移动智能终端平台基于行为的远程证明方案①

詹世才, 秦 宇, 初晓博

(中国科学院软件研究所 可信计算与信息保障实验室, 北京 100190)

摘 要: 移动智能终端平台集通信、社交、网上购物、娱乐等众多功能于一身, 恶意程序对相关服务的破坏可能威胁到用户财产和个人隐私的安全. 远程证明是可信计算的核心功能之一, 它使得移动智能终端能向远程服务提供方证明平台运行状态的安全性. 传统的远程证明方案主要应用于计算机平台, 无法很好的适应软件频繁更新、多方服务共同运行的移动智能终端环境. 针对移动智能终端环境的特点, 本文设计了一种基于行为的远程证明方案, 通过软件开发人员定义软件行为列表, 终端系统强制实施行为限制, 服务提供方自定义策略对终端环境进行验证的方式, 满足了服务提供方保障其服务安全运行的需求. 方案原型的实现和评估表明本文方案兼具较强的安全能力和较高的性能.

关键词: 移动安全; 可信计算; 远程证明; 行为

Behavior-Based Remote Attestation for Mobile Platforms

ZHAN Shi-Cai, QING Yu, CHU Xiao-Bo

(Trusted Computing and Information Assurance Laboratory, Institute of Software, the Chinese Academy of Sciences, Beijing 100190, China)

Abstract: Mobile intelligent terminal platform combines telephone communication, social contact, online shopping and many other functions in one, the destruction of related services by malicious software could undermine the safety of personal privacy and property. Remote attestation, which is a core function of trusted computing, provides a scheme to allow mobile intelligent terminal to assure the secure running state to remote service provider. Traditional remote attestation model is mainly used in computer platform, which is unable to adapt to the environment of smart device where software updates frequently and works with many different service running together. To address on these issues, this paper presents a new behavior-based attestation model for smart device. Way from software developers to define behavior limitation, OS to enforce the limitation, service provider to custom strategy to attest smart device, successfully meets the requirement that corresponds service and should run in a secure environment. The implementation and evaluation of the model prototype shows that the model has strong security capabilities as well as efficiency.

Key words: mobile security; trusted computing; remote attestation; behavior

1 引言

近年来,移动智能终端的安全性得到了充分的关注. 相对于传统的计算机平台, 移动智能终端通常集通信、社交、网上购物、娱乐等众多功能于一身, 同时作为多个不同服务的载体. 这一方面导致移动智能终端平台通常会安装数量众多的分属于不同服务提供方的软件, 导致其运行环境复杂, 软件运行环境容易被破坏; 另一方面对于社交和购物等类型的软件, 被恶意软件攻击可能威胁到用户隐私和财产的安全, 从而保障这些软件运行环境的安全性尤为重要. 现有的移动智能终端安全软件和安全机制目标在于保障终端的安全性, 而无法向服务提供方证明终端运行环境的安全性.

远程证明提供了移动智能终端向远程服务提供方证明其运行环境安全可信的技术, 能很好的满足服务提供方保障其软件运行环境安全性的需求. 远程证明

① 收稿时间:2015-12-22;收到修改稿时间:2016-01-25 [doi:10.15888/j.cnki.csa.005303]

概念起源于可信计算技术领域[1]. 在典型的远程证明过程中, 证明者(移动终端)收集平台相关信息后发送给挑战者(服务提供方), 由挑战者根据这些信息判断证明者运行环境的安全性, 从而决定是否为挑战者提供相应服务. 远程证明技术一方面能保护服务提供方数据只能被安全可信的终端访问, 从而保障服务提供方的数据安全性; 另一方面, 阻止不安全的终端中运行相关服务也能防止用户的隐私信息被恶意程序访问, 增加用户的隐私和信息安全性.

远程证明方案可以根据证明内容分为以下三种类型: 基于二进制值的远程证明方案、基于属性的远程证明方案和基于行为的远程证明方案[2]. 二进制证明方案通过对软件进行度量以保障其完整性; 基于属性的证明方案通过可信第三方评估系统的安全属性并颁发相应的属性证书, 在验证时由挑战者验证证书的有效性;基于行为的证明方案通过记录系统的行为, 从而确保系统是在安全状态之间转换.

由于移动智能终端软件更新频繁, 终端的运行环境持续变化, 多个服务提供方对其服务软件的运行环境要求不同, 传统的远程证明方案并不适用. 基于二进制值的方案存在软件更新困难、暴露证明者配置隐私以及标准完整性值管理繁杂的问题. 基于属性的证明方法虽然可以较好的保护证明者隐私, 但其本质上仍然是基于二进制值的方法, 因此仍然存在难以克服的应用局限. 基于行为的证明方法从概念上拥有最好的实用性, 但是现有工作大都停留在模型设计层面, 尚缺乏通用、高效的方案设计以及完整的实验验证.

针对现有方案的不足, 本文旨在为移动智能终端平台提供一种通用的基于行为的远程证明方案. 利用本文方案, 服务提供方不仅能确保己方软件没有被篡改, 也可以要求终端在访问其服务时验证软件当前的运行环境是否满足服务提供商的安全策略. 例如, 网络银行服务可以禁止其他程序访问用户输入的账号密码信息和验证码短信, 自动导航软件在提供服务时可以拒绝终端同时运行多媒体服务等等. 我们的方案设计为: 第一, 服务提供方在终端运行相应服务软件时对其运行环境进行验证, 根据终端的状态决定是否提供服务, 并在拒绝提供服务时反馈具体原因; 第二, 终端根据服务提供方的反馈对调整运行状态进行调整以满足服务提供方对运行环境的策略需求, 然后重新发起远程证明. 通过这种方式, 既有效的保障了服务提供商的服务安全运行, 也能有效的保护的用户的隐私数据不被恶意程序窃取.

