计算机科学 2002Vol. 29№. 10

PCC 技术在移动 Agent 系统安全中的应用初探

Some Research on Proof-Carrying Code in Mobile Agent Security

冯扬悦　陶先平　吕　建

（南京大学计算机软件新技术国家重点实验室　南京 210093）

Abstract　Proof-carrying code (PCC for short)is a software mechanism that allows a host system to determine with certainty that it is safe to execute a program supplied by an untrusted source. Unlike traditional cryptography-based strategy, PCC helps a code consumer establish trust towards the code itself but not necessarily to its producer, thus it is useful for language interoperability, OS kernel and mobile code security etc. This article emphasizes on PCC as one of the technologies for the latter, introduces its primary ideas and implementing issues, and tries to exploit its application in mobile agent security.

Keywords　Mobile code security, PCC, Safety policy, Safety proof, Proof validation, Mogent

1　引言

移动代码指可以在一台或多台远程主机上而不仅限于在源主机上运行的程序。从 Java applet、ActiveX 程序直至 90 年代中期提出的移动 Agent 技术均属于这一范畴。其思想是化远程调用为本地执行，从而具有减少网络负载、避免网络延时、动态适应、异步自主执行、支持异构体系、健壮性和容错性等优点。可以说，代码的可移动性是对传统非分布环境中可编程性的很大扩展。然而，高效与灵活的增长并非不无代价：分布式环境中代码的移动使安全威胁增多，安全保护更为复杂，安全课题也更为重要。

图 1 是对移动代码系统的安全课题的内容概览。包含的内容分为两类：保护远程主机的环境和资源不受移动代码的攻击；保护移动代码及其数据不受恶意主机摧毁和未授权修改。近年来，对主机的保护已有较为透彻的研究，主要从两方面着手：移动代码下层基础设施的增强和移动代码本身语义的验证。很多成熟的技术业已出现并付诸应用，例如沙盒技术、数字签名、访问控制、代码验证，以及设置路径历史、使用解释语言、进行状态评估等等。相比之下，由于运行环境对移动代码具有完全的控制，对移动代码本身的保护要比保护主机困难得多，研究起步也较晚。按照移动程序包含的信息，可将问题分为代码保护和数据保护两部分；按照保护的目的，可分为隐私性维持与完整性维持。目前尚无成熟的成果，已有的一些方法包括局部结果封装、相互记录路径、执行跟踪、环境钥匙、加密函数、混淆代码等等。

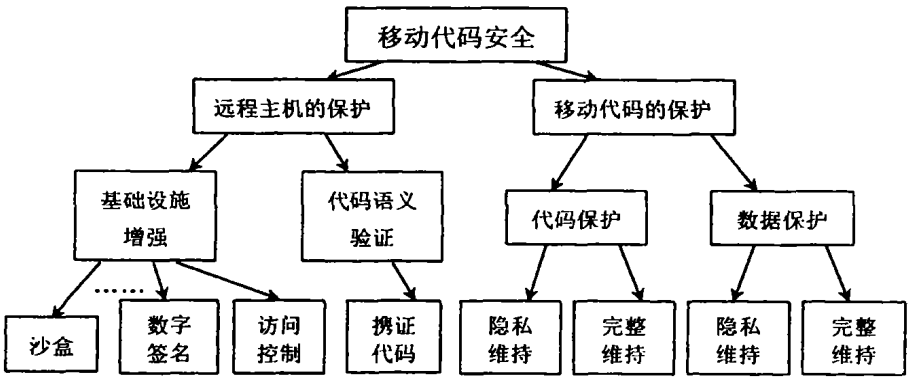

图 1　移动代码系统的安全

我们的注意力集中在代码语义的验证，它是通过对移动代码的语义（结构或行为）的分析提供更高层安全保障的一种方法。这方面所取得的较好结果是携证代码，或称 PCC (Proof-Carrying Code)的研究，其肇始于卡内基·梅隆大学的 G. C. Necula 和 P. Lee 等人，是将形式语义学、类型系统、程序验证和逻辑框架理论有机结合的一种基于程序设计语言的安全解决方案，在移动代码安全领域具有相当意义。下文我们将介绍它产生的动因、技术的特点以及系统的实现，并研究它在移动 Agent 系统安全中的应用。

2　携证代码

2.1　动因

携证代码(PCC)初步的概念，顾名思义，即是附带有对本身安全性之证明的代码；整体含义上则指生成和使用这种代码的整个体系所构成的软件框架。可用如下几点作一描述：

场景：存在代码执行供求关系的两方，即代码提供者和代码执行者，前者产生的代码需要在后者处执行。

目的：保护接收代码执行的一方不受提供代码的一方之伤害。

方法：欲执行代码的一方预先制定安全策略，代码提供者据此赋予代码一个安全证明以说明其遵守安全守则；再把该证明送至代码执行者带的证明确认器；证明被确认为对的，则说明真的符合安全需求，方可将此代码交付执行。

