(19)中华人民共和国国家知识产权局

(12)发明专利

(10)授权公告号 CN 105706048 B
(45)授权公告日 2019.06.04

(21)申请号 201480052380.1

(22)申请日 2014.07.23

(65)同一申请的已公布的文献号
申请公布号 CN 105706048 A

(43)申请公布日 2016.06.22

(30)优先权数据
61/857656 2013.07.23 US

(85)PCT国际申请进入国家阶段日
2016.03.23

(86)PCT国际申请的申请数据
PCT/US2014/047830 2014.07.23

(87)PCT国际申请的公布数据
WO2015/013412 EN 2015.01.29

(73)专利权人 爱立信股份有限公司
地址 美国马萨诸塞州

(72)发明人 M.米克海洛夫 R.奈尔

(74)专利代理机构 中国专利代理(香港)有限公司 72001
代理人 杨美灵 张懿

(51)Int.Cl.
G06F 7/04(2006.01)

(56)对比文件
WO 2012106097 A2,2012.08.09,说明书第1页第31-33行,第2页第1-3行,第3页第27-31行,第5页第7-10行,第6页第5-11,24-31行,第7页第1-9,19-22行.
CN 102982257 A,2013.03.20,全文.
US 2005/0144439 A1,2005.06.30,全文.
US 2009/0276617 A1,2009.11.05,全文.
US 2008/0082828 A1,2008.04.03,全文.
US 2013/0152180 A1,2013.06.13,说明书第[0001],[0010]-[0036],[0041]-[0045]段,图1-4.
US 2012/0096560 A1,2012.04.19,全文.
US 2012/0303951 A1,2012.11.29,全文.

审查员 王黎

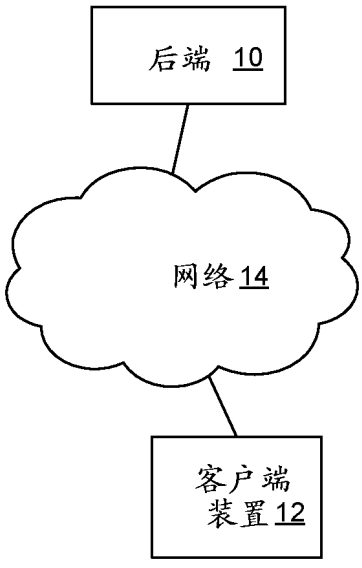
权利要求书2页 说明书13页 附图7页

(54)发明名称

使用硬件信任根的媒体客户端装置鉴权

(57)摘要

用于媒体重放的客户端装置包括实现数字权利管理(DRM)系统的客户端侧的用户可安装媒体客户端应用程序。客户端装置采用安全引导,并且检验由用户安装的应用程序。应用程序经增强以防止反向工程,并且它利用客户端装置提供的特殊API以绑定到安全引导,桥接在安全引导与应用程序内包含的DRM系统的客户端侧之间的间隙。



CN 105706048 B

1.一种用于证实真实性的方法,客户端装置根据所述方法向要在所述客户端装置上由用户安装的媒体客户端及以通信方式耦合到所述客户端装置的权利管理服务器证实其真实性,包括:

参与安全引导过程,以确认持久存储在所述客户端装置中并且包括固件以便执行的镜像明确加有密钥以便在权利管理方案中使用,所述权利管理方案采用在所述权利管理服务器的私有加密密钥和在所述镜像中安全存储的对应公共加密密钥,所述固件配置成且可操作以在执行时用于通过返回使用所述公共加密密钥加密的装置注册消息,响应来自所述媒体客户端的请求,所述安全引导过程包括(1)使用在所述客户端装置的一次性可编程(OTP)存储装置中安全存储的第一检验密钥,检验所述存储的公共加密密钥的签名,(2)使用在所述OTP存储装置中安全存储的一个或更多个第二检验密钥,将所述镜像中包含的加密对称密钥解密,并且检验所述解密的对称密钥的签名,以及(3)使用所述解密的对称密钥,检验所述持久存储镜像的签名;

在所述安全引导过程成功完成时,加载和执行所述固件;以及 在随后的操作期间通过所述固件,并且响应来自所述媒体客户端的请求,使用所述持久存储的公共加密密钥来创建所述加密的装置注册消息,并且将所述加密装置注册消息返回到所述媒体客户端,以便作为装置鉴权过程的一部分转发到所述权利管理服务器。

2.如权利要求1所述的方法,其中所述安全引导过程包括处理器级引导和使用所述镜像中包含的固件引导例程的随后固件级引导,所述固件级引导执行所述加密对称密钥的所述解密和所述持久存储镜像的所述签名的所述检验,并且还包括由所述处理器级引导检验所述固件引导例程的签名,并且在所述签名的检验成功时,随后启动所述固件引导。

3.如权利要求1所述的方法,其中所述镜像存储用于所述镜像的所述存储的公共加密密钥、所述加密对称密钥和固件组件的相应存储的签名,以及其中检验用于给定密钥或固件组件的签名包括(1)计算用于所述给定密钥或固件组件的签名,以及(2)比较所述计算的签名和所述存储的签名的一个相应签名。

4.如权利要求1所述的方法,其中所述媒体客户端实现数字权利管理(DRM)系统的客户端侧,并且在所述客户端装置上由用户安装。

5.如权利要求4所述的方法,其中所述客户端装置的所述固件检验所述客户端装置上所述由用户安装的媒体客户端。

6.如权利要求4所述的方法,其中所述媒体客户端利用所述客户端装置提供的特殊API以绑定到所述安全引导,桥接在所述安全引导与应用程序内包含的所述DRM系统的所述客户端侧之间的间隙。

7.一种客户端装置,包括:

一个或多个处理器;

存储器;

输入/输出接口电路;以及

一个或更多个数据总线,将所述处理器、存储器和输入/输出电路耦合在一起以便在其之间实现高速数据传送,

所述存储器存储由所述处理器执行的计算机程序指令,以促使所述客户端装置执行向要在所述客户端装置上由用户安装的媒体客户端及以通信方式耦合到所述客户端装置的

权利管理服务器提供其真实性的方法，所述方法包括：

参与安全引导过程，以确认持久存储在所述客户端装置中并且包括固件以便执行的镜像明确加有密钥以便在权利管理方案中使用，所述权利管理方案采用在所述权利管理服务器的私有加密密钥和在所述镜像中安全存储的对应公共加密密钥，所述固件配置成且可操作以在执行时用于通过返回使用所述公共加密密钥加密的装置注册消息，响应来自所述媒体客户端的请求，所述安全引导过程包括(1)使用在所述客户端装置的一次性可编程(OTP)存储装置中安全存储的第一检验密钥，检验所述存储的公共加密密钥的签名，(2)使用在所述OTP存储装置中安全存储的一个或更多个第二检验密钥，将所述镜像中包含的加密对称密钥解密，并且检验所述解密的对称密钥的签名，以及(3)使用所述解密的对称密钥，检验所述持久存储镜像的签名；

在所述安全引导过程成功完成时，加载和执行所述固件；以及

在随后的操作期间通过所述固件，并且响应来自所述媒体客户端的请求，使用所述持久存储的公共加密密钥来创建所述加密的装置注册消息，并且将所述加密装置注册消息返回到所述媒体客户端，以便作为装置鉴权过程的一部分转发到所述权利管理服务器。

8.如权利要求7所述的客户端装置，其中所述安全引导过程包括处理器级引导和使用所述镜像中包含的固件引导例程的随后固件级引导，所述固件级引导执行所述加密对称密钥的所述解密和所述持久存储镜像的所述签名的所述检验，并且其中所述方法还包括由所述处理器级引导检验所述固件引导例程的签名，并且在所述签名的检验成功时，随后启动所述固件引导。

9.如权利要求7所述的客户端装置，其中所述镜像存储用于所述镜像的所述存储的公共加密密钥、所述加密对称密钥和固件组件的相应存储的签名，其中检验用于给定密钥或固件组件的签名包括(1)计算用于所述给定密钥或固件组件的签名，以及(2)比较所述计算的签名和所述存储的签名的一个相应签名。