本文设计了一种基于行为的远程验证方案. 通过由软件开发者来定义软件可能触发的行为列表, 系统安全模块强制实施行为限制的方式, 将传统二进制或属性证明方案中对静态特性的评估改成了动态特性的限制. 在软件更新时, 不需要任何改动或只需要少量改动其行为列表定义, 从而能很好的应对软件频繁更新的问题. 移动智能终端平台通常会运行多种服务, 不同服务提供方对运行环境有不同需求, 本文方案中由移动智能终端操作系统记录运行环境的进程可能触发的行为, 并由服务提供方自定义验证策略用于限制运行环境中的其他进程能触发行为, 使得它们不会破坏服务提供方相应服务正常运行, 从而允许多个服务提供方灵活的定义其对终端运行环境的需求.

本文的主要贡献如下:

1) 本文设计了一种移动智能终端的行为远程证明方案, 通过行为定义、实施和验证的方式, 能很好的应对软件频繁更新和多种不同安全策略需求的问题.

2) 本文基于开源的引导程序 Uboot 和 Linux 的安全模块 Smack, 在 BeagleBone Black 开发板上实现了完整的方案原型, 对原型的测试表明此方案具有良好的性能.

3) 本文详细的评估了方案的安全性, 分析表明此方案能很好的防范 Rookit 和间谍软件等恶意程序的攻击.

本文的组织如下: 第二节讨论了移动智能终端平台已有的远程证明方案并分析它们的优缺点, 第三节描述本方案的设计, 包括系统架构的描述, 行为的定义和验证方式, 终端和服务提供方之间的交互协议等, 第四节中描述了方案原型的实现, 第五节对方案的性能进行了详细的测试, 并评估了方案的安全性.

2 相关工作

远程证明技术能使终端向远程服务提供方证明其运行状态的安全性, 建立网络通信实体之间信任, 确保网络通信安全, 在计算机平台就得到不断的发展. 随着移动智能终端的兴起, 其安全问题越来越受到人们的关注, 远程证明作为一种保障服务安全运行的方法, 在移动智能终端的应用也得到广泛的研究. 针对移动智能终端和传统平台环境和需求的差异, 不同机

构做了相关的理论研究并提出不同的解决方案.

二进制证明方案目前最为成熟, 其中 IBM 提出的 IMA 方案[3]已经成为 Linux 可选安全模块之一. IMA 方案的思路为度量系统关键信息的完整性, 挑战者根据验证者发送的完整性信息判断操作系统的相关程序是否被篡改, 从而判定验证者系统的安全状态. IMA 方案通常会使用安全芯片(如 TPM, TCM 等), 通过安全芯片身份认证功能和 PCR(Platform Configuration Register)扩展度量值的方式来保证验证过程的安全性. 在智能手机平台, 基于二进制方案的改进方案也被提出. Mohammad[4]等人针对 Android 操作系统中应用 APP 运行在 Java 虚拟机之上, 导致传统 IMA 方案无法验证应用层软件完整性的缺陷, 对 Dalvik 虚拟机进行修改加上相应的钩子函数实现了对 Android 应用层 APP 的完整性度量. 针对智能终端平台资源受限, 二进制方案效率不高的问题, Zhang Xinwen 等人[5]通过在操作系统中建立完整性模型, 限制低完整性实体对高完整性实体的访问, 并排除对系统中只读实体和不可信实体的度量, 从而大大缩小了需要度量的实体的范围, 减小了传统方案的资源需求并增强了其证明效率. 二进制方案要求服务提供方保存完整性度量值的标准值列表, 因而很难应对移动智能终端平台软件频繁更新的问题, 此外, 由于其需要将系统的所有完整性度量值发送给服务提供方, 会导致平台的配置信息泄露, 造成终端平台隐私泄露.

基于属性的远程远程证明方案最初设计以克服二进制方案的证明效率低下和隐私泄露问题[6,7]. 其主要思路为将系统的配置信息传递给可信第三方(TTP, Trusted Third Party),由第三方进行评估得出其具有的属性并发布相应的证书. 在基于属性的远程证明方案中, 平台的配置信息被转化成一系列属性, 挑战者只需验证平台具有的安全属性证书是否合法, 从而判断其具有的安全属性. 这种方案的缺陷在于: 第一, 由于引入 TTP 来产生属性证书, 从而将 TTP 纳入了可信计算基(TCB, Trust Computing Base)中, 加大了 TCB 的复杂性, 此外在移动智能终端通常作为多种不同服务的载体, TTP 的角色由谁来担任也是一个亟待解决的问题. 第二, 基于属性的远程证明方案面临属性难定义(从系统配置到安全属性的转化), 属性证书评估和撤销频繁的问题. Chen Liqun 等[8]针对难以选定 TTP 的问题, 设计了一种不需要不需要独立的可信第三方的远程证明方案. 在移动平台的基于属性的远程证明方案文献较少, 其主要原因在于基于属性的远程证明方案模型只能验证系统的静态信息, 从而在移动环境缺乏实用的应用场景. 文献[9]中指基于属性的远程证明方案模型只能验证系统的静态信息, 在移动环境缺乏适用的应用场景, 从而导致具体需要证明的属性和由谁来建立这种证明服务都不明确.