PCC 机制的产生，缘于开放与安全的冲突与权衡。一个

系统出于功能或性能的考虑会希望开放，但开放之后安全即成为关键的问题。例如，高级程序设计语言的类型结构是良定的，但为扩充程序能力而将其与某些低级语言如 C 或汇编同用时，类型安全就无法保证。又如，为提高运行功效，操作系统内核可能会将低端地址的控制权限开放给某些应用，但这样后者可对其执行环境为所欲为，甚至造成系统瘫痪，而限制此类应用或使用昂贵防火墙都非可取之道。在 Internet 环境下，这种冲突更为激化：共享的网络使得一台宿主机可以允许任意外来代码在其上动态地安装和运行以灵活地访问内部资源，可是如何保证该外来代码不会破坏宿主机？移动代码的出现更加剧了这对矛盾，随着移动 Agent 技术的发展，在异构的网络环境中可能出现如下计算模式：巨量的跨域的匿名 A-gent 向开放的宿主机发送巨量的移动代码对象，这些代码可能穿过路由器，在宿主机上安装并运行。随即可能带来的也是大量与安全相关的问题：诸如宿主机如何保证这段代码的执行不会损坏自己的数据结构、不会占用太多资源、或不会影响整机的性能？综上所述，需要一种既利于开放又保障安全的机制。

以加解密为核心的密码学（如访问控制中的认证口令、安全协议中的加密算法、数字签名等）是信息安全技术的原理之一，大大有助于构建封闭系统的安全体系，但在移动代码安全研究中，基于密码学的安全机制于许多方面有所欠缺。首先，密码学依赖于个人信任，只能保证外来代码确实由某个可信赖的对方提供，而不能保证这段代码本身无害，即它提供的是一种“对人不对事”的信任，然而“人”可能犯无意或故意的错误，甚至被伪造。其次，对某人身份认证或签名验证的前提是必须存在其先前的信息，然而在动态的环境中遇到的可能是大量没有先前历史记录的陌生或匿名访客。再次，密码策略的结果往往只是“可”或“不可”，而对代码是否满足某些特定的安全特质（安全特质，指的是宿主机需要判断外来移动代码是否满足的一组性质，可以是有关该移动代码行为的断言，如只能对集 X 操作，至多能对集 Y 操作 w 次等等；也可以是有关可信度的问题，如该程序由 X 创造，可由宣告具有性质 Z 的 Y 检查等等。）则无法加以刻画。

与基于密码学的方法相比，PCC 可谓建立“对事不对人”的信任：不是本方向外来方索要口令或密钥，而是外来方主动向本方提供出证明，亦即就事论事地对外来程序论证安全性，只要它满足事先规定的安全特质就可以运行，这种脱离了个人的信赖是可靠的。其次，并不关心外来者以前的信息、真实的身份、程序的来源，而只考虑本次访问的可靠性，使社会关系处理简单化。再次，安全策略可定制，故可要求外来方满足不同方面不同层次的安全特质。总之，较之于传统的基于密码学的安全手段，PCC 技术更为适应开放目标下的安全需求。

2.2 框架

就静态结构而言，携证代码生成和使用的典型系统框架如图 2 所示。

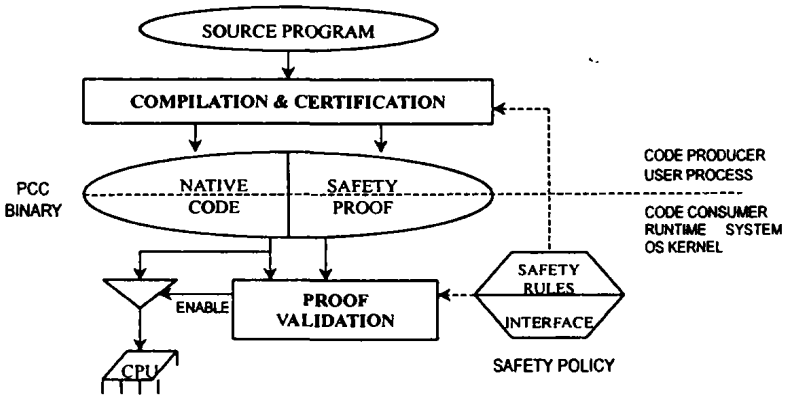

图 2 携证代码系统框架

整个体系包括安全策略、编译与证明、证明确认三个主要模块及相关控制成分，围绕的核心是由代码执行者定义并公布的安全策略，通过它精确地描述在何种情形下可认为外来程序是安全的。其包括两个部分：安全守则和接口，前者描述了所有的授权操作及相应安全的前后置条件；后者描述代码执行者与外来程序的调用约定，即调用前后的不变式。与类型理论相比拟，安全守则对应于类型规则，接口对应于模块编译时参照的型构。