10.如权利要求7所述的客户端装置，其中所述媒体客户端实现数字权利管理(DRM)系统的客户端侧，并且在所述客户端装置上由用户安装。

11.如权利要求10所述的客户端装置，其中所述客户端装置的所述固件检验所述客户端装置上所述由用户安装的媒体客户端。

12.如权利要求10或11所述的客户端装置，其中所述媒体客户端利用所述客户端装置提供的特殊API以绑定到所述安全引导，桥接在所述安全引导与应用程序内包含的所述DRM系统的所述客户端侧之间的间隙。

使用硬件信任根的媒体客户端装置鉴权

发明内容

[0001] 本文公开了用于经在客户端装置上安装的应用程序,在客户端装置与服务器之间建立信任的技术。客户端装置适用于再现或重放媒体。它能够是具有IP连接的机顶盒(STB)。它也能够是诸如智能电话等移动装置或任何其它类型的客户端装置。在客户端装置上运行的操作系统能够是Android、Linux、Windows 8或任何其它操作系统。

[0002] 客户端装置具有烧录到硬件中的一个或更多个机密值,使得它们始终存在,并且不能被移除或改变。安全引导过程依赖这些秘密值以确保诸如另外的秘密、引导加载程序、操作系统本身及其各种组件等持久、非易失性存储的某些组件能够在引导时被检验,并且能够示为是正版,如同在工厂或在经授权的升级过程期间安装的一样,并且未被篡改。

[0003] 一旦检验了装置和操作系统的完整性,应用程序(自动或由用户启动)便使用系统提供的特殊应用编程接口(API)在客户端装置与服务器之间建立信任,该API利用在装置上可用并且在安全引导期间被检验的秘密值。

[0004] 实现数字权利管理(DRM)系统的客户端侧的应用程序在客户端装置上可由用户安装/更新。客户端装置采用安全引导,并且检验由用户安装的应用程序。应用程序可经增强以防止反向工程,并且它使用客户端装置提供的特殊API以绑定到安全引导,桥接在安全引导与应用程序内包含的DRM系统的客户端侧之间的间隙。

附图说明

[0005] 从如附图所示本发明的特定实施例的以下描述中,将明白上述和其它目的、特征和优点,图中类似的标号表示在不同视图内的相同部分。图1是用于内容输送和重放的连网系统的框图;

[0006] 图2是计算机化装置的硬件组织的框图;

[0007] 图3是客户端装置的软件/固件组织的框图;

[0008] 图4是在客户端装置中安全处理器和闪存可编程(闪速)存储器的内容的示意图;

[0009] 图5是安全引导过程的高级流程图;

[0010] 图6是用于装置注册过程的消息流程图;

[0011] 图7是表示代码混淆技术中关系的图形;

[0012] 图8是用于代码混淆技术的消息流程图。

具体实施方式

[0013] 图1是示出用于存储,输送和播放诸如加密视频文件等受保护内容的连网系统的有关组件的简化视图。在此简化视图中,系统包括经网络14连接到客户端装置12的一组后端服务器或“后端”10。下面更详细地描述后端10。客户端装置12通常是具有包括加密内容文件的解密等重放能力的计算机化装置。客户端装置的示例包括个人计算机、平板计算机、智能电话等。用于将加密内容文件解密的解密密钥由后端10提供到客户端装置12。在如下更详细描述的操作中,客户端装置12向后端10进行自我鉴权,并且提供信息,从而建立其播

放识别的加密内容(例如,特定视频)的授权。后端10的响应是提供一个或更多个解密密钥,允许客户端装置12将用于视频的内容文件解密。客户端装置12从后端10的内容服务器(未示出)获得加密内容文件,使用解密密钥将文件解密,并且随后再现(播放)解密内容。

[0014] 后端10可使用一个或更多个服务器计算机实现,这些计算机可位于相同位置(例如,在数据中心),或者以某一方式在多个位置分布。一些或所有服务器可以是内容输送网络(CDN)的一部分。在操作中,可获得来自内容发布者的内容,并且随后将其分段以便实现向客户端装置12的基于段的输送。媒体准备引擎从后端10的数字权利管理(DRM)服务器获得加密/解密密钥,并且使用密钥将内容加密以便以加密形式存储和以后输送。后端10可采用权利服务器作为用于DRM有关操作和通信的焦点,在此情况下,DRM服务器可经更专门调整以适合使用适当网络协议的加密密钥生成、存储和检索。

[0015] 图2是计算机化装置的一般化示图,计算机化装置诸如可用于认识客户端装置12和后端10的服务器。它包括通过一个或更多个数据总线28耦合在一起的一个或更多个处理器20、存储器22、本地存储装置24和输入/输出(I/O)接口电路26。如本领域通常熟知的一样,I/O接口电路26将装置耦合到一个或更多个外部网络(如网络14)、另外的存储装置或系统及其它输入/输出装置。如本文中所述装置的系统级功能由执行计算机程序指令(软件)的硬件提供,指令一般存储在存储器22中,并且由处理器20检索和执行。本文中执行功能的软件组件的任何描述将被理解为在执行软件组件的指令时对计算机或计算机化装置的操作的简略引用。此外,图2中组件的集合可称为“处理电路”,并且在执行给定软件组件时可视为功能专业化电路,例如,在执行实现内容播放器功能的软件组件时的“播放器电路”。如下所述,客户端装置12包括用于安全性目的的更专业化硬件组织。

[0016] 在一个实施例中,客户端装置12具有适用于包括媒体输送和重放的DRM方面的敏感应用的专业化组织。具体而言,客户端装置12可在安全执行环境与普通或非安全环境之间划分电路和功能性。硬件组件可包括在非安全环境中的应用处理器和在安全环境中的单独安全处理器。非安全环境中的操作软件可包括操作系统(O/S)和内容播放器应用程序(称为“app”)。在一个实施例中,操作系统是用于移动装置的Android®操作系统。安全环境中的组件负责与后端10(图1)建立信任根,以允许客户端装置12获得用于将内容解密的解密密钥。安全环境包括安全内核和安全存储器。客户端装置也包括媒体客户端,媒体客户端将注册装置12的请求发送到后端10,获得用于媒体对象的重放的权利对象,以及执行允许媒体对象的解密和播放的其它功能。媒体客户端可具有相应地在安全与非安全环境之间划分的单独安全和非安全部分。

[0017] 在一个实施例中,客户端装置12的安全环境可采用所谓的TrustZone系列的组件,包括根据ARM体系结构实现的安全处理器及特别调整以适合安全性有关的使用的安全内核和安全存储器。建立信任根可部分基于在用于构建装置12(例如,移动电话手机)的电路板中嵌入的安全处理硬件提供的安全性特征。芯片集制造商提供硬件,并且装置制造商(OEM)加载某些固件(代码),诸如下面更详细描述的。

[0018] 图3从软件角度示出客户端装置12的组织,这也反映了在安全与非安全环境之间的上述划分。它包括媒体播放器30、媒体客户端32、操作系统(OS)内核34和安全固件36。媒体客户端32具有到后端10的功能连接38。在操作中,媒体播放器30在诸如显示器等客户端装置12的适合设施上再现诸如视频等媒体。如本领域通常熟知的一样,媒体播放器30也包

括允许用户控制媒体的选择和重放的图形用户接口。媒体客户端32执行与媒体的下载以便重放(再现)有关的各种功能,包括装置注册的总体控制、加密密钥的输送和媒体(内容)从网络14的下载。下面更具体描述装置注册功能性及OS内核34和安全固件36的有关功能性。

[0019] 图4示出在客户端装置12中的某些结构和数据项目,包括安全处理器40和非易失性闪存可编程(闪速)存储器42。在下述操作中,安全处理器40执行代码并且访问在闪存42中存储的数据项。然而,安全处理器40也具有相关更低级(硬件)组件和操作。具体而言,处理器40包括存储在重置或者接通关闭电源时自动执行的小处理器引导(PROC引导)例程44的只读存储器(ROM)。它也包括用于某些基本数据项的永久性硬件级存储的一次性可编程(OTP)存储46,包括根公共密钥(PuK)KEY1、BL根密钥KEY4和密钥划分(part.)密钥KEY5。OTP存储装置可使用例如在客户端装置12制造时选择性打开(“熔断”)的熔丝阵列实现。