基于行为的远程证明方案思路通常是通过一个行为度量模块记录系统的行为信息, 并将对平台可信的证明转化为对平台系统的历史的行为序列的证明[10]. 这种远程证明方案有很多优点: 通过对系统的行为的管理和控制能很好的保护用户平台配置的隐私性, 同时允许灵活的对平台进行配置; 由于其根据软件或者系统行为的动态特性而不是度量值等静态特性, 可以很好的应对软件的频繁更新问题; 在证明时只需要对和平台可信状态相关的系统行为进行分析, 从而能减小运算量增强效率. 目前基于行为的远程证明方案[10,11]中缺乏明确的行为定义, 并且没有阐明挑战者如何根据验证者发送的行为序列判定验证方是否处于安全状态. Shahbaz 等人[12]针对医院职工的移动智能终端设备需要满足多个利益相关方(如医院, 设备使用者, 银行服务)的安全策略需求的场景, 设计了一种由多个利益相关方来协同制定设备上的强制访问控制策略的动态行为验证方案. 然而这种方案中远程服务方的权力太强, 且终端需要维护服务提供方的身份证书, 不能被普通用户接受, 并不适用于通用的场景.

3 系统架构

本文的系统总体框架如图 1 所示. 移动智能终端上需要实现三个主要模块: BAM(行为分析模块, Behavior Analyze Module)、BRM(行为报告模块, Behavior Report Module)和 BEM(行为实施模块, Behavior Enforce Module). 本文架构下, 软件开发人员发布软件时同时发布其行为列表, BAM 在软件加载时解析其行为列表, 并将相应行为限制加载到内核的行为池; 内核层的 BEM 强制实施软件行为限制, 确保软件可能触发的行为在其行为列表描述范围内; 当需要远程证明时, BRM 收集系统的当前运行状态信息并向服务提供方发起远程证明. 移动智能终端服务申请流程如图 2 所示. 在本文架构中, 服务软件通过运行凭证获得相应的服务, 其中运行凭证为与系统验证时

状态和验证时时间绑定的实体, 当存在有效凭证时终端可以直接运行服务程序, 在凭证不存在或失效时终端需要向服务提供方发起远程证明申请运行凭证. 远程证明过程中, 服务提供方根据移动智能终端发送的运行状态信息和预先定义的验证策略判断终端运行状态是否符合服务提供方对终端运行环境的要求, 由此决定是否返回运行凭证. 此外, 本文方案使用完整性校验机制来保证操作系统内核和 BAM, BRM 等核心模块功能的安全性, 通过建立从信任根到引导程序、系统内核和系统核心模块的信任链, 确保本文行为管控和验证机制的正确运行.

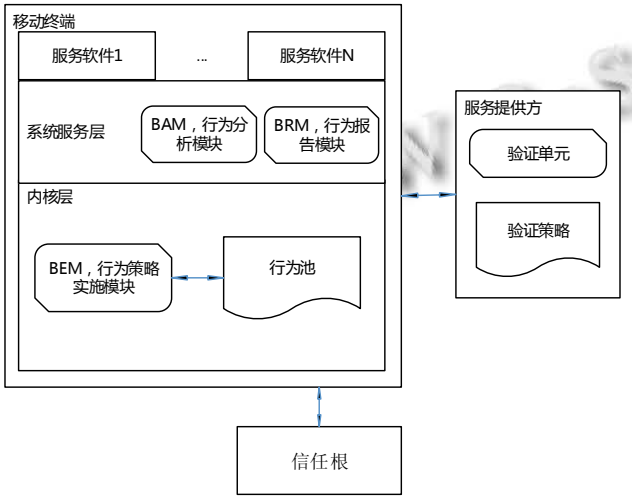

图 1　系统框架图

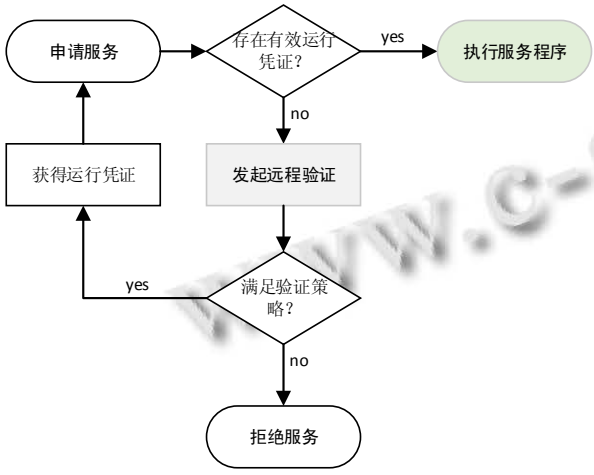

图 2　申请服务流程图

3.1 移动智能终端内部模块的完整性度量

为保障本文架构所依赖的相关模块的功能正确运行, 本方案中采用可信计算[13]中的信任链建立机制来保障本文方案所依赖的基础模块的正常运转. 信任建立从 RTM(可信度量根, Root-of-Trust for Measurement)开始, 由 RTM 度量引导程序的完整性, 引导程序度量操作系统内核的完整性及内核度量其相应的模块并记录完整性度量值[14], 由完整性值信息判断从 RTM 开始建立的整条信任链是否被破坏. RTM 通常为固化的只读程序, 由此保证 RTM 本身的安全性.

