(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 103532707 B
(45)授权公告日 2018.01.26

(21)申请号 201310426022.X

(22)申请日 2008.05.16

(65)同一申请的已公布的文献号
申请公布号 CN 103532707 A

(43)申请公布日 2014.01.22

(30)优先权数据
60/938,994 2007.05.18 US

(62)分案原申请数据
200880016538.4 2008.05.16

(73)专利权人 美国唯美安视国际有限公司
地址 美国加利福尼亚

(72)发明人 R·T·库拉科斯基 C·莫特纳
J·B·费伊 J·布朗特
G·哈钦斯

(74)专利代理机构 中国国际贸易促进委员会专利商标事务所 11038
代理人 赵科

(51)Int.Cl.
H04L 9/18(2006.01)
H04L 29/06(2006.01)

(56)对比文件
CN 1137670 A,1996.12.11,
US 2001025340 A1,2001.09.27,
CN 1303065 A,2001.07.11,

审查员 肖小义

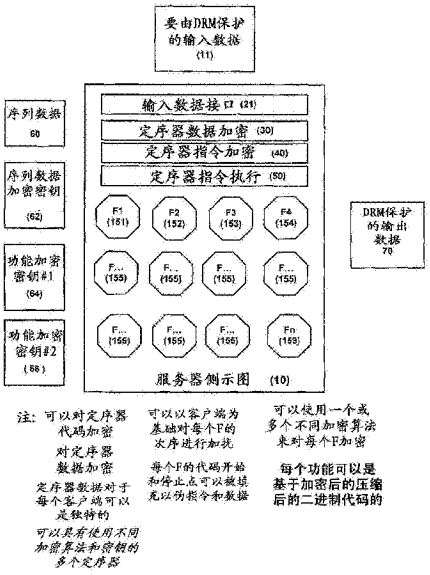
权利要求书2页 说明书23页 附图9页

(54)发明名称

用于确定在保护数据时应用的可编程处理步骤的系统和方法

(57)摘要

用于对正在客户端与服务器之间发送的数据进行保护的系统和方法具有定义在保护该数据时由服务器应用的可编程处理步骤和在不对数据进行保护时由客户端应用的相同步骤的能力。可以为每个客户端独有地定义可编程处理步骤，并且从使用定义了处理步骤的序列数据的多个功能中选择可编程处理步骤。可编程处理步骤允许每个客户端以不同的方式处理加密数据，并且由称为数字版权管理(DRM)的序列密钥来定义可编程处理步骤，这样该系统和方法引入可添加密钥的DRM，从而可以通过独有(或近似独有)的方式处理每个DRM消息。



CN 103532707 B

1.一种对要发送到多个客户端设备的数据进行保护的方法,所述方法包括:

接收待保护的输入数据;

生成定序器数据,所述定序器数据描述所选择的保护功能将被应用于所述输入数据的次序;

将定序器数据提供给每个客户端设备;

根据所述定序器数据描述的次序对所述输入数据依次应用所选择的保护功能,以生成受保护的输出数据;

将所述受保护的输出数据发送到所述多个客户端设备;

在所述多个客户端设备中的每个客户端设备处,基于相应的客户端设备根据定序器数据选择保护功能所使用的映射,映射定序器数据,其中,所述多个客户端设备中的每个客户端设备具有用于根据定序器数据选择保护功能的唯一映射;以及

将对应的、经映射的定序器数据发送到所述多个客户端设备中的每个客户端设备,

其中,被提供给每个客户端设备的定序器数据是相同的。

2.如权利要求1所述的方法,其中,所述定序器数据是使用有关客户端设备的信息生成的。

3.如权利要求2所述的方法,其中,所述有关客户端设备的信息包括有关在客户端设备中可用的保护功能的信息。

4.如权利要求3所述的方法,其中,通过将定序器数据发送至每个客户端设备来将定序器数据提供给每个客户端设备,并且所述方法还包括在定序器数据被发送到客户端设备之前对该定序器数据进行加密。

5.如权利要求1所述的方法,其中,多个保护功能中所选择的保护功能是动态的。

6.如权利要求1所述的方法,其中,定序器数据的映射是动态的。

7.一种对要发送到多个客户端设备的数据进行保护的方法,所述方法包括:

接收待保护的输入数据;

对于所述多个客户端设备中的每个客户端设备,生成定序器数据,所述定序器数据描述所选择的保护功能将被应用于所述输入数据的次序;

基于所述多个客户端设备中的每个客户端设备选择多个保护功能中所选择的保护功能将被应用于所述输入数据的次序所使用的映射,针对所述多个客户端设备中的相应的客户端设备映射定序器数据,其中,所述多个客户端设备中的每个客户端设备具有用于根据定序器数据选择保护功能的唯一映射;

根据经映射的定序器数据描述的次序对所述输入数据依次应用所选择的保护功能,以生成针对每个客户端设备的受保护的输出数据;

