一种安全风险可控的弹性移动云计算通用框架

李新国[1] 李鹏伟[2,3] 傅建明[2,3,4] 丁笑一[2,3]
(深圳数字电视国家工程实验室股份有限公司 深圳 518057)[1] (武汉大学计算机学院 武汉 430072)[2]
(武汉大学空天信息安全与可信计算教育部重点实验室 武汉 430072)[3]
(武汉大学软件工程国家重点实验室 武汉 430072)[4]

摘 要 弹性移动云计算(Elastic Mobile Cloud Computing,EMCC)中,移动设备按照当前需求将部分计算任务迁移到云端执行,无缝透明地利用云资源增强自身功能。首先,在总结现有EMCC方案的基础上,抽象出通用的EMCC框架;指出EMCC程序中敏感模块的迁移会给EMCC带来隐私泄露、信息流劫持等安全风险;然后设计了融合风险管理的弹性移动云计算通用框架,该框架将安全风险看作EMCC的一种成本,保证EMCC的使用对用户来说是有利的;最后,指出风险管理的难点在于风险量化以及敏感模块标注。对此,设计了风险量化算法,实现了Android程序敏感模块自动标注工具,并通过实验证明了自动标注的准确性。

关键词 移动云计算,Android,模块分配,安全威胁,风险控制
中图法分类号 TP309 **文献标识码** A

Risk-controllable Common Elastic Mobile Cloud Computing Framework

LI Xin-guo[1] LI Peng-wei[2,3] FU Jian-ming[2,3,4] DING Xiao-yi[2,3]
(Shenzhen Digital TV National Engineering Laboratory Co. Ltd. Shenzhen 518057, China)[1]
(School of Computer, Wuhan University, Wuhan 430072C, China)[2]
(Key Laboratory of Aerospace Information Security and Trusted Computing, Ministry of Education, Wuhan 430072, China)[3]
(State Key Lab of Software Engineering, Wuhan University, Wuhan 430072, China)[4]

Abstract Elastic mobile cloud computing(EMCC) enables mobile devices to seamlessly and transparently use cloud resources to augment the capability by moving part of mobile devices' execution tasks to cloud on demand. At first, based on the summary of existing EMCC programs, the common EMCC implementation framework was build. Then we pointed out that the execution of EMCC applications may lead to privacy leakage and information flow hijack. Then an EMCC framework was proposed in which security risks are seen as costs of EMCC, and this framework can ensure the use of EMCC makes benefits for the mobile device user. Since the major difficulties of the implement of this framework are risk quantification and security-sensitive modules annotation, at last, a modules of risk quantification was designed and a tool which can annotate security-sensitive methods automatically was implemented. The validity of this tool was proved by experiments.

Keywords Mobile cloud computing, Android, Module allocation, Security threats, Risk-controllable

移动网络正在迅速发展。据统计,2014年中国手机网民规模达7.29亿[1]。无线通信、社交网络、游戏娱乐、移动金融/医疗/教育、现实增强等各种新兴任务对移动设备的性能要求越来越高。但移动设备受限于便携性、易用性、成本、散热等,其计算、存储、网络能力逊色于同时期固定设备;同时,电量对移动设备应用时间和范围的制约也日益凸显[2],电量问题甚至被认为是阻碍智能手机取代PC的主要原因[3]。WLAN、2G/3G/4G、WiMAX、CDMA2000等无线网络技术的发展为移动设备和云的结合提供了条件。移动云计算(Mobile Cloud Computing,MCC)中,计算任务可以在移动设备之外进行,云端为移动设备提供各种服务,从而突破移动设备的性能、电量瓶颈,极大地扩展移动设备的使用范围。MCC正在迅速发展,Heavy Reading的一份报告[4]预测,移动云计算的市场的直接收入将在2017年达到680亿美元。

现有MCC通常为C-S模式[5],移动任务被分为两份:由部署在移动设备中的客户端获取本地信息和用户输入,云端的服务器预先安装好相应软件,提供指定的服务。该模式的MCC中,任务的划分是固定的。为了能够更加灵活、高效地利用云端的计算资源,文献[6-13]设计实现了新的MCC方案:移动设备保管完成任务所需要的所有数据和代码,未能连

本文受国家自然科学基金(61373168, 61202387),教育部博士点基金(20120141110002)资助。
李新国(1976-),男,博士,主要研究方向为数字多媒体安全,E-mail:xgli@neldtv.org;**李鹏伟**(1987-),男,博士生,主要研究方向为软件安全、系统安全,E-mail:xu_pang@163.com(通信作者);**傅建明**(1969-),男,教授,博士生导师,主要研究方向为网络安全、软件安全;**丁笑一**(1991-),男,硕士,主要研究方向为Android安全。

接到云端时，移动设备自行完成计算；能够连接到云端时，移动设备将数据和代码切分为多个模块，按需将其中一部分模块迁移到云端，雇用云端来完成计算，以节省电量、提高运算速度。在程序执行过程中，移动设备可以依照具体执行环境（网络状况、电量、任务性质、用户模式等）对任务进行按需、灵活的动态调整。这种新的 MCC 模式中，移动设备对云端计算资源的索取是按需自助、弹性可变的，因此称此类新的移动云计算为弹性移动云计算（Elastic Mobile Cloud Computing，EMCC）。图 1 给出了一个简单的例子：一个程序被切分为 A、B、C、D、E、F 6 个模块，其中 A、F 被分配到移动设备，B、C、D、E 被分配并迁移到云端，然后依次执行各个模块。在执行过程中，用户需求发生变化，分配结果也随之变化，E 被重新分配到移动设备执行。执行完成后，由移动设备显示结果。