对完整性值的校验有以下两种常用方式. 在配备 TPM, TCM 等安全芯片的设备中, 可在远程阶段由芯片对完整性值签名后由远程验证方对完整性值进行校验, 由远程挑战者判断系统是否被篡改; 在没有配备或不便使用安全芯片的设备中, 也可通过在 RTM 中预置公钥并验证随系统发布的对完整性度量值的签名, 从而确保系统没有被篡改, 如 UEFI 安全启动方案. 由于移动智能终端平台空间有限, 通常不会配备安全芯片, 本文采取上述第二种方式来校验相应模块的安全性. 本文方案要求 OS 提供商发布系统同时提供内核和核心模块程序的完整性值列表并对其签名, 并在引导阶段和操作系统加载相应模块时验证完整性, 从而保证方案的正确运行.

3.2 软件行为定义及强制实施

目前判断预测软件可能触发行为主要有两种方式, 一种通过静态代码分析或动态运行历史预测软件可能触发的行为; 另一种是由软件开发人员声明软件可能触发行为, 操作系统强制实施行为限制, 从而在一定程度上保护用户终端的安全性, 如 Android 的权限申请机制. 使用静态代码分析的缺陷在于无法准确的判定软件的行为, 如访问资源时对应的资源路径可能是通过计算而来, 而静态代码分析无法准确的判定其访问的资源对象. 在远程验证方案使用动态行为历史进行判断会面临 TOCTOU(Time of Check to Time of Use)攻击,即平台在证明时刻会尚未触发可能违背验证方策略的行为被判定安全, 而当服务软件运行时破坏其运行状态. 采用预定义行为列表的方式, 既避免了行为的不可预测性, 也能很好的应对传统远程证明方案中面临的软件频繁更新问题. Android 权限机制并不适用于本文方案, 一方面是因为它提供的管理粒度过粗, 对于网络和短信等资源只能粗粒度的选择是否能访问, 而不能具体到某个网络资源或短信资源对象[15], 另一方面在于它没有提供一种机制向服务提供方报告当前系统的运行状态.

本文将行为定义为 (s,c,o,p) 四元组: 其中 s 表示

主体名称, 软件主体名称由 BAM 在软件加载时对其进行度量生成, 从而允许服务提供方验证其相应软件的完整性, 系统核心程序或模块的主体名称由操作系统赋值为 SYSTEM; c 用于表示客体类型, 可以是系统调用或某种资源类型, 不同的资源类型通过公共约束的方式形成唯一标识; o 表示客体名称, 通过操作系统指定, 也可由资源的提供者对其标记唯一的名称, 资源可以由客体类型和客体名称来唯一标识: 如网络资源的可通过指定类型为网络和指定被访问方的IP来唯一指定; p 表示主体对客体可进行的操作的集合, $p \subseteq \{r,w,x,a\}$, 其中 r,w,x,a 分别表示读, 写, 执行, 追加操作, 当客体类型 c 为系统调用时, p 可以为 $\varnothing$ 或 $\{x\}$.

软件开发者发布软件时需将行为列表作为软件的一部分发布. 当操作系统加载软件时, 由行为分析模块 BAM 将行为列表解析成对应的 (s,c,o,p) 四元组并加入系统行为池. 行为池中存储软件的行为限制, 由 BEM 模块强制实施. 当软件触发其定义的行为列表之外的行为时, 相应的资源访问或系统调用返回错误并终止软件运行. 对行为进行描述时, 允许对客体名称, 操作类型使用“*”通配符. 当客体类型为不需要访问资源的系统调用时, 操作类型默认为“*”表示软件可以执行相应的系统调用; 当客体类型代表某种资源时, 客体名称的“*”表明主体对相应类型的所有资源具有操作类型所指定的操作权限. 操作集合定义为“*”时表示相应的 $p=\{r,w,x,a\}$, 表明相应的主体能能对相应的资源执行任意操作. 预定义行为列表需要包含对客体类型、客体名称、操作类型三者的定义. 列表可以使用 xml 文件描述, 其示例文件如下所示.

```
<?xml version="1.0" encoding="utf-8"?>
<action-list>
<action>
    <object-type>network</oubject-type>
    <oubject>*</object>
<access>r</access>
</action>
<action>
    <object-type>file</oubject-type>
    <object>/sdcard</object>
<access>*</access>
</action>
……
</action-list>
```

此例中软件要求能读取所有的网络通信内容, 且能读写 sd 卡等等.

3.3 服务提供方验证策略定义及验证交互流程

服务提供方自定义验证策略对移动智能终端的运行环境进行验证. 相较于二进制方案中校验移动智能终端上运行软件的完整性、基于属性的远程证明方案中判断终端具备的安全属性, 本文的方式允许服务提供方对移动智能终端的运行环境提出更灵活的需求.

验证策略的定义使用黑名单机制. 服务提供方在提供服务时, 可以限制其他进程对特定的资源的访问, 禁止其他进程调用某个系统调用, 或禁止终端上特定程序的运行. 我们将验证策略描述成服务提供方禁止移动智能终端触发的行为的集合, 禁止的行为同样表示成 (s,c,o,p) 四元组, 其含义与上节一致. 在定义验证策略时, 行为的描述中允许使用“*”来实现粗粒度的管理, 如将主体名称设置为“*”表示限制终端上运行的其他所有应用软件对相应资源的访问. 将客体名称设置为“*”可以同时限制主体对一类资源的访问等等. 服务提供方可以直接使用文本文件描述其验证策略, 示例文件如下所示.

