一个具有高安全性的移动 Agent 系统模板结构*

杨 鲲, 刘大有, 郭 欣

(教育部符号计算与知识工程重点实验室,吉林 长春 130012);
(吉林大学 计算机科学与技术学院,吉林 长春 130012)
E-mail: yk_yangkun@hotmail.com; guoxinyk@mail.jl.cn
http://www.jlu.edu.cn

摘要: 在分析现有的移动 Agent 系统的特点以及 MASIF(mobile agent system interoperability facility)规范的基础上,给出了一个具有高安全性的移动 Agent 系统 Jamogents 及其模板结构,描述了其工作流程,并在重载 Java 类 java.lang.SecurityManager 的基础上实现了一种用于加密和数字签名的算法 RIM(RSA+IDEA+MD5).

关 键 词: Agent;移动 Agent;移动 Agent 服务环境;移动 Agent 系统;体系结构;安全性
中图法分类号: TP311 文献标识码: A

日益庞大的计算机网络及其异质性对网络管理和互操作提出了新的挑战.如何合理而有效地利用 Internet 上的巨大资源是计算机工作者们关注的重要问题.目前,国内外研究者们常常使用移动 Agent 技术来解决这些问题.移动 Agent(mobile agent,简称 MA)是具有移动性(mobility)的 Agent,它可以自主地在网络上从一台主机移动到另一台主机,并代表用户完成指定的任务.MA 具有更大的灵活性和更高的效率.这种灵活性为网络环境,尤其是 Internet 环境下的应用程序提供了很多潜在的优点[1].

使用 MA 技术的关键是给出一个完善、通用的 MA 系统的体系结构.目前已有不少成功的 MA 系统[1],这些系统或者是用专门语言(如 TCL,APRIL)实现的,无法进行跨平台操作;或者虽然采用了跨平台语言 Java,但在设计上往往基于专有的软硬件系统,其通用性受到较大的限制.这些 MA 系统在体系结构和系统实现上都存在着较大的差异,而且多数没有采用软构件技术,对其改造和重用非常困难,这就阻碍了 MA 系统的互操作和 MA 技术的推广.Open Group 等曾提出了有关移动 Agent 系统互操作的规范建议书 MASIF(mobile agent system interoperability facility)[2].MASIF 主要是从 MA 和 MA 系统的命名、MA 系统类型和定位、MA 的转移以及 MA 的管理等几个方面给出了一系列建议,但它没有给出一个较为明确的 MA 系统体系结构的框架,也没有定义 MA 的结构,而 MA 的结构无疑也存在着标准化的需求.

本文综合 MASIF 建议和相关研究成果[1,3~5],给出一个 MA 系统(称为 Jamogents)的模板结构,并在此基础上进一步给出了 MA 及其 MA 服务环境的模板结构,描述了 Jamogents 的工作流程,并针对 MA 系统的安全性问题给出了一种解决方法,即使用一种加密和数字签名的混合算法 RIM(RSA+IDEA+MD5).Jamogents 的实现语言是 Java.Java 的跨平台特性和面向对象的特点为 Jamogents 实现的构件化和精巧化提供了极大的方便.所谓“模板”的含义是指我们试图给出一个有一定通用意义的 MA 系统框架模型,以支撑设计和建造适于多种应用需求的 MA 系统.

* 收稿日期: 1999-09-14; 修改日期: 2000-08-01
基金项目: 国家 863 青年基金资助项目(863-306-QN2000-1)

