一种 CA 私钥安全管理方案

蔡永泉, 杜秋玲
(北京工业大学计算机学院, 北京 100022)

摘 要: CA(certificate authority)是 PKI 中的重要组成部分, 负责签发可以识别用户身份的数字证书. CA 的私有密钥一旦泄露, 它所签发的所有证书将全部作废. 因此, 保护 CA 私钥的安全性是整个 PKI 安全的核心. 本文介绍的 CA 私钥安全管理方案主要基于门限密码技术. 通过将不同的密钥份额分布在不同部件上、任何部件都无法重构私钥, 来确保在密钥产生、分发及使用过程中, 即使部分系统部件受到攻击或系统管理人员背叛, 也不会泄漏 CA 的私钥, CA 仍可以正常工作.

关键词: 认证机构; 密钥管理; 容忍入侵
中图分类号: TP309 文献标识码: A 文章编号: 0372-2112 (2005) 08-1407-04

A Secure Scheme for Managing a CA Private Key

CAI Yong-quan, DU Qiu-ling
(*Institute of Computer science, Beijing University of Technology, Beijing* 100022, *China*)

Abstract: CA (certificate authority) is an important component in PKI (Public Key Infrastructure), and its main task is to issue and sign digital certificates that can identify different users. When the private key of a CA is compromised, all the certificates that are issued by this CA would be revoked. So, keeping the private key secret is the core of the whole PKI security. The secure managing scheme for protecting the private key of a CA recommended in this article is based on threshold cryptography. By storing the private key of a CA in more than one components and by ensuring that any component of the CA is unable to reconstruct the private key, this scheme makes sure that even if some components are compromised or some system administrators betray the private key of the CA would not be leaked and the CA can still work normally in the process of generating, distributing and using the private key.

Key words: CA(certificate authority); key management; intrusion tolerant

1 引言

基于网络的电子商务、电子政务等正在蓬勃迅猛的发展, 给人类的生活带来巨大的变化. 而电子商务和电子政务等面临的最大问题是如何建立相互之间的信任关系以及如何保证传输信息的真实性、完整性、机密性和不可否认性. 公钥基础设施 PKI 是目前被广泛采用的解决这一系列问题的安全解决方案.

公钥基础设施 PKI 通过使用公开密钥技术来确保系统信息安全. 然而, 在开放的网络环境中, 如何鉴别公钥的真实性成为此方案所要解决的主要问题, 否则供给者就能篡改他人的公开密钥导致保密通信机制的崩溃. 因此, PKI 把权威的、可信赖的、公正的第三方机构 CA 作为其不可缺少的重要组成部分, 其他设备或人之间的通信和验证都依赖于 CA 所颁发的数字证书. 数字证书也就是将一个公开密钥和身份信息绑定在一起, 用 CA 的私钥签名后得到的数据结构. 从中我们可以看到, 保护 CA 私钥不泄露是整个 CA 安全的基础.

当攻击者攻击一台 CA 成功时, 攻击者有可能获得该机器内的资源, 从而找到 CA 的私钥, 这对于 PKI 系统来说是非常致命的. 同时, 应当保证无论任何人(包括内部工作人员)控制其中一台设备时, 都无法获得 CA 的私钥. 另外, 在 CA 私钥的产生过程中不应当存在唯一的可信者, 当 CA 私钥进行分发时, 应当保证即使分发者被黑客攻击成功, 也不会泄露 CA 的私钥. 由于硬件错误或其他原因, 导致一台或多台设备出现故障时, 应该保证整个 PKI 的正常运作.

本文针对文献[5]的弹性 CA 方案进行了改进, 较好的解决了上述问题. 具有以下特点:

(1) 系统中不存在唯一的可信者. CA 私钥的产生由多个 Share Server 共同完成, 不存在唯一的可信者. 当进行密钥分发时, 即使分发者被黑客攻击成功, 也不会泄露 CA 的私钥.

(2) 同时具备文献[5] CA 方案的诸多优点: (*a*) 攻击一个系统并不能得到私钥. 在 $t-k$ 方案中, 即使在全部掌握所有的 Share Server 时, 也不能得到 CA 的私钥. 掌握小于 t 台 Share Server 加上任何一个 Combiner 也不能得到 CA 的私钥; (*b*) 部

收稿日期: 2005-02-05; 修回日期: 2005-04-16
基金项目: 北京市重点学科建设(No. 4010001202111)

分设备损坏不影响正常的服务. 当一台或多台 Share Server 损坏时, 仍旧可以由其他的 Share Server(> = t 台) 来提供正常的签名服务; (c) 当一台 Share Server 开始计算时, 无需要知道他的合作者是谁, 也就是说不需要预先同步. (d) 该系统的算法和原理较简单.