Subject	Object-Type	Object	Access
S1	*	*	*
S2	*	*	*
*	network	124.167.232.125	*
*	sms	10086	w
*	systemcall	ptrace	*
*	systemcall	setpriority	*

此例中描述的服务提供方验证策略为: 禁止移动智能终端上名称为 S1 和 S2 软件运行, 拒绝其他进程对 10086 的 SMS 信息写入, 获得与 124.167.232.125 IP 地址对应的网络通信方的独占交互, 同时不允许其他进程调用跟踪和改变优先级的系统调用.

移动智能终端向服务提供方申请验证以获得相应服务时, 由 BRM 模块与 BEM 模块交互获得系统当前的运行状态 $\boldsymbol{S}$. 服务提供方首先验证其客户端软件是否被篡改, 然后将客户端的运行状态 $\boldsymbol{S}$ 与其预先定义的验证策略 $\boldsymbol{\mathcal{B}}$ 对比, $\boldsymbol{\mathcal{B}}$ 为服务提供方禁止的行为的集合, 通过判断集合 $\boldsymbol{S}$ 和 $\boldsymbol{\mathcal{B}}$ 是否有交集即可判定移动智能终端当前运行状态是否满足服务提供方的需求. 验

证过程 *Attest* 的伪代码如下:

算法: *Attest*

输入: 正在进行验证的软件 s_0, 移动智能终端的状态 $\boldsymbol{S}$, 服务提供方的验证策略集合 $\boldsymbol{\mathcal{B}}$, s_0 正确度量值集合 $\boldsymbol{\mathcal{M}}$

输出: 终端当前违背验证方策略的主体集合 $\boldsymbol{\mathcal{R}}$

```
1   R ← ∅
2   if   s0 ∈ M
3        R ← R ∪ {s0}
4      return  R
5   end if
6   for each  (s1, c1, o1, p1) ∈ B
7       for each  (s2, c2, o2, p2) ∈ S
8           if  (c1 = * or c2 = * or c1 = c2) and
9               (o1 = * or o2 = * or o1 = o2) and
10              (s1 = * or s2 = * or s1 = s2) and
11              (p1 ? p2)
12                R ← R ∪ {s2}
13      end for
14 end for
15 return  R
```

为了避免终端每一次申请相应服务时都要重新发起验证, 本方案中软件通过验证运行凭证的有效性来获得相应服务. 当 Attest 返回的集合 $\boldsymbol{\mathcal{R}}$ 为空集时, 表明终端当前的运行环境满足服务提供方的需求. 此时服务提供方发送返回移动智能终端一个运行凭证. 凭证与验证发生的服务器时间 t 和移动智能终端验证时状态 $\boldsymbol{S}$ 绑定, 并包含凭证有效时间信息. 验证凭证的有效性即为判断验证时终端状态是否和 $\boldsymbol{S}$ 相同并且验证时时间是否在凭证的有效期内; 当返回集合 $\boldsymbol{\mathcal{R}}$ 不为空时, 表明终端当前的运行环境不符合服务提供方验证策略的要求, 此时集合 $\boldsymbol{\mathcal{R}}$ 中的内容为移动智能终端当前运行的软件中不满足服务提供方策略的主体集合, 移动智能终端获得服务提供方返回的 $\boldsymbol{\mathcal{R}}$ 集合后, 由终端用户对运行环境调整后重新发起验证.

终端与服务提供商的验证交互协议流程如图 3 所示, 详细过程描述如下:

① 移动智能终端申请服务, 由 BRM 模块向服务提供方发起验证申请;

② 服务提供方生成一个随机数 n 并将其返回终端, 同时记录当前时间 t;

③ BRM 模块向 BEM 模块请求系统运行状态;

④ BEM 模块将系统当前状态 $\boldsymbol{S}$ 返回 BRM;

⑤ BRM 模块对 $\boldsymbol{S}, n$ 签名后, 将 $\text{sign}(\boldsymbol{S}, n)$ 和 $\boldsymbol{\mathcal{R}}$ 发送给服务提供方. 签名密钥对可通过交互生成, 也可将公钥其预置于软件中;

⑥ 服务提供方根据移动智能终端发动的状态 $\boldsymbol{S}$ 和预先定义的验证策略 $\boldsymbol{\mathcal{B}}$ 通过 Attest 过程计算 $\boldsymbol{S}$ 中违背验证策略的主体集合 $\boldsymbol{\mathcal{R}}$. 若 $\boldsymbol{\mathcal{R}}$ 为 $\varnothing$, 服务提供方生成与状态 $\boldsymbol{S}$ 和当前时间 t 绑定的凭证 $cre(\boldsymbol{S}, t)$ 后将其返回终端, 否则直接返回 $\boldsymbol{\mathcal{R}}$.