[0020] 闪存42存储以下项及相应签名(SIG)50:

[0021] 1.外部根公共密钥48(KEY2)和签名50-1

[0022] 2.快速引导代码52和签名50-2

[0023] 3.加密对称密钥54(KEY2)和签名50-3

[0024] 4.引导变元56和签名50-4

[0025] 5.恢复代码58和签名50-5

[0026] 6.内核代码60和签名50-6

[0027] 7.系统代码62和签名50-7

[0028] 图5概括示出安全引导过程。

[0029] 在70,处理器引导代码44使用以下特定操作检验KEY2的签名50-1:

[0030] 1.计算外部根公共密钥(KEY2)的校验和的安全哈希H2(例如,SHA-256哈希)

[0031] 2.读取OTP 46中的根公共密钥(KEY1),并且使用KEY1将KEY2的签名50-1解密,并且与H2进行比较

[0032] 在72,处理器引导代码44使用KEY2检验快速引导镜像52的签名50-2。这通过使用KEY2将签名50-2解密,并且将结果与快速引导镜像52的计算的安全哈希进行比较来完成。

[0033] 在此阶段,快速引导镜像52加载到存储器中并且开始执行。由于随后的步骤要求访问OTP 46,这只能够由安全处理器进行,因此,快速引导由安全处理器执行。

[0034] 在74,将加密KEY3 54解密,并且检验其签名。使用以下步骤:

[0035] a.首先,使用在OTP 46中存储的BL根密钥KEY4,将加密KEY3解密

[0036] b.接着,使用从KEY4和KEY5推导的中间密钥,计算解密的KEY3的AES CBC-MAC签名

[0037] c.使用密钥划分密钥KEY5,将KEY3的存储的签名50-3解密

[0038] d.比较步骤b和c的相应输出以检验KEY3的完整性

[0039] 在76,快速引导使用KEY3校验闪存42的剩余分区:

[0040] 1.校验引导变元56的签名50-4

[0041] 2.校验恢复代码58的签名50-5

[0042] 3.校验内核60的签名50-6

[0043] 4.校验系统代码62的签名50-7

[0044] 在上述安全引导过程后,控制传递到内核60。

[0045] 在一个实施例中,密钥KEY1和KEY2长度为2048个比特。它们可由诸如此专利的受

让人等DRM基础设施的特定提供商提供。诸如智能电话等否则常规的客户端装置12可能需要修改以只允许执行使用来自指定证书机构(包括来自受让人)的证书签名的app。

[0046] 图6示出如下的随后装置注册过程：

[0047] 1.媒体客户端32使用特殊内核调用请求并且获得装置序列号(S)。在一个实施例中,序列号S形成为manufacturer_id(4字节)+device_model(4字节)+batch_no(8字节)+serial_no(8字节)的级联

[0048] 2.媒体客户端32将包含以下内容的鉴权请求向上发送到后端10:S+App_Id(4字节)+Android_Time_Stamp(14字节)+random_nonce_1(16字节)

[0049] 3.后端10使用对应于外部根公共密钥(KEY2)的2048比特私有密钥,将收到的信息加密

[0050] 4.加密消息由后端10向下发送到装置12

[0051] 5.媒体客户端32请求来自内核34的KEY2

[0052] 6.媒体客户端32使用KEY2将来自后端10的消息解密,并且随后检验时戳、随机数等。

[0053] 7.媒体客户端32请求来自内核34的装置注册消息

[0054] 8.内核34要求安全固件36创建装置注册消息,在一个实施例中,装置注册消息包括使用KEY2加密的(装置id令牌+'#'+Android_Time_Stamp+'#'+random_nonce_1),其中,装置id令牌是S+ChipID(4字节)+IN(24字节)的安全哈希,IN是个性化数字,它是芯片集制造商id(4字节)+batch_no(8字节)+serial_no(12字节)

[0055] 9.将装置注册消息发送到后端10

[0056] 10.后端通过对应于KEY2的私有密钥将消息解密,检验时戳和随机数,并且使用装置id令牌作为用于装置注册的装置标识。

[0057] 11.后端10将装置注册的结果返回到媒体客户端32

[0058] 用于App鉴权的鲁棒混淆(Robust Obfuscation)

[0059] 图7和8用于描述基于代码以难以复制的方式的混淆的诸如媒体客户端32等敏感应用程序的鉴权的技术。

[0060] App鉴权通过白盒解密器的组合将用于建立双向经鉴权安全信道的私有密钥解密来实现。可证为困难的混淆器被用于防止反向工程和延伸用于防篡改。混淆器使用与诸如3SAT等已知困难问题的组合的控制流程平整(flattening),3 SAT是能够简化成在平整的控制流程中的可达性问题的NP完全(NP-complete)。换而言之,知道控制流程相当于解决已知困难问题。

[0061] 混淆过程(在代码开发时完成)

[0062] 使用此过程混淆在许可密钥检索和媒体重放中涉及的媒体客户端代码32的部分,这包括由代码开发者在其开发环境中执行的以下步骤。

[0063] 步骤1:选择不透明判定簇。这些判定用于创建如下在步骤5中所示的条件代码。此类条件在后面描述的代码平整过程中使用,该过程采用线性代码并且创建大的分支结构。可用簇是：

[0064] 素数剩余根(RPR)：

[0065] 这定义如下-

[0066] 对于任何素数P＝4X+3，并且任何A＜P，判定生成模板为“{(A^(X+1))^2-A}%P”，其中，%表示模运算。这通过改变全部评估为0或假(FALSE)的A和X而生成判定簇。

[0067] 类似地，在P＝8X+5时，对应模板为“{((4*A)^(X+1)/2)^2-A}%P”。

[0068] 通常，这些判定具有“L-R”的形式，其中，L是((4*A)^(X+1)/2)^2，并且R是A。我们随机挑选带有相同素数剩余的此簇的两个实例，并且如在L-L′中一样使用两个左侧部分，其中，L′来自带有不同A′和P′的簇，其中，A%P＝A′%P′。很明显，L-L′评估为0。

[0069] 判定也可在表达式中混合，其中，A%P不等于A′%P′以便获得非零表达式。获得非零表达式的另一方式是使用如下所示始终不为真的恒等式。

[0070] 简单二次多项式(SOP)：

[0071] 这定义如下-

[0072] “$7*Y^2-1-X^2$”对于任何整数X和Y始终不为0。

[0073] 此外，此判定用作是非零表达式的L-R。此类型的判定用于引入条件。

[0074] 步骤2:随后，创建混淆变量的集合，这些变量随后用于构成图形结构的实例，如下步骤3中所述，该结构随后嵌入代码混淆中。图7中示出此类图形结构的示意图。可存在一些所选择数量的变量，诸如例如1024。通过使用带有CTR_DRBG的加密属性的掷币选择，将这些变量任意划分成两个组G_1和G_2。选择是基于生成的数的奇偶性;仅将奇数选入G_1中。G_1或G_2中的成员从未显露，并且仅为程序员所知。要注意的是，代码没有宣布任一集中成员的语句;如下所述，仅程序员使用成员构建由k个子句构成的3 SAT问题$3SAT_m$的实例。

[0075] $3SAT_m = (a \vee s_1 \vee t_1) \wedge (\neg t_1 \vee s_2 \vee t_2) \wedge (\neg t_2 \vee s_3 \vee t_3) \ldots (\neg t_{k-1} \vee s_k \vee \neg a)$

[0076] 其中，如用于下面的证明需要的一样，k＞3。

[0077] 其中，k是集G_1的大小，并且每个s_i从集G_1中抽取并且设为真(TRUE)。剩余的a和t_1到t_{k-1}设为在运行时计算的随机布尔值。此设置将始终满足$3SAT_m$。