就动态流程而言，PCC 二进制码的生存周期经历三阶段：证明，确认与执行。

1）*证明阶段*：代码提供者通过编译（或装配）对源程序生成一个“本程序遵守安全策略”的证明。一般说来，基本的方法是根据安全策略给出的规格说明作程序验证。这个证明采用合适的格式，连同源代码共同构成 PCC 二进制码，送交代码执行者。

2）*确认阶段*：代码执行者将源代码暂存，将证明送往证明确认器。通过一个快速直接、实现坚固、安全策略可信的算法进行确认。确认工作是离线的，且只需进行一次。

3）*执行阶段*：如果证明确认为正确，暂存的源代码即可送交 CPU 执行。可以执行多次而不再需要检测，因为上一阶段已保证了此代码不会违背安全策略。

2.3 实现

任何对 PCC 的实现至少需要四个部分：一个用以刻画安全策略的形式的规格说明语言；一个外来代码所用语言的形式语义（通常是一个将程序关联到规格说明的逻辑体系）；一个用以表达证明的语言；以及一个用以确认证明的算法。

在 Necular 与 Lee 所作的实现中，外来代码是类似汇编语言的 DEC Alpha 处理器机器码，上述四部分分别选取为：

1）*规格说明语言*：采用扩展的一阶谓词演算逻辑，增加了与应用安全需求有关的公式及推导规则。

2）*将程序关联到规约的逻辑系统*：目标是将一段代码转换成一个逻辑公式，这段代码是否具有特定安全属性就对应于这个公式是否永真；即需要一个程序的逻辑语义产生器，实现程序世界到逻辑世界本质不变的转换。采用的是 Floyd 验证条件生成器（VCG），它是一个函数，将安全需求与一段程序同时输入，输出一个相关的谓词公式（VC），该程序是否满

足安全需求由这个谓词的真假性所反映，从而把验证程序是否具有安全属性转换为证明谓词是否为真。

3）证明的格式：采用爱丁堡逻辑框架(LF)对证明编码，其本质上是带类型的λ演算，即根据 Curry-Howard 同态，把谓词公式视作类型，把证明视作表达式。

4）证明的确认：由于采用 LF 对证明编码，则验证一个证明即是对一个表达式作 LF 的类型检查，所用算法应简单高效，并独立于特定的安全策略和使用的逻辑。

进一步考察安全策略，它由安全守则和接口组成，细言之，安全策略由如下三部分组成：

1）Floyd 验证条件生成器，它由前后置条件和待证明代码产生一个一阶谓词公式，称为安全谓词；

2）证明系统，是一组公理的集合，用于证明安全谓词；

3）前置条件和后置条件（实际运算中往往固定其中一者以考察另一者，例如设后置条件为真，考察相应前置条件。）集，分别刻画执行者调用 PCC 二进制码前后必须成立的状态条件，即对安全需求的语义描述。

实验中，系统构建过程为：首先定义一个抽象机以刻画 DEC Alpha 机器码的操作语义；由此给出了安全谓词、前后置条件的描述语法，包括表达式和类型，可分为一阶逻辑语法和应用相关语法；其次给出形式证明系统的核心——公理集，包括算术公理与类型公理；最后定义计算安全谓词的 VCG。实例运行过程为：将一个外来程序与前后置条件一起输入 VCG，生成一个安全谓词，使用公理对其自动或人工证明；用 LF 对此证明编码后送交类型检查器，返回结果决定上述程序是否可在本机安全运行。

整个 PCC 系统的实现是成功的，并在若干实例上表现出了较好的性能。它的正确性，包括公理系统的坚固性和 LF 能力的充足性都得到了形式化的证明。

3 应用初探

3.1 研究价值

我们初步研究了 PCC 技术在一个移动 Agent 系统——Mogent 系统安全模块中的应用。Mogent 系统是由南京大学计算机软件新技术国家重点实验室自行研制的移动 Agent 系统，以 Internet 为基础网络环境，以 Java 语言及其环境为基本的语言支撑。其中的安全子系统的设计目标是系统性、保密性、完整性和可用性，目前采用了安全传输通道、基于信任传递的授权模型和电子货币等机制。我们仍希望能对其进一步完善，如通过形式化提高安全程度等。

首当探讨的是研究的价值：是否存在一个典型应用例，使得 PCC 使用的性价比最高。因为 PCC 的劣势在于实现复杂，运行效率低，涉及自动生成等困难问题，代价较大。我们认为，同时符合以下两个条件的情形 PCC 较为适用：一、外来代码的执行者具有重要的私有核心部分，它有可能暴露给外界而导致不安全；二、执行者面对的代码提供者是一个无安全感或不值得信任的群体，而非已经事先具有可信赖背景的熟知。可以说，私有核心越紧要，交往对象越不可信或不可知，就越适于使用 PCC。