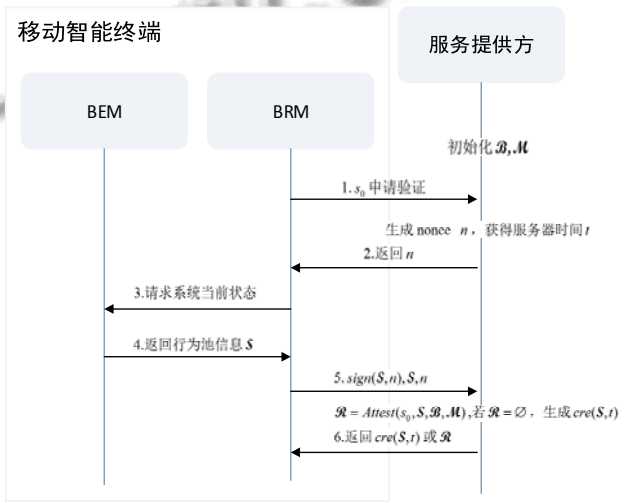

图 3　协议交互图

4　方案原型实现

本节介绍我们在嵌入式平台上实现的基于行为的远程证明方案的原型, 主要包含以下三部分的内容: ① 通过信任链建立实现从可信度量根到操作系统内核的层层度量, 从而保证操作系统内核不被篡改; ② 基于 Linux 安全模块之一 Smack 源码实现对系统核心模块(如 BAM, BEM)的基于完整性的保护和对应用程序的行为管控; ③ 实现终端到服务端之间的远程验证协议. 本节介绍前两部分的实现. 我们使用 Beaglebone Black 作为我们的实验平台. BeagleBone Black 开发板具有体积小、性能强大、扩展性强、价格低廉等优点, 其处理器为 AM335x 1GHz ARM Cortex-A8, 具有 512MB DDR3 RAM 内存, 非常适合作为我们的实验平台. 另外, 我们使用 U-boot 2013-04 版本作为我们的基础引导程序. Linux 内核版本为 4.1.2, 在实验过程中, 我们使用了 Debian7.8(BeagleBone Black-4GB SD)镜像作为我们实验使用的基础系统镜

像.

4.1 可信引导实现

我们实现的安全引导方案通过在引导程序中加入完整性验证和签名验证来实现. BeagleBone Black 属于 am335x 系列开发板, 采用 Uboot 作为引导程序时其引导过程分为以下三个阶段: 第一阶段代码固化在 rom 中, 通过板子上的电平开关判断 SPL(第二引导阶段, Second Program Loader)程序的位置. 第二阶段和第三阶段的引导程序由 Uboot 源码编译生成, 对应文件名分别为 MLO 和 uboot.img. 其中, MLO 文件会被映射到 CPU 的 RAM 中执行. 通过 MLO 加 uboot.img, uboot.img 加载操作系统内核的方式最终实现操作系统的引导. 由于 rom 区无法修改, 我们选择 MLO 作为我们方案的可信度量根. 在实际使用中可以通过将 MLO 文件存储在只读 TF 卡区从而确保其不被篡改.

对 Uboot 源码的修改主要包含以下两部分内容: 一, 在源码中加入 RSA 验证算法库和哈希算法库; 二, 在关键位置加入对镜像签名的验证逻辑. 由于 MLO 文件大小需要严格的限制, 加入额外的算法库都会导致其体积过大, 且引导程序通常不需要更换, 我们方案中 MLO 对 Uboot 的完整性校验直接通过哈希值比对的方式实现, 在加载时通过将 Uboot 的完整性值度量后硬编码到 MLO 中并在加载时判断其值是否一致, 从而确保 Uboot 引导程序的安全性. Uboot 对内核的完整性判断通过 RSA 签名验证算法实现, 当其加载操作系统内核时, 同时读取系统发布方提供的内核签名文件, Uboot 启动内核镜像时度量内核的完整性值并验证完整性度量值签名, 从而确保内核的安全. 整个过程需要进行两次编译, 第一次编译生成 Uboot 并通过外部程序计算其完整性值, 第二次将其完整性值写入 MLO 对应的源码中生成相应的 MLO. 修改后的引导示意图如图 4 所示.

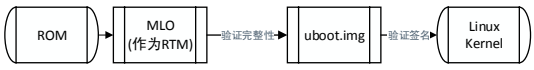

图 4 可信引导流程图

4.2 基于 Smack 实现完整性度量和行为监控

当操作系统内核加载完成后, 下一步需要实现对 BAM、BRM 等应用层系统核心程序的保护和行为监控模型. 对于前者, 我们采用完整性校验的方式来确保其功能的正确运行. 系统核心模块或程序的完整性值需由 OS 发布商发布时提供并对其签名, 并默认赋予其 SYSTEM 主体标签. 对于非 SYSTEM 程序我们将其加入行为管控的监测范围.

我们使用 Smack 作为我们方案的基础. Smack 本身实现了 Linux 上资源的强制访问控制机制. 通过在系统中定义标签和标签之间的访问控制规则, 并对程序赋予主体标签和对资源赋予客体标签和客体标签, 实现了实体之间的访问控制. 我们对 Smack 的主要修改如下: 第一, 我们在 Smack 中默认加入了 SYSTEM 主体标签, 内核中默认定义 SYSTEM 主体无行为限制. 第二, Smack 的访问控制策略为(主体标签, 客体标签, 访问控制权限)三元组, 为了实现我们方案中的(主体名称, 客体类型, 客体名称, 操作类型)四元组策略, 进而实现对系统调用的监控和不同粒度的策略管理, 我们在其标签基础上加入了所属类型, 并在系统中预定义了如文件, 网络, 系统调用等类型; 第三, Smack 本身只是为了实现系统的强制访问控制, 并不关心哪些规则实际生效, 我们在其标签的基础上加入了引用计数, 并在进程运行和退出时改变引用计数, 从而 BRM 向远程服务提供方验证时只需验证实际在系统中运行的进程对应的行为策略. 第四, 我们在用户层实现了 BRM 和 BAM 的原型,由 BAM 在程序运行时往内核行为池中加入相应的行为管控, BRM 从内核中获得系统当前行为行为池信息并由此向远程服务提供方发起验证.

我们修改过的系统中软件的运行流程如图 5 所示.