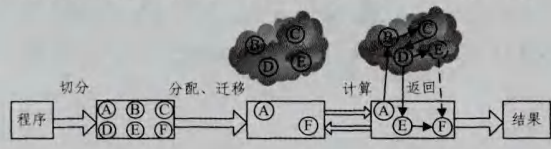

图 1　EMCC 程序执行过程示例

安全问题被看作是阻碍云计算发展的首要障碍[14]。移动设备的移动性、开放性、不稳定性等使得 MCC 的安全问题比传统云计算更复杂[35]。无线信道的窃听/拦截、云端所面临的外部与内部攻击、云服务提供商（Cloud Service Provider，CSP）窃取/出卖信息等均对 EMCC 构成安全威胁。例如，图 1 中，模块 B、C、D、E 被迁移到云端执行，则这些模块在迁移、执行的过程中可能会遭受信息窃取、数据修改等威胁。

EMCC 还处于发展初期，现有研究主要致力于实现细粒度的、对用户透明的 EMCC。EMCC 安全问题通常被列为下一步研究内容。本文对 EMCC 安全问题进行分析总结，指出提高 EMCC 安全性的主要方法是尽可能避免对保密性、完整性、可用性要求较高的“敏感模块”在风险过高的场景下进行迁移。基于这一原则，设计实现了一种风险可控的弹性移动云计算模块分配框架。该框架是通用的，与各 EMCC 的实施细节无关。最后的分析和实验证实了本文所述框架的可用性。

本文主要贡献如下：

1）总结出当前 EMCC 方案所共用的基本框架及其威胁模型。指出移动应用敏感模块的迁移会给 EMCC 带来隐私泄露、信息流劫持等安全风险。

2）设计了面向 EMCC 的风险量化方法。将安全风险看作 EMCC 中的一种成本，设计了一种融合风险管理的 EMCC 框架，保证 EMCC 的使用对用户来说是有利的。指出了实现该框架的难点及相应的解决方案。

3）指出实现 EMCC 风险量化的关键难点在于，标注出移动应用中各个模块的迁移可能会带来什么样的安全后果。针对这一问题，设计了面向 EMCC 的 Android 应用敏感方法标注方法，并通过实验证明了该方法的有效性。

1　EMCC 通用框架及其威胁模型

现有研究设计实现了多种不同的 EMCC 方案，但整体而言，大多数 EMCC 方案都使用了包含云环境构建、程序切分、模块分配、模块迁移、程序执行、结果返回等基本环节的通用框架。同时，各 EMCC 面临着类似的安全问题。

1.1　EMCC 通用框架

1.1.1　云环境构建

实现计算迁移的前提是构建用于辅助移动设备完成计算的云环境。这里的云环境包括云计算资源和云执行环境。可以利用公有云、私有云/专用云、本地计算机、附近的移动终端等构建云计算资源。如表 1 所列，各类云计算资源具有不同特点。

表 1　云计算资源的类型及其特点

云计算资源类型	相应方案	优势	缺陷
公有云	文献[6-13]	近乎无限的资源	网络质量制约服务质量[15]；存在用户信任问题
本地云	文献[15]	较高带宽、较低延迟网络	需要较高的管理成本；资源有限
本地 PC	文献[16,31]	较低成本；较小延迟；方便	稳定性、安全性较差
本地移动终端	文献[17,18]	较方便；较低成本	易泄露隐私；需提供用户开放计算资源的动机

为移动设备提供云端支持的可以是多个云服务提供商，还可以是多种资源的混合体[19]，例如可以通过本地 PC、本地云、公有云、闲置智能手机等共同构建资源池，按需灵活地为移动设备提供云计算、存储资源。

一般情况下，云服务器和移动设备的指令集架构不同（前者通常为 X86，后者为 ARM），需要在云端提供支持任务模块执行的环境，例如可利用 Android x86 平台为用户 EMCC 程序提供执行环境。资源共享能够有效地利用资源：如果资源可共用，则云端只需要提供各个结点总需求的峰值而不是各个节点需求峰值的叠加。表 2 列出了各类云执行环境的类型及其特点。

表 2　云执行环境的类型及其特点

云执行环境	示例	特点
共享模块	共享可执行模块[10]	资源利用率高，相应速度快，不安全
共享执行环境	多个用户的模块共享 Android 虚拟机[6,7]	资源利用率较高，安全性较差
独占执行环境	每个用户独占一台 Android 虚拟机[8,15]	资源利用率低，相应速度慢，较安全

云计算资源、云执行环境的多样化带来了多种安全问题。一方面，云端安全缺陷可能引发安全问题。例如，虚拟机系统漏洞、多租户问题可能导致用户信息被窃取、服务不可用等。特别是在共享模块或者共享执行环境的场景下，迁移到云端的移动应用计算模块可能需要和来自其他用户的计算模块共享 CPU、内存、时钟以及 Android 系统底层的 Linux 内核、库文件等，该共享为攻击者实施侧信道攻击等提供了便利。另一方面，恶意的云资源所有者容易通过下层的信息和资源窃取用户数据、破坏用户程序的执行。

1.1.2　程序切分

程序切分即将任务分为若干个可执行模块并分别标注各个模块的属性。现有研究中的切分主要有两种方法。1）重新设计适合在 MCC 中运行的程序架构与程序开发方式。例如，文献[6]中的 EMCC 程序由一个或者多个 Weblet、UI 和描述程序的声明组成。Weblet 是一种可以在移动设备或者云端运行的程序片段，可以独立完成某种特定的功能并提供与其他 Weblet 或者 UI 交互的接口。Thinkair[8] 提出面向

