一种入侵容忍的 CA 方案*

荆继武[1], 冯登国[1,2]

[1](中国科学院 研究生院 信息安全国家重点实验室,北京 100039);
[2](中国科学院 软件研究所 信息安全国家重点实验室,北京 100080)
E-mail: jing@is.ac.cn; isjing@nisdata.com
http://www.is.ac.cn

摘要: CA(certificate authority)是 PKI 中的关键设施.CA 的私有密钥一旦泄露,该 CA 签发的所有证书就只能全部作废.保护在线服务 CA 的私钥也就成为一个非常重要的课题.不是从保护系统或检测入侵出发来保证 CA 的安全,而是确保当少数部件被攻击或占领后,CA 系统的机密信息并没有暴露.通过将私钥分发给不同的部件,并保证任何一个在线的部件无法恢复 CA 的私钥,从而保护了 CA 私钥的保密性.

关 键 词: 入侵容忍;弹性;CA;数字签名;RSA
中图法分类号: TP309 文献标识码: A

PKI 是公开密钥基础结构的简称,它基于公开密钥密码算法.在 PKI 系统中,CA(certificate authority)又是一个域中的信任中心.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在一起,用 CA 的私钥签名后得到的数据.简单来说,一方要验证另一方的身份时可以通过两个步骤来完成.首先验证证书的签名是否正确.由于签名只能由 CA 的私钥来完成,所以只有 CA 才能够进行签名.第 2 步就是验证对方是否拥有与证书中公钥对应的私钥,如果成立,则与公钥绑定的身份数据就是对方的.

从中我们可以看到,CA 的私钥是 CA 安全的核心.保护私钥不泄露是整个 CA 域安全的基础.一般来说,CA 必须是一个在线的网络设备,特别是直接面向用户的 CA,以便自动地提供相应的证书服务.现在,很多方案中已将上级 CA(给 CA 发证书的 CA)和根 CA 设计为离线方式,以保证 PKI 结构的安全.但用户 CA 基本上无法离线工作.连网设备遭遇网络攻击是不可避免的.当一个黑客攻击一台 CA 成功时,攻击者就有可能获得机器内的资源,从而找到 CA 的私钥.这对于 PKI 系统来说是非常致命的.一个内部的雇员对系统的攻击也是应该预防的.当一个雇员完全控制其中一台设备时,也应该保证他没有获得 CA 的私钥.由于硬件错误或其他原因,包括恶意攻击,使一台或多台($<t$ 台)设备瘫痪时,应该保证整体 PKI 的基本运作的正常.

我们提出的安全弹性 CA 方案较好地解决了以上问题.

本文所描述的方法是基于 RSA 算法的 CA 的弹性方法.该方案具有以下特点:

(1) (scalability)系统很容易地扩充.当需要增加一个 CA 服务器时,只需对此服务器进行一次密钥的分发,并通知所有 Combiner.增加 Combiner 也是容易的,只需传送正确的 Combiner 的数据就可以了.所有过程都由 Distributer 来完成.

(2) (security)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.

* 收稿日期: 2001-05-31; 修改日期: 2001-08-30
基金项目: 国家重点基础研究发展规划 973 资助项目(G1999035802);国家杰出青年科学基金资助项目(60025205)
作者简介: 荆继武(1964－),男,湖南洪江人,教授,主要研究领域为信息安全;冯登国(1965－),男,陕西靖边人,博士,研究员,博士生导师,主要研究领域为信息与网络安全,编码理论与技术.

(3) (redundancy)一个或多个设备损坏不影响正常的服务.当一个 Share Server 开始计算时,无须要知道他的合作者是谁,故不需要一个合作群的指定机构(预先同步).考虑到一个 Share Server 可能损坏,指定合作者可能会因为一个错误的 Share Server 而导致一次计算的失败.

(4) (simplicity)该系统的算法和原理都非常简单.Share Server 和 Combiner 的计算量相差不大.

(5) (low cost)基于现有的商用系统构建.