2 CA 弹性方案[5]

该方案的系统结构如图 1 所示, 主要包括的组件有: Share Server(共享服务器)、Key distributer(分发者)、Combiner(合成器)、RA Agent(注册代理机构) 和 Repository Agent. 其中, RA Agent 是与 RA 的接口, 负责与 RA 进行保密通信, 并检查 RA的签名. 在用户直接申请证书时, 也通过此接口. Share Server 用于CA 签名, 每一个 Share Server 都有自己的 ID 号. RA Agent 通过广播信道 B1 与 Share Server 相连. 当有证书 cer 需要CA 签名时, RA 为此签名设定一个可以区分的任务号 $Task_{(cer)}$. RA Agent 通过广播将需要签名的信息与设定的任务号 $Task_{(cer)}$广播到 Share Server 的网络上. Share Server i 收到广播后, 根据自己的忙闲程度来完成部分密钥的签名. Combiner 是最后的计算单元, 将各服务器的部分签名及自身的签名结果合并得到最终的签名. Repository Agent 是与数据库的接口, 同时赋有系统错误检测和告警的任务. 当 Combiner 计算验证错误时会通知 Repository Agent. Repository Agent 可以通过其他 Combiner 的计算结果查找问题的服务器或 Combiner. Key distributor 通常是非在线的, 采用带外的方式进行密钥注入. 它负责分发 Share Server、Combiner 的密钥. 其过程如图 1 所示.

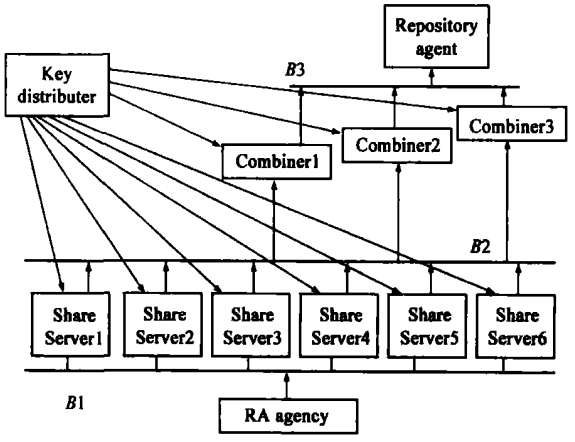

图 1 原 CA 系统结构图

该方案的优点是明显的, 它具有简单的结构, CA 私钥的存储及签名的过程采用双层弹性结构, 具有较好的安全性. 其不足之处在于, 密钥的分发是由唯一的可信者 distributer 来完成的. 而单一的分发者可能会引进单点攻击, 从而危及整个 PKI 系统的安全. 为了使密钥产生、分发具有更好的容忍入侵性, 本文对该方案进行了改进.

3 改进方案

3.1 改进 CA 弹性方案

原方案在 CA 私钥的产生、分发过程中存在唯一的可信者, 容易引进单点威胁, 而私钥的存储及使用私钥进行签名的过程确具有很好的安全性, 故改进方案重点对私钥的产生、分发方法进行改进, 其系统结构与原方案基本一致, 其中, 可信的密钥产生、分发者 Key distributer 由部分密钥分发者 Helper 取代, 同时, 组件的功能也发生了变化.

在原方案中, 私钥的产生、分发是由唯一的可信者 Key distributer 来完成的, 一旦攻击者攻破了 Key distributer, 就可以获得认证机构 CA 的私钥, 从而可以伪造证书、欺骗使用该证书的用户. 针对这种攻击的可能性, 改进方案采用分布式 RSA 算法来产生私钥. 该算法[4] 是由兼具存储私钥分额功能的 k 台共享服务器, 共同协作完成的. 最终产生私钥 d 和模数N (N, d 满足$C = M^d \bmod N$). 产生 N 和 d 的算法如下:

(1) 计算 N

(a) 每台 Share Server 随机地选两个整数 q_i, p_i 且不泄漏自身的q_i, p_i. 然后, k 台 Share Server 各自产生两个 l 次多项式f_i、$g_i \in Z_p[x]$ (其中, $f_i(0) = p_i$, $g_i(0) = q_i$, $l = \lfloor \frac{k-1}{2} \rfloor$), 一个 $2l$ 次多项式 $h_i \in Z_p[x]$ (其中, $h_i(0) = 0$).