[0078] 如下所示，使用不透明判定，隐藏变量的实际值。

[0079] 例如，我们使用RPR进行以下操作而不设置s_i＝TRUE_

[0080] 设置s_i＝1-({((4*A_i)^(X+1)/2)^2-A_i}%P_i)，其中，对于一些A_i&P_i，A_i＜P_i。

[0081] 使用相同生成模板将剩余字面值(literal)tj设成随机值，但如上所述，使用掷币过程选择Aj小于或大于Pj。

[0082] 字面值的计算由软件开发者在整个代码内分布。进行设置的分布是为了实现被认为对代码混淆重要的非局部性的属性。

[0083] 接着，使用$3SAT_m$的字面值，计算图形G。顶点能够着色k+1颜色，使得当且仅当$3SAT_m$满足时，没有边缘在其末端具有相同颜色。随后，由于s_i设为真满足了$3SAT_m$，因此，我们也知道图形是可着色(k+1)颜色。此知识仅为开发者所知，因此能够用于检查图形中随机边缘末端的颜色，以确定随后用于保证代码段的着色的状态。

[0084] 图形G构建如下：

[0085] • 每个字面值t_i、$\neg t_i$、S_j、a和$\neg a$是图形G中的节点

[0086] • 每个子句C_i是着色颜色i并且连接到不在C_i中的字面值的节点

[0087] • 每个节点S_i着色颜色i并且连接到所有其它s个节点和所有字面值t_j和$\neg t_i$，其中，$j \neq i$

[0088] 每个节点t_i和$\neg t_i$通过边缘连接。每个节点t_i着色颜色i，并且节点$\neg t_i$着色颜色k+1。类似地，节点a和$\neg a$通过边缘连接，并且分别着色颜色1和k+1。我们将t_i和S_i表示为“真”节点。要注意的是，在程序的执行期间，随机改变t_i和$\neg t_i$的值，并且视值分别为真或假而定，对应着色从i改变为到(k+1)。[注:我们将k+1颜色表示为假颜色。]

[0089] 论点:图形G可着色(k+1)颜色。

[0090] 证明：

[0091] 由于每个子句Q具有最多3个字面值，并且有至少4个子句，因此，对于至少一个j，每个字句必须连接到t_j和$\neg t_j$。因此，无子句能够着色k+1颜色。

[0092] 因此，如果在每个C_i中存在其颜色为i的真节点，则图形G可着色(k+1)颜色。由于我们知道S_i始终为真，因此，图形G可着色(k+1)颜色。[注意，每个子句必须着色从1到k颜色之一，即，真节点之一的颜色。]

[0093] 步骤3:接着，划开未混淆代码的各种部分发以便处理。这在代码创建期间手动进行。如接下来所述，使用在以后为代码平整读取的特殊注释指示符，标记这些部分。要注意的是，代码平整的过程将带来分支代码及嵌入的实例的问题，该问题使得恢复正确控制流程的过程等效于解决图形着色的一般实例-NP完全问题。实例使用上面在步骤2中描述的字面值构建。另外，如上所述，即使带有随机颜色指派和t个节点的重新链接，实例也可着色k+1颜色。换而言之，由于如图所示的对称性，节点能够在图形G中的组2与3之间迁移。通过使用下面定义的一些着色安全操纵，在混淆中利用此属性-

[0094] 着色安全操纵

[0095] 1.在随机时间，在组2与3和组1与C之间交换节点，并且相应地调整指针；

[0096] 2.以与着色同构的方式跨组1、2、3和C及在1内移动指针

[0097] 注:操纵应只寻址图形中的特定节点，并且用于节点选择的逻辑在代码创建时完成。换而言之，组成员在代码中从不明确显示。

[0098] 在实现中，通过在节点与图形G的组之间的随机映射，在阵列外分配节点。仅代码混淆生成器程序知道3-SAT实例。混淆的程序只具有(k+1)色实例图形G。换而言之，G的解的可用性只为开发者所知，并且从未向监视器显露。

[0099] 步骤4:通过将语句的线性序列变换成分支形式，实现代码平整。因此，从以下序列开始

[0100] XI;X2;X3;...;Xn

[0101] 注:如下所示，通过使用诸如来自SQP等的不透明判定，可引入死码以增大段的长度：

[0102]

"如果(L-R) == 0，则使用生成模板，如更早所述地将任何字面值设成随机值。"

得到以下结构 -

条件指派 S = {Li, Lj, ... } ((color(t_a) == color(not t_a)) (color(t_m) ==color(s_n)) ...))

/** 这意味着 if(color(t_a) == color(not t_a))，then S = Li;

if (color(t_m) == color(t_n)) then S = Lj; 并以此类推

这些值中仅一个值将产生 S 的正确值 **/

L: Switch (S)

{

范例 1: 由不透明判定保护(

X1;

经条件指派，计算新 S;

转到 L;

范例 2: 由不透明判定保护(

X2;

经条件指派，计算新 S;

转到 L;

等等

}

示例: 采用要混淆的以下代码:

Func_123 (a, b, c):

这通过死码扩展如下:

XI : If (Ll-Rl) then

{

```
            Func_123 (a, c, b) /**      类似但稍微不同于在 Xn 下的
正确项 **/
            设置 s_k 成 1-L2-R2
        }
X2: if (L2-R2) then
        {
        Func_123 (a, b, b)
        设置 s_m 成 L5-R5
        }
  ...等等...
Xn      if (Ln==Rn) then /** 其中，n 是比如 3 的倍数，即，n = 3k **/
        {
            Func_123(a,b,c); /** 原代码 **/
            设置 s_x 成 Lm-Rm
        }
```

[0103] 这变得平整成以下结构。

```
/** 将 n 个执行步骤分成 m <= n 个序列。**/
/** 随机选择 N 个标签的集 C，并且选择这些标签的置换
P=Pl ..Pm **/
/** 现在，将随机置换 R（嵌入 P）挑选为在平整的结构中
“执行”的顺序 **/
/** 要注意的是，每次通过结构只执行 m 个序列之一 **/
设置起始标签 S = 条件指派 {004, 120, ... } (颜色 tl == 颜
色 t5, 颜色 C50 != 颜色 t50, ...)
/** 这设置 S 为 120，但静态分析不能确定该情况 **/
L: switch (S) {
        范例 0:
        {
            /** 对随后 S 值的条件指派后的分段 **/
```

```
            }
            ***等等***
            范例 1:
            {
                  /** 如果 S 值为假，则这就是动作，在此情
                  况下，它是欺骗 **/
                  X5' /** 进行模仿序列 X5 的操作 **/
                  挑选一些节点 t_i，并且随机改变它们和调整
                  指针
                  S 的条件指派
                  转到 L
            }
            ***等等***
            范例 P1: /** 用于 clause_num(Pl)的正确情况 **/
            { XI /** 如果这是用于此颜色的正确序列 **/
            S 的条件指派
            转到 L
            }
            }
      }
   ***等等以涵盖直到 N 个范例 **
   }
```

[0104]

[0105] 如示例中所示，选择了N个标签的特定置换。从置换中的第一标签开始，计算对应于用于交换到其范例主体的开关变量的正确设置的值，该主体计算生成在置换中下一元素的正确设置的值的代码，并以此类推。此变量的设置由条件代码保护，该条件代码测试由3SATm实例映射的图形的节点的颜色设置。只有在知道首色问题的解的设置的情况下才使得能正确选择范例，而无需求助于暴力搜索。

[0106] 因此，仅范例语句内的计算知道将执行的下一地址的值(范例的标签)。此外，如下所述，每个这些范例计算另一受保护语句内下一标签的值。有N个范例，其中，在每个循环上，我们设置下一范例标签，并且只对于正确的组合才将保证用于k种颜色的每个颜色的正确设置。对于标签的错误设置，范例语句将计算用于下一迭代的S的错误值。此外，只存在对标签的值的一次正确遍历，并且我们将在以后证实，正确的遍历是着色问题且又是3 SAT问题的解的知识的证明。换而言之，唯一的另一方式是随机猜测，这是NP困难(NP-hard)。