1 现有的方法

现有的密钥保护的入侵容忍方案都采用了门限密码的技术,文献[1]给出了一个较为全面的介绍.针对 RSA 计算来说,最容易想到的就是将秘密密钥 d 拆成 t 个随机数之和.文献[2]描述了这样一种通用的入侵容忍方案,该方案具有简单的结构和容易证明的安全性,通过将私钥 d 分解成若干个数的和 $d=d_1+d_2+\ldots+d_t$,再将 d_i 分到第 i 个服务器中去,当需要签名时,客户机将需要的签名信息 HASH 结果 M 发送到这 t 个服务器中,各服务器将计算结果 $M_i = M^{d_i}$ 送回客户机,客户机再计算

$$S = \prod_{i=1}^{t} M^{d_i} = M^{\sum_{i=1}^{t} d_i} = M^{d},$$

就得到了需要的结果.文中通过多组 d_i 来产生冗余配置.即随机找到若干组数,如

第 1 组:$d=d_{11}+d_{12}+\ldots+d_{1t}$;

第 2 组:$d=d_{21}+d_{22}+\ldots+d_{2t}$;

…

然后将它们分到不同的服务器;每个服务器得到多个 d_{ij},但同组的数据只得到 1 个.如对于 4 个服务器,t=3 的情况,分配方案可以如下:

Server 1	Server 2	Server 3	Server 4
d_{11}	d_{12}	d_{13}	d_{13}
d_{23}	d_{21}	d_{22}	d_{23}

当客户机需要计算时,客户机选定 t 个完好的服务器,然后告诉这些服务器使用第几组参数.于是服务器就可以计算相应的部分签名了.文献[2]还描述了如何通过一种部分签名方案来检测失效的服务器等技术细节,在此不再描述.

该方案的优点明显,它具有简单的结构和很好的安全性.其不足之处有如下几点:

(1) 部分密钥的分发和管理比较困难,当增加一个服务器时,必须对每一个在线的服务器分配密钥数据;同时客户机也必须知道这样的增加.

(2) 当服务器很多时,服务器的密钥存储会迅速增加.设服务器的数量为 k,则每个服务器存储密钥的数量至少为 C_k^{t-1} 个.这里,C 表示组合.当 k=10,t=3 时,数量为 45 个.

(3) 存在必须先同步的问题.在进行计算前,必须先选定 t 个服务器.当 t 个服务器选择完成后,必须找到与这 t 个服务器匹配的数据组并通知它们.当其中有一个服务器被破坏时,必须重复从选择服务器开始的整个过程.

让客户机来选择服务器对于一个容错系统来说不是一个完美的方案.我们很容易联想到,在 Shamir 的秘密共享方案中,任意取 t 个部分密钥就能够生成秘密密钥.我们也希望在入侵容忍的 CA 设计中达到这种效果.但 Shamir 的方案中必须先恢复秘密密钥,这是我们所不希望的.我们希望在任何情况下都不恢复秘密密钥.许多文章都在讨论基于 Shamir 秘密共享的方案,文献[3,4]都给出了这样的讨论.给出一个多项式 $f(x) = \sum_{i=0}^{t-1} a_i x_i$,利用 LaGrange 插值公式,我们有

$$f(x) = \sum_{i=1}^{t} \left(f(x_i) \prod_{j=1,t}^{j\neq i} \frac{x - x_j}{x_i - x_j} \right). \tag{1}$$

任选 t 个 x_i 和 $f(x_i)$,我们就可以得到

$$a_0 = f(0) = \sum_{i=1}^{t} \left(f(x_i) \prod_{j=1,t}^{j\neq i} \frac{-x_j}{x_i - x_j} \right). \tag{2}$$

我们可以设置 a_0 为秘密密钥 d.此时,对一个 HASH 值 M 的签名计算为

$$M^d = M^{f(0)} = \prod_{i=1}^{t} M^{f(x_i)\prod_{j=1,t}^{j\neq i}\frac{-x_j}{x_i-x_j}} = \prod_{i=1}^{t} M^{b_i}, \tag{3}$$

其中

$$b_i = f(x_i) * c_i = f(x_i)\prod_{j=1,t}^{j\neq i}\frac{-x_j}{x_i-x_j}. \tag{4}$$

这样可以将秘密 d 分到 k 个服务器中去($k\leq t$).每个服务器计算 M^{b_i},然后由一个 Combiner 将结果乘起来就得到 M^d,而任何服务器都不会泄露秘密 d.由于 b_i 中有除法计算,这很容易让人想到找一个域或环 Z_v 来进行.其中,必须满足下面的条件 1.