(b) 每台服务器计算各自的 $p_{i,j} = f_i(j)$, $q_{i,j} = g_i(j)$, $h_{i,j} = h_i(j)$ 及 $N_i = \left(\sum_{j=1}^{k} p_{j,i}\right)\left(\sum_{j=1}^{k} q_{j,i}\right) + \sum_{j=1}^{k} h_{j,i} (\bmod P)$ ($j = 1, 2, \cdots, k$), 将其传递给其他服务器. 令 $\alpha(x) = \left(\sum_j f_i(x)\right)\left(\sum_j g_j(x)\right) + \left(\sum_j h_j(x)\right) (\bmod P)$, 如果 $\alpha(x)$ 满足 $\alpha(i) = N_i$, 由此可得 $N = a(0)$.

(2) 计算 d

(a) k 台共享服务器测试N 是否满足两个素数的乘积.

(b) 由第一台服务器计算 $\Phi_1 = N - p_1 - q_1 + 1$, 其他服务器计算 $\Phi_i = -p_i - q_i$, 并使得 $\Phi(N) = \sum \Phi_i$. k 台服务器共同计算$l = \Phi(N) \bmod e$, 令 $\gamma = l^{-1} \bmod e$. 最后, 每台服务器计算 $d_i = \lfloor \frac{-\gamma * \Phi_i}{e} \rfloor$, 作为各自的私钥份额.

最终, k 台服务器都获得私钥 d 的一个分额, 而每台服务器都不知道其他服务器的私钥分额信息. 同时, k 台服务器知道N 满足两个素数的乘积, 但都不知道它的因式分解. 在此时间段内, 攻破任何少于 k 台服务器的攻击者都无法获得 CA 私钥的信息. 从而避免了单一分发者带来的单点威胁, 增强了认证中心在私钥管理过程中的安全性.

另外, 为防止 Helper 成为唯一的密钥分发者, 在它为 Share server 进行部分密钥 d' 的分发之前, 需先将密钥的另一部分 d'' 存储到Combiner 内. 为了保证 Helper 和任意一台 Combiner 合谋也不会获取 CA 私钥 d, 需要将部分密钥 d'' 共享存储到多台 Combiner 内[2]. 采用(t', k') 门限存储方案, 当 $t' = 2$, $k' = 3$ 时, 攻击者要获取 CA 部分密钥 d'', 必须先攻破 3 台 Combiner 中的任意 2 台. 其具体的分配过程如下: 在使用 Helper 进行部分密钥 d' 的分发之前, 服务器要将自己 d_i 的一个随机数 r_i 发送给 Combiner(要保证每个 Combiner 都能收到一个 r_i, 部分 Combiner 会收到多个 r_i), Combiner 将收到的所有 r_i 相加作为自身的密钥份额. d''_j. 此时实现了(3, 3) 共享存储, 为达到容忍性, 需实现(2, 3) 共享存储, 如图 2 所示. 其中, $d'' = d''_1 + d''_2 = d''_3 + d''_4$. 首先, 根据此结构, 每台 Combiner 构

建自己 d''_j 的(2, 3)共享份额, $d''_j = d''_{j1} + d''_{j2} = d''_{j3} + d''_{j4}$ (j = 1, 2, 3), 之后, Combiner j 即 C_j 将份额 d''_{jl} (l = 1, 2, 3, 4)发送给其他 Combiner. 最后,每台 Combiner 将收到的密钥份额按组合相加, 作为自己最终的密钥份额 d'_j. 实现了攻击者攻破任何少于 2 台 Combiner 都不会获得该 CA 的部分密钥 d', 从而避免了 Helper 与任一台 Combiner 合谋, 获得 CA 的私钥.