在 Mogent 系统中，我们亦从保护站点和保护 Agent 两个方面考虑安全问题，它们分别面临的各种安全威胁细分之后如表 1 所示。

一方面，由于站点存放大量核心数据，并与众多陌生或匿名对象进行交互，符合上述两个条件，较为适合应用 PCC 技术，从而我们的重点定位在运用 PCC 保护站点。另一方面，虽然对 Agent 的保护要比对主机的保护困难得多，有的学者甚至认为这是不可能有效解决的问题，但这项研究是必要的：首先从安全漏洞角度，存在主机或其他 Agent 恶意干扰它或篡改其数据的可能；其次从应用需求角度，Agent 和主机概念上是对等的，即资源的供求双方可能互换，因而有保护的需要；此外从系统角度这也是安全的另一方面课题，因此我们也尝试借鉴 PCC 的思想进行研究。

表 1 移动 Agent 系统面临的安全威胁

保护对象	攻击源	安全威胁
主机平台	本地来访 Agent	伪装 Agent 的攻击；DoS 攻击；未授权访问
	外部实体	伪装；未授权访问；DoS 攻击；窃取复制并重演
移动 Agent	其他 Agent	伪装 Agent 的攻击；DoS 攻击；抵赖；未授权访问
	主机平台	伪装平台的攻击；DoS 攻击；窃听；修改

3.2 保护主机

目前 Mogent 系统中安全部分的设计和实现主要是围绕保护站点不受恶意 Agent 攻击，总体目标是维持保密性、完整性和可用性，由六大块及一些辅助设施组成：安全传输通道、密钥管理、密写操作、电子银行和电子货币、授权机制、访问控制。整个安全体系以密钥管理和密写操作为中心，通过授权与访问控制限制对站点资源的获取，通信在 TCP 传输层之上一个类似 SSL 的可认证、保密的信道中完成，可使用付费方式以抵御 DoS 攻击。

移动 Agent 一次旅行会经历多台主机，途中有可能被恶意主机改动过，故在服务主机的授权模型中不仅要考虑与来访 Agent 的直接信任关系，还要考虑与它路径中所经过的所有主机的间接信任关系。这种不仅基于用户信任关系、还基于访问历史的授权模型存在着一些缺陷：首先，Agent 访问的成功度与效率降低：如果 Agent 路径上曾经访问一个被服务器认为不可靠的站点，本次旅行就整个被认为是不可信赖的，拒绝为之服务，这大大影响了 Agent 工作的效率和主机服务的可用性；其次，就 Agent 自身的隐私而言，访问授权也需要保护一个 Agent 的隐私，Agent 在到达一个主机前去过什么地方属于其隐私信息，不应透露给本地主机；再次，路径历史的数据完整性也必须保证，目前采用的是联锁机制，即每次每表项中记录本节点地址和下一节点地址，但只能实现部分防篡改；另外服务器端控制的复杂程度高、开销大也是实现中的一个问题。

而 PCC 技术可以解决这种授权模型的问题。服务器可以制定一系列安全策略，来访的 Agent 在事先知道这些安全需求的前提下预先对代码做一个证明，迁移至该站点时提交证明，站点使用快速可靠的算法确认该证明为真，才同意 Agent 使用其资源。这种方式的优点在于：首先，服务器无需担心 Agent 在迁移途中曾被恶意主机窜改，只要提交的证明能通过验证，即说明这段代码可以安全地执行。这种情况下主要应考虑证明本身在 Agent 迁移途中的隐私性和完整性保护。其次，大工作量的证明工作交给访问者，主机只需相对简单地确认，减轻了负担与复杂度。

图 3 是 PCC 技术应用于 Mogent 系统中的设计构架，值

得注意的是PCC并不排斥原先的授权机制。从系统的功效与灵活性考虑，允许两种安全机制并存：对某些不可信的来访Agent或比较脆弱的资源使用PCC；而对于某些可信赖的对方的Agent（例如假定从某一特定站点来的Agent均认为非恶意的），或者访问的站点资源具有较好的坚固性，可以采取原先的授权与访问控制。