条件 1.v 为素数,或者保证所有 x_i 构成的 t 阶 Vandermonde 矩阵的行列式的值与 v 互素.

在一般情况下,分开计算 M^{b_i} 后带来的结果就是 $\prod_{i=1}^{t} M^{b_i} = M^{d+wv}$.如何去掉 v 的影响,许多人想到了 $\Phi(N)$,因为 $M^{\Phi(N)}=1$.但选择 $v=\Phi(N)$使我们的 x_i 选取大大受限于条件 1,并且知道了某元素 o 与其对 $\Phi(N)$的逆 o^{-1},就可以求出 $\Phi(N)$.显然这是不安全的.

文献[4]提出让多项式的系数 a_i 在$\{0,L,\ldots,2L^3N^{2+e}t\}$中,其中 $L=k!$,并取 x_i 属于[1,2,…,k−1].由于所有 $f(x_i)$都能够整除 L,故 b_i 的计算去掉了求逆的操作,可以在整数中进行.该方案可以对一般的 RSA 算法进行而不需要 RSA 的强素数.由于其参数的选取受到极大的限制,带来了算法原理及安全证明的复杂性.当 b_i 由 Share Server 进行计算时,也存在着如文献[2]所提方法中的同步问题.因为 b_i 依赖于 x_i 的选取.

文献[3]中使用 RSA 强素数的方案,其保密的素数 $p=2p'+1$,$q=2q'+1$.所有的插值方程都在模 $m=p'q'$的环中进行.因为 M^{4m} 模 N 等于 1,分开计算时增加一次平方,Combiner 再分别对各个结果平方一次从而得到 $M^{4\Delta(gm+d)}$.其中 $\Delta=(k!)^2$ 而 g 为整数.该方案中 c_i 是由 Combiner 完成的,从而消除了计算前的同步问题,但带来了 Combiner 的计算难度,使 Combiner 的计算性能下降.该方案中,Combiner 必须计算:

$$W = y_{i_1}^{2\lambda c_{i_1}} y_{i_2}^{2\lambda c_{i_2}} \ldots y_{i_t}^{2\lambda c_{i_t}},$$

其中 y_i 为各 Share Server 的计算结果,$\lambda=k!$.可以看到,整个计算接近或相当于 t 次签名.Combiner 的计算量远远大于 Share Server 的计算量.

文献[3,4]都给出了较好的部分签名验证方法,这里就不再陈述.

2 弹性系统结构

我们的 CA 是以 RSA 算法为基础的 CA,其基本原理是基于文献[2]的想法.但我们消除了预先同步的问题,添加了系统连接的层次,减少了管理开销.图 1 演示了安全弹性 CA 系统的结构.下面介绍图 1 中各部件的作用.

RA Agent 是与 RA 的接口,它负责与 RA 进行保密通信,并检查 RA 的签名.同时,它也是通向外部网络的一个途径.在用户直接申请证书时,也通过此接口.

Share Server 1–n,这些服务器是用于 CA 签名的.每一个服务器都有自己的 ID 号.RA Agent 通过广播信道 $B1$ 与这些服务器相连.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和设定的算法,计算自己的忙闲因子 F_i(Task(cer)).然后向网络 $B1$ 广播自己对应此任务的 F_i(Task(cer)).当收到其他服务器关于此任务的忙闲因子 F_j(Task(cer))之后,与自己计算的忙闲因子 F_i(Task(cer))进行比较.当发现有 m 个服务器的忙闲因子小于自己的忙闲因子时,抛弃此任务.参数 m 是由安全和运行策略进行控制的,但不应该小于参数 t.