将相应的受保护的输出数据发送到所述多个客户端设备中的每个客户端设备;以及

将定序器数据发送到所述多个客户端设备,所述定序器数据用于由所述多个客户端设备中的每个客户端设备解密受保护的输出数据,

其中,被发送到每个客户端设备的定序器数据是相同的。

8.如权利要求7所述的方法,其中,所述定序器数据是使用有关客户端设备的信息生成的。

9.如权利要求8所述的方法,其中,所述有关客户端设备的信息包括有关在客户端设备

中可用的保护功能的信息。

10.如权利要求7所述的方法，还包括：在定序器数据被发送到客户端设备之前对该定序器数据进行加密。

11.如权利要求7所述的方法，其中，多个保护功能中所选择的保护功能是动态的。

12.如权利要求7所述的方法，其中，定序器数据的映射是动态的。

13.一种用于使用加密数据的计算机实现的系统，包括：

多个保护功能；

输入数据接口模块，用于接收受保护的数据和定义所述多个保护功能中的所选择的保护功能的顺序的定序器数据；以及

定序器指令执行模块，用于按照解密的定序器数据定义的顺序对受保护的数据依次地应用所述多个保护功能中所选择的保护功能，

其中，由定序器数据定义的顺序包括由定序器数据的一部分标识的功能，通过对定序器数据的至少一部分执行所述功能生成自修改定序器数据，并且所述自修改定序器数据需要调用正确的功能。

14.如权利要求13所述的系统，其中，所述多个保护功能中的至少一个被分解，并且分解的部分按加扰次序存储在存储器中。

15.一种计算机实现的加密系统，包括：

输入数据接口模块，用于接收待保护的输入数据；

多个保护功能；

定序器数据，所述定序器数据定义所述多个保护功能中所选择的保护功能的顺序；以及

定序器指令执行模块，被配置成通过按照所述定序器数据定义的顺序依次地应用所述多个保护功能中所选择的保护功能来对输入数据进行加密，以产生用于发送到客户端设备的受保护的输出数据，

其中，由定序器数据定义的顺序包括由定序器数据的一部分标识的功能，通过对定序器数据的至少一部分执行所述功能生成自修改定序器数据，并且所述自修改定序器数据需要调用正确的功能。

16.如权利要求15所述的系统，还包括定序器数据加密模块，所述定序器数据加密模块被配置成加密用于发送至客户端设备的定序器数据。

17.如权利要求16所述的系统，其中，在与发送受保护的输出数据的消息分离的消息中发送加密的定序器数据。

18.如权利要求15所述的系统，其中，所述客户端设备是多个客户端设备中所述计算机实现的加密系统为其产生受保护的输出数据的那个客户端设备，并且所述多个客户端设备中的每一个具有唯一的定序器数据。

用于确定在保护数据时应用的可编程处理步骤的系统和方法

[0001] 本申请是申请日为2008年5月16日的题为“用于确定在保护数据时应用的可编程处理步骤的系统和方法”的发明专利申请 200880016538.4的分案申请。

技术领域

[0002] 本发明涉及数据安全以及用于对在设备之间传输的数据进行加密和解密的系统和方法。

背景技术

[0003] 数字盗版和计算机黑客行为是一个公共问题，特别是现在，数字内容经常通过因特网被公众得到。数字盗版发生在内容创建地、内容准备地以及内容分发网络中。

[0004] 对于内容分发系统，诸如数字下载服务和有线电视公司来说，数字内容的盗版或窃取是值得注意的问题。存在一些可用来保护数字内容文件安全的安全技术，包括加密技术、水印技术等。尽管存在这些实施，由于黑客行为、窃取等原因，数字盗版依然存在。

[0005] 因此，需要一种系统和方法来减小或克服这些重大问题以及在如上所述的传统系统中存在的其他问题。

发明内容

[0006] 在此公开的一些实施例提供了用于对正在客户端与服务器之间发送的数据进行保护的系统和方法，具有确定在对数据进行保护时由服务器应用的可编程处理步骤的能力，并且相同步骤在对数据进行去保护(unprotecting)时由客户端应用。可以为每个客户端分别确定可编程处理步骤，并且可编程处理步骤从使用定义处理步骤的定序数据(sequencing data)的多个功能中选择。可编程处理步骤允许每个客户端以不同的方式处理加密数据，并且由所谓的数字版权管理(DRM) 定序密钥来确定可编程处理步骤，并且，该系统和方法引入支持密钥的(key-able)DRM，从而可以以独特(或准独特)的方式处理每个DRM消息。DRM序列密钥是用来定义处理步骤的序列的数据和对在此所描述的DRM处理过程所保护的输入数据执行的密钥数据。在本专利申请中，可互换地使用DRM序列密钥和DRM序列数据。