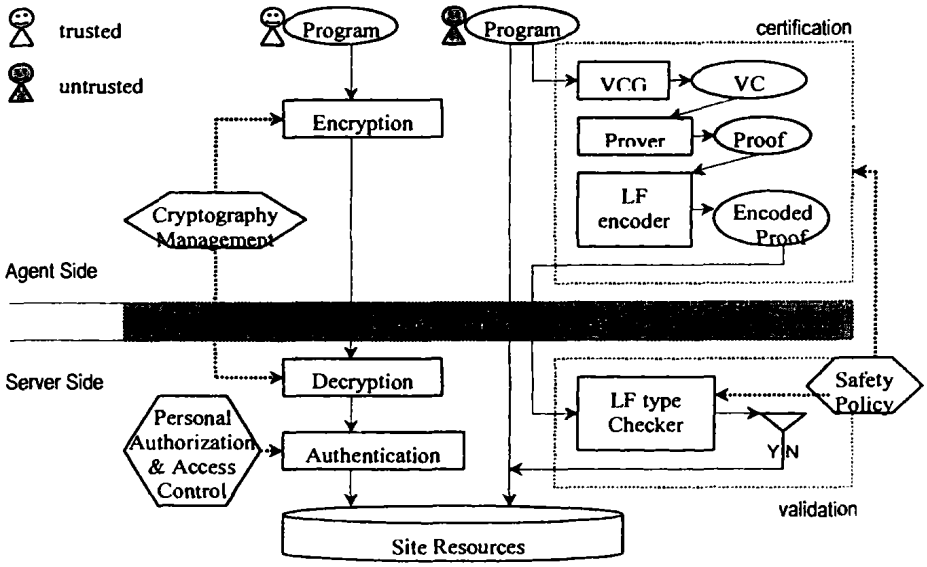

图3 用PCC技术保护站点

深入研究PCC与访问授权机制的并存，我们认为其结合点在于彼此存在某种互补的关系。传统的授权机制在一个封闭的环境中可以很好地运作，因为授权模型可以做到相当细的粒度，可以实现比较好的控制；而PCC机制更有利于开放的环境。于是，如果通过事先的身份认证能发现来访Agent具有一个确定的已知身份，则可以采用传统的方法；反之，此Agent属于陌生范畴，此时可以看其是否带有PCC证明：如果有的话，进入PCC框架，如果既没有确定身份又没有对自身的证明，则只能归入不可信的类别，仅具有最低的权限。图4显示了这样一个容许两种安全防护机制并存的主机决策判定框架，在这个决策模型中，进一步把“可信”与“不可信”取代为四个概念：经过第一步的身份认证后出来的有两种结果“经授权的”与“未知身份的”，后者可区分为“可以进行PCC检验的”（亦即带有合法的PCC证明）与“不可信的”（既无身份又无证明），将可以由PCC检验的程序输入PCC模块，得到的结果可能是“可信的”（证明通过确认）或“失败的”（亦即不符合安全策略的）。对于“经授权的”与“可信的”都有很明确的资源访问策略，对“不可信的”与“失败的”只能有最低的访问级别。此时有一个问题即两种安全机制的冲突：一个Agent可能同时是“经授权的”和“失败的”，到底以谁为主线？这并非两种机制孰是孰非的问题，而是在不同的环境下会有不同的要求，甚至可以作为一个选项留给主机用户决定。

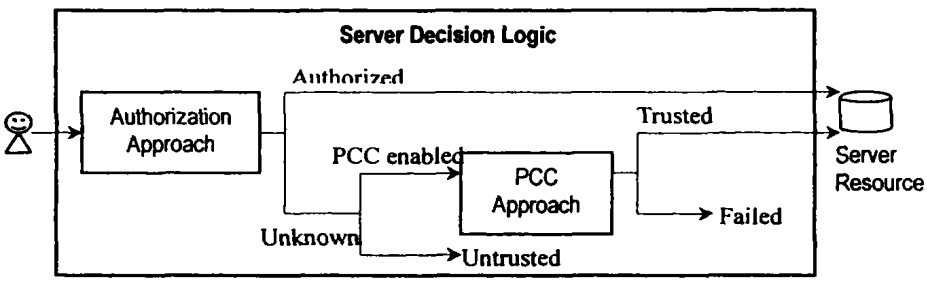

图4 两种机制并存的主机判定逻辑

我们的设计原则中尤为重要的是灵活性与通用性，并不只是为少数问题或应用特制，而是为广泛的应用提供通用的模型：PCC的安全策略应是可拆卸和可动态配置的。另外关于安全策略本身的刻画，近来还有学者对“信任”进行更深入的探查，认为对某人某事持“信”或“不信”过于刚性，从而通过模拟社会学研究，建立更为自然的信任概念，这也值得借鉴。

3.3 保护Agent

移动Agent面临的攻击源主要来自恶意的主机或恶意的Agent；它携带的信息一般包括代码、静态数据、收集数据及动态数据（状态），因此可将问题分为代码保护和数据保护两部分；其目的是：维持隐私性与维持完整性。具体含义如表2。

表2 移动Agent保护的内容

内容	隐私性维持	完整性维持
代码保护	防止代码的语义泄漏给潜在的恶意主机或Agent	代码在主机上执行或与其他Agent交互后的语义与原来一致
数据保护	使需要保密的数据不被潜在的恶意主机或Agent读取	不可变更的数据不能被潜在的恶意主机或Agent修改