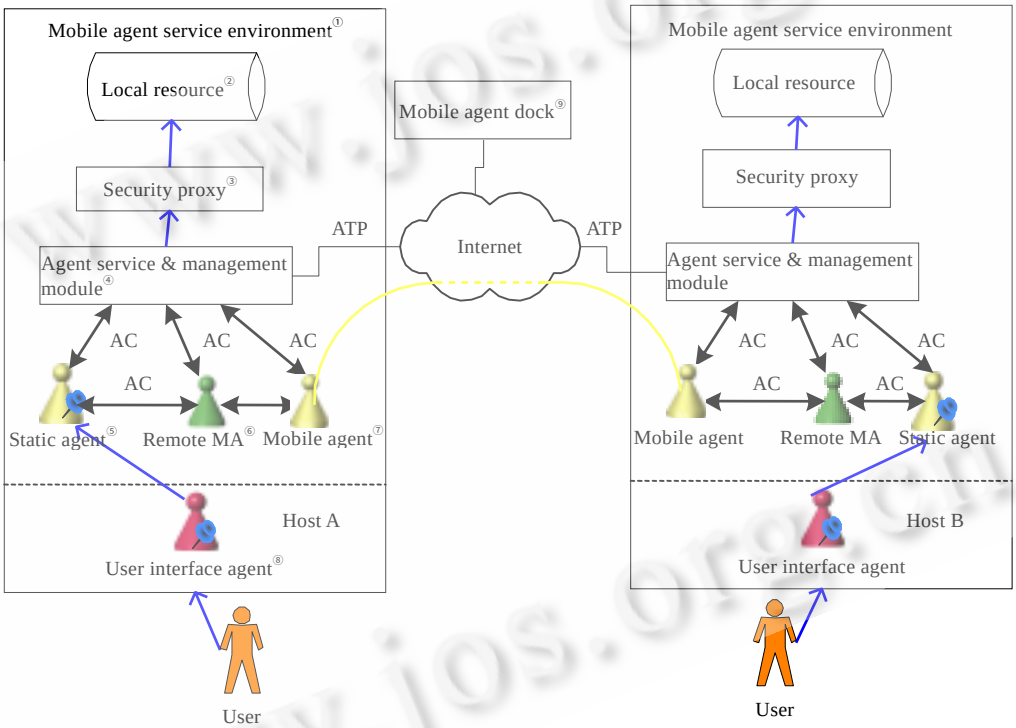
作者简介: 杨鲲(1969－),男,河北张家口人,博士,讲师,主要研究领域为分布式人工智能,移动 Agent 技术,Internet 网络管理技术;刘大有(1942－),男,吉林长春人,教授,博士生导师,主要研究领域为分布式人工智能,知识工程,多 Agent 系统;郭欣(1971－),女,吉林长春人,工程师,主要研究领域为移动 Agent 技术,分布式对象技术,网络技术.

1 移动 Agent 系统 Jamogents 的模板结构

1.1 Jamogents模板结构概览

Jamogents 由 4 部分组成:用户接口 Agent(user interface agent,简称 UIA)、服务请求端创建的要移动的 Agent(即 MA)、服务器端的 MA 服务环境(或称 MA 服务器、MA 主机)和 MA 停泊码头(dock).

UIA 是静态 Agent,用户通过它与 MA 进行交互和管理,其重点在于采用先进的人机接口技术,充分满足用户个性化的需求.MA 服务环境利用 Agent 传输协议 ATP(agent transfer protocol)来实现 Agent 在主机之间的移动,并为其建立远程执行环境和各种服务接口.MA 在服务环境中执行,以完成指定的任务,并通过 ACL(agent communication language)与其他 MA 通信或者访问 MA 服务环境所提供的服务.MA 停泊码头(MAD)是为了适应低可靠性网络和解决网络拥塞而设置的 MA 转接系统,向 MA 提供暂存服务,MAD 常常是网络中的一台主机.这里,MA 执行环境是 MA 系统的关键.这 4 个部分及其相互关系构成了 MA 系统的模板化体系结构,如图 1 所示.



①移动 Agent 服务环境,②本地资源,③安全代理,④Agent 服务与管理模块,⑤静态 Agent,⑥远程移动 Agent,⑦要移动的 Agent,⑧用户接口 Agent,⑨移动 Agent 停泊码头.

Fig.1 Template architecture of mobile agent system

图 1 MA 系统模板结构

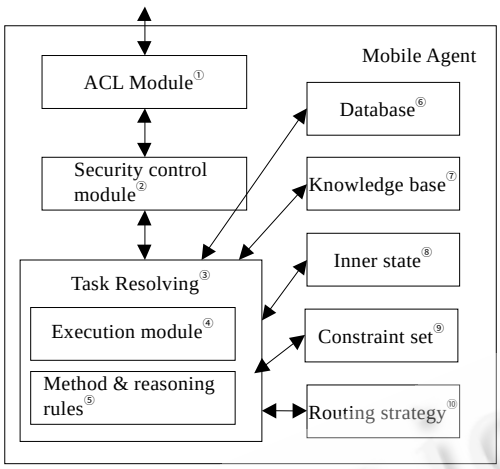
下面,我们重点探讨一下 MA 和 MA 服务环境的体系结构和功能,并在描述 Jamogents 的工作流程时给出各个部分的相互作用关系.