MCC 的 Android 程序开发方案并提供了编译器，程序开发者对可迁移到云端的方法(Method)标注@Remote，以方法为粒度对 Java 程序进行切分。2)基于分析的切分。Clonecloud[7]通过对现有程序进行分析找到可以对 Android 程序进行切分的点，然后自动切分现有程序，可将部分线程迁移到云端上执行。

1.1.3 模块分配

与固定设备相比，移动设备程序的执行环境更具多变性。在不同时间，移动设备的网络状况、电量、CPU/内存利用率等各不相同。因此，具体哪些模块需要被迁移到云端执行应该由当时的执行环境决定，且可随执行环境的变化而变化。

假设程序被切分为 n 个模块，且存在约束，有 k 个模块不能被迁移，则有 2^{n-k} 种模块分配方式。例如图 2 中，$n=4$，$k=2$，模块 1、4 不可迁移，则有 4 种模块分配方式。较多的分配方式保证了任务分配的灵活性。模块分配就是在这 2^{n-k} 种可能的分配方式中找出最好或较好的分配方式的过程。表 3 中列出了部分现有方案中的模块分配方法。

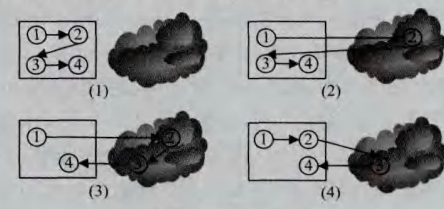

图 2 模块分配方式

表 3 现有 EMCC 方案中的模块分配方法示例

方案	切分粒度	目标	考虑因素	求解方法
Weblet[2]	Weblet	电量、资金、性能	电量、网络、负载、性能	机器学习，朴素贝叶斯模型
Clonecloud[7]	线程	时间、电量	网络、CPU 状况	消耗模型，最优选择器
Thinkair[8]	方法	负载、期限、硬件环境	网络状态、硬件状态、历史记录等	PowerTutor 模型、对比
AlfredO[11]	自定义模块	时间	模块的资源使用	资源消耗图
文献[10]中方案	组件	CPU 利用率，网络状态	最大吞吐量	遗传算法
文献[32]中方案	组件	移动设备性能、电量；云端消耗	执行环境，云端并行性、弹性等	模糊逻辑模型和实证学习

1.1.4 模块迁移

模块迁移即将一个或多个被分配到云端的模块迁移到云端，通常通过无线网络实现。由于迁移过程中模块/参数往往以明文方式存在，且相对于双绞线、光缆网络、无线网络(电话服务、短信服务、3G、WIFI、蓝牙等)更容易被监听[20]或干扰抑制[21]，攻击者可能会利用硬件进行信息窃取、信息屏蔽、发起传统拒绝服务攻击等。

1.1.5 程序执行与结果返回

程序执行即依次执行被分配到移动设备和云端的各个模块。可以通过虚拟机的使用来实现资源的灵活分配[11]，通过并行计算来提高计算速度[6]，通过资源共享来提高资源利用率[10]。云端的模块的执行完毕后，将结果返回给移动设备，由移动设备将最后的结果展示给用户。

1.2 EMCC 威胁模型

由上文分析可知，现有各 EMCC 方案虽然实现细节各异，但是其基本实现框架大致相同。现有 EMCC 均面临着移动设备的系统漏洞和恶意代码、无线信道的带宽/安全问题、云端所面临外部/内部攻击、恶意云资源提供者等安全威胁。此处，假设 EMCC 中被迁移到云端的模块可以执行所有的操作，例如可以发送短信。尽管发短信这一行为可能最终需要在移动设备上执行，但是由于发送短信之前的处理(例如收件人、短信内容的生成)是在云端完成的，因此，从安全角度，可以把“发短信”这一动作看成是在云环境中完成的，它会受到无线信道和云环境的威胁。

1.2.1 隐私泄露

对于攻击者来说，移动设备中包含有大量有价值的信息，包括用户信息、传感信息、系统状态、用户输入信息等。共享执行环境时，其他程序有可能通过底层资源窃取信息；恶意资源管理器可以通过记录内存、网络、CPU 数据窃取信息。无线信道的脆弱性、隐通道、对虚拟机或虚拟机管理器的攻击、不安全执行环境、不安全资源等导致保护被上传的信息是困难的。

1.2.2 信息流劫持

在云端或者无线信道中，攻击者有可能会对迁移到云端的模块或信息流进行修改，破坏程序执行过程的完整性。云端攻击者通过劫持此类程序信息流/控制流可以轻易实现多种攻击。例如，假设模块 A 生成一系列参数，模块 B 利用这些参数对移动终端中的系统状态或者用户信息进行修改，如果 A 被迁移到云端后被修改，则可能导致 B 对移动终端系统或用户信息的修改是错误的。此类攻击可能会导致病毒蠕虫传播、系统状态/用户信息被破坏、程序向错误对象发送特定短信或访问错误的网络地址，从而实现拒绝攻击、网络钓鱼、扣费攻击、电子诈骗等。现有安全方案中的验证，例如文献[25]中的模块间认证，只能保证启动时完整性，不能保证运行时完整性，不能防御此类攻击。

现有 EMCC 方案提到了一些简单的安全措施。例如，文献[6]中 EMCC 方案规定涉及本地数据的 Weblet 只能在本地执行，Clonecloud[7]在用到本机数据时将必须线程切回移动设备。但是，一方面，这些方法没有明确分析出哪些模块包含了“本地数据”，而且可能对程序的安全性造成威胁的模块不仅仅是包含本地数据的模块；另一方面，并非所有包含本地数据的模块都不应被迁移到云端执行，而应该是在用户的安全性和方便性之间寻求一个最优解。