其中，代码的隐私性维持是相当困难的问题，因为传统上都基本假定执行环境是安全可靠的，代码的语义可以被其知晓而无需维持隐私。对于完整性维持，一般认为：移动Agent本身

难以阻止窜改，但可以采取措施检测是否被窜改了。

起初，我们尝试用 PCC 的思想来解决 Agent 与主机之间的安全保护，因为存在 Agent 受恶意主机干扰和篡改的可能，且 Agent 本身也会携带重要的隐私数据，例如电子市场中外遣 Agent 携带的银行支票、私钥等，这符合使用 PCC 的条件。但在实现上却存在很多困难：首先，主机对 Agent 具有完全的控制，当一段 Agent 代码迁移到一主机平台并进驻其内存一块空间后，主机可对这块空间任意读取甚至修改，因而理论上隐私性的维持几乎不可能，而完整性的维持也只能通过检测的方法，对有害行为无法避免；其次，如果将主机针对 Agent 驻存空间的操作序列看作一个“程序”，但这就可能涉及主机所有的命令，使得该程序非常庞大，证明的工作量将会太重；其三，我们规定由 Agent 拥有者制定安全策略，主机对其操作“代码”作执行前或执行后的证明，但该证明的可靠性难以保证；其四，证明的确认方式，若移动 Agent 自身携带确认器，则存在的问题是确认器的保护，若使用主机提供的确认器，存在信任度问题，若由可信赖的第三方确认，则增加了开销；此外，即使主机的行为经过检验，也难以保证实际执行时的兑现。

Agent 与 Agent 之间的交互也需要考虑安全问题。两个移动 Agent 之间的交互可能发生在其所有者需要谈判、协商或议价的场合，例如电子市场中买方和卖方各自派遣移动 Agent 外出商谈价格；或是两个 Agent 本身便是软件服务提供者与消费者的关系；或者它们需要相互协同完成任务，等等。为了进行交互它们往往会登录到一台可信的第三方主机上，此主机提供类似沙盒的场所给其通信、运算和使用公共服务，而本身并不参与洽谈。由于进驻统一的内存区，Agent 体可能暴露给对方而造成不安全，例如一个恶意的 Agent 可能发出一条读/写某一块内存的指令，而这一块内存放的正是另一个 Agent 体，这样前者就可以通过分析窃取出后者的私密数据，或是修改后者的内容。即便限制两个 Agent 各自的内存区，使其进驻两个沙盒，恶意的 Agent 仍可借助主机上的通信环境对另一 Agent 做非法操作：例如抵赖、伪装身份、发出 DoS 攻击、非授权地调用另一 Agent 的公共方法等等。为此，我们如果能令第三方主机根据待保护 Agent 的安全策略，对可疑 Agent 的程序语义做检查，即提高主机场所的安全性，这样的安全漏洞就可以很大程度地避免。

我们把这样的主机称为 PCC 主机，看待它与两个 Agent 三者之间的关系可有两种视角。一是仍然看作独立的三方，主机为其上两个 Agent 的交互提供安全的场所：它提供一个可信任的证明生成器，用来自动地把一段程序转换成一个谓词并生成该谓词的证明；提供一个可调整的证明确认器，其中采用的算法可以根据用户要求变动；还提供一个可动态配置的安全策略装载器，需要保护的 Agent 可以把其安全策略装在 PCC 主机上，以指导对可疑 Agent 的检查。这样 PCC 主机的结构示意图参见图 5-1，这里只画出单方面校验的情形，而实际上交互双方可能都要求检验对方。

实际上，可以将登录到主机上、有待保护的一个 Agent 看作主机资源的一部分，另一个有待查证的 Agent 看作唯一的外部实体，这样三者之间的安全关系可以转化为主机与待查 Agent 两者之间的安全关系，从而可借用 3.2 节保护主机的模型，见图 5-2。我们把主机上的资源看作是带“插槽”的可扩展体，由两部分构成：固定部分是主机自己的资源如内存、文件系统、CPU 等等；移动部分是迁移进来有待保护的 Agent。同样在 PCC 模块中，安全策略也分为主机自己固定的安全策略和可动态插卸的 Agent 安全策略，从而 PCC 证明、确认工作均是增量型的。

