Java卡的可信代码装载机制设计

胥怡心 张其善 刘建伟

(北京航空航天大学 电子信息工程学院, 北京 100191)

摘 要: 对满足可信计算平台(TCP, Trusted Computing Platform)需求的代码管理问题,提出了以Java卡为核心,以代码签名技术为主要手段,以卡内的安全域(security domain)作为各个软件权威(Java卡设备发行方和应用提供方)安全策略执行者的可信代码管理框架.新机制支持设备发行方软件的装载和更新,完善了应用提供方安全域的装载和更新流程,并对下载命令数据结构进行了扩充.解决了在复杂应用环境中,多个软件权威相互独立的限制条件下,代码的免人工广播式发布问题,扩展了传统Java卡软件装载模式,提高了系统的安全性,为使用Java卡平台进行可信计算提供保障.

关 键 词: Java卡; 可信计算平台; 代码下载; 代码签名

中图分类号: TP 316

文献标识码: A 文 章 编 号: 1001-5965(2009)01-0032-04

Design of trusted Java card code bading mechanism

Xu Yixi Zhang Qishan Liu Jianwei

(School of Electronics and Information Engineering, Beijing University of Aeronautics and Astronautics, Beijing 100191, China)

Abstract Based on the requirements of the trusted computing platform (TCP) program code management, a trusted code management framework was proposed. The Java card technique was the core of the framework. The code signature technique was adopted for the program bading authentication while the on-card security domains were created as the off-card software authorities' security policies executors. The novel mechanism supported the installation and hot update of the device issuer software, promoted the installation and update procedure of the application provider software, and expanded the code-loading command data structure. The framework implemented the unmanual code broadcast distribution with the limitation of mutual independency of all software authorities under the hostile environment. This advanced Java card software loading mode improves the system's security, and enables the trusted computing on the Java card platform.

Key words Java card; trusted computing platform; code-loading; code signature

Java卡是斯伦贝谢于1996年推出的智能卡平台,它通过Java虚拟机技术屏蔽智能卡硬件细节,并在卡内提供应用编程接口(API, Application Programming Interface),使得开发人员无需了解智能卡硬件和专用技术就可以进行相关应用的开发. Java卡支持多应用,所有的应用可以在发卡后下载安装到卡内,这比传统智能卡软件开发更具灵活性.

传统智能卡芯片操作系统(COS, Chip Operating System)是在发卡前写入卡内的,代码的安全性比较高,但灵活性很差,只适合单一的应用系统. Java卡虽然在代码装载机制上更灵活一些,但也不能满足复杂的应用环境以及可信计算对代码管理的要求.如何在发卡后对发卡方程序进行更新?如何让多个软件权威(包括发卡方和应用提供者)相互信任?如何在应用代码更新时处理应用相关的敏感数据?虽然这些问题在现有的Java卡应用中并不突出,因而还没有得到重视,但随着对分布式计算安全性要求的提高以及可信计算平台研究的发展,使用Java卡技术的设备也需要提

收稿日期: 2008-01-23

作者简介: 胥怡心(1978-),男,四川剑阁人,博士生, xyx buaa@gmail.com.

供更多的安全服务. 将 Java卡系统构建成便携式的可信计算平台, 就是本文要设计可信的代码装载机制的动机.

论文的第 1节介绍可信计算平台研究现状以及对软件设计的要求; 第 2节介绍 Java卡设备能成为可信计算平台的原因; 第 3节分析改进的 Java卡软件结构和可信的代码装载机制; 最后得出 Java卡完全可以满足可信计算应用对代码安全的需要.

1 可信计算平台

1.1 可信计算平台的现状

近几年来, 以可信计算平台联盟 /可信计算组织 (TCPA/TCG, Trusted Computing Platform Alliance/Trust Computing Group)、微软公司的下一代安全计算基 (NGSCB, Next-Generation Secure Computing Base)和英特尔公司的 LaGrande等研究机构为代表的工业界不断努力, 大大推动了“可信计算平台”概念的发展. 这些平台通过硬件和软件技术相结合, 尝试针对不同种类的敌手, 解决多方信任问题. 当前关于可信计算平台的讨论焦点主要集中在 TCPA/TCG规范的发展、可信计算平台的设计、以及潜在社会影响等方面[1-4].