Key Distributor 平时是离线的.由安全策略和管理规定与 Share Servers 的连网时间和方式.连接的方式可以采用经典的密钥注入的方式.Key Distributor 负责更换 Share Servers 的密钥.Combiner 是最后的计算单元.

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.

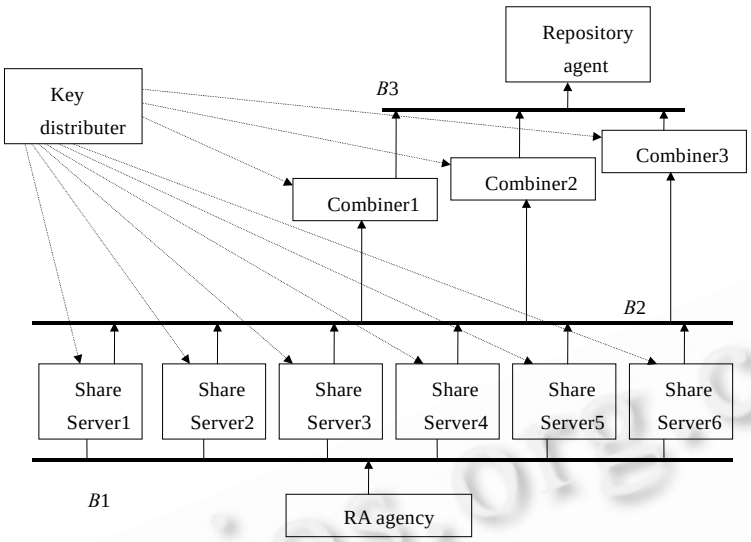

Fig.1 Structure of a resilient CA system

图 1 弹性 CA 系统结构

3 工作协议简介

我们考虑设计 k 个 Share Server 取 t 个得到签名的操作模式.CA 的私钥为 d,公开密钥为 e 和 N,$\Phi(N)=(p-1)(q-1)$也是应该保密的.Share Server i 记为 S_i.

准备:

Key Distributor 随机选定 k 个小于(d/t)的随机数 d_i,该数可以远小于 d 但有效长度不应低于 200Bits.Distributor 根据针对每个任意 t 的组合计算:

$$c_j = d - (d_{i_1} + d_{i_2} + d_{i_3} + ... + d_{i_t}),$$

并保存 $i_1,i_2,i_3,\ldots,i_t$和对应的 c_j.对 k 个 d_i 来说,共有 C_k^t 个组合,即计算 C_k^t 个 c_j.当 k=10,t=3 时,共有 120 组.将这 C_k^t 组数$(i_1,i_2,i_3,\ldots,i_t,c_j)$的部分送到一个 Combiner,而将其他组合送到其他 Combiner.其中 $i_1,i_2,i_3,\ldots,i_t$ 为 Share Server 的 ID 号.

第 1 步.RA Agent 分配任务号.每一个 RA Agent 都有自己的惟一标识 x,每一个 RA Agent 也维持一个序列号,当分发一个签名任务时,其任务序列号加 1.标识 x 组合上自身的任务序列号就构成了任务号.这个任务号是全网惟一的.

第 2 步.Share Server i 计算忙闲因子 F_i(Task(cer)).自己正在执行的任务数乘以本机的性能指标得到忙闲因子 F_i(Task(cer)).其中性能指标是在空闲状态下计算一次标准升幂的时间.

第 3 步.接收针对此任务的忙闲因子的广播,抛弃那些从同一个 IP 地址来的重复报文,并将接收到的忙闲因子按大小排序.如果机器空闲,直接进入第 5 步.

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 5 步.计算 $y_i = (\text{HASH(cer)})^{d_i}$.其中 HASH 是文摘函数.Share Server 将计算的结果连同 cer、任务号 Task(cer)和自己的代号 i 送往广播信道 $B2$.