[0107] 步骤5:通过使用以下结构,实现使用不透明判定(X)的保护。

```
If (A-B = 0) then
    {
        要混淆的代码
    }
  else
        {
                看上去像正确代码，但有使其不正确的小变化
的欺骗代码。
        }
```

[0108] 备选形式为,

```
If (A-B+C = 0) then /** 其中 C 是来自 G1 的字面值之一 **/
        {
            要混淆的代码
        }
  else
        {
            看上去像正确代码，有使其不正确的小变化的
欺骗代码。
        }
```

[0109] 鉴权协议

[0110] 鉴权协议在图8中概括示出并且描述如下:

[0111] 1.开发者创建代码,包括:

[0112] a.使用128比特AES密钥(WB-AES-Key-2)加密的2048比特公共密钥(PUB)(2048比特长,并且在开发服务器上使用/dev/random生成)以及

[0113] b.白盒解密器(其密钥是WB-AES-Key-2)

[0114] [注:如下面解释的一样,将公共密钥加密的原因是确保其真实性。]

[0115] 2.[下述内容可具有针对Android®平台的细节]在匹配开发服务器的安全性准则的安全服务器上,使用/dev/random生成对APP特定的2048比特长私有签名密钥(APP_PRK)。对应的公共密钥(APP_PUB)用于生成使用APP_PRK签名(即,自签名)的X.509证书(APP_CERT)。通过使用PGP以确保传送的安全,此证书经电子邮件安全地输送到开发者。APP_CERT使用WB-AES-Key-2加密,并且存储在AMC中。

[0116] 3.app如上所述混淆,并且如下一步骤中所示,准备经app商店分发。

[0117] 4.[也用于Android]内容提供商(或代表内容提供商的app开发者)使用其Google

Play Android Developer Console帐户提交签名的app。签名的证书对提交者的id进行鉴权。内容提供商/app开发者必须使用“Keytool”实用程序在“KeyStore”中存储APP_CERT。最后，使用APP_CERT封装app，以便使用“jarsigner”实用程序经Google Play app商店分发，该实用程序将KeyStore和密钥别名作为输入，并且将APP_CERT插入app包中。

[0118] 5.[也用于Android]用户在其装置上下载，安装和启动app。在安装时，Android OS校验附连到App的证书的有效性。这作为安装过程的一部分进行。另外，如接下来所述，我们执行证书的真实性检验。

[0119] 6.对于app的每次启动，必须检验附连到app的证书。app的实例(也称为“客户端”)启动并且使用白盒解密器以将在AMC中存储的加密APP_CERT解密。从此解密字符串中，客户端如下所示使用(我们信任的)APP_PUB*对app的签名进行鉴权。

[0120] 如下所示，通过经packageInfo api获得签名，验证附连到app的证书。首先，比较证书中的公共密钥和APP_PUB*，并且如果它们匹配，则继续检验签名。使用APP_PUB*将附连到证书的签名解密，以获得随后能够直接检验的证书的校验和的加密哈希。如果此签名通过鉴权，则客户端继续与后端设置相互鉴权的SSL会话。要注意的是，每次启动app时进行此步骤。

[0121] [也用于Android]PackageInfo.signatures api可用于从app签名获得公共密钥。它只用于对开发者的id进行鉴权，并且能够与APP_PUB*的受信任值进行比较。

[0122] public void onCreate(Bundle savedInstanceState){

[0123] super.onCreate(savedInstanceState);

[0124] PackageManager pm＝this.getPackageManager();

[0125] String packageName＝this.getPackageName();

[0126] int field＝PackageManager.GET_SIGNATURES;

[0127] PackageInfo packageInfo＝pm.getPackageInfo(packageName,field);

[0128] Signature[]signatures＝packageInfo.signatures;

[0129] //并且此处我们具有DER编码的X.509证书

[0130] byte[]certificate＝signatures[0].toByteArray();

[0131] }

[0132] 7.客户端使用由开发者通过在开发服务器使用熵的/dev/random生成的2048比特密钥签名的X.509证书(CLIENT CERT)。CLIENT CERT嵌在app中。通过使用AES-128白盒解密器将私有SSL密钥(SSL)解密并且使用它建立安全信道，实现双向经鉴权的安全信道的建立。SSL长度为2048比特，并且在开发服务器上使用熵的/dev/random生成。SSL通过白盒WB-AES-Key2(与文档中更早在“装置注册”部分下更早所述的相同)加密存储(加密是保护私有密钥SSL的机密性所必需的)，并且考虑到AES加密的强度和鲁棒混淆，将被视为是安全的。服务器使用其自己的证书与客户端进行鉴权。

[0133] 8.后端借助于经SSL的相互鉴权，对客户端进行鉴权。

[0134] 9.接着，客户端如上所述进行装置注册。可无需推送通知或推送随机数。

[0135] 10.后端随后向下发送使用WB-AES-Key-2加密的内容密钥。

[0136] 11.密钥仅在静态存储器中解密，并且从不在磁盘中未加密存储。此外，仅为内容解密将密钥解密，并且通过如下所述使用鲁棒、安全的比特模式进行改写，安全地擦除存储

器。

[0137] 12.WB_AES-Key-2和APP_PUB通过强制app升级进行更新。

[0138] 其它措施

[0139] 在使用后使用鲁棒、安全的比特模式,改写包含解密会话密钥的缓冲器。我们不得使用堆存储器,并且只使用本地缓冲器(即,在堆栈变量中),且在控制传递到例程外时安全地擦除它们。各种比特模式(0xF6、0x00、0xFF、随机、0x00、0xFF、随机)按顺序写入缓冲器中。未加密密钥从不外写到磁盘。

[0140] 图形着色的执行发现的等效物:

[0141] 技术的特征部分在于以下两个断定:

[0142] 1.执行语句X1...Xn的正确序列暗示基于G的着色的条件(在保护中使用)的正确答案的知识。

[0143] 2.有关通过平整的结构,经执行遇到的G的着色的条件的答案的正确集产生了正确的执行序列X1...Xn。

[0144] 第一断定从混淆的构造中得出。换而言之,正确的执行序列X1...Xm示出图形着色问题的实例G的解的先验知识。这是因为图形的结构和着色在检查的时间前未知。它在代码的不同部分不同,包括在序列X1...Xn内。在无解的先验知识的情况下,我们将要知道如何解决难以计算的一般着色问题。

[0145] 对于第二断定,显示与条件的正确设置有关系的任何执行序列必须产生执行的正确顺序已是足够。这将建立图形着色问题的执行发现的等效物,其中,后者的解已知为NP完全。

[0146] 基础:E=1。由于只有一个顺序用于此序列,因此,命题是平凡真(trivially true)。

[0147] 归纳假设(I.H.):假设命题对于一些E=m为真,其中,m>1,即,正确回答有关G的图形着色的条件的任何执行序列也是X1...Xn的正确执行。

[0148] 随后示出命题对于E=m+1成立。考虑其执行序列Y长度为m+1的问题的实例。

[0149] 序列Y的任何元素的结构如下所示-

[0150] A1 A2 A3 A4 A5 (1)

[0151] 其中

[0152] 1.A1和A3是由假不透明判定保护的欺骗代码的一个或更多个实例;

[0153] 2.A2是需要执行的实际代码;

[0154] 3.A4是switch语句的下一值S的条件指派;以及

[0155] 4.A5是GoTo

[0156] 假设Y的最后两个元素是y′和y″。现在,我们找到组合这些元素的新方式以便产生与前一执行序列Y的执行图同构(即,是X1...Xn的正确执行序列)的长度为m的执行。我们如下定义此类组合:

[0157] A1′A1″A2′A2″A3′A3″Combined(A4′,A4″)A5″ (2)

[0158] 其中,A分量分别属于由素数符号指示的y′和y″。

[0159] Combined(U,V)是从在U和V中单独条件的条件和对应指派语句的聚合产生的条件指派。很明显,上面的(2)中的组合具有与(1)相同的结构。因此,它能够替换元素y′和y″,产