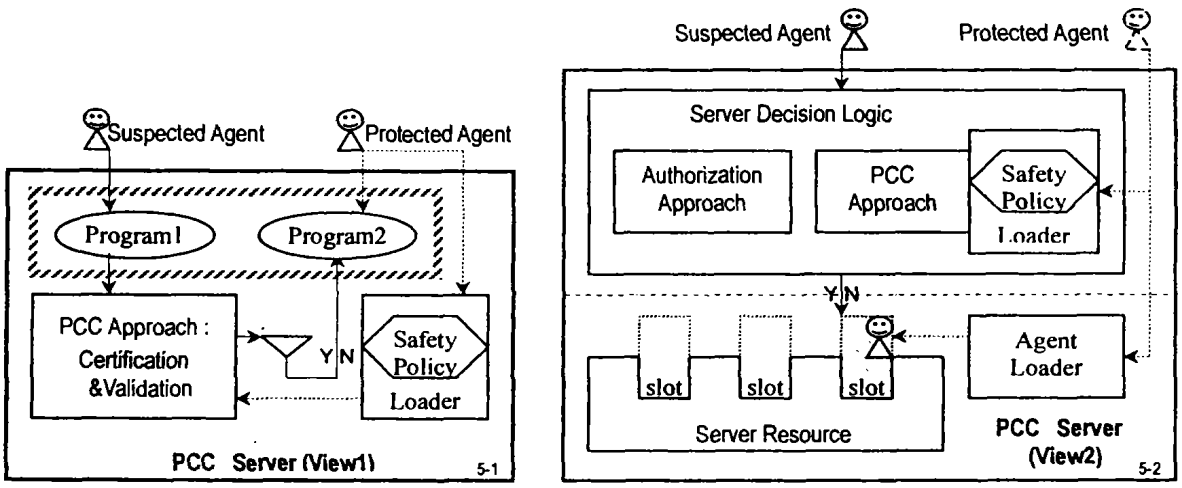

图 5　用 PCC 主机保护 Agent 交互安全(两种视角)

PCC 主机的好处是提高了 Agent 交互的开放性、安全性与可靠性。首先，Agent 相互操作的行为得到了预先的检查，避免了非法和恶意的攻击；其次，同样不需要了解交互对象的名称、来源和历史，更符合移动 Agent 计算的开放和分布特性；另外，证明的生成和确认都由可信的第三方主机做，可靠程度高，当然还可以考虑到 PCC 主机服务的收费代价。

此外，对于移动 Agent 与主机之间的保护，PCC 另一种可能的应用是验证防篡改硬件(Tamper-Proof Hardware，简称 TPH)是否符合安全要求。TPH 与主机的区别在于：TPH 一旦售出，任何组织(包括其制造者)都无法修改其操作行为，故 Agent 与已验证过的 TPH 的交互较为安全；而主机的操作程序在理论上可由其所有者在任何时间作任何可能的修改，相应保护的难度就大得多。

总结与展望　移动代码系统使得互联网上计算的效率与灵活得到了提高，但同时其安全亦成为一个重大课题。携证代码(PCC)是解决移动代码系统安全的技术之一，是集形式语义学、类型系统、程序验证和逻辑框架多种理论为一体的基于

(下转第 5 页)

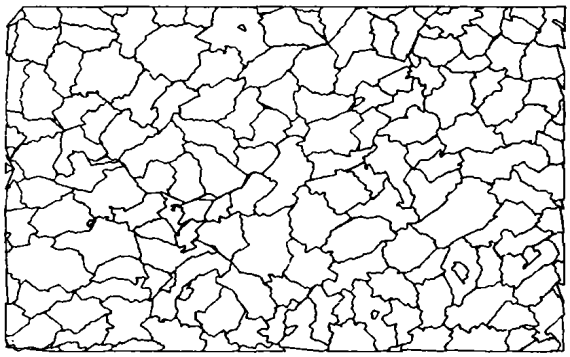

图 9　面积阈值为 0.1 的情况

测试结果比较情况参见表 3。

表 3　宽带面积去值的测试结果

	数据量(K)	计算值	近似度	计算时间(s)
原图层	1029	24.00000	100%	100.125
宽带面积:0.001	208	23.84722	100%	60.34
宽带面积:0.01	169	22.44723	106%	58.23
宽带面积:0.1	166	21.37943	110%	58.00

3.2.4　*基于数据精度转换的方法*　在空间图层中,通常用浮点数据形式来表示一个位置,由于浮点运算数占用空间多,计算比较复杂。基于精度转换的方法,将原有的图层数据用整型数据来表示空间中的点。可以达到减少数据存储空间,提高计算速度的目的。

利用该方法对上述数据进行测试,结果如表 4 所示。

表 4　基于精度转换方法的测试结果

	数据量(K)	计算值	近似度	计算时间(s)
原图层	1029	24.0000	100%	100.125
整型表示	256	23.8812	100%	64.38

总结　空间数据仓库和空间 Cube 的计算作为有效支持空间数据分析的一种手段,目前越来越受到学术界和工业界的高度重视。本文在介绍空间数据仓库和空间 Cube 概念的基础上,针对空间信息数据量大、空间聚集操作复杂的特点,给空间 Cube 的高效计算带来了一些问题,提出了扩充的 MBR、无损精度压缩以及多级抽点方法研究空间 Cube 三种近似计算方法,在减少空间数据的存储空间的同时,提高了空间 Cube 计算的精度和速度。