C_1	C_2	C_3
d''_1	d''_2	d''_1
d''_3		d''_4

图 2　共享存储结构

3.2　改进方案的系统工作协议

CA 的私钥为 d, 公开密钥为 e 和 N. 设共有 k 台 Share Server, 对应的密钥份额为 d_i, Share Server i 记为 S_i (i = 1, 2, 3, …, k), $i_1, i_2, i_3, \cdots, i_k$ 为 Share Server 的 ID 号, k' 台 Combiner, $j_1, j_2, j_3, \cdots, j_{k'}$ 为 Combiner 的 ID 号.

第一步: k 台 Share Server 采用分布式 RSA 算法共同产生 CA 私钥, 运算结束时, 每台共享服务器 S_i 都持有 CA 私钥的一个分额 d_i, 但却无法获悉其他服务器的分额信息.

首先对分布式 RSA 算法的产生步骤进行一下简要的介绍, 具体细节请参见[4]:

(1) 每台 Share Server 随机地选两个整数 q_i, p_i, 并且保持它们的机密性.

(2) k 台 Share Server 通过一个秘密的分布式计算方法, 计算满足 $C = M^d \bmod N$ 的 N 值, 且不泄漏自身的 q_i, p_i 值.

(3) k 台共享服务器共同参与一个分布式计算方法来测试 N 是否满足两个素数的乘积. 如果测试失败了, 协议从第一步开始执行, 否则执行第 4 步. 同样, 这一步仍旧不会显示关于各服务器私钥分额的信息.

(4) 给定公钥 e, k 台服务器使用分布式算法产生各自的私钥份额 d_i.

第二步: 各服务器选定一个小于本地私钥分额的随机数 r_i, 获得新的私钥份额 $d'_i = d_i - r_i$, 将 r_i 通过安全的通道发送给 Combiner, Combiner 将收到的所有 r_i 相加作为自身的密钥份额. 每台 Combiner 构建各自 d'_j 的共享份额, $d'_j = d''_{j1} + d''_{j2} = d''_{j3} + d''_{j4}$ (j = 1, 2, 3). 并按设计的存储结构将自己的份额 d''_{jl} (l = 1, 2, 3, 4)发送给其他 Combiner. 最后, 每台 Combiner 将收到的密钥份额按组合相加, 作为自己最终的密钥份额 d'_j.

第三步: 各服务器将自己的私钥分额 d'_i 通过安全的通道发送给 Helper, 并撕毁原来的私钥分额. Helper 将 k 台服务器发来的 k 个私钥分额相加得到 $d' = \sum_{i=1}^{k} d'_i$. Helper 为每台服务器随机选定 1 个小于 (d'/t) 的随机数 d'_i (i = 1, 2, …, k), 再将 d'_i 通过安全的通道发送给对应的服务器作为私钥份额. 这些随机数可以远小于 d', 但有效长度不应低于 200 Bits. 从 k 个 d'_i 中, 任选 t 个, 共有 $j = C_k^t$ 种组合. 当 k = 5, t = 3 时, 共有 10 组. 设 $i_1, i_2, i_3, \cdots, i_t$ 为 Share Server 的 ID 号, $d'_{i_1} d'_{i_2} \cdots d'_{i_t}$ 是 ID 号为 $i_1, i_2, i_3, \cdots, i_t$ 的 Share Server 的私钥份额. Helper 针对每种组合计算: $c_j = d' - (d'_{i_1} + d'_{i_2} + \cdots + d'_{i_t})$, 并保存 $i_1, i_2, i_3, \cdots, i_t$ 和对应的 c_j. 之后, 将这 C_k^t 组数组 $(i_1, i_2, i_3, \cdots, i_t, c_j)$ 的部分送到一个 Combiner, 而将其他组合送到其他 Combiner.

至此, 密钥分发完成之后, 当用户提出签名请求时, Share Server 和 Combiner 将协同产生签名, 其工作步骤(详见文[5])简单介绍如下:

第 1 步: RA Agent 为证书 cer 分配任务号 $Task_{(cer)}$

第 2 步: 计算忙闲因子 $F_i(Task_{(cer)})$.

第 3 步: 接收针对此任务的忙闲因子的广播. 如果机器空闲, 直接进入第 4 步. 如果存在 m 个服务器的忙闲因子小于自己的忙闲因子时, 抛弃该任务. 否则执行下一步.

第 4 步: 计算 $y_i = (HASH_{(cer)})^{d'_i}$, 其中 $HASH$ 是文摘函数. Share Server 将计算的结果发送出去.