[0007] 在阅读以下的详细描述和附图之后，本发明的其他特征和优势对于本领域技术人员来说更明显。

附图说明

[0008] 图1是根据一个实施例的系统的框图。

[0009] 图2是图1的服务器10的功能框图。

[0010] 图3是在客户端设备上的客户端库结构的功能框图。

[0011] 图4是处理原语(processing primitive)的表示。

[0012] 图5示出了在由DRM处理过程来保护数据时所执行的处理步骤。

[0013] 图6是用于在DRM序列密钥被发送到客户端之前对DRM序列密钥进行加密的处理过程的流程图。

[0014] 图7A、图7B和图7C是示出了由三个不同客户端应用到输入数据的功能序列的表示。

[0015] 图8是用于对图7中的处理流程进行加密的处理过程的流程图。

[0016] 图9是用于将图8中所示的处理过程进行逆向处理的处理过程的流程图。

[0017] 图10是用于保护主内容加密密钥的处理过程的表示。

[0018] 图11是对一特定客户端独特地加密的客户端特定虚拟机或序列数据的简化框图。

[0019] 图12是三个功能的分解的图形表示。

具体实施方式

[0020] 在阅读本说明书之后，如何在各种可选实施例和可选应用中实现本发明对于本领域技术人员将变得明显。然而，尽管将在此描述本发明的多种实施例，但是应当理解，这些实施例仅以示例、而非限定方式给出。同样地，各种可选实施例的这种具体描述不应被解释为限制本发明的范围或宽度。

[0021] 在此公开的一些实施例提供了用于对正在客户端与服务器之间发送的数据进行保护的系统和方法，具有确定在保护该数据时由服务器应用的可编程处理步骤的能力，并且相同的步骤在对数据进行去保护时由客户端应用。可以独特地为每个客户端定义可编程处理步骤，并且从使用定义这些处理步骤的定序数据的多个功能中选择可编程处理步骤。可编程处理步骤允许每个客户端以不同的方式处理加密数据，并且由所谓的数字版权管理(DRM)定序密钥来确定可编程处理步骤，并且该系统和方法引入支持密钥的DRM，从而可以按独特(或准独特) 的方式处理每个DRM消息。DRM序列密钥是用来定义处理步骤的序列的数据和对由在此所描述的DRM处理过程保护的输入数据执行的密钥数据。在本专利申请中，可互换地使用DRM序列密钥和DRM序列数据。

[0022] 在本专利申请中，使用以下术语，这些术语具有以下含义。

[0023] DRM处理过程：数字版权管理(DRM)处理过程是被执行以保护正在客户端与服务器之间发送的数据的处理。DRM处理过程包括由服务器执行以保护(加密和置乱(obfuscate))数据、并且然后由客户端执行以去除或取消应用于数据的受保护处理的(可逆的、密码的和其他的)功能序列的应用。在此所描述的DRM处理过程是用于保护数据的步骤序列，诸如应用具有一个或多个密钥的加密算法、散列函数、数据转换步骤(线性或非线性)、数据映射函数(其中，输入数据被映射到不同的输出数据)、数据移位、数字置换、包括轮密钥 (round key)的功能(其中，算法的一个轮回获得一新密钥，或者从服务器数据得到轮密钥)、数据移位、异或运算(XOR)、位反转、位转换，应用多轮回加密算法的一个或多个轮回，以及其他用于保护数据的功能。

[0024] DRM处理过程可以由单个密码处理步骤组成或优选由多个密码处理步骤组成，多个处理步骤的应用包括使用一个或多个标准加密算法，该标准加密算法具有被应用以进一步对数据进行置乱的附加密码功能。DRM处理过程可以被应用于任何系统数据，包括用于数字内容或媒体的内容使用权信息、内容控制信息(单次播放，不允许复制，有效期，账单数据，电影访问标准等)、用于对加密内容进行解密的内容密钥；和在服务器与客户端之间任