1.2 可信计算平台对软件开发和部署的需求

可信计算平台除了对硬件设备有很高的安全要求以外, 对软件设计也有以下需求[5]:

1) 必须容易编程;

2) 必须有通用操作系统, 并带有调试支持;

3) 必须支持大量的开发, 并有发布应用程序与操作系统代码的权威;

4) 具有安全的代码管理机制, 允许操作系统和应用来自不同的, 甚至是相互不信任的设计者, 而不会互相干扰;

5) 在敌对环境中, 没有可信的安全人员在场的情况下, 更新应该尽可能远程执行.

2 Java卡平台

2.1 Java卡技术

Java智能卡技术是 Java技术和智能卡技术的有机结合体, 是在智能卡硬件系统的基础之上在卡片内通过软件构建的一个支持 Java程序下载、安装和运行的软、硬件系统. SUN公司发布了有关 Java卡技术的规范[6], 并为 Java卡设备的软件开发提供开发工具包.

2.2 Java卡设备硬件结构

将 Java卡芯片和扩展存储器安全封装在一起, 增加窜改检测电路, 以通用串行总线为外部接口, 这样的基于 Java卡技术的便携设备 (以下统称为设备)就构成了可信计算的物理平台, 它具备如下安全特性: ①窜改检测功能, 阻止暴力打开设备获取存储内容; ②受保护的安全存储区, 阻止对敏感数据和代码进行非授权访问; ③硬件密码算法协处理器, 提供对称加密算法 (如 DES, Data Encryption Standard)和公钥密码算法 (如椭圆曲线公钥算法)支持.

设备的硬件安全特性可以基本满足可信计算平台的物理完全要求. 它所提供的公钥密码算法是实现可信代码装载的基础.

3 平台软件体系结构

可信计算平台要具有安全的多应用管理机制, 允许操作系统和应用来自不同的, 甚至是相互不信任的设计者, 而不会互相干扰. Java卡技术的应用 (Applet)防火墙机制虽然保证了安装后各个应用之间相互隔离, 防止应用间非授权访问, 但是为了保证下载的代码是真实完整的, 还需要对代码装载机制进行改善才能满足可信计算平台的代码安全需求.

3.1 平台的软件结构

将设备上的软件结构进行划分, 如图 1所示.

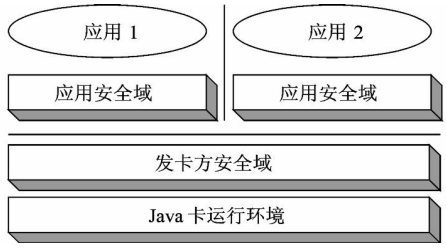

图 1　Java卡设备软件结构

1) Java卡运行环境: 包括 Java卡本地方法集、Java卡虚拟机、应用编程接口, 它提供 Java字节码解释、命令分派, 应用选择、下载和安装等功能.

2) 安全域: 为上层应用提供密码服务, 每个安全域代表一个应用提供者, 是应用提供者方的安全策略在卡中的体现. 其中设备发行方安全域必须在设备发行前由发行方创建, 它将执行发卡后的安全策略.

3) 应用: 其中每个应用都必须与同属一个应用提供方的安全域相关联, 使用该安全域的安全服务.

3.2 可信代码的安全下载管理

代码下载管理是基于如下限制性原则[5]:

1) 免人工广播式发布. 设备发行方和应用提供者没有必要定义好设备的代码加载, 并且代码能够通过公共的网络发送.

2) 可更新. 可更新已安装在设备上的代码.

3) 互不信任, 独立的软件提供者. 在特定设备中, 各个软件层 (如 Java卡虚拟机层、Applet层等)可能由互不信任的软件权威控制, 这些权威对软件的更新可能有不同的意见和策略.

4) 敌对环境. 不假定设备位于可信的环境, 或者有可信的安全管理员.

上述限制原则要比设备传统的应用环境更复杂, 在现有规范[7]中, 虽然规定了应用的下载机制, 但对至关重要的安全域下载和安装没有明确说明. 可信代码安全下载管理就是对原有下载机制的改善, 支持设备发行方软件的装载和更新, 完善了应用提供方安全域的装载和更新流程, 并对下载命令数据结构进行了修改以满足更新策略.