第 6 步.Combiner 取用 t 个结果数据报.并寻找组合$(i_1,i_2,i_3,\ldots,i_t,c_j)$.如果找到,就计算

$$R = (\text{HASH(cer)})^{c_j} * \prod_{i=i_1}^{i_t} y_i,$$

R 就是应该得到的签名.如果没找到,重新取用另外 t 个结果数据报,重复第 6 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 7 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)重新计算.

第 8 步.Repository Agent 将正确的 cer 与 R 一起送往数据库存档和被查,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 Combiner 的问题

Combiner 可以安装在 Share Server 的后面,比 Share Server 更难受到攻击.但即使敌人全部掌握一个 Combiner,也不能推出秘密密钥 d.即使一个 Combiner 拥有了全部的组合,也无法求解 d.可以证明,所有具有形如 $c_j = d - (d_{i_1} + d_{i_2} + d_{i_3} + ... + d_{i_t})$ 的方程合在一起,其系数矩阵的秩为 k,而变量个数为 k+1.虽然 Combiner 从广播信道上获取信息不能推出任何 d_i,但是,当一个 Combiner 得到所有的组合后,Combiner 能够计算出任意的 d_i–d_j 的值.也就是说,他能够通过 d_i 求出 d_j.这样,一个 Combiner 就能够与一个 Share Server 合作,求出秘密 d.这是我们所不希望的.

我们通过将不同的组合分配给不同的 Combiner 来阻止 Combiner 与 ShareServer 的合谋攻击.我们提出一个 Combiner 的安全条件,即在同一个 Combiner 内,任何方程的线性组合得到的新方程,其变量的个数大于值 t.这样就保证了一个 Combiner 必须同 t 个 Share Server 合谋才能得到秘密密钥.当每个 Combiner 只有一个方程时,条件就得到了满足.这说明本方案在理论上可行的.我们进行了分析计算,对 k=5,t=3 来说,需要 8 个 Combiner 才能放进所有的组合,并且保证满足 Combiner 的安全条件.为此,可以采用每个 Share Server 多个子密钥的方案.计算表明,在每个 Share Server 两个子密钥的条件下,当取 k=6,t=3 时,只要两个 Combiner 就能够放进所有的针对 Share Server 的组合,并且满足 Combiner 的安全条件.由于 Share Server 中的密钥长度远小于签名密钥的长度,所以,增加子密钥基本不影响 Share Server 的计算性能.而对于 Combiner 来说,升幂计算只进行 1 次.由于方程个数并没有增加,故查找匹配的时间也没有太多的变化.

限于篇幅,具体的方案在此不再详述.

5 安全性分析

该系统的安全性是可以保证的.

首先是一个 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,即条件信息熵 $H(d|d_i)=H(d)$.由于 d_i 与 d_j 当 $i\neq j$ 时是独立的随机变量,所以有 $H(d|d_i,d_j)=H(d)$.即多个随机的 d_i 也不反映 d 的任何信息.理论上说,任意个 Share Server 的泄露都不泄露秘密密钥 d.

其次,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 $y_i = (\mathrm{HASH(cer)})^{d_i}$ 能够反映秘密密钥,但通过 y_i 求 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.这对于 Combiner 的弹性是有帮助的.

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.我们已经说明,所有方程合起来的秩只有 k 而变量的个数却是 k+1,所以无法找到秘密 d.当每个 Share Server 采用两个以上的子密钥时,变量的个数增加了 k 个,但方程的数量却可以不增加.这更增大了 Combiner 合谋攻击的难度.

由于系统满足 Combiner 安全条件.一个 Combiner 与少于 t 台 Share Server 合谋不能对系统构成威胁.但多台 Combiner 与多个 Share Server 合谋可能会对系统形成威胁.在满足 Combiner 安全条件时,合谋的攻击也是有条件的.合谋攻击成功的条件与对 Combiner 组合的发放算法密切相关.合理的组合发放算法能够增加合谋攻击的难度.利用 Share Server 的多个子密钥和 Combiner 的安全条件,抵制多个 Combiner 和一个 Share Server 合谋攻击是容易的.理论上说,随着单个 Share Server 子密钥的增加,变量个数会成倍增加,而放在 Combiner 的方程的个数却可以不增加.这样,单个子密钥在 Combiner 中的方程中出现的次数就会很少,极限的情况就是每个子密钥只在一个方程中出现一次.在这种极限情况下,所有的方程组合就满足 Combiner 的安全条件.也就是说,多个 Combiner 联合就相当于满足 Combiner 安全条件的一个 Combiner.