2 一种风险可控的 EMCC 通用框架

为保证 EMCC 的使用对用户来说是利大于弊的，可以将安全风险看作 EMCC 中的一种成本，设计了一种融合风险控制的 EMCC 框架，尽量避免“将敏感模块迁移到云端”的高风险行为的发生。

2.1 EMCC 的收益-成本控制

对于移动终端用户来说，安装具有危险权限组合的程序，例如同时具有 INTERNET 权限和 ACCESS_COARSE_LOCATION 权限的程序显然是不安全的，但 Google Play 中此类程序得到了大量的下载安装[27]；对 Android 手机进行 root 显然会带来安全风险，但仍有大量手机被 root。可以看出普

通用户对安全的需求不是绝对的。因此，本文假设用户愿意为了功能的提升而接受一定的安全风险(用户在处理安全要求非常高的任务时可放弃使用 EMCC)，将安全风险视为 EMCC 的一种支出。

在不同时间，用户可能通过不同的网络连接方式以不同带宽连接到多种不同的云资源在不同的云环境中完成计算。对于用户，部分任务模块被迁移到云端执行提高了这些模块的执行速度，节省了移动设备电量；但同时，模块分配、迁移、结果返回等都需要消耗时间、电量，而且带来耗费网络流量、承担安全风险等其它支出。因此，应计算出各收益与支出，如果收益小于支出，则应放弃 EMCC，在移动设备本地完成程序执行。

本文所设计的通用的、试图确保 EMCC 的使用所带来的收益大于其支出的 EMCC 整体框架如图 3 所示。其中，“融合风险控制的模块分配”类似于 1.1.3 节中常见的分配方式，但是增加了对安全风险的考虑，是本节的主要内容；“风险标注”指的是标注出包含敏感操作或者可能会对敏感操作产生影响的模块，具体方法在第 3 节进行说明。

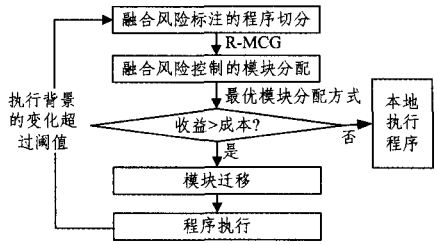

图 3　融合风险管理的 EMCC 通用框架

2.2　风险量化

不同模块的迁移所带来的安全风险各不相同，其具体风险值与模块包含信息、涉及的权限、所在程序、实时执行环境等相关联。风险由危害发生的可能性和危害程度两个方面决定，若能对这两个方面进行量化评估，就能完成对风险的量化评估。假设 R 表示风险，P 表示危害发生的可能性(威胁发生概率)，V 表示危害程度则上述关系可用式(1)表述：

$$R=V\times P \tag{1}$$

首先对隐私泄露的危害程度即保密性(Confidentiality)风险 R_C 的危害程度 V_C 进行分析。同一信息对不同用户的危害程度不同，例如用户甲认为照片信息泄露的危害大于短信内容泄露的，用户乙可能与之相反；而且，对于用户输入信息，只有用户知道自己所输入的信息的敏感程度。此外，不同信息的泄露，其危害程度不同，例如泄露短信息联系人数据的危害性不同。因此，要获取用户对泄露各信息的担心程度，首先需要了解用户的认知，其次需要分析出具体泄露了什么信息。对于前者，可以参考现有的风险评估方法[26,27]，通过统计分析、询问用户或者机器学习等方法得到结果；对于后者，则需要分析出具体涉及了什么信息，第 3 节将具体说明实现这一分析的方法。

威胁发生的概率主要与无线信道类型 WC、云执行环境 CP、云计算资源 CR 等相关联。当前的无线信道类型、云执行环境、云计算资源窃取信息 IN 的概率分别为 $Pc_{WC}(IN)$、$Pc_{CP}(IN)$、$Pc_{CR}(IN)$，修改操作 OP 的概率分别为 $Pi_{WC}(OP)$、$Pi_{CP}(OP)$、$Pi_{CR}(OP)$，则信息窃取概率 $P_C(IN)=Pc_{WC}(IN)+Pc_{CP}(IN)+Pc_{CR}(IN)$；信息劫持概率 $P_I(OP)=Pi_{WC}(OP)+Pi_{CP}(OP)+Pi_{CR}(OP)$。其具体数值可以通过分析具体情况、统计历史数据得出，本文不做具体分析。

设模块 M 中敏感信息 IN 被窃取、敏感操作 OP 被劫持造成的危害分别为 $V_C(IN)$，$V_I(OP)$；假设在 M 之前没有包含 IN/OP 的模块被迁移，则包含 IN/OP 模块 M 的迁移带来的安全风险的计算如式(2)所示。对模块迁移引发的信息流劫持威胁即完整性(Integrity)风险 R_I 危害程度 V_I 的计算与上文类似。对危害程度进行判断的关键点在于分析得到程序进行了什么样的敏感操作。

$$R(M)=P_C(IN)V_C(IN)+P_I(OP)V_I(OP) \tag{2}$$

2.3　风险可控的模块分配

设程序被切分为 n 个模块，模块分配即从 2^n 种可能的执行方式中找出最好或较好的执行方式(模块分配方法)。此处，模块分配的目标是尽可能找出具有最小风险 R、最小时间 T、最少耗电量 E、最小移动终端流量消耗 F 的分配方法。为方便描述，表 4 中列出了涉及到的元素的符号表示。其中分配方法 $X=\{x_1,x_2,\cdots,x_n\}$ 是长度为 n 的二进制序列，$x_i=1$ 表示模块 m_i 在云端执行，$x_j=0$ 表示模块 m_j 在本地执行。