一方向上发送的任何其他数据;或者被应用于在包括由DRM处理过程使用的数据的系统中所用的任何数据。

[0025] 状态表:术语“状态表”或定序器用于定义任何类型的软件结构、方法、或可用来定义指令序列、函数类或计算机设备执行的处理的数据。术语“状态机”、“虚拟机”、或“VM”、“VM指令集”、“定序器”、“控制环”、“编译器”、“解译器”等是在执行代码时处理软件指令的当前软件方法的示例,并且该处理由状态机定义、虚拟机指令集、软件设计、控制环的软件设计、编译器或解译器的设计等确定。在本专利申请中,可以使用任何形式的虚拟机、状态机、定序器、控制环、或任何其他结构来用由适合所用软件方法的状态表所定义的顺序步骤来对处理过程所执行的指令流进行定序。例如,用于虚拟机的状态表将不同于用于控制环或解译器的状态表;然而,用于每种方法的状态表定义用于该特定方法的软件定序,并且可以是状态表格式的,但并非必须为状态表格式的。应当想到的是,可以使用任何形式的顺序控制来定义处理时被执行或应用的计算步骤(或计算流程)。定义所执行的序列的一些控制或全部控制被共同地或单独地定义在虚拟机指令集、计算机指令集、状态机状态表、解译器指令集或控制定序数据中,并且,在本专利申请中这被称为状态表或状态机或虚拟机。可以利用对于所有客户端公共的密钥来对在此使用的任何实施例中的状态表(VM、定序器、字节码解译器等)进行进一步加密,或者可以针对每个客户端对状态表进行独特地加密。用来对实际上实现状态表的代码进行解密的状态表解密密钥可以是包含在客户端库代码中的独立密钥或者是与客户端库代码分开的密钥,或者其可以是 DRM定序密钥信息的一部分或者是客户端库的一部分或者与客户端库分开并且以任意方式传送。这意味着,DRM序列密钥还可以包括对加密后的状态表执行代码进行解密所需的密钥,使得客户端设备中的 CPU(中央处理单元)可以执行基于未加密的状态表的处理序列。在一个实施例中,可以存在一个或多个状态表或定序器,并且其中每一个可以都是不同的,并且可以具有不同的用于对用来实现状态表处理或定序器或VM的代码进行保护的加密算法和密钥,并且不同加密中的每一个都可以具有公共密钥、独有密钥或状态表独有密钥,并且这些密钥中的任何一个都可以在客户端库中,或者在客户端库之外或者这两者的组合,并且还可以被包含在DRM序列密钥数据中。事实上,可以使用多个密钥,并且状态表或VM内的指示器可以指出在对状态表或VM数据进行解密时应该使用哪个密钥,并且甚至也可以使用不同的加密算法。

[0026] 状态机:该术语是指多个软件方法中的任何一个,其或者使用表数据、指令集、虚拟机指令、编译器技术、控制环设计方法、解译器设计,或者使用定义由计算机的中央处理单元(CPU)所使用的指令序列的任何软件方法来定义软件处理序列。状态机的示例包括但不限于:C或C++的switch-case语句构造、任何类型的虚拟机、IF/Then类型软件控制流、字节码解译器、编译器(例如,C、C++、C#、Java、 Forth、Basic、Fortran等)、状态机定义语言、任意类型的解译器(例如,Java、C#、BASIC、Forth、Basic等),或可以执行处理序列或提供应用程序接口(API)(如Soap、.net、CGI等)的任何脚本控制方法,或者编程实用程序(诸如Ruby-on-Rails、Perl、PHP、JavaScript 等)。在本发明的各个实施例中的状态机可以利用对于所有客户端公共的密钥而被进一步加密,或者状态机可以对于每个客户端独特地加密。状态机解密密钥可以是独立密钥,或者其可以是DRM定序密钥信息的一部分,并且可以作为客户端库的一部分或者与客户端库分开地以广播或单播方法被传送,并且可以如上面的状态表部分中所定义地、或者利用任何类型的密钥控制(keying)方法或密钥交换方法来

使用一个或多个密钥。另外，在本发明中使用的任何实施例中的状态机 (VM，定序器，字节码解译器等)可以利用对于所有客户端公共的密钥而被进一步加密，或者可以针对每个客户端独特地对状态机进行加密。状态机解密密钥可以是独立密钥，或者其可以是DRM定序密钥信息的一部分，或者可以是客户端库的一部分或者与客户端库分开，与客户端库分开意味着与客户端库在不同的时间下载并且不包含作为客户端库的一部分。在有些实施例中，可以存在一个或多个状态机或定序器或VM，并且每一个都可以是不同的，并且可以具有不同的用来对用来实现状态机处理或定序器或VM的代码进行保护的加密，并且不同加密中的每一个都可以具有公共密钥或状态机独特密钥。事实上，可以使用多个密钥，并且状态机或VM内的指示器可以指出在对状态机或VM数据进行解密时应当使用哪个密钥，并且甚至还可以针对不同处理部分使用不同的加密算法。

[0027] 图1是根据一个实施例的系统的框图，其中服务器10和一个或多个客户端设备12a、12b和12c通过通信网络14通信。通信网络例如可以是局域网、广域网、因特网或这些网络与其他允许服务器10和客户端设备12交换信息的网络的组合。网络可以是有线和/或无线的。服务器 10可以是通用计算机或专用计算机或这样的计算机的组合。类似地，客户端设备12可以是机顶盒(STB)(通常用于有线和卫星电视系统)、包括允许通过网络进行通信的可得到的各种计算设备中任何设备的通用计算机或专用计算机，例如，家用计算机、笔记本计算机、个人数字助理或移动电话。