第 5 步: C_j 取用 t 个结果数据报, 并寻找组合 $(i_1, i_2, i_3, \cdots, i_t, c_j)$. 如果找到, 计算 $y'_j = (HASH_{(cer)})^{d''_j}$, 并向同一组合的 $t-1$ 个 Combiner 发出签名请求, 收到响应后使用 c_j 计算签名结果 $R = (HASH_{(cer)})^{c_j} * \prod_{i=i_1}^{i_t} y_i * \prod_{j=j_1}^{j_t} y'_j$. 如果没找到, 重新取用另外 t 个结果数据报, 重复第 5 步. 若穷举所有可能的结果组合, 仍旧没有找到匹配的 c_j 则抛弃该任务.

第 6 步: Combiner 利用公开密钥验证 R 的正确性, 验证正确就将 cer、各服务器的 ID 号与 R 一起通过广播信道送往 Repository Agent, 否则向广播信道广播任务失败消息. 其他 Combiner 收到任务失败消息后将启动计算, 重新计算该任务, 验证合格后都通过广播信道送往 Repository Agent. 计算失败的 Combiner 重新选择另外不同的 t 个结果(如果有的话), 并重新进行计算.

第 7 步: Repository Agent 将正确的 cer 与 R 一起送往数据库存档, 并通知其他 Combiner 停止该任务. Combiner 收到任务完成消息后即抛弃该任务的数据.

4　安全性分析

(1) 在私钥产生过程中, 采用的是分布式 RSA 算法, 由 k 台服务器共同产生, 可以保证服务器之间并不知道他人的私钥份额, 所以即使攻破其中的部分服务器, 也无法获得 CA 的私钥信息.

(2) 在分发的过程中, 由于各服务器已经将产生的随机数发送给了 Combiner, Helper 负责分发的私钥并不是 CA 的真正私钥, 而只是其中的一部分. 所以, 即使它被攻破, 也不会泄露私钥信息. 而另一部分密钥份额共享存储在 Combiner 内, 即使 Helper 与任一台 Combiner 合谋, 也不会获得 CA 的私钥.

(3) 在签名过程中的 Combiner 问题, 可通过使系统满足 Combiner 安全条件来解决(详见文[5]).

(4) 同时该系统满足原弹性 CA 方案所具备的安全性: 一个 Share Server 的泄露和被敌人掌握只能泄露其掌握的 d'_i, 因为 d'_i 是随机选择的, d'_i 与 d 没有任何关系. 所以, 一个 d'_i 不暴露秘密密钥 d 的任何信息; 通过 Share Server 到 Combiner 的广播信道也不能掌握秘密信息 d. 在广播信道上, 只有各服务器的部分签名结果能够反映密钥, 但通过部分签名结果来求秘密份额 d'_i 与 RSA 算法的困难性是相同的. 也就

是说,通过广播信道得不到关于任意一个 d'_i 的信息;在得不到任何 d'_i 的情况下,任意多个 Combiner 合谋进行攻击也只能得到最多所有组合的方程,但都无法找到密钥 d.

5 结束语

公开密钥基础设施(PKI)已成为信息安全领域研究的热点,而其中认证机构 CA 私钥的安全是 PKI 安全的核心,国内在这方面的研究方案很少.本文通过研究国内外已有的方案[2~5],对 CA 的私钥管理进行精心的设计,以提高其产生、分发及使用过程中的安全性,从而确保整个 PKI 设施的正常运转.

参考文献:

[1] Shamir A. How to share a secret[J]. Communication of the ACM, 1979, 22(9): 612- 613.

[2] Malkin M, Wu T, Boneh D. Building intrusion tolerant applications [A]. The 8th USENIX Security Symposium[C]. Washington: USENIX Association, 1999. 79- 91.

[3] P Gemme. An introduction to threshold cryptography [EB/OL]. http://citeseer.ist.psu.edu/context/515121/0.

[4] Malkin M, Wu T. Experimenting with shared RSA key generation[J]. Network and Distributed System Security, 1999, 17(1): 43- 56.

[5] 荆继武,冯登国. 一种入侵容忍的 CA 方案[J]. 软件学报, 2002, 13(8): 1417- 1422.

作者简介:

蔡永泉 副教授、博士,主要从事计算机网络协议开发、网络安全及其算法的研究. E-mail: cyq@bjut.edu.cn.

杜秋玲 在读硕士研究生,主要从事网络安全的研究.