表 4　符号表示

元素	符号表示
模块集	$M=\{m_1,m_2,\cdots,m_n\}$
需要被迁移的模块集	$M',M'\subset M$
分配方法	$X=\{x_1,x_2,\cdots,x_n\}$
模块 m_i,m_j 之间的数据量	$D(m_i,m_j)$
模块 m_i 的计算量	$C(m_i)$
模块 m_i 的长度	$L(m_i)$
移动终端计算能力(单位时间内计算量)	P_{md}
云计算能力	P_{could}
网络带宽	B
模块 m_i 的对外数据访问量	$D(m_i)$
模块 m_i 包含的敏感信息	$IN(m_i)$
模块 m_i 包含的敏感操作	$OP(m_i)$

EMCC 带来的风险 R 由保密性风险 R_C、完整性风险 R_I、其他风险 3 部分组成，其具体计算如式(3)所示，r 表示与具体方案相关的其他风险。

$$R=\sum_{i=1}^{n}x_iV_C(IN_i)P_C(IN_i)+\sum_{j=1}^{n}x_jV_I(OP_j)P_I(OP_j)+r \tag{3}$$

时间消耗 T 包含在移动终端进行计算花费的时间、在云端计算花费的时间、迁移模块花费时间、模块间通信花费时间等。T 的计算方法如式(4)所示，其中 t 表示与具体方案相关的其他时间消耗，如运行 EMCC 方案本身的时间消耗等。

$$T=\sum_{i=1}^{n}\left[(x_i+1)\frac{C(m_i)}{P_{md}}+(x_i-1)\frac{C(m_i)}{P_{cloud}}+(x_i-1)\frac{L(m_i)}{B}\right]+\sum_{i=1}^{n}\sum_{j=1}^{n}(x_i-x_j)^2\frac{D(m_i,m_j)}{B}+t \tag{4}$$

电量消耗 E 包括移动终端进行计算、对外网络访问、模块迁移、模块间交互所消耗的电量。E 的计算如式(5)所示，K_1、K_2 分别表示单位时间计算耗电系数和单位时间网络传输耗电系数，e 表示与具体方案相关的其他电量消耗。

$$E=\sum_{i=1}^{n}\{k_1(x_i+1)\frac{C(m_i)}{P_{md}}+k_2(x_i+1)\frac{D(m_i)}{B}+k_2(x_i-1)\frac{L(m_i)}{B}\}+k_2\sum_{i=1}^{n}\sum_{j=1}^{n}(x_i-x_j)^2\frac{D(m_i,m_j)}{B}+e \tag{5}$$

流量消耗 F 包括被分配到本地的模块进行网络访问的流量消耗、模块迁移的流量消耗和模块间交互的流量消耗。其具体计算如式(6)所示，f 为与具体方案相关的其他流量消耗。

$$F=\sum_{i=1}^{n}(x_i+1)D(m_i)+\sum_{i=1}^{n}x_iL(m_i)+\sum_{i=1}^{n}\sum_{j=1}^{n}(x_i-x_j)^2 D(m_i,m_j)+f \quad (6)$$

上文中的 t,e,r,f 可以根据具体方案、历史记录得出。其他因素，如云费用等，可通过类似方法算出。

明确求解目标之后，模块分配问题变为多目标求最优解问题。各个求解目标的比重可以根据用户当前的电量需求、时间需求、执行环境等决定。

3 面向 EMCC 的 Android 应用敏感模块标注

EMCC 风险量化的难点在于标注出移动应用中各个模块的迁移可能会带来什么样的安全后果。此处，假设移动设备使用的是 Android 程序，通过静态分析实现对敏感模块的标注。

3.1 敏感 API 定位

首先需要确定程序的切分粒度。此处假设程序切分采用类似 Thinkair[8] 的程序切分方案，其粒度是方法(Method)。这是因为该切分方法的粒度较小、较为精确，且多个方法可以组成组件、Weblet 或者其他自定义模块，从而实现对多种切分粒度的支持。

然后需要找出敏感 API，即能够获取敏感数据或者执行敏感操作的 API。通常来说，这些 API 都是受到权限保护的。A. P. Felt 等人建立了 Android 中系统权限与 API 直接的映射表[29]，AndroGuard 中的 api_permissions. py 工具融合了这一研究成果，本文中所用的敏感 API 首先吸纳了该工具中所列出的 API。没有受到权限保护的敏感 API 主要为从敏感区域读取数据(例如读剪贴板数据、读取 password 风格的文本框中的数据)或者将数据写到公共区域的 API(例如发送 intent)，我们结合 SUSI[38] 中的研究成果，对此类 API 进行了总结。最终的敏感 API 列表主要包含了上述两类敏感 API。

最后，反编译 Android 程序并将其转换为 Java 代码，构建程序的调用关系图(Call Graph，CG)，然后遍历该图即可得到敏感 API 的位置并获得其所在的方法。现有多项 Android 静态分析方向的研究工作提供了实现该分析的工具。