[0028] 使用所述系统和方法执行的DRM定序密钥或DRM序列密钥步骤对于系统中每个客户端可以是独特(或近似独特)的，其中服务器和客户端两者在利用DRM处理过程准备或处理(加密或解密)数据时将使用从对于客户端独特的DRM定序密钥或DRM处理序列密钥导出的对于客户端独特的处理步骤。当该系统和方法被应用于对在客户端与服务器之间发送的数据进行保护时，该系统和方法产生由对于客户端独特的一组DRM处理步骤所保护的加密的密钥数据(或任何类型的数据)。这可以利用下载到系统中所有客户端的单个公共代码来实现。用于每个客户端的DRM处理过程，或者甚至每个消息交换可以是独特的，因为能够利用DRM定序密钥配置被应用来对数据进行保护或加密的各个功能。在此描述的系统和方法允许从公共固件或者发送到所有客户端的软件代码图像导出对于客户端独特的DRM处理方法。这可以消除将独特的客户端固件图像分发到每个客户端设备的需要。可以包括产生每个客户端设备的独特的客户端软件代码图像(意味着，用于每个客户端设备的运行时软件代码图像)的软件方法。可以使用以下任一方法或这两种方法，即公共固件图像下载和客户端独特的固件下载图像。在许多广播网络中，如有线和卫星视频网络，将数百万的独特客户端传送到数百万的客户端设备(例如STB)所需的网络带宽和时间是不可得到的。然而，在双向网络(如因特网)中，以及在由每个客户端单独下载软件(例如初始设置)的情况下，传送到每个客户端的代码图像可以是独特的。在卫星和有线(单向)网络中，公共代码图像可以被发送到所有客户端。在客户端设备固件更新过程期间，客户端图像可以被客户端设备进一步加扰(scramble)，从而使得每个客户端代码图像是独特(或近乎独特)的。客户端对代码图像加扰的方式对于服务器是已知的，从而服务器可以对于各个客户端正确地处理输入数据。

[0029] 在一个示例应用中，在客户端设备启动时，没有客户端库DRM 处理流程数据(DRM处理过程或DRM定序密钥(定序器数据60)、或VM指令解密密钥、或状态机密钥控制数据)包

含在客户端库代码图像中。这意味着，在没有附加的数据被从服务器单独地发送到客户端库的情况下，客户端不知道如何对消息进行解密。用于客户端库的处理序列或软件代码流由序列数据60所定义的DRM定序密钥来定义，其中序列数据60包含虚拟指令数据、序列数据、状态表或将被称为DRM定序密钥的状态表密钥控制数据，DRM定序密钥在客户端加电或复位时，或者在客户端需要用于解密数据的DRM定序密钥时从服务器下载，或者DRM定序密钥信息被存储、并从智能卡中的数据来提供，或者通过网络安全地发送，或者通过其他任何数据交换方法获得，包括通过网络、从硬件获得、使用客户端特定的标识符数据的散列而获得等。由于不将DRM定序密钥存储在客户端固件代码图像中，所以不能利用静态代码分析工具(如IDAPro)对客户端库实施黑客行为，并且不存在电脑黑客可用来弄清客户端库的操作的运行时功能调用树形数据。

[0030] 图2是图1的服务器10的功能框图。输入数据11是要由DRM处理过程保护的数据。该数据可以是任何类型的数据，例如数字媒体内容 (诸如音乐或电影)或其他信息。输入数据接口模块21是用于接收将要保护的输入数据11的接口。定序器数据(或DRM序列密钥)60是定义关于如何执行功能F1(151)到Fn(159)的序列流的数据。定序器数据60是按与客户端设备中所使用的状态机或状态表的形式兼容的格式的，或者定序器数据60将是这样一种格式，即在通过功能或功能集转换时，该格式产生与客户端设备兼容的格式的数据。定序器数据60 也可以被用来修改包含在客户端设备中的实际代码(例如，客户端设备上以及如下所述的功能F1(151)到Fn(159)中的一个或多个)。这产生自修改的修改软件，其中使用DRM序列密钥或定序器数据60 来修改创建由DRM序列密钥60数据驱动的自修改代码的客户端库软件指令。自修改代码可以在运行时或在客户端库在安装过程期间被下载时使每个客户端是独特的。

[0031] 可以对定序器代码加密，对定序器数据加密。定序器数据对于每个客户端可以是独特的，可以具有使用不同加密算法和密钥的多个定序器。可以以客户端为基础对每个F的次序进行加扰，每个F的代码开始和停止点可以被填充以伪指令和数据。可以使用一个或多个不同加密算法来对每个F加密。每个功能可以是基于加密后的压缩后的二进制代码的。

[0032] 一实施例中的一可选步骤是在将序列数据60发送到客户端之前使用定序器数据加密模块30来对序列数据60进行加密。定序器数据加密30所使用的密钥可以是对于每个客户端是独特的，或者密钥可以是对于所有客户端或对于多组客户端是公共的，或者它们可以是对于每个客户端独特的和对于所有客户端或对于多组客户端公共的一些密钥的组合。用于定序器数据加密30的密钥可以被包括作为客户端库版本的一部分，或者它们可以在服务器与客户端之间以一单独的消息被发送，或者它们可以作为附加数据被添加到从服务器发送到客户端的消息。定序器数据加密30也可以被用来利用一个或多个加密密钥、或者利用一个或多个加密算法、或者利用这两者来对功能F1(151)到Fn (159)进行加密。可选地，可以包括如功能加密密钥#1(64)和功能加密密钥#2(66)所示的附加加密密钥来对功能F1(151)到Fn(159) 进行加密。在一个系统中可以使用一个、二个、或更多个功能加密密钥(64和66)，其中以任何方式(优选为加密地)将功能加密密钥传送到客户端，例如使用安全单播连接、通过单向信道、通过双向信道传送到客户端，或者被集成到客户端库代码中，或者被集成到能由客户端设备访问的硬件中，或者集成到智能卡或安全存储器中。

