理的效率，一些相应的CRL改进方案被提了出来，主要包括增量CRL(Delta-CRL)、分段CRL(Segment-CRL)、重复颁发CRL(Over-issued CRL)[5]。

1.1 增量CRL(Delta-CRL)

缩短CRL的颁发周期可以减少CRL时延给用户带来的风险，但频繁发布庞大的CRL给CA和用户都带来了沉重的负

作者简介：赵 明(1978－)，男，硕士生，主研方向：密码学，信息安全；刘 佳，博士生
收稿日期：2006-08-14 **E-mail：**wsyw2000@sina.com

担。解决CRL时延问题的一种新机制是Delta-CRL，它包含两种类型的CRL：基准CRL和增量CRL。基准CRL发布注销的证书信息，增量CRL在每两次基准CRL更新期间，再以较小的时间间隔颁发自上一次基准CRL颁发以来新增加的注销证书供用户下载。Delta-CRL提高了CA发布注销证书的频率，同时也减小了用户下载尺寸，但是每一次Delta-CRL的发布也会出现类似CRL的峰值请求高峰；而且在用户请求完一次Delta-CRL时，还需请求基准CRL，增加了用户的平均请求率。

1.2 分段 CRL(Delta-CRL)

按照某种分类方式将 CRL 分段发布，可以减少 CRL 长度。如可以按照地区分段，用户只需下载该地区的 CRL 段，无需下载全部 CRL。每段 CRL 的存放地点在颁发证书时，可以在扩展字段中指明其存放地址。但是，证书一旦颁发，CRL 段的存放地址就已经固定。为了改变这种不利于扩展的局限性，提出了重定向 CRL，即在证书扩展项中不是指明 CRL 段的存放地点，而是指出可以找到 CRL 段地址的地址。这种方法也称为动态分段 CRL。

分段 CRL 可以发布在不同的证书库中，它减少了用户下载 CRL 长度，但不会降低峰值请求，而且在用户需要查询多个段时，反而增加了用户的平均请求率和等待时间。

1.3 重复颁发 CRL(Over-issued CRL)

将 CRL 交错颁发，即在旧的 CRL 还未到期时，即发布新的 CRL，使用户缓存中的 CRL 在不同时间到期，来降低峰值请求。如第一个 CRL 在 0：00 发布，24：00 到期，每隔 6 小时，再发布一次新的 CRL，即 6：00 发布，次日 6：00 到期，依次类推用户下载 CRL 后将其缓存，只有在 CRL 过期时，才会去下载新的 CRL。将 CRL 到期时间交错开可以降低用户的峰值请求。

1.4 方案的性能指标

一个良好的证书注销体系在安全性、可扩展性、服务器性能、用户查询代价方面能够满足应用需求。首先要保证所发布消息安全可靠，不会遭到不可信证书库篡改或伪造；当 PKI 规模扩张时，容易扩展适应新的规模；服务器性能主要是指峰值请求和网络带宽；用户查询代价指用户所需下载的 CRL 大小、请求响应延迟及信息是否最新等几个方面。通过分析可以看出虽然上述的 3 种改进方案解决了 CRL 中某些方面的问题，但是每种改进方案都有其自身的局限性。由此，本文提出了一种基于 Hash 算法的部分签名协议，并在此基础上设计了采用基于 Hash 表的证书撤销维护方案，分析表明此方案在各项性能指数上都有较大的提高。

2 基于 Hash 算法的部分签名协议(HPSP)

2.1 哈希表的建立

协议中对属性证书的撤销管理基于Hash表机制[6]，这里采用链表模式的Hash表进行证书撤销列表的存储。步骤如下：

(1)对撤销的证书的序列号进行 Hash 运算确定其在 Hash 表中的地址，之后在相应地址处插入结点；

(2)Hash表中每一个结点不仅有指向下一个结点的指针，数据区中还包括相应证书的序列号ID_i和一个Hash值h_i。Hash值的参数包含本结点的证书序列号ID_i和前一结点的Hash值h_{i-1}，即$h_i = Hash(ID_i, h_{i-1})$。如果结点中指向下一个结点的指针为空，则说明该结点是链表中的最后一个节点。链表中的首结点(表头结点)的Hash值参数只有该结点的证书序列号ID_1，即$h_1 = Hash(ID_1)$。

(3)AA按照上面的规则建立Hash表后，对每一个链表使用AA的私钥进行签名，其中要包括链表地址a中最后一个结点的Hash值h_n，链表的签发时间t_1以及下次更新时间t_2，即$S_a = Sig(Hash(h_n a, t_1, t_2))$。这样对链表中的任意一个结点的数据区进行篡改，都不能通过AA签名的验证。对于没有结点的空地址AA也要进行签名，内容包括链表地址a签发时间t_1，下次更新时间t_2。即$S_a = Sig(Hash(a, t_1, t_2))$。