6 实现与结论

由于该方案无法使用中国剩余定理进行加速,单次签名的速度比用中国剩余定理加速时慢一倍左右.但由于方案中引入了冗余和并行操作,速度的影响是可以忽略的.整个系统的原型开发是利用同一个局域网进行的,即在原型阶段,广播信道 $B1$,$B3$ 在一个物理网段上,$B2$ 在一个独立的物理网段上.广播通信采用 UDP 协议.服务器采用 NT 系统完成.在最终应用时可以采用层次的网络结构以使整个系统更加安全.

我们采用了 5 个 Share Server 和 3 个 Combiner 的结构.每个 Share Server 有两个子密钥.采用穷举的办法进行 Combiner 的发放,在保证满足 Combiner 安全条件的前提下,只需两个 Combiner 就完全包括了所有 Share Server 组合,共 10 种.而这两个 Combiner 中还有足够的空间放其他的组合以增加冗余能力.最后,在第 1 个 Combiner 中放入了 8 个组合,第 2 个 Combiner 中放入了 11 个组合,第 3 个 Combiner 与第 1 个相同.这样,我们在满足 Combiner 安全的条件下,又保证了冗余和容错的能力.

由于篇幅的关系,实现的其他许多细节不再描述,如,服务器的部分密钥更新、如何判断损坏的服务器、怎样更新一个损坏的服务器、如何通报损坏的服务器、如何添加一台服务器等.

致谢 中国科学院研究生院的叶顶峰教授对本文的工作给予了指导,中国科学院软件研究所的张立武博士对本文的完成提出了很多有益的建议,在此一并表示感谢.

References:

[1] Gemmell, P.S. An introduction to threshold cryptography. CryptoBytes, 1977,2(7):7~12.

[2] Wu, T., Malkin, M., Boneh, D. Building intrusion-tolerant applications. In: Proceedings of the USENIX Security Symposium. 1999. 79~91.

[3] Shoup, V. Practical threshold signatures. In: Proceedings of the Eurocrypt 2000. Bruges (Brugge): Springer-Verlag, 2000. 207~220.

[4] Frankel, Y., Gemmell, P., MacKenzie, P.D., *et al*. Optimal-Resilience proactive public-key cryptosystems. In: IEEE Symposium on Foundations of Computer Science. 1997. 384~393.

An Intrusion Tolerant CA Scheme*

JING Ji-wu[1], FENG Deng-guo[1,2]

[1](*State Key Laboratory of Information Security*, *Graduate School*, *The Chinese Academy of Sciences*, *Beijing* 100039, *China*);

[2](*State Key Laboratory of Information Security*, *Institute of Software*, *The Chinese Academy of Sciences*, *Beijing* 100080, *China*)

E-mail: jing@is.ac.cn; isjing@nisdata.com

http://www.is.ac.cn

Abstract: CA (certificate authority) is a critical component in PKI. When the private key of a CA is compromised, all the certificates issued by that CA should be revoked. Keeping the private key secret while providing service on line is very important for a CA. Rather than prevent intrusions or detect them after the fact, the project ensures that the compromise of a few system components does not compromise the private key of the CA. The private key is protected by distributing it across a few servers. The private key is never reconstructed at a single online location.

Key words: intrusion tolerant; resilience; CA (certificate authority); digital signature; RSA

* Received May 31, 2001; accepted August 30, 2001

Supported by the National Grand Fundamental Research 973 Program of China under Grant No.G1999035802; the Outstanding Youth Foundation of the National Science of China under Grant No.60025205