[0033] 定序器指令执行模块50是对功能F1(151)到Fn(159)的执行进行定序的代码。附加

的可选步骤是对定序器指令加密，并且这被表示为定序器指令加密模块40，并且该步骤对用于实现定序器50的实际代码进行加密。定序器指令加密40可以使用一个或多个密钥和加密方法来对定序器指令集和控制数据进行加密。

[0034] 功能F1(151)到Fn(159)表示一组用来根据序列数据60保护输入数据11的处理功能。这些功能是算法的一部分或者是整个算法，包括以下算法类型：加密算法、散列函数、数据转换步骤(线性和非线性)、数据映射函数(其中，输入数据被映射到不同的输出数据)、数据移位、数字置换、包括轮密钥的功能(其中算法的一个轮回获得新密钥，或者从服务器数据获得轮密钥，并且轮回是通过单或多传递算法的一个通过)、数据移位、异或运算(XOR)、位反转、位转换，应用多轮回加密算法的一个或多个轮回、以及其他用来保护数据、对 DRM处理进行置乱、添加附加安全性、混洗(shuffle)存储器中的数据、转换和/或混洗存储器中的数据的功能，以及其他功能。这些功能可以使用用于使用需要密钥的算法的一个或多个密钥，并且这些功能可以包括其他数据来在适当时为这些功能提供附加的可编程性。例如，当使用功能(如F3(153))来混洗数据时，执行混洗的方式可以基于功能自身内的数据，或者其他功能所提供的数据，或者服务器10所提供的数据，或者嵌入在客户端库内的数据。

[0035] 功能(功能处理)应用的输出是DRM保护的输出数据70。用于产生DRM保护的输出数据70的服务器侧处理可以总结如下：

[0036] 1.要由DRM保护的输入数据11被提供到输入数据接口21。

[0037] 2.产生定序器数据60，其包含以下数据，即该数据描述应该对功能(F1151到Fn159)序列如何进行定序来处理输入数据11的方式。定序器数据60也可以包含在处理时由功能 F1(151)到Fn(159)使用的密钥或其他数据。

[0038] 3.定序器数据60被用作定序器指令执行50的输入，来定义功能F1(151)到Fn(159)的执行序列。

[0039] 4.在完成定序器指令执行50后，处理后的数据被输出作为 DRM保护的输出数据70。

[0040] 上面的步骤3(执行定序器指令执行50)工作如下：

[0041] a.使用序列数据60的第一部分执行与定序器数据60的第一部分相关的功能。

[0042] b.使用定序器数据60的下一部分执行与定序器数据60的下一部分相关的功能。

[0043] c.重复上面的步骤b直到所有处理已经被执行。例如，如果定序器数据60由128位组成并且每个客户端具有16个功能组(4位)，则会有128/4＝32个功能被定序器选择。步骤1将使用序列数据60的4位来选择16个功能中的一个，步骤2 将使用序列数据60的下一4位来执行16个功能组中的第二个功能，等等。应当注意，如上所述的序列数据60的划分仅仅是定序器数据60可以如何被映射到处理执行的一个示例。存在本领域技术人员能实现这样的定序器的许多其他不同的方式。

[0044] 为了进一步说明定序器操作，考虑具有512个不同功能(F1(151) 到F512(159))的库。定序器数据可以是包含用于功能号的二进制值以及它们应当被运行的顺序的简单数值。例如，用于一个客户端的输入数据的DRM处理可以使用以下功能序列：

[0045] 处理序列：F14、F57、F396、F127、F241、F501、F8、F72

[0046] 然后，使用简单的数据结构，其中该序列被定义为用于功能号的二进制值，以下序列数据将向定序器指令执行模块50指示DRM处理过程功能处理：

[0047] 十进制的序列数据:014、057、396、127、501、008、072

[0048] 在以上示例中,序列数据60被显示为3个字符的十进制数。然而,在一个示例中,序列数据60不是包含要顺序执行的功能号的简单数据结构。在该示例中,序列数据60被加扰和置乱,使得不容易确定功能 (F1(151)到Fn(159))的调用序列,并且迫使黑客在进行系统黑客行为期间进行更复杂的软件分析。另外,可选地,序列数据60可以被加密,以及对用来识别系统内任一功能的代码进行加密。可以使用任何其他类型的数据结构、指令集定义、指令集分组或映射、数值散列或压缩来把序列数据60扩展到用于在准备DRM保护的输出数据70时实现DRM处理过程的实际指令和功能。