在建立 Hash 表时可以：

(1)为不同的链表设定不同的更新时间，将各个链表到期时间交错开后，便可降低用户的峰值请求。

(2)针对分布式的发布机构将 Hash 表进行分布式存储，这样可以减少每个发布机构存储 Hash 表所占用的空间及访问量。用户可根据需要到特定的发布机构下载链表。

为尽量避免出现太多的地址冲突，Hash 地址的算法应该使 Hash 表分布均匀。笔者建立的 Hash 表结构如图 1 所示，其中每一个结点由属性证书序列号、Hash 值及下一结点指针组成。

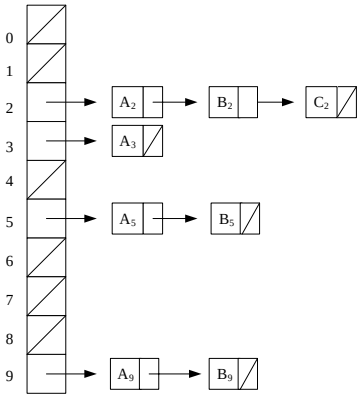

图 1 哈希表结构

2.2 属性证书的查询

用户进行属性证书撤销情况的查询时，首先将要查询的证书序列号ID_i发送给发布机构，发布机构根据证书序列号计算其在Hash表中的地址a，之后将该地址中的整条链表，包括结点的数据区、AA的签名S_{ai}以及a、t_1、t_2发送给用户。用户根据返回的链表判断该属性证书是否被撤销，随后可根据需要对链表进行存储。

若该地址中没有结点，则将AA对该地址的签名S_a以及a、t_1、t_2返回给用户。

2.3 撤销列表的更新

AA中维护一个与发布机构相同的Hash表，正常更新时AA不需要向发布机构传送整个Hash表的结构。更新的具体过程如下：

(1)更新时AA向发布机构指出新增的撤销证书序列号，发布机构在Hash表中增加与之对应的结点。每个链表中新增加的结点放置在链表的末端，之后按照Hash表建立的方式为该结点计算Hash值，并将链表中末端结点的Hash值提交给AA；

(2)已经被撤销或超出有效期自然失效的证书，发布机构在Hash表中删除与之对应的结点，之后重新计算链表中的Hash值，并将表头结点的Hash值提交给AA。

AA 获得该 Hash 值h_n''后，用同样的方法更新自己的 Hash 表并与自身的相应结点 Hash 值h_n'进行比较，若$h_n'' = h_n'$，则根据该 Hash 值产生新的签名$S_a' = Sig(Hash(a, t_1', t_2', h_n''))$，将签名与新的更新时间 a、$t_1'$、$t_2'$一同发送给发布机构。发布机构获得新的签名后就完成了一次更新；若不同则要求发布机构重新生成 Hash 值。

2.4 属性证书的检验

检验返回的查询结果是否有效，此过程通过计算返回的每一个结点数据区的 Hash 值来完成。用户从路径的表头结点

开始，计算数据区的 Hash 值，与下一结点保存的 Hash 值相比较，如果相同，则说明该结点数据没有被改动是可信的。以此类推，直到末端结点，末端结点的真实性依靠 AA 的签名来检验。如果返回数据通过上述检验，说明查询结果是有效的。该模型如图 2 所示。

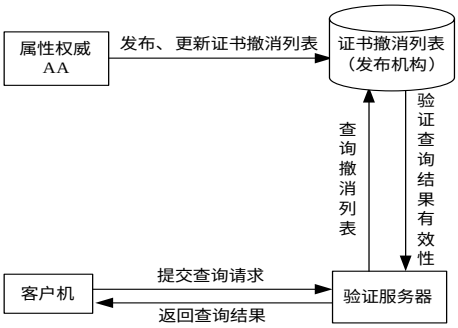

图 2 方案结构模型

3 协议分析

3.1 有效性分析

(1)如果证书已经被撤销，则在Hash表中有对应的结点，可以在返回的链表中找到对应的结点。

(2)如果一个证书没有被撤销，则在 Hash 表中没有对应的结点，在返回的链表(可能为空链表)中没有对应的结点。

3.2 安全性分析

(1)AA对链表末端结点的数据进行签名，末端结点数据区的Hash值与该链表中所有的结点数据都相关。根据Hash函数的特点，所有对结点数据的篡改都将会被发现，所以能够检测出任何对结点的非法修改。

(2)无论是发布机构还是攻击者，不能够截短查询结果中的返回路径。截短的路径可以通过第一步AA签名的检查，但是由于数据区被截断的结点对应的Hash值不为空，根据Hash函数的特点，将被发现是非法截断的路径。具体过程如图3所示。