1.2 移动Agent

综合相关研究成果[1,3~5],我们给出移动 Agent 的体系结构,如图 2 所示.

ACL 模块:实现 ACL 的语义,保证使用相同 ACL 的 MA 与 MA 之间、MA 与 MA 执行环境之间的正确通信和协商,而且通信内容的语义应当与 ACL 无关.ACL 可以采用隐式通信和显式通信两种方式,前者是指 MA 能够自觉感知环境和其他 MA 的变化并作出反映,后者是指通信双方直接交换信息.在 Jamogents 系统中,我们

采用了一种简化的 KQML(knowledge query and manipulation language).它既具有 KQML 与环境无关、语法语义一致的特点,又达到了信息快速传输的效果.



①ACL 模块,②安全控制模块,③任务求解模块,④运行模块,⑤方法及推理规则,⑥数据库,⑦知识库,⑧内部状态集,⑨约束条件集,⑩路由策略.

Fig.2 Mobile agent architecture
图 2 MA 体系结构模型

安全控制模块:提供 MA 自身的保护,防止外部环境对 MA 的非法访问.它常常要完成加密、数字签名等任务.本文第 3 节给出了一种用来完成加密和数字签名任务的方法.

MA 的任务求解模块包括 MA 的运行模块和 MA 任务相关的推理方法与规则.运行模块包括 MA 的初始化程序和事件处理程序,前者在初始或移动到另一节点后启动事件处理线程,后者持续自主运行,感知外部环境的请求,并依据内部的规则和状态产生动作.MA 运行模块可以设计为任务独立的模块.任务相关性由不同的推理方法和规则集来实现.

数据库:保存 MA 运行时所需的数据、中间结果以及由 MA 采集处理并将要发送回用户的数据.知识库为 Agent 所感知的世界和自身模型,并保存在移动过程中获取的知识和任务求解结果.内部状态集是 MA 执行过程中的当前状态,它影响 MA 的任务求解过程,同时,MA 的任务求解又作用于内部状态.内部状态必须实现持久化支持跨平台的持续运行.

约束条件集是 MA 创建者为保证 MA 的行为和性能而设置的约束参数的集合,如返回时间、站点停留时间和任务完成程度等.一般只有创建者拥有对约束条件的修改权限,创建者可以通过用户接口 Agent 来实现对约束条件集的设定和修改.这些约束条件一方面由 MA 实现,一方面通过 MA 服务环境来保证.

路由策略决定 MA 要移动到的目标主机的有序列表.具体的移动路径由网络的 IP 协议决定,不属于 MA 研究的范畴.对于简单、明确的任务求解过程,可以采用静态的路由策略,即事先给出一张目标主机的有序列表.这张列表可以由 MA 用户通过 UIA 直接给出,也可以由 MA 发起方的服务环境给出.但对于网络结构和信息内容都动态变化的 MA 应用平台以及复杂和非确定性的任务求解过程,则要采用动态的路由策略.MA 的路由策略保证了 MA 可以自主地移动.

1.3 移动Agent服务环境

Jamogents 系统的移动 Agent 服务环境的结构如图 3 所示.

ATP 模块:ATP 定义了 MA 在各主机之间移动的语法和语义,ATP 模块具体实现 Agent 在服务环境之间的移动机制,包括移出和移入.

ACL 模块:采用 ACL 完成除了 MA 移动以外的其他通信任务,主要用于 MA 之间的通信.其功能和实现与 MA 结构中的 ACL 模块相同.

MA 管理与控制模块:它是 MA 服务环境的中心部件,将有关 MA 正常运行所需的各项服务正确分配给相应的模块,如,将有关 MA 执行环境的建立、启动的服务交给基本服务模块,将 MA 身份认证的服务交给安全控制模块,将 MA 要完成的特殊任务交给定制服务模块,将重新移动的任务交给 ATP 模块,将 MA 之间通信的任务交给 ACL 模块等,并协调各个模块的正常运行.该模块的另一个重要功能是实施 MA 的约束机制,并根据约束条件控制各个模块的运行.

基本服务模块:提供基本的 MA 服务,包括 MA 初始参数的设置、安全管理器的设置、MA 的启动、保证 MA 的正常运行等;另外,还完成与其他模块的交互,如调用安全控制模块进行 MA 的可靠性验证、通过 MA 管理与控制模块与其他 MA 通信等.