3.2 敏感模块标注

假设包含敏感操作的模块 A 中包含一条发送短信的 API：*SendTextMessage*($V1$,$V2$,$V3$,$V4$,$V5$)，其中 $V3$ 为短信的内容。但是，$V3$ 不一定是在模块 A 中定义的，有可能是在模块 B 中定义，然后经过了模块 C、D、E 的传播与修改才到达模块 A。此时，模块 B、C、D、E 和模块 A 一样应该被视为敏感模块。类似地，除敏感数据获取模块之外，该模块所获取的敏感信息流经过的模块也应该被视为敏感模块。

因此，需要对程序进行数据流分析，找出敏感操作 API 中的参数依赖于哪些语句、这些语句所在的方法、所获取的敏感信息、这些信息被复制到了哪些语句或者方法。此处，利用 Android 程序静态数据流分析工具 AmanDroid[36] 实现这一分析。AmanDroid 能够将 Android 程序转化为便于静态信息流分析的 Pilar 语言，然后实现上下文敏感、流敏感的数据流分析，为 Android 程序构建组件间的数据流图(Inter-Component Data Flow Graph，IDFG)。对于敏感信息获取 API，首先在 IDFG 上找到该 API 及其参数，然后进行前向(Forward)数据流分析，找到敏感信息传播过程中所涉及的语句，然后找到这些语句所在的方法；对于敏感操作 API，进行后向(Backward)分析，找到敏感操作 API 相关参数的定义点，然后输出分析过程中所涉及的方法。

4 实验与分析

由于目前并没有相关研究或者产业进行类似的工作，只能通过手动分析验证本文所设计实现的敏感模块标注方法的有效性。而手动验证需要阅读程序源码。因此首先使用的 FlowDroid 提供的开源测试集 DroidBench[37] 进行测试。DroidBench 最近版提供了 120 个程序，覆盖了 Android 程序传送敏感数据的多种方法。使用 DroidBench 进行测试有以下 3 个原因：1) DroidBench 是开源的，便于验证标注的准确性；2) DroidBench 本身是面向静态数据流分析，覆盖了 Android 程序传送敏感数据的多种方法；3) 大部分 DroidBench 都包含了敏感数据获取 API 以及传出敏感数据的 API。由于 DroidBench 中程序结果较为相似、人工分析工作量较大，本文选取了 DroidBench 中 15 个具有较好代表性的程序进行分析。

首先人工分析各程序源码，统计出各程序的用于获取敏感数据的 API 数、敏感操作相关 API 数、总方法数、在实际运行过程中不可能被调用的无效方法数、敏感数据获取相关方法数、敏感操作相关方法数。结果如表 5 所列。然后利用第 3 节中所述的敏感模块标注工具统计各程序的相关信息，结果和正确性分析如表 6 所列。其中，“虚拟方法数”指的是在编译过程中创建的、在源码中看不到的方法，如类的初始化等。

表 5 对 DroidBench 中的程序进行手动敏感方法标注的结果

程序名称	总方法数	无效方法数	敏感数据获取 API 数	敏感操作 API 数	敏感数据获取相关方法数	敏感操作相关方法数
DirectLeak1	1	0	1	1	1	1
Obfuscation1	3	1	1	1	1	1
PrivateDataLeak1	11	3	1	2	5	8
PrivateDataLeak2	1	0	1	1	1	1
PrivateDataLeak3	3	0	1	2	1	2
Button1	2	0	1	1	2	2
LocationLeak3	9	0	2	2	2	2
MethodOverride1	2	0	1	1	1	1
RegisterGlobal1	17	0	1	11	1	11
FieldSensitivity1	7	2	1	1	4	5
IntentSink2	2	0	1	1	1	1
ActivityLifecycle1	3	0	1	1	2	2
ApplicationLifecycle1	3	0	1	1	2	2
Reflection1	4	1	1	1	2	2
Reflection2	4	0	1	1	3	3

表 6　对 DroidBench 中的程序进行自动敏感模块标注的结果

程序名称	分析覆盖的方法数	虚拟方法数	分析覆盖的敏感信息获取 API 数	分析覆盖的敏感操作 API 数	标注的敏感数据相关方法数	标注的敏感操作相关方法数	误报数	漏报数
DirectLeak1	4	3	1	1	1	1	0	0
Obfuscation1	5	3	1	1	1	1	0	0
PrivateDataLeak1	12	4	1	2	5	8	0	0
PrivateDataLeak2	4	3	1	1	1	1	0	0
PrivateDataLeak3	6	3	1	2	1	2	0	0
Button1	6	4	1	1	2	2	0	0
LocationLeak3	11	4	2	2	2	2	0	0
Ordering1	5	3	1	1	1	1	0	0
RegisterGlobal1	10	3	1	1	0	6	0	6
FieldSensitivity1	9	4	1	1	2	3	0	4
IntentSink2	5	3	1	1	1	1	0	0
ActivityLifecycle1	7	4	1	1	2	2	0	0
ServiceLifecycle1	6	3	1	1	2	2	0	0
Reflection1	5	3	1	1	1	1	0	2
Reflection2	5	3	1	1	1	1	0	4

由表 6 可知，对 15 个程序的标注过程没有出现误报，4 个程序出现了漏报。具体分析漏报发生的原因发现，出现这些漏报的原因均为 AmanDroid 未能处理这些程序所使用的特定数据结构。例如 RegisterGlobal1 出现漏报的原因是 AmanDroid 未能处理程序中的全局回调函数；FieldSensitivity1 出现漏报的原因是 AmanDroid 没有能够正确处理部分域结构；Reflection1、Reflection2 出现漏报的原因是静态分析方法本身难以处理反射机制。所有漏报出现的原因均为现有的 Android 静态数据流分析还存在一定的局限性，随着近年来 Android数据流分析方面的研究工作的快速发展，将来可以实现更准确的标注。