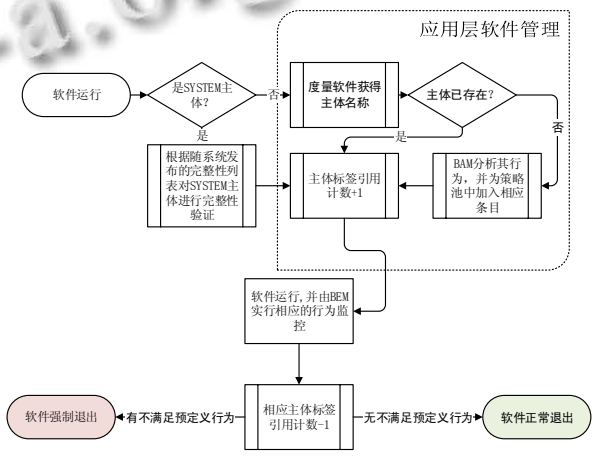

图 5 软件运行过程图

5 方案评估

我们在 BeagleBone Black 开发板实现了方案原型

系统. 本节将从引导程序大小, 引导程序性能, 行为检测相关模块性能和其对程序运行性能造成的影响等方面对方案进行全面的评估. 最后, 我们针对常见的系统攻击类型分析此方案的安全能力.

5.1 系统性能评估

1) MLO 文件大小

MLO 文件在我们方案中充当可信度量根的角色, 在运行过程中会被映射到 CPU 的 RAM 区执行, 而 CPU 的可用 RAM 区通常都很小, 因此对 MLO 文件的大小有严格的限制. 我们使用的 BeagleBone Black 开发板对应的 AM335x 1GHz ARM Cortex-A8 型号的 CPU 可用于存储 MLO 的最大空间为 109KB[15]. 在实践中, 我们分别测试了在 SPL 阶段加入 sha1 算法库和 sha256 算法库对 MLO 文件大小的影响. 实验结果图 6 所示.

图 6　MLO 大小

实验结果表明在 SPL 阶段加入 SHA1 算法库或 sha256 算法对文件造成的大小变化仍然在可接受的范围之内. 使用 SHA256 算法将使生成的 MLO 大小达到我们实验所用的开发板 BeagleBone Black 所能接受的最大值.

2) 引导程序性能

为了测试在引导阶段加入完整性验证对系统引导时间的影响, 我们测试了不同算法下 uboot 开始运行到操作系统内核开始执行的时间. 对每一种算法, 我们均测试了 10 次并分别记录了其最大值, 最小值, 平均值, 不同算法下其执行时间表 1 所示.

表 1　不同算法引导时间的影响(单位 s)

	平均值	最大值	最小值
正常	7.047	7.659	5.65
RSA1024_SHA1	13.453	13.665	12.644
RSA1024_SHA256	14.892	15.005	13.989
RSA2048_SHA1	13.178	13.596	11.564
RSA2048_SHA256	14.934	15.313	13.292

实验中我们使用的内核镜像大小为 4Mb 左右, 在不使用完整性保护时从第三阶段引导开始运行到内核镜像加载完成所需的时间约为 7s, 加入签名验证的过程后会造成大概 6 到 8 秒的附加引导时间开销, 而 sha256 比 sha1 算法需要大约多使用 2s 的时间, 其中 RSA 算法签名长度的选择对引导时间基本无影响. 实验过程中从嵌入式开发板上电到操作系统开始运行的总时间开销大约为 26s, 故增加 6s 到 8s 的引导时间实现对内核的完整性保护是完全可以接受的.

3) 程序度量时间

本文方案要求软件在加载时计算其完整性度量值信息, 并以此作为其主体名称, 在验证时由服务提供商校验软件的完整性是否被破坏. 我们分别测试了使用 sha1 算法和 sha256 算法时 BAM 模块对不同大小文件的度量所需的时间, 结果图 7 所示.

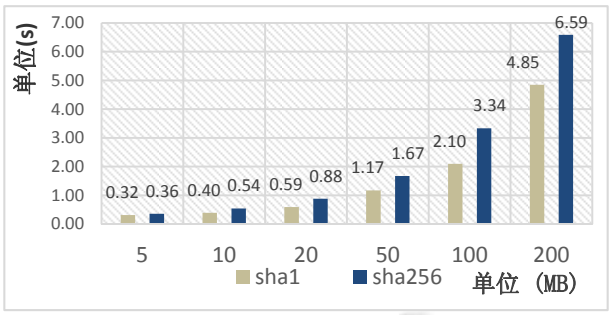

图 7　度量时间

分析上图可知, 程序启动时度量完整性会在一定程序上影响其启动时间. 对于 CPU 性能为 1GHZ 的 BeagleBone Black 开发板, 加载常见的 50MB 大小软件的时间延迟在 1s 左右.

4) 加入行为监控后对程序运行性能的影响

本文方案中, 内核层 BEM 模块根据内核中行为池中的内容对应用软件触发的软件进行审核, 确保相应软件不会触发在其行为定义列表之外的行为. 对行为的审核会对程序的运行性能造成一定影响. 本节测试 BEM 模块对程序运行效率的影响. 根据谷歌发布的智能手机用户研究报告[16], 智能手机中平均安装的 APP 个数在 20 到 50 之间, 我们分别测试了系统中运行有 20,30,40,50 个不同软件, 假定每个软件对应的行为列表条目数量为 500,1000,1500,2000 时, 实现行为检测的关键的函数 smk_access 所需的执行时间, 结果如表 2 所示.