定制服务模块:为 MA 提供领域相关的任务求解服务,它能以组件的形式出现,以充分利用第三方产品.

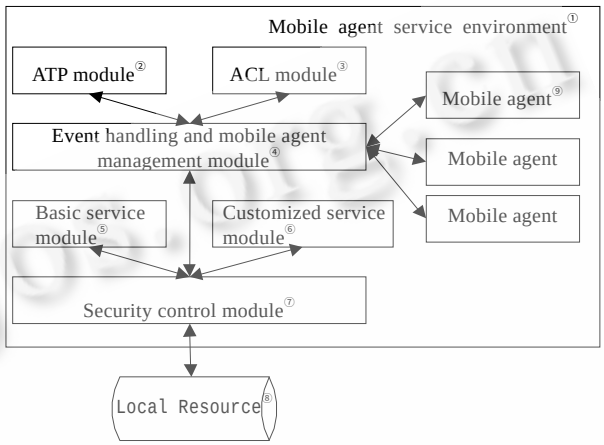
安全控制模块:主要用于实现主机的安全性策略.如进行数字签名验证、MA 代码的解密/解压缩等工作,还控制 MA 对本地资源的安全访问,进行付费检查等.

MA 服务环境的分配有两种策略,一种是为每一个 MA 分配单独的服务环境,另一种是为所有的 MA 分配同一个执行环境.显然,前一种分配策略具有更强的安全性,但会占用更多的资源.

2 Jamogents 的工作流程

现在,我们以图 1 所示的结构来简述 Jamogents 的工作流程.

(1) 用户首先根据自己要完成的任务,通过用户接口 Agent 对要创建的 MA 进行内部状态、数据库、知识库、约束条件等的初始化.将代表用户目标的特征表示和匹配尺度写入 MA 的知识库,并设置最长运行时间、每个站点的停留时间、任务完成度、搜索范围、经由节点的最大数目等约束条件.若采用静态路由策略,则还需要设置 MA 要访问的路由表.



①移动 Agent 服务环境,②ATP 模块,③ACL 模块,④事件处理与移动 Agent 管理模块,⑤基本服务模块,⑥定制服务模块,⑦安全控制模块,⑧本地资源,⑨移动 Agent.

Fig.3 Architecture of mobile agent service environment
图 3 MA 服务环境结构

(2) 调用“路由策略”模块进行路由选择.可以先指定一个目标主机,当移动到该主机上运行结束以后,再根据网络的当前情况和任务的完成情况决定下一个要移动的目标;也可以一次选择出所有要访问的目标主机列表.缺省时使用后一种方法.当采用静态路由策略时,此步可以忽略.假设被选中的目标主机为 A.

(3) 根据 MA 的安全策略,利用 MA 的“安全控制模块”对 MA 进行加密和数字签名.

(4) MA 利用“ACL 模块”与其本地的服务环境通信,并由本地的 MA 服务环境利用 ATP 协议将该 MA 通过网络(Internet)移动到主机 A.

(5) MA 到达主机 A 以后,A 通过其服务环境的“ATP 模块”对 MA 进行接收,并交给“MA 管理与控制模块”.

(6) “MA 管理与控制模块”首先调用“安全控制模块”对 MA 进行签名验证.若验证不成功,则拒绝该 MA 并在通知主机 A 以后将该 MA 删除;或者拒绝 MA,并通过其 ATP 模块返回到主机 A,以供主机 A 查明原因.若验证成功,则进行下一步.

(7) MA 在“MA 管理与控制模块”的控制下在 A 上运行,调用“基本服务模块”和“定制服务模块”执行相应的任务,通过服务环境的“安全控制模块”访问主机 A 上的资源,通过“ACL 模块”与其他 MA 进行通信.

(8) MA 在主机 A 上运行完毕之后,根据约束条件、路由策略、网络状态等条件决定下一步的行为:若任务完成或超出了指定的约束条件(如到达返回时间)则转步骤(12);否则,根据路由策略选择下一个要移动到的主机(假设为 B),接着执行下一步.

(9) MA 保存相应的历史记录和当前状态以后阻塞自身的工作,通过 ACL 模块向主机 B 发出移动请求,当得到允许以后,通过 ATP 模块移动到主机 B(此前也要进行安全性控制).

(10) MA 在主机 B 中重新进入运行态,继续执行任务(在主机 B 上的具体执行过程与步骤(5)~(7)完全相同).