为分析上述标注方法对于实际应用程序是否可行，选取了 5 个正常的 Android 应用程序和 5 个恶意代码进行测试。由于手工分析较为困难，选取了 10 个较小的程序(1.5MB 以下)。选取恶意代码的原因是，一般来说，恶意代码中会包含较多的敏感模块，能够更好地验证标注的准确性。实验结果如表 7 所列，对 10 个程序的自动标注工作均在 1 分钟内完成，手动验证分析结果发现，只有一个程序的标注存在错误。与上节中情况类似，该程序分析出错的原因也是 AmanDroid 的数据流分析未能覆盖所有的方法。实现结果说明，利用上述方法标注现实程序是可行的。

表 7　对现实程序的敏感模块标注结果

程序名称/MD5	方法数	敏感数据相关方法	敏感操作相关方法	是否正确标注	用时/s
一键锁屏	6	2	2	Y	21
安卓手电筒	48	0	4	N	16
推金币	53	38	13	Y	38
海卓上网大师	15	3	2	Y	19
真心话大冒险	63	13	12	Y	43
F31FD6D32609F6FF1C7C397A821D5FC9	175	74	4	Y	49
3323A2BD2E10FB7CAF2D9DA3823E5CE9	22	5	12	Y	28
5441DF458FCB198BC65B2B2797975375	93	2	49	Y	53
C6675832F44B225379B60CEFD7C80D36	16	1	2	Y	13
342C5D9F4DC064419544735F89C89658	9	3	2	Y	15

整体而言，本文所提框架可以将 EMCC 带来的风险控制在用户可接受的范围内，并尽可能防止 EMCC 的使用出现“得不偿失”的现象。该框架中，为提高安全性所做的工作主要有两部分：1)找出程序中的敏感模块；2)计算每一种分配方案的风险值。本节的实验证实标注的成本是较低的。对于风险值的计算，假设采用穷举各种分配的方式求最优解，每次求风险值的计算量为 N，对于切分为 n 个模块的程序，最多进行 2^n 次计算，如果 n 值较大则风险值计算的成本较高($N\times 2^n$)。但是，程序执行场景的可能数是有限的，例如，设用户的电量需求、时间需求、网络带宽、云服务状态、流量计划各有 4 种可能，则执行场景的可能性共 $M=4^5=1024$ 种。如果可以为这 1024 种场景分别求出最优的模块分配方案并保存，则不需要在每次运行程序时都进行求最优解的计算，计算量为 $M\times N$ 且只计算一次，该计算成本是可接受的。

结束语　近年来，国外各研究机构设计实现了多种 EMCC 方案。本文对这些方案共同面临的安全问题进行较为具体的分析总结并给出了通用的、易于实施的解决思路。本文首先建立了现有 EMCC 方案的通用实施框架(云环境构建、程序切分、模块分配、模块迁移、程序执行、结果返回)，然后从移动设备用户角度分析总结了 EMCC 带来的主要安全问题：隐私泄露、信息流劫持，指出可以利用风险管理来控制 EMCC 程序执行带来的安全风险。其中，对风险的控制可以融合到 EMCC 通用框架中的“模块分配”环节，实现风险管理的关键在于对各分配方案带来的风险的量化。针对该问题，设计了面向 EMCC 模块分配的风险量化方法，并解决了该量化方法面临的主要难点，即进行敏感模块标识。该框架可以为将来的 EMCC 方案的设计提供参考。

参 考 文 献

[1] 中国移动互联网用户行为统计报告 2015[EB/OL] http://mt.sohu.com/20150318/n409959259.shtml

[2] Barbera M V, Kosta S, Mei A, et al. To Offload or Not to Offload? The Bandwidth and Energy Costs of Mobile Cloud Computing[C]//Proc. of IEEE INFOCOM. 2013

[3] Rahimi M R, Ren J, Liu C H, et al. Mobile Cloud Computing: A Survey, State of Art and Future Directions[J]. Mobile Networks and Applications, 2014, 19(2): 133-143

[4] The mobile cloud market outlook to 2017[R]. Reading Real World Research, 2013

[5] Huang D, Xing T, Wu H. Mobile Cloud Computing Service Models: A User-Centric Approach[J]. IEEE Network, 2013, 27(5): 6-11

[6] Zhang X, Kunjithapatham A, Jeong S, et al. Towards an elastic application model for augmenting the computing capabilities of mobile devices with cloud computing[J]. Mobile Networks and Applications, 2011, 16(3): 270-284

[7] Chun B G, Ihm S, Maniatis P, et al. Clonecloud: elastic execution between mobile device and cloud[C]// Proceedings of the Sixth Conference on Computer Systems. ACM, 2011: 301-314

[8] Kosta S, Aucinas A, Hui P, et al. Thinkair: Dynamic resource allocation and parallel execution in the cloud for mobile code offloading[C]// 2012 Proceedings IEEE INFOCOM. IEEE, 2012: 945-953

[9] Cuervo E, Balasubramanian A, Cho D, et al. MAUI: making smartphones last longer with code offload[C]// Proceedings of the 8th International Conference on Mobile Systems, Applications, and Services. ACM, 2010: 49-62

[10] Yang L, Cao J, Yuan Y, et al. A framework for partitioning and execution of data stream applications in mobile cloud computing [J]. ACM SIGMETRICS Performance Evaluation Review, 2013, 40(4): 23-32

[11] Rellermeyer J S, Riva O, Alonso G. AlfredO: an architecture for flexible interaction with electronic devices[C]// Proceedings of the 9th ACM/IFIP/USENIX International Conference on Middleware. Springer-Verlag New York, 2008: 22-41