生长度为m的执行序列。

[0160] 从I.H.中，我们知道其条件正确满足G的着色的任何此类执行序列(长度为m)是X1...Xn的正确执行。我们也知道此类序列的条件映射到Y的对应执行，其中，条件拆分为步骤y′和y″。我们只需要示出此执行序列也正确。

[0161] 具体而言，我们知道组合的序列正确。因此，我们现在需要示出分解成Y的原序列将如何保留执行的正确性。排除y″为空的情况，我们观察到如果在y″中有满足有关着色的问题的条件，则这些条件必须包括按构造执行的正确步骤。要注意的是，y′和y″的其它分量是着色安全的，即，它们不改变条件中颜色有关问题的结果。

[0162] 图8示出上述技术，包括以下操作：

[0163] 1.创建2048比特密钥对(APP_PUB,APP_PRK)；生成APP_CERT；

[0164] 2.X＝使用WB-AES-Key-2的加密(PUB)；Y＝用于WB-AES-Key-2的白盒解密器；将X和Y嵌入app中；使用3-SAT实例混淆app；

[0165] 3.发送混淆的app

[0166] 4.使用APP_CERT为app签名，并且提供到App Store

[0167] 5.下载；安装；启动app

[0168] 6.在启动时，AMC使用Y将X解密以获得APP_PUB*；使用APP_PUB*以验证app签名；使用Y将SSL解密

[0169] 7.创建与后端相互鉴权的SSL会话。发送带有独特id的装置注册请求。

[0170] 8.在播放请求时，要求后端提供内容密钥

[0171] 9.向下发送使用WB-AES-Key-2加密的内容密钥

[0172] 在图8中，用于项7-9的虚线指示带有2048比特密钥的SSL会话。

[0173] 简而言之，下述内容是本公开方法和设备的重要方面：

[0174] 实现DRM系统的客户端侧的应用程序在客户端装置上可由用户安装/更新

[0175] 客户端装置采用安全引导，并且检验由用户安装的应用程序

[0176] 应用程序经增强以防止反向工程

[0177] 应用程序利用客户端装置提供的特殊API以绑定到安全引导，桥接在安全引导与应用程序内包含的DRM系统的客户端侧之间的间隙。

[0178] 虽然本发明的各种实施例已特别示出和描述，但本领域的技术人员将理解，在不脱离如所附权利要求书定义的本发明的精神和范围的情况下，可在形式和细节上进行各种更改。