(11) 重复执行步骤(8)~(10),直到任务完成或超出了指定的约束条件(如到达返回时间)为止,转步骤(12).

(12) MA 返回源主机(即用户端).在返回之前先判断网络的连通性,若连通,而且用户端的 MA 服务环境已经启动,则 MA 可以直接返回;否则,MA 移动到 MA 停泊码头(MAD)上暂存.MA 移动到 MAD 以后,MAD 会根据网络和自身的负载情况,将 MA 部分或全部从内存卸载到硬盘上,并替 MA 监视源主机.一旦具备传送条件,就激

活 MA,并将其传回到源主机.对于移动用户或设备,可由源主机(位置已经发生变化)访问 MAD.

(13) 当到达源主机以后,MA 将任务的求解结果提交给用户,并将自身全部卸载,进入终止态(进行一些善后处理工作,如保存获得的新知识等),结束该任务周期.

3 Jamogents 的加密和数字签名算法——RIM

MA 系统中的安全性威胁既可能来自于恶意的 MA,也可能来自于恶意的目标主机[3,5].所以,一个有效的安全保障机制应当从主机和 MA 两个方面着手解决,体现在 Jamogents 中就是 MA 服务环境和 MA 中的"安全控制模块".为了阻止恶意 MA 的非法执行,在主机上常常采用如下安全检测机制:用户身份认证、代码验证、授权认证和付费检查.为保证 MA 在传送和远程执行时的安全,可以采用加密技术、身份认证技术和数字签名技术等.其中,代码验证技术的难度很大,超出了本文的研究范围[5].授权认证主要检查 MA 对主机资源的各种访问许可,这可以通过建立一个类似于 OS 中使用的存取控制矩阵(表)的方法来实现.付费检查可以在 MA 的约束条件中给出 MA 所携带的电子货币的数量,并通过 ACL 与 MA 服务环境的"MA 管理和控制模块"进行协商和交易.本文集中讨论 MA 系统的两个最基本而且最重要的要求——安全传输和用户身份认证,即加密技术和数字签名技术,并实现了一种加密和数字签名的二合一算法——RIM.Java 语言本身也提供了一定的安全机制,但过于简单,而且很多方法只是抽象方法.我们对 java.lang.SecurityManager 类进行了重载,并在其基础上实现了 RIM.

在加密技术方面,目前大多采用 RSA 算法.它的安全性很高,但所需计算量太大,不适合于大数据量的加密.在 Jamogents 中,我们对需要大量传输的正文采用计算量相对较小(与 RSA 相比)的对称加密算法 IDEA (international data encryption algorithm),而对 IDEA 算法的密钥则采用 RSA 进行加密.在数字签名方面,我们用单向散列算法(本文采用 MD5(Message Digest 5))产生信息摘录,并用 RSA 对摘录加密(用私有密钥),从而得到签名.这样的链式加密方法既有 RSA 体系的保密性,又有 IDEA 算法的快捷性.我们将该算法称为 RIM(RSA+IDEA+MD5).

RSA 算法用到的是两个非常大的质数的乘积,对其进行攻击的方法是使用因数分解算法.而因数分解算法基本上是指数级的,当密钥长度足够长时(如超过 128 位),用目前的计算机进行运算基本上是很困难的.IDEA 算法是一种使用 128 位密钥、包含 17 个循环的分块加密/解密的对称加密算法.目前还没有办法对 IDEA 进行密码学分析,对 IDEA 的攻击方法只能采用密钥穷举,难度很大.IDEA 的密钥空间(密钥长度)为 128 位,其密钥个数是 2 128,从这些密钥中试探出一个密钥几乎是不可能的.MD5 是一种单向散列算法,已经有人证明,找到一个与原件具有相同特征精华的信息数据段的概率小于 1/265,这已经是足够安全了.RSA、IDEA 和 MD5 是成熟的算法,本文不再赘述.由以上分析可知,RIM 算法是足够安全的.