[12] Petitprez N, Rouvoy R, Filip K, et al. Opportunistic Offloading of Mobile Applications in Pervasive Environments[C]// 29th Symposium on Applied Computing(SAC). 2014: 1-6

[13] Shiraz M, Gani A. A lightweight active service migration framework for computational offloading in mobile cloud computing [J]. The Journal of Supercomputing, 2014, 68(2): 1-18

[14] Ren K, Wang C, Wang Q. Security challenges for the public cloud[J]. IEEE Internet Computing, 2012, 16(1): 69-73

[15] Satyanarayanan M, Bahl P, Caceres R, et al. The case for vm-based cloudlets in mobile computing[J]. IEEE Pervasive Computing, 2009, 8(4): 14-23

[16] Chen E Y, Itoh M. Virtual smartphone over IP[C]// 2010 IEEE International Symposium on a World of Wireless Mobile and Multimedia Networks(WoWMoM). IEEE, 2010: 1-6

[17] Huerta-Canepa G, Lee D. A virtual cloud computing provider for mobile devices[C]// Proceedings of the 1st ACM Workshop on Mobile Cloud Computing & Services: Social Networks and Beyond. ACM, 2010: 6

[18] Marinelli E E. Hyrax: cloud computing on mobile devices using MapReduce[D]. Carnegie-Mellon Univ, Pittsburgh PA, 2009

[19] Kaewpuang R, Niyato D, Wang P, et al. A Framework for Cooperative Resource Management in Mobile Cloud Computing [J]. IEEE Journal on Selected Areas in Communications, 2013, 31(12): 2685-2700

[20] Suo H, Liu Z, Wan J, et al. Security and Privacy in Mobile Cloud Computing[C]// International Wireless Communications & Mobile Computing Conference. 2013: 655-659

[21] Pelechrinis K, Iliofotou M, Krishnamurthy S V. Denial of service attacks in wireless networks: The case of jammers[J]. IEEE Communications Surveys & Tutorials, 2011, 13(2): 245-257

[22] Enck W, Ongtang M, McDaniel P. On lightweight mobile phone application certification[C]// Proceedings of the 16th ACM Conference on Computer and Communications Security. ACM, 2009: 235-245

[23] Enck W, Gilbert P, Chun B G, et al. TaintDroid: An Information-Flow Tracking System for Realtime Privacy Monitoring on Smartphones[C]// OSDI. 2010: 255-270

[24] Bugiel S, Davi L, Dmitrienko A, et al. XmAndroid: A new Android evolution to mitigate privilege escalation attacks: TR-2011-04[R]. Technische Universität Darmstadt, 2011

[25] Zhang X, Schiffman J, Gibbs S, et al. Securing elastic applications on mobile devices for cloud computing[C]// Proceedings of the 2009 ACM workshop on Cloud Computing Security. ACM, 2009: 127-134

[26] Theoharidou M, Mylonas A, Gritzalis D. A Risk Assessment Method for Smartphones[M]// Information Security and Privacy Research. Springer Berlin Heidelberg, 2012: 443-456

[27] Mylonas A, Theoharidou M, Gritzalis D. Assessing privacy risks in Android: A user-centric approach[M]// Risk Assessment and Risk-Driven Testing. Springer Berlin Heidelberg, 2013: 21-37

[28] Google Inc. androguard [EB/OL]. https://code.google.com/p/androguard/, 2012

[29] Felt A P, Chin E, Hanna S, et al. Android permissions demystified [C]// Proceedings of the 18th ACM Conference on Computer and Communications Security. ACM, 2011: 627-638

[30] Zhou X, Demetriou S, He D, et al. Identity, location, disease and more: inferring your secrets from Android public resources[C]// Proceedings of the 2013 ACM SIGSAC Conference on Computer & Communications Security(CCS2013). ACM, 2013: 1017-1028

[31] Hassan M A, Chen S. An investigation of different computing sources for mobile application outsourcing on the road[M]// Mobile Wireless Middleware, Operating Systems, and Applications. Springer Berlin Heidelberg, 2012: 153-166

[32] Flores Macario H R, Srirama S. Adaptive code offloading for mobile cloud applications: Exploiting fuzzy sets and evidence-based learning[C]// Proceeding of the fourth ACM Workshop on Mobile Cloud Computing and Services. ACM, 2013: 9-16

[33] Gu Q, Guirguis M. Secure Mobile Cloud Computing and Security Issues[M]// High Performance Cloud Auditing and Applications. Springer New York, 2014: 65-90

[34] Zhang Y, Yang M, Xu B, et al. Vetting undesirable behaviors in Android apps with permission use analysis[C]// Proceedings of the 2013 ACM SIGSAC conference on Computer & Communications security(CCS2013). ACM, 2013: 611-622

[35] 李瑞轩，董新华，辜希武，等. 移动云服务的数据安全与隐私保护综述[J]，通信学报，2013，34(12)：158-166

[36] Wei F, Roy S, Ou X. Amandroid: A Precise and General Inter-component Data Flow Analysis Framework for Security Vetting of Android Apps[C]// Proceedings of the 2014 ACM SIGSAC Conference on Computer and Communications Security. ACM, 2014: 1329-1341

[37] DroidBench[EB/OL]. http://sseblog.ec-spride.de/tools/droidbench/

[38] Rasthofer S, Arzt S, Bodden E. A machine-learning approach for classifying and categorizing android sources and sinks[C]// 2014 Network and Distributed System Security Symposium(NDSS). 2014