3.2.1 代码装载管理策略

代码装载管理使用代码签名技术, 强调安全域作为各个软件权威 (发行方和应用提供方)的安全策略执行者的地位, 其基本规则如下:

1) 所有装载操作都由安全域进行检查验证;

2) 应用提供方安全域的装载必须由发行方和应用提供方联署签名, 以增强设备的安全性;

3) 在装载操作中增加目标设备标识和软件配置标识数据, 使软件权威能够确信命令仅在可信环境下执行;

4) 权威通过验证签名收据来确定代码已经被加载;

5) 在装载应用提供方安全域的命令中增加的信任参数数据项, 用来标识应用提供方对发行方软件更新的看法, 如总是信任发行方软件更新或总是不信任发行方软件更新, 以此避免新的发行方软件内在缺陷或恶意后门造成应用秘密信息泄露的危险.

考虑到设备有限的计算能力, 代码签名技术采用 SHA-1算法计算数据的哈希 (HASH)值, 使用有限域 GF(2^n)上的椭圆曲线数字签名算法进行代码签名验证.

代码装载命令中使用的数据定义如下:

R: 随机数, 用于保证命令的新鲜性;

M: 目标数据, 包括目标设备标识和目标设备软件配置标识;

C: 代码;

H: 命令数据的 HASH 值;

S: 对 HASH 值的数字签名;

I: 发行方;

A: 应用提供方;

E: Java卡设备;

K_{pr}: 私钥;

K_{pu}: 公钥;

SHA-1( ): SHA-1哈什计算;

$Sig_{K_{pr}}(\ )$: 使用私钥 K_{pr}计算数字签名.

3.2.2 发行方软件的装载和更新

发行方软件即为图 1中的 Java卡运行环境层软件以及发行方安全域. 其中 Java卡运行环境层软件根据发行方的策略以及芯片存储器容量决定是放置到只读存储器 ROM (Read-Only Memory)中不可更新, 还是放置在可重写 Flash中允许更新; 发行方安全域作为发行方的安全策略体现, 装载到 Flash中, 允许更新. 另外, 发行方的公钥 $K_{pu}(I)$也要在发行前装载到发行方安全域中. 发行方将在设备内产生密钥对 $K_{pr}(E)$和$K_{pu}(E)$, 保存到发行方安全域中, 同时对外公开公钥, 由发行方证书机构 CA (Certification Authority)在此基础上添加设备身份标识信息和软件配置信息等, 为这个设备签发证书, 并将证书返回给设备由发行方安全域控制使用.

当设备发行后需要更新 Java卡运行环境层软件, 装载命令的数据为

$$D_{CMD} = R \,||\, M \,||\, C \,||\, H \,||\, S$$

其中

$$H = \text{SHA-1}(R \,||\, M \,||\, C)$$

$$S = Sig_{K_{pr}(I)}(H)$$

设备接收到下载命令, 将完成如下操作: ①检查自己的标识是否属于目标数据标识的集合; ②使用 $K_{pu}(I)$对命令签名进行验证; ③校验 HASH 值, 检查代码完整性; ④使用设备私钥签发签名收据, 即 $R_{ec} = Sig_{K_{pr}(E)}(R)$. 发行方安全域的更新, 标志着发行方安全策略的变化, 所以下载命令数据要包括新的发行方公钥, 即

$$D_{CMD} = R \,||\, M \,|\, C \,||\, K_{pu}'(I) \,||\, H \,||\, S$$

其中

$$H = \text{SHA-1}(R \,||\, M \,||\, C \,||\, K_{pu}'(I))$$

$$S = Sig_{K_{pr}(I)}(H)$$

发行方安全域更新后, 原来其管理的密钥、证书都将作废, 需要重新生成或由外部重新导入, 设备内的应用也需要重新下载.

3.2.3 应用提供方安全域 (AP. SD)装载

AP. SD是应用提供方的安全策略在设备内的体现. 为了防止与发行方的安全策略发生冲突, 应用提供方的安全策略必须接收发行方检查, 并对装载命令进行联署签名. 在 AP. SD装载之前, 必须由发行方为 AP. SD装载公钥 ($K_{1pu}(A)$)签发

证书导入设备,由发行方安全域管理. 该公钥专门用于验证 AP. SD的装载命令. AP. SD的装载流程如图 2所示.