为确保在正式通信以前通信双方的正确性,通信双方首先要进行两阶段握手,即接收方认证和发送方认证.同时也进行双方公开密钥的交换.下面给出 RIM 的简单工作过程(假设 A 向 B 发送数据 Information):

(1) A 使用 MD5 产生数据 Information 的信息摘录 D;

(2) A 使用自己的私有密钥 SKA 对摘录 D 进行加密,得到签名 S;

(3) A 随机产生一个 IDEA 的密钥 K;

(4) A 使用 IDEA 算法对 Information+S 使用 K 进行加密,结果为 KIS;

(5) A 使用 RSA 算法,用 B 的公开密钥 PKB(在两阶段握手时获得)对 K 进行加密,结果为 PK;

(6) A 将处理后的信息 PK+KIS 发送给 B (通过移动 Agent);

(7) B 收到信息 PK+KIS 以后,使用 RSA 算法,用自己的私有密钥 SKB 解出 K;

(8) B 使用 K 对加密信息 KIS 进行解密,得到信息 Information 和签名 S;

(9) B 使用 A 的公开密钥 PKA 对 S 进行解密以后得到由 A 传送过来的信息摘录 D;

(10) B 利用 MD5 对收到的信息 Information 计算其信息摘录,得到新的信息摘录 D′,比较 D 和 D′进行签名判定.

4 结 论

与其他 MA 系统相比,本文提出的 MA 系统体系结构有以下特点:(1) 采用了模板结构模型,这样可以有效地提高 Agent 软件的模块性、可重用性、可扩充性和通用性;(2) 给出了完整的 MA 和 MA 服务环境的结构及其工作流;(3) 整个 MA 系统用 Java 语言实现,保证了跨平台移动;(4) 重载了 Java 的标准类 java.lang.SecurityManager,给出了一种高效、实用的安全算法 RIM;(5) 通过用户接口 Agent,可以在一定程度上实现对 MA 的管理与控制.在 Jamogents 中,MA 之间的交互方法基本上与 MASIF 建议相同;在 MA 的结构及其实现上借鉴了 Aglets 和 Mole[6]系统的特点,但在模块和功能上有更多的扩充.

我们今后的工作是在 Jamogents 中增加域名服务以及故障恢复和容错等功能.

References:

[1] Dale, J. A mobile agent architecture for distributed information management [Ph.D. Thesis]. Southampton: University of Southampton, 1997. 65~98.

[2] Milojicic, D., Breugst, M. MASIF: the OMG mobile agent system interoperability facility. In: Breugst, M., ed. Proceedings of the 2nd International Workshop on Mobile Agents. LNCS 1477, Berlin: Springer-Verlag, 1998. 50~67.

[3] Gray, R.S. Agent TCL: a flexible and secure mobile agent system. Technical Report, PCS-TR98-327, Hanover, NH: Department of Computer Science, Dartmouth College, 1998.

[4] Liberman, B. Griffel, F. Java-Based mobile agents——how to migrate, persist, and Interact on electronic service markets. In: Rothermel, K., Popescu-Zeletin, R., eds. Mobile Agents: the 1st International Workshop on Mobile Agent (MA'97). LNCS 1219, Berlin: Springer-Verlag, 1997. 27~38.

[5] Farmer, W., Guttman, J., Swarup, V. Security for mobile agents: issues and requirements. In: Vitek, J., Tschudin, C., eds. Proceedings of the 19th National Information Systems Security Conference. Baltimore, 1996. 591~597.

[6] Straßer, M., Baumann, J., Hohl, F. Mole——a Java-based mobile agent system. In: Andersen, B., Baquero, C., eds. Proceedings of the 2nd ECOOP Workshop on Mobile Object Systems. Berlin: Springer-Verlag, 1996. 28~35.

A Template Architecture for Mobile Agent System of High Security*

YANG Kun, LIU Da-you, GUO Xin

(*Key Laboratory for Symbolic Computation and Knowledge Engineering of Ministry of Education, Changchun* 130012, *China*);

(*Institute of Computer Science and Technology, Jilin University, Changchun* 130012, *China*)

E-mail: yk_yangkun@hotmail.com; guoxinyk@mail.jl.cn

http://www.jlu.edu.cn

Abstract: Based on the analysis of the current mobile agent systems and MASIF (mobile agent system interoperability facility), a template architecture for mobile agent system of high security called Jamogents including the architectures of both mobile agent itself and its supporting environment (MASE) are proposed in this paper. The working flow of Jamogents is also described. To make Jamogents secure, an algorithm called RIM, which implements both encryption and digital signature, is also discussed and implemented.

Key words: agent; mobile agents; mobile agent service environment; mobile agent system; architecture; security

* Received September 14, 1999; accepted August 1, 2000

Supported by the Youth Foundation of the National High Technology Development 863 Program of China under Grant No.863-306-QN2000-1