[0049] 在功能运行时,可以通过被调用功能有效地创建自修改序列数据 60来修改在此之前未被使用的一些序列数据60,其中自修改序列数据 60将需要调用正确的功能以便产生重新更新的正确的剩余序列数据 60。类似的处理可以应用于整个DRM处理,其中将需要调用功能(F1 151到Fn159)的正确序列来准备用来处理输入数据(11)的数据以及用来修改序列数据(60)的数据。

[0050] 有很多方法可以把功能分解成功能块,下面将示出一个示例。对于DRM系统来说,重要的是建立DRM处理的密码强度以保证系统的密码体系满足最小安全标准。在一个示例中,该系统的一个或多个功能为系统提供最小密码强度。这意味着:系统的DRM处理最低也要包括具有已知的可接受的密码强度的至少一个功能。例如,一个功能可以是在此级别建立最小加密强度的128位或192位AES算法。存在其他功能来提高最小加密强度,但是通过描述系统密码强度强于公知的可接受的标准(如128位AES加密)就容易理解和说明系统的最小密码强度。另外,在不止使用单个加密功能时,可以使用一个以上可接受的加密算法来建立密码强度。例如,系统可以使用128位AES和192位AES 以及三重DES(Triple-DES)两者。而且,还包括附加的安全性更低的功能来帮助对DRM处理置乱并且增加对DRM处理过程的逆向工程和黑客行为的复杂度。定序器数据60指示待运行的功能,并且在有些实施例中,具有各种密码强度的功能将被使用,被从每个都具有不同密码强度的一种或多种功能类别中选择。例如,假定通常存在四种功能类别,其中每个类别都具有相似的加密强度,这些类别可以布置如下:

[0051] 类别1:全128位AES、全192位AES、三重DES、BlowFish等。

[0052] 类别2:11轮的AES、5轮的三重DES、17轮的AES、6轮的BlowFish 等。当只使用特定多轮算法的多个轮回时,序列数据60可以确定被执行的实际轮回数量。

[0053] 类别3:XOR、位移、转换、映射、数学函数、散列、鉴权、块处理等。

[0054] 类别4:数据转置、数据混洗、数据存储器混洗、数据地址混洗、数据加扰。

[0055] 如上所示的每个类别中的每个功能实际上都被分解成数十个或数百个小功能块,从而需要运行大量的小功能块以实现在上面的类别中示出的功能。例如,全128位AES算法将被分解成许多实现AES算法的一个或多个步骤的较小功能,其中这些功能小到工作于算法的一行代码、或者1个轮回的一部分、或者1个轮回、或任何其他算法或代码切割(dissection)大小。DRM处理过程和DRM序列密钥或DRM序列数据60将包含将使得从一个或多个类别执行一个或多个功能的数据。当功能被分解成一组较小功能时,DRM序列密钥或DRM序列数据60 将包含执行功能的所有功能块的数据,或者定序器指令执行(50)的设计将执行功能的被创建作为将功能切割成较小块的部分的所有功能块。

[0056] 图3是诸如图1的客户端设备12这样的客户端设备上客户端库结构的一个实施例的功能框图。待处理或待解密的输入数据110经由输入数据接口模块121被输入到客户端代码库或模块100。定序器数据160 指示定序器指令执行模块150如何利用功能F1(151)到Fn(159)的序列来处理输入数据110，或者定序器数据160是用于定序器指令执行模块(50)的关于如何通过使用功能F1(151)到Fn(159)的序列来处理输入数据110的指令。定序器数据160被用于选择将对图2中所述的服务器侧DRM处理所添加的保护进行解密或去除的功能。

[0057] 可以对定序器代码加密，对定序器数据加密。序列数据对于每个客户端可以是独特的，可以具有使用不同加密算法和密钥的多个定序器。可以以客户端为基础对每个F的次序进行加扰，每个F的代码开始和停止点可以被填充以伪指令和数据。可以使用一个或多个不同加密算法来对每个F加密。每个功能可以是基于加密后的压缩后的二进制代码的。

[0058] 在完成如定序器数据160所指示的大部分或全部DRM处理后，处理后的输入数据110使大部分或全部DRM处理去除，并且输出是未受保护的输出数据170。在有些实施例中，以上处理执行大部分DRM处理，因为在有些系统中，优选要输出执行了大部分而非全部DRM处理的数据，从而数据不以明文被输出，并且输出数据170将仍需要由使用输出数据170的处理过程或功能进行的附加处理。这使得输出数据在被输出到另一功能或接口时也被保持为部分受到保护。接收输出数据170 的功能或接口将应用剩余的必要步骤来去除在可以使用数据之前需要执行的任何部分处理。可替代地，也可以不从客户端库100输出未受保护的输出数据170，而是输出由接收输出数据170的功能或处理过程已知的加密处理加密的数据。

[0059] 尽管在图2的服务器侧框图和图3中的客户端代码库100中都显示了相同的功能F1(151)到Fn(159)，但是它们不必是相同的。当它们不同时，将需要提供映射信息到客户端(明确或暗含地)，使得DRM 处理可以利用重映射功能并调整序列数据(60和160)来对到客户端的输入数据(110)进行去保护。