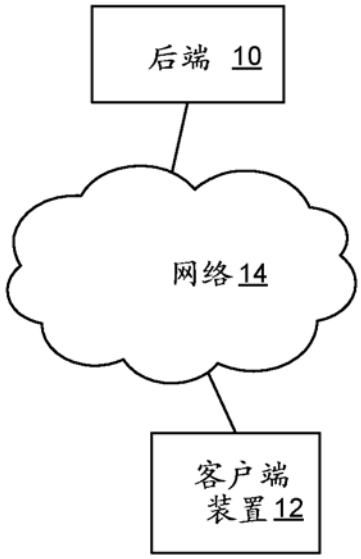

后端 10
网络 14
客户端
装置 12

图 1

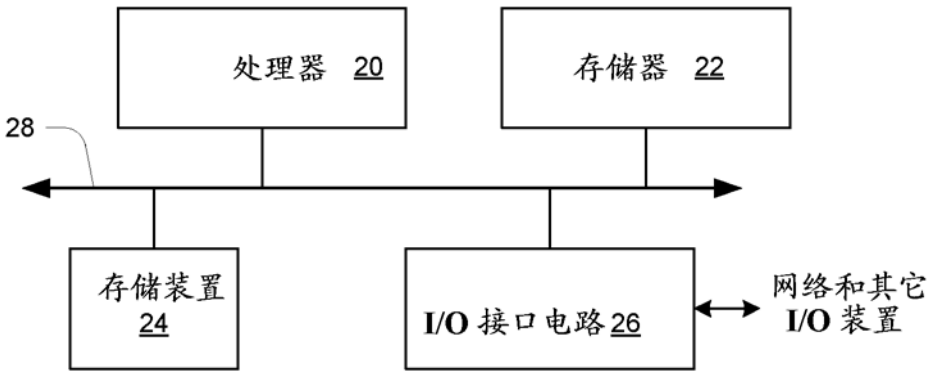

处理器 20
存储器 22
28
存储装置
24
I/O 接口电路 26
网络和其它
I/O 装置

图 2

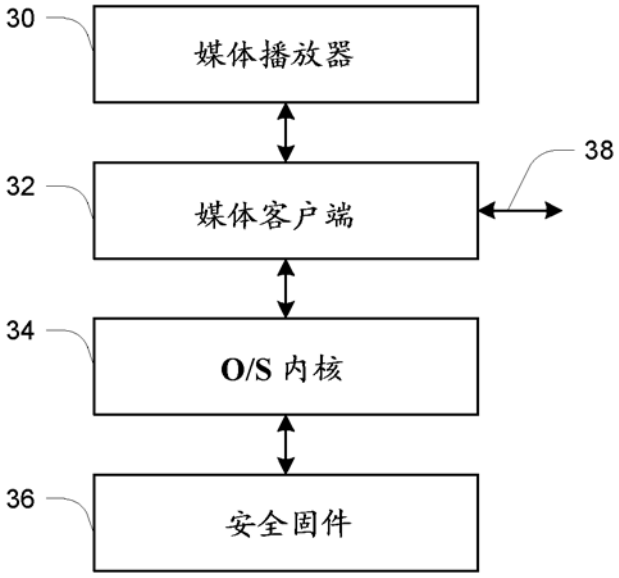


图 3

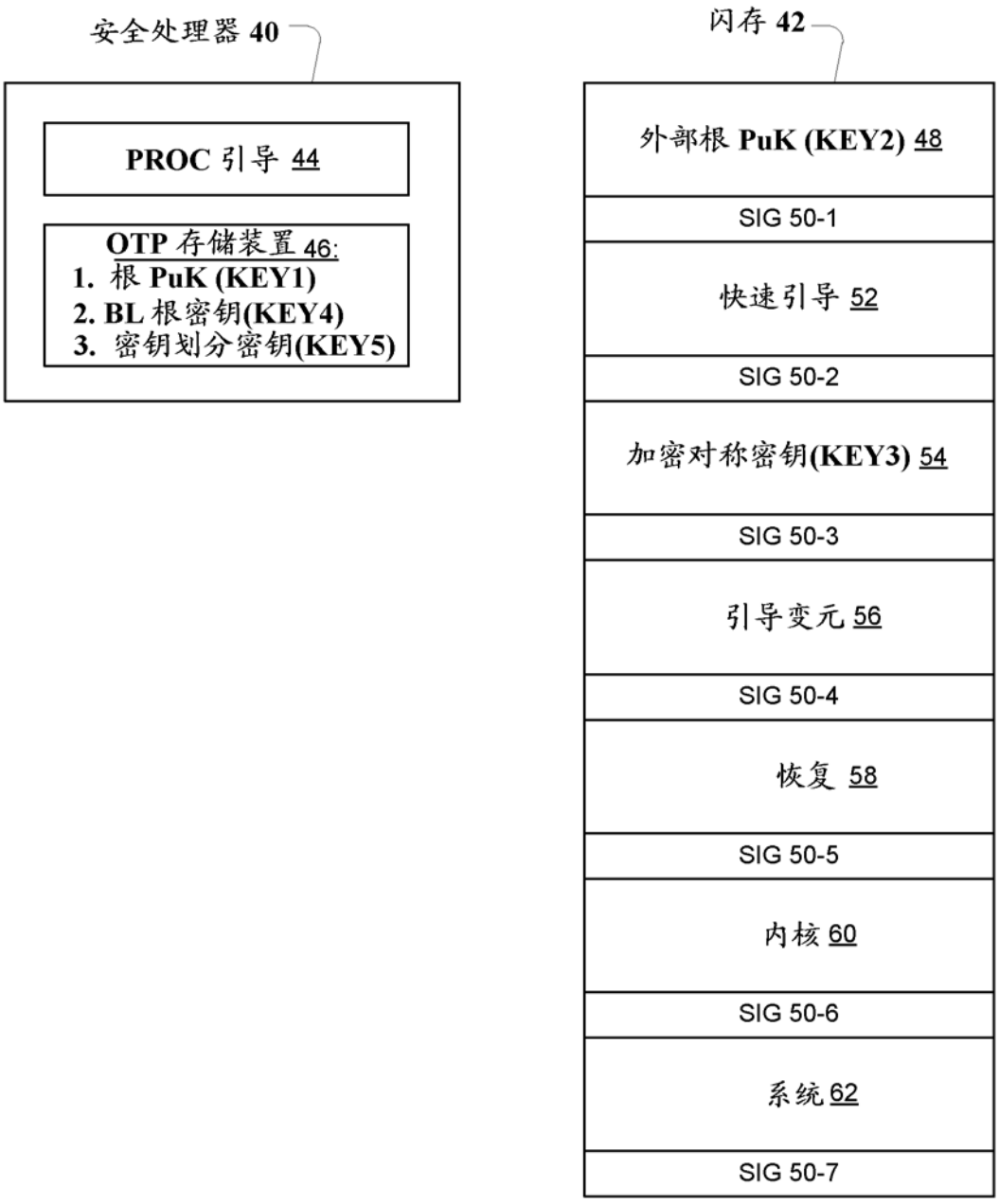

安全处理器 40
PROC 引导 44
OTP 存储装置 46:
1. 根 PuK (KEY1)
2. BL 根密钥(KEY4)
3. 密钥划分密钥(KEY5)
闪存 42
外部根 PuK (KEY2) 48
SIG 50-1
快速引导 52
SIG 50-2
加密对称密钥(KEY3) 54
SIG 50-3
引导变元 56
SIG 50-4
恢复 58
SIG 50-5
内核 60
SIG 50-6
系统 62
SIG 50-7

图 4

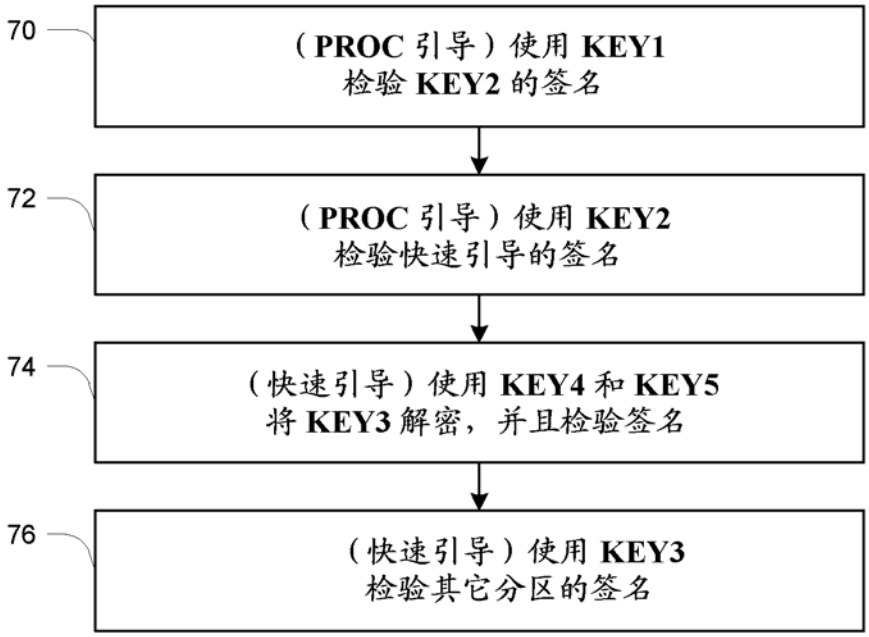

70
（PROC 引导）使用 KEY1
检验 KEY2 的签名
72
（PROC 引导）使用 KEY2
检验快速引导的签名
74
（快速引导）使用 KEY4 和 KEY5
将 KEY3 解密，并且检验签名
76
（快速引导）使用 KEY3
检验其它分区的签名

图 5

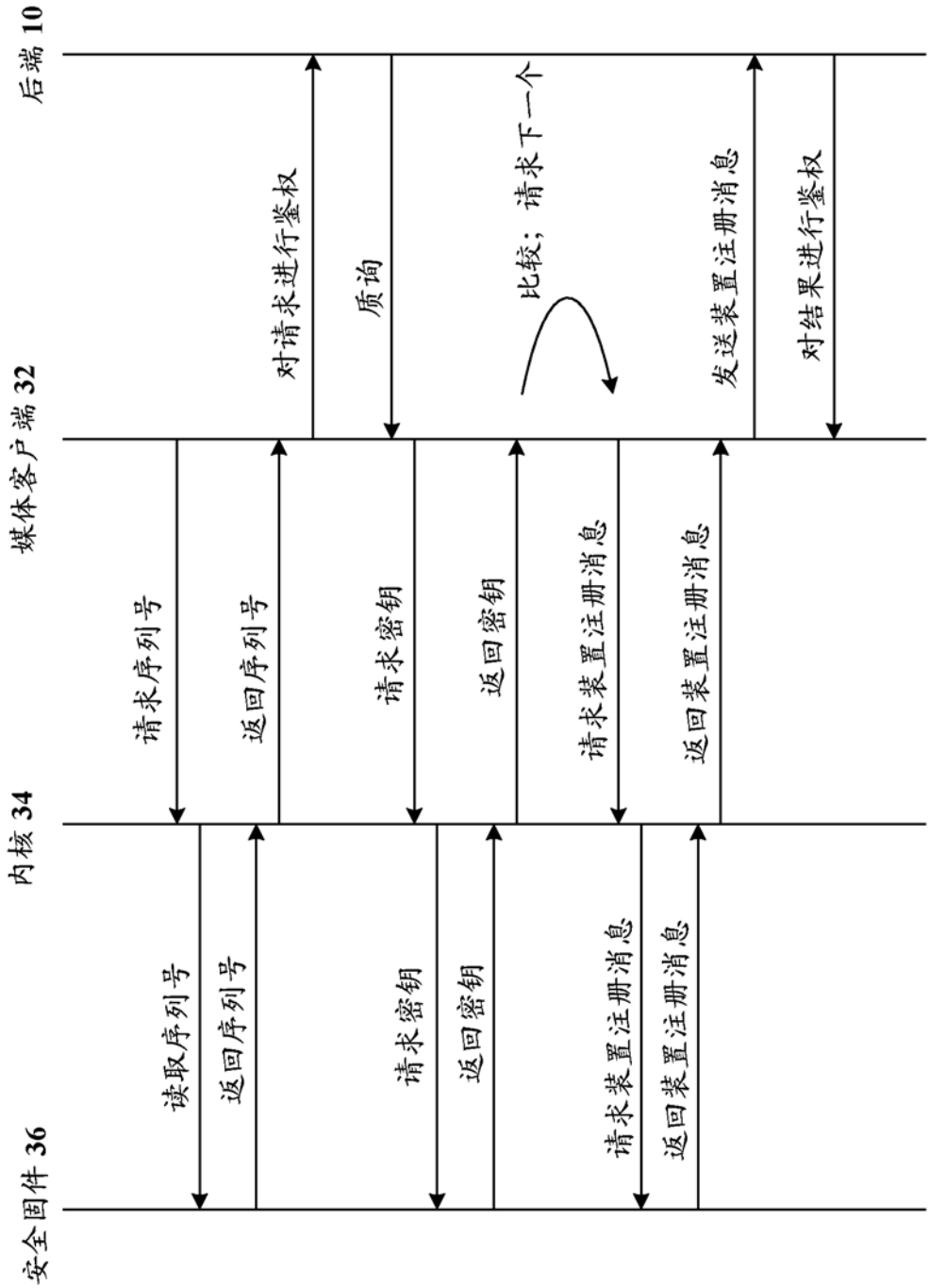

后端 10
媒体客户端 32
内核 34
安全固件 36
对请求进行鉴权
质询
比较；请求下一个
发送装置注册消息
对结果进行鉴权
请求序列号
返回序列号
请求密钥
返回密钥
请求装置注册消息
返回装置注册消息
读取序列号
返回序列号
请求密钥
返回密钥
请求装置注册消息
返回装置注册消息