表 2 行为判断时间(单位 ms)

	500 条规则	1000 条规则	1500 条规则	2000 条规则
20 个 app	0.046	0.143	0.380	0.651
30 个 app	0.045	0.143	0.381	0.698
40 个 app	0.046	0.141	0.380	0.697
50 个 app	0.051	0.144	0.382	0.699

分析表 2 可知, 运行的软件的个数对行为判断所需时间基本无影响. 当软件的行为条目增加时, 判定时间会相应延长, 但即使软件的行为条目达到 2000 条, 所需的行为判断时间也不足 1ms, 从而本文方案对程序运行的性能影响是可以接受的.

5.2 系统安全性分析

本文方案的主要目标为保证只有合法的用户在合适的运行环境下才能访问服务提供方的服务, 保障相应服务的安全运行, 防止恶意程序窃取用户个人隐私和数据. 以下分析本文方案针对常见攻击的防护能力:

Rootkit 攻击: Rootkit 指成功侵入目标系统的情况下, 保持对目标系统的最高访问权限. Rootkit 有内核级和应用级两种, 通过对内核或用户层程序的篡改实现程序的隐藏. 本文架构中, 内核和核心模块通过完整性度量进行保护, 对应用层程序有严格的行为监控, 从而能有效的遏制 Rootkit 攻击.

间谍软件: 如 Android 间谍软件 SpyEye, 在用户访问网络银行服务时潜藏在后台收集用户的银行账户资料和用户输入的账号密码等等. 针对这类后台监控类恶意软件, 本方案中服务提供方可以通过验证策略的设置禁止恶意软件在终端与其交互时访问和收集相关的信息, 从而能有效的保护用户的资料信息.

假冒攻击: 如 Android 系统目前存在软件市场混乱的问题, 用户从非官方下载的应用可能是被篡改过的恶意应用, 从而在与服务提供商交互时窃取用户信息和资料等等. 本文方案由于主体标签由系统对程序进行度量而生成, 并由服务提供方验证完整性度量值的正确性, 从而能有效的鉴别此类假冒软件.

TOCTOU 攻击: 远程证明中常见的问题是 TOCTOU 攻击, 即平台在证明时刻满足服务器的策略需求, 但在运行时状态被破坏. 本文通过运行凭证和系统状态绑定, 在运行时判定凭证有效性的方式, 能有效防止 TOCTOU 攻击.

6 结语

本文为移动智能终端设计了基于行为的远程证明方案. 通过对软件行为的定义和限制, 服务提供方自定义验证策略对终端环境进行验证, 有效应对了移动智能终端环境中软件频繁更新问题和多个服务提供方对终端运行环境有独立的需求的问题. 此方案能有效防御常见的 Rootkit, 间谍软件等类型的攻击. 我们设计完成了方案原型系统, 系统测试结果表明此方案对系统和软件运行性能的影响较小. 本方案适用于服务提供商对客户端的服务运行环境有一定要求的场景.

参考文献

1 冯登国,秦宇,汪丹,初晓博.可信计算技术研究.计算机研究与发展,2011,48(8):1332–1349.
2 陈婷,王永全.远程证明方法的研究综述.世界科技研究与发展,2009,31(6):1069–1073.
3 Sailer R, Zhang X, Jaeger T, Van Doorn L. Design and implementation of a TCG-based integrity measurement architecture. Proc. of the USENIX Security Symposium. 2004.
4 Nauman M, Khan S, Zhang X, Seifert JP. Beyond kernel-level integrity measurement: Enabling remote attestation for the android platform. Trust and Trustworthy Computing. Springer. 2010: 1–15.
5 Zhang X, Acıiçmez O, Seifert J-P. Building efficient integrity measurement and attestation for mobile phone platforms. Security and Privacy in Mobile Information and Communication Systems. Springer. 2009: 71–82.
6 Sadeghi AR, Stüble C. Property-based attestation for computing platforms: caring about properties, not mechanisms. Proc. of the 2004 Workshop on New Security Paradigms. 2004.
7 秦宇,冯登国.基于组件属性的远程证明.软件学报,2009, 20(6):1625–1641.
8 Chen L, Löhr H, Manulis M, Sadeghi AR. Property-based attestation without a trusted third party. Information Security. Springer. 2008: 31–46.
9 Kostiainen K, Asokan N, Ekberg JE. Practical property-based attestation on mobile devices. Trust and Trustworthy Computing. Springer. 2011: 78–92.
10 李晓勇,左晓栋,沈昌祥.基于系统行为的计算平台可信证明.电子学报,2007,35(7):1234–1239.
11 Huanguo Z, Fan W. A behavior-based remote trust attestation model. Wuhan University Journal of Natural Sciences, 2006, 11(6): 1819–1822.
12 Khan S, Khan S, Nauman M, Ali T, Alam M. Realizing dynamic behavior attestation for mobile platforms. Proc. of the 7th International Conference on Frontiers of Information Technology, 2009.
13 TCG. TCG specification architecture overview. TCG Specification Revision, 2007: 1–24.
14 Maruyama H, Seliger F, Nagaratnam N, Ebringer T, Munetho S, Yoshihama S, Nakamura T: CiteseeTrusted platform on demand (TPod)[Technical Report], Submitted for Publication, 2004.
15 Instruments T. AM335x ARM Cortex-A8 Microprocessors (MPUs). Technical Reference Manual, 2013.
16 Google. Global Perspectives: The Smartphone User & The Mobile Marketer, 2011.