参 考 文 献

1　Inmon W H. Building the Data Warehouse. Second Edition. John Wiley & Sons, March 1996

2　Chaudhuri S, Dayal U. An Overview of Data Warehousing and O-LAP Technology. ACM SIGMOD Record, 1997, 26(1): 65～74

3　Güting R H. An Introduction to Spatial Database Systems. VLDB Journal, 1994, 3(4): 357～399

4　Han J, Stefanovie N, Koperski K. Selective Materialization: An Efficient Method for Spatial Data Cube Construction. In: Proc. 1998 Pacific-Asia Conf. on Knowledge Discovery and Data Mining (PAKDD'98), Melbourne, Australia, April 1998. 144～158

5　Clement G, et al. OGDI: Toward Interoperability among Geospatial Databases. SIGMOD Record, 1997, 26(3): 18～23

6　http://www.opengis.org, 2000

7　唐世渭,杨冬青,徐其钧,杨继国,谢昆青,裴健,陈伟毅. 支持数字地球信息集成与共享的空间数据仓库体系结构. 中国图像图形学报, 1999, 4(A 版增刊): 64～68

8　裴健,杨冬青,唐世渭. 基于数字地图的空间联机分析处理和空间数据挖掘. 中国图像图形学报, 1999, 4(A 版增刊): 59～63

9　Barclay T, Gray J, Slutz D. Microsoft TerraServer: A Spatial Data Warehouse: [MSR-TR-99-29]. Accepted by ACM SIGMOD 2000

10　Han J, Koperski K, Stefanovic N. GeoMiner: A System Prototype for Spatial Data Mining. In: Proc. 1997 ACM-SIGMOD Int'l Conf. on Management of Data (SIGMOD'97), Tucson, Arizona, May 1997, (System prototype demonstration)

11　Gray J, et al. Data Cube: A Relational Aggregation Operator Generalizing Group-by, Cross-tab and Sub-totals. Data Mining and Knowledge Discovery, 1997, 1(1): 29～54

12　Harinarayan V, Rajaraman A, Ullman J D. Implementing Data Cubes Efficiently. SIGMOD'96, Montreal, Canada, June 1996. 205～216

13　Agarwal S, et al. On the Computation of Multidimensional Aggregates. In: Proc. 1996 Int. Conf. Very Large Data Bases, Bombay, India, Sept. 1996. 506～521

(上接第 24 页)

程序设计语言的软件机制。其基本思想是旨在建立程序执行者对程序本身而非程序所有者的信任,使宿主机系统能籍此判定外来代码可否在本机上安全执行。宿主机预先制定安全策略,访问者据此赋予自己的代码一个证明,以说明其遵守安全守则;再把该证明提交给宿主机带的证明确认器;证明被确认为对的,则说明真的符合安全需求,方可将代码交付执行。它能克服传统基于密码学的安全机制的一些缺陷,可适用于语言互操作、操作系统内核、尤其是移动代码安全等领域。本文简要介绍了它的概念、动因及实现,并探寻其在 Mogent 系统安全模块中的应用,分别从保护主机站点与保护移动 Agent 两方面进行了分析与设计。

PCC 有一些自然的优点:主机端工作简单,因为证明由外来访问者做,主机无须关心如何做而只需快速自动地确认;防篡改,一旦代码变动后就要重新证明;先检查后执行,避免产生不良后果,也无运行时刻的开销;检查一次通过可执行多次,有利于工程化设计;自含且兼容,不必借助密码学或第三方,但也不排斥同时与之联合使用。然而,在实现中也存在一些未决的限制与难点,诸如证明的自动生成、公理化、规约逻辑、实用化甚至哲学问题,这些都将是未来研究努力的方向和突破口。

参 考 文 献

1　Loureiro S. Mobile Code Protection. Thèse présentée pour obtenir le grade de docteur, de l'Ecole Nationale Supérieure, des Télécommunications, Jan 26, 2001

2　Jansen W, Karygiannis T. Mobile Agent Security. In: NIST special publication 800-19

3　Necular G C, Lee P. Proof-Carrying Code: [Technical Report CMU-CS-96-165]. School of Computer Science, Carnegie Mellon University, Sep. 1996

4　Lee P, Necular G. Research on Proof-Carrying Code for Mobile-Code Security. A position paper in DARPA Workshop on Foundations for Secure Mobile Code, March 1997. 26～28

5　Feigenbaum J, Lee P. Trust Management and Proof-Carrying Code in Secure Mobile-Code Applications. A position paper in DARPA Workshop on Foundations for Secure Mobile Code, March 1997. 26～28

6　李新,吕建,曹春,冯新宇,陶先平. 移动 Agent 的安全性研究. 软件学报, 2002, 13(4)