图 6

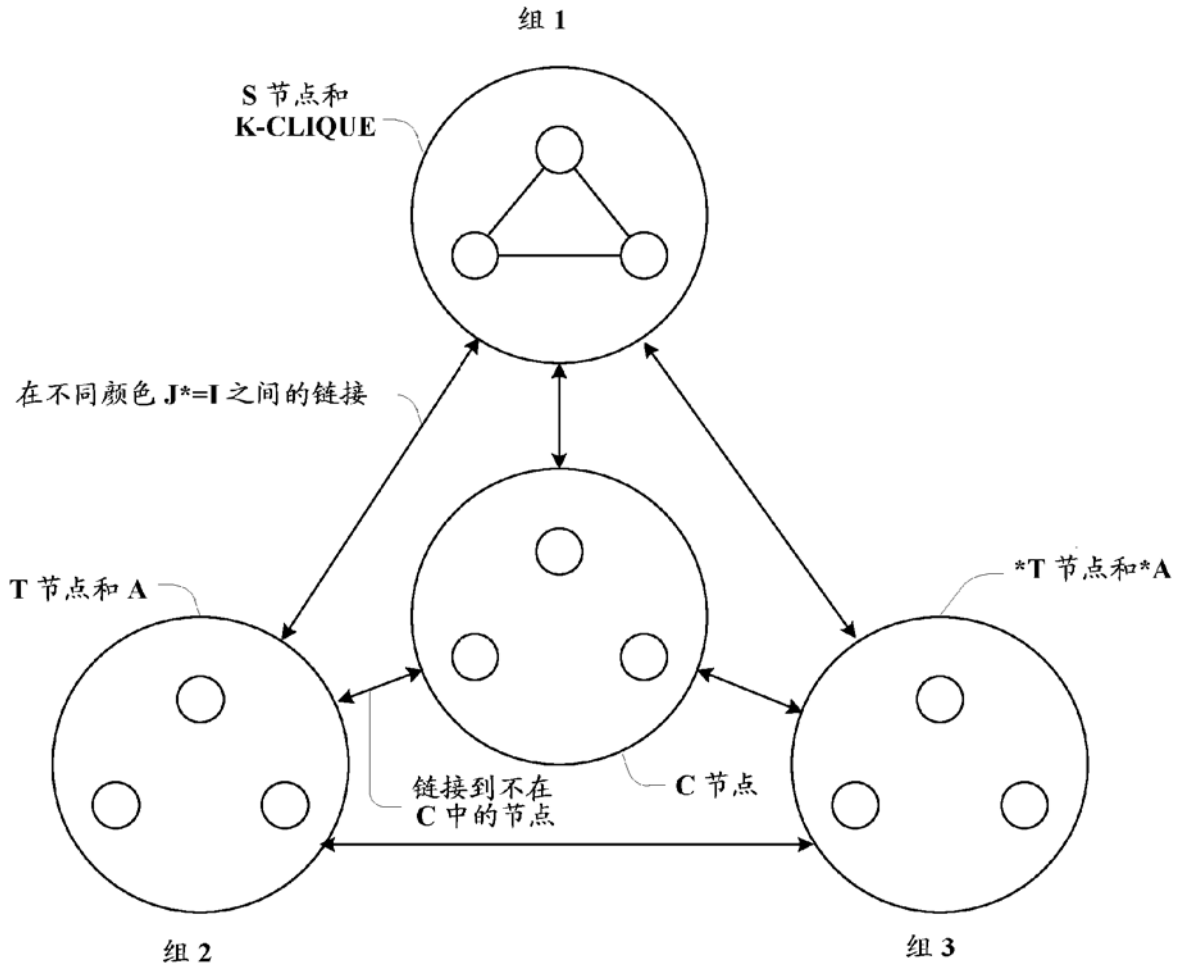

组 1
S 节点和
K-CLIQUE
在不同颜色 J*=I 之间的链接
T 节点和 A
*T 节点和*A
链接到不在
C 中的节点
C 节点
组 2
组 3

图 7

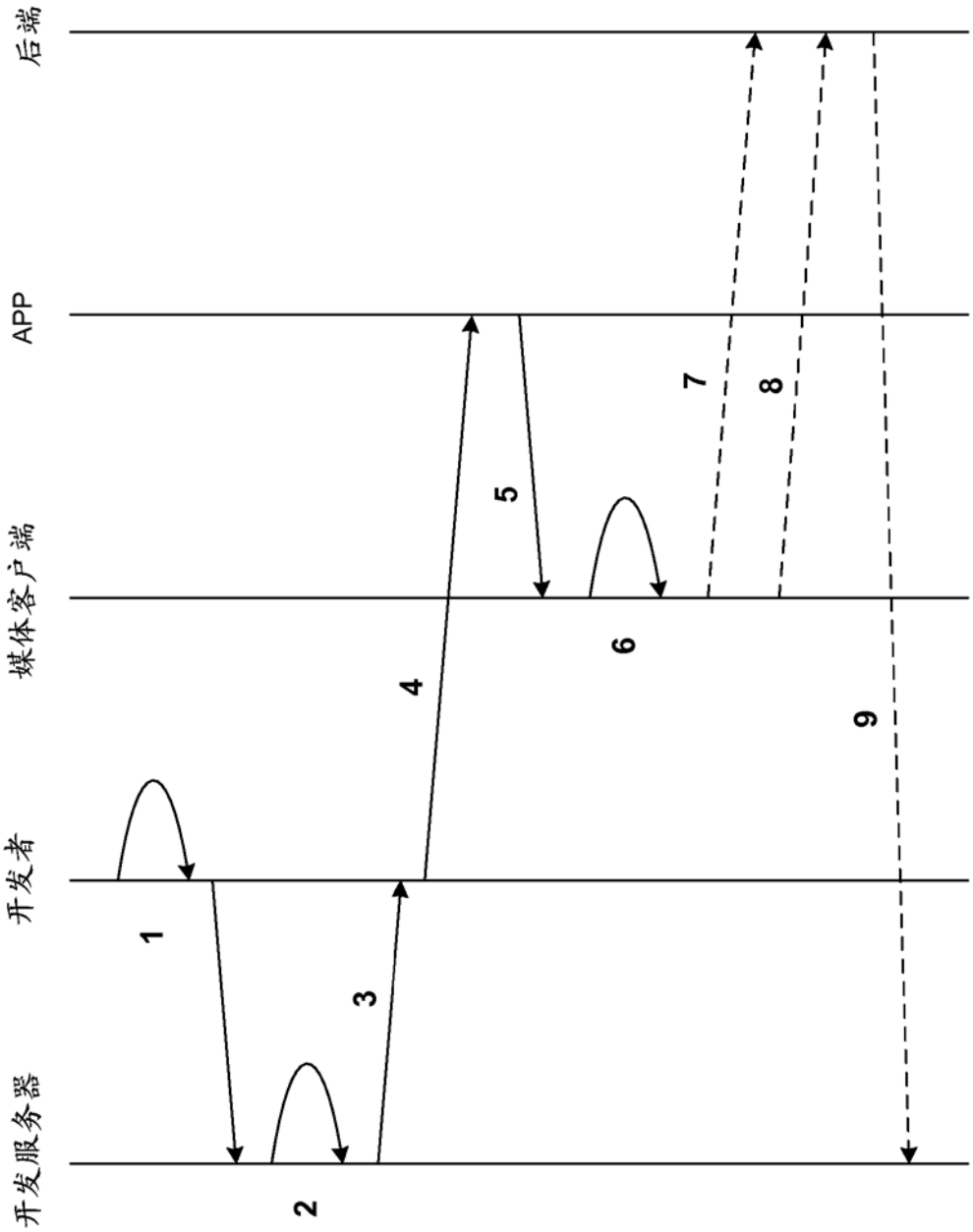


图 8