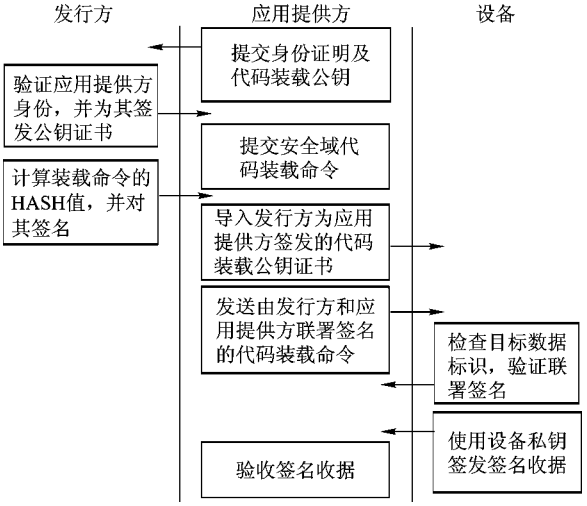

图 2　应用提供方安全域装载流程

图 2中应用提供方递交给发行方的装载命令数据为

$$D_{CMD} = R \| M \| CE \| K_{2pu}(A) \| H \| S$$

其中 $K_{2pu}(A)$表示应用提供方用于装载 Applet的公钥(装载 AP. SD的密钥对可以与装载 Applet的密钥对相同),而

$$H = SHA\text{-}1(R \| M \| C \| K_{2pu}(A))$$

$$S = Sig_{K_{1pr}(A)}(H)$$

由发行方返回给应用提供方的装载命令增加了发行方对原命令的签名,即

$$D_{CMD}' = Sig_{K_{pr}(I)}(H_{CMD})$$

其中　　$H_{CMD} = SHA\text{-}1(D_{CMD})$

而设备返回的签名收据与更新发行方软件时相同.

设备完成 AP. SD装载后,公钥 $K_{2pu}(A)$保存到 AP. SD中,以备验证 Applet装载命令使用. 同时设备将生成 AP. SD密钥对,私钥保存到 AP. SD中,公钥对外公布. 该密钥对是用于对 Applet装载命令签发签名收据的.

3.2.4　应用装载

每个应用都与和它同属一个应用提供方的 AP. SD关联,并由 AP. SD控制其装载过程,与发行方无关.

装载命令数据为

$$D_{CMD} = R \| M \| C \| H \| S$$

其中　　$H = SHA\text{-}1(R \| M \| C)$

$$S = Sig_{K_{2pr}(A)}(H)$$

设备接收到下载命令,将由 AP. SD完成如下操作: ①检查自己的标识是否属于目标数据标识的集合; ②使用应用提供方的应用装载公钥对命令签名进行验证; ③校验 HASH值,检查代码完整性; ④使用 AP. SD私钥签发签名收据.

4　结　论

本文提出的可信代码装载机制根据可信计算平台的软件设计原则,对 Java卡技术中的代码装载机制进行了补充和完善,增加了对发行方软件更新的支持,强化了安全域装载的认证力度,使用各个软件权威装载到卡中的安全域集中管理秘密数据和证书并执行权威的安全策略. 改进后的 Java卡代码装载机制,解决了论文开始提出的发行方软件更新、软件权威间的信任及应用代码与应用敏感数据分离管理这三个问题. 结合 Java卡设备本身具备的高安全性特点和应用防火墙机制,这种可信代码装载机制可满足可信计算平台对代码安全性的要求,这是进一步将 Java卡设备构建成便携式可信计算平台的研究基础.

参考文献 (References)

[1] Kent S. Protecting externally supplied software in small computers [D]. Massachusetts, USA: Massachusetts Institute of Technology, 1980

[2] Whit S R, Comerford L D. ABYSS: a trusted architecture for software protection[C] // Proceedings of the IEEE Symposium on Security and Privacy. New York, USA: IEEE Computer Society Press, 1987. 38-51

[3] White S, Weingart S H, Arnold W, et al. Introduction to the citadel architecture: security in physically exposed environment [R]. RC16672, 1991

[4] Yee B S. Using secure coprocessors [D]. Pittsburgh, USA: Department of Computer Science, Carnegie Mellon University, 1994

[5] Sean W S. 可信计算平台: 设计与应用 [M]. 冯登国, 徐震, 张立武译. 北京: 清华大学出版社, 2006: 54-55
Sean W S. Trusted computing platforms: design and applications [M]. Translated by Feng Dengguo, Xu Zhen and Zhang Liwu. Beijing: Tsinghua University Press, 2006: 54-55 (in Chinese)

[6] Sun Microsystems, Inc. Java card platform specification 2.2.2 [EB/OL]. New York, U.S: Sun Microsystems, Inc, 2005. http://www.javasoft.com

[7] Global Platform, Inc. Global platform card specification 2.1 [EB/OL]. Readwood, U.S: GlobalPlatform, Inc, 2001. http://www.globalplatform.org

(责任编辑: 娄　嘉)