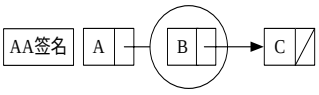

图 3 非法截短路径的检测

如图3，假设发布结构或攻击者将结点B从查询结果中截断，则申请者在进行查询结果的检验时根据A的Hash值以及C的证书序列号计算Hash值时会发现计算结果与结点C中的Hash值不相等，从而发现路径被非法截断。

(3)笔者认为发布机构是不可信的，如果出现发布机构不诚实，想通过返回非法的路径来欺骗申请者的情况，在本协议中也将被发现。因为AA对链表中的每个地址都进行了签名。对于有链表存储的地址，AA对末端结点的Hash值以及更新时间等参数进行了签名，在这种情况下，根据Hash函数及数字签名的性质，发布机构将不能对查询结果进行欺骗。对于不包含结点的地址，由于AA也对其进行了签名，发布机构同样不能对返回结果进行欺骗。

由此可见，本方案不仅可以抵抗来自发布机构外部的攻击，也可以抵抗来自发布机构内部的破坏，具有较高安全性。

3.3 性能分析

本节将HPSP协议与其他的证书撤销方案进行性能方面的比较：

(1)在HPSP中可以对Hash表按地址进行分段，采用分布式存储的方式，减少了单个发布机构的访问流量和存储空间。同时避免了用户要下载整个撤销列表的情况，只需下载与用户相关的撤销链表；

(2)证书更新采用了增量更新的方式，可以缩短撤销列表的更新周期，减少时延带来的风险，同时避免了每次更新需要发布庞大的撤销列表。

(3)对每个地址中的链表可以定义不同的发布时间和下次更新时间，使用户缓存的链表在不同时间到期，降低了峰值请求。

(4)支持离线、在线两种方式，用户可根据自身环境选择是否存储撤销链表。

表 1 给出了 HPSP 与其它方案的性能对比情况。

表 1 证书撤销方案性能比较

撤销方法	访问量与峰值	合时性	扩展性	安全性	用户端要求
CRL	流量高，存在峰值问题	延迟较大	适合小规模或撤销频率低的系统	由AA签名保证安全性	支持离线使用
增量CRL	相对CRL访问率和流量有改善	经常发布新的撤销信息，合时性较高	相对CRL有所改善	与CRL相同	支持离线使用，需经常下载撤销信息
分段CRL	单个CRL分布访问率和流量减少	相对CRL，传输验证时间变短	适合大规模、高撤销频率	与CRL相同	不适合离线使用
重复颁发CRL	解决峰值问题，性能显著改善	相对CRL，发布延迟减小	相对CRL有所改善	存在多个信息源弱化不可否认服务	支持离线使用
HPSP	解决峰值问题，性能显著改善	能够经常发布新的撤销信息，合时性较高	适合大规模、高撤销频率系统	由Hash运算及AA签名保证其安全性	支持离线使用

由表1可以看出：与其它属性撤销方案相比，CRL规则在性能、合时性、扩展性、安全性以及用户端要求方面都有了明显的改善和提高，是一种较为理想的属性证书撤销列表的维护方案。

此外还将对撤销列表更新时的通信量以及用户证书是否已经被撤销时的通信量进行分析。因为进行Hash表更新时，AA只需指出新增与删除哪些结点，而不用传输整个Hash表，所以只要很少的通信量。

每次查询最多需要向用户传输k个结点的内容以及AA对表头结点的签名，将对表头结点的签名长度记为 l_{sig}。每个结点的内容包括该证书的序列号、指向下个结点的指针以及本结点的Hash值。假设证书的序列号长度为20B，Hash函数的输出也是20B，那么一个结点需要传输的内容为40B。

假设共有N个需要撤销的证书存储到M个链表中，则在一个具有M个链表、N个结点的Hash表中，每个链表的结点数在N/M常数倍内的概率趋近于1。这是因为已知表中具有k个项的概率P为

$$P = \binom{N}{k}\left(\frac{1}{M}\right)^k\left(1-\frac{1}{M}\right)^{N-k}$$

其中有一个基本参数。从N个项中选出k个项：这k个项Hash到已知链表的概率为$1/M$，而其他N-K个项不Hash到已知链表的概率为 $1-(1/M)$。令 $\alpha = N/M$，上式可以写成

$$P = \binom{N}{k}\left(\frac{\alpha}{N}\right)^k\left(1-\frac{\alpha}{N}\right)^{N-k}$$

根据泊松近似，上式小于 $\frac{\alpha^k e^{-\alpha}}{k!}$。从这个结果中可以推知，链表中项多于 $t\alpha$ 的概率小于 $\left(\frac{\alpha e}{t}\right)^t e^{-\alpha}$。对于实际参数值域，这个概率非常小。例如，如果链表平均长度是20，那么，通过Hash运算得到链表中长度大于40的概率小于 $(20e/2)^2 e^{-20} \approx 0.0000016$。

(下转第201页)