[0060] 在一个实施例中，每个客户端获得运行时客户端特定信息，该信息以被称为序列数据160的DRM定序密钥的形式指示定序器150应该如何对功能进行定序。序列数据160可以由服务器与特定客户端之间的单播消息序列提供，或者序列数据160可以由客户端从服务器下载，或者其可以被提供作为客户端软件安装的一部分，或者作为多部分软件下载处理的一部分，其中DRM加载器处理的两个阶段中的第一阶段将单个公共代码图像通过多播或轮播(carousel)型下载分发到所有客户端，而2阶段DRM加载器的第二阶段用独特代码或状态序列指令流数据以单播方式进行，其中独特代码或状态序列指令流数据对于在第二阶段中被分发的每个客户端是独特的。对说明处理和实际步骤的两阶段加载器的描述可以被合并到单软件下载中，其中下载包括客户端公共代码基以及特定于客户端的密钥(或多个密钥或DRM密钥或序列数据60和160)，其定义每个客户端如何执行客户端库处理流程。当然，如果系统包含足够的带宽来下载网络上每个客户端的对于客户端独特的库，则客户端库代码图像也可以对于每个客户端是独特的。何时可以传送特定于客户端的代码库的一个示例是当个人计算机下载应用代码图像时，其中代码图像的下载是安装过程的一部分。可选地，序列数据160可以以被置乱的形式被包含作为客户端库的一部分。

[0061] 在本申请中，术语“智能卡”是指一种智能卡型装置，其包含安全微处理器、或具有安全存储器的芯片、或安全身份、或具有独特标识符的加密处理机构、或包含安全处理支持

或安全存储器的个人计算机中央处理单元。用于安全性的这些硬件和软件处理技术中的任何技术在本专利申请中被通称为智能卡。

[0062] 当客户端设备(诸如机顶盒)包含智能卡时,用于客户端库的代码执行的一部分将需要从智能卡获得的或者用包含在客户端智能卡中的数据处理后的特定于客户端的密钥控制数据。这意味着:公用客户端软件图像可以被分发到在网络上运行的所有客户端设备,并且基于存储在芯片的安全部分中或存储在安全处理器中或智能卡中或安全软件狗中或其他安全设备中的DRM定序密钥数据,客户端库的软件运行对于每个客户端将是独特的。

[0063] 当服务器10(参见图1)与某客户端12通信时,通过执行从可能对于每个客户端来说是独特的DRM定序密钥所导出的特定于客户端的处理步骤,服务器将知道具体如何为客户端加密数据或准备密钥。通过应用特定于客户端的DRM定序密钥,客户端将会知道如何处理从服务器接收的数据。这意味着,在对用于某客户端的数据进行加密时,服务器产生和使用DRM定序密钥,或者在对某客户端的数据进行加密时,服务器使用已经定义的DRM定序密钥。当然,在加密数据时可以使用其他非特定于客户端的密钥,并且密钥可以作为在加密算法(诸如高级加密系统(AES))中使用的普通密钥(标准算法中使用的密钥),并且也可以定义除了算法密钥数据之外应用的功能的客户端库特定定序。

[0064] 使用DRM序列密钥来从多个软件功能中进行选择,这些软件功能也被称为处理原语、代码块、或处理功能或转换或者可以是非可逆或可逆功能或转换的处理元素。图4中示出了处理原语的示例。可以有无限多个处理原语,例如在DRM序列密钥的长度为3位时,可以有8 个处理原语,或者可以有数百个处理原语在库中被使用。处理原语可以是独特的(诸如异或和移位),或者它们可以包括多组相似的运算 (诸如移位功能的16个不同变体,或者异或功能的32个不同变体)。如在本专利申请使用的那样,处理原语也可以包括算法的完整实现(诸如AES算法的整个实现)或者算法的一定数量的轮回(诸如AES算法的“n”轮(AES.n314))。通过将整个算法(诸如完整的AES实现) 并入作为许多处理序列步骤之一,DRM处理的安全审查将展示出 DRM至少与结合有附加的对于客户端独特的置乱技术和处理步骤的 AES加密算法一样强大。当然,可以将多个可信的安全加密算法结合以增强DRM的基本安全强度。对于此的一个例子使用AES和三重DES 作为DRM处理序列中许多处理步骤中的两个。

[0065] 在图2和图3中,通过为处理块添加具有一个或多个密钥的加密,可以使上述的核心处理块更安全。例如,如图3所示,序列数据160可以用序列数据解密密钥162来加密,并且为了执行正确的处理,将会需要知道如何对加密后的序列数据(160)进行解密。如图2中所示的服务器在把序列数据60发送到客户端和客户端代码库100时将利用序列数据加密密钥62和加密算法(未示出)来对序列数据60进行加密。尽管序列数据160可能通过安全的被加密的通信链路来发送,但是黑客将需要跟踪序列数据160的解密来了解客户端的操作。当加密后的序列数据60被图3中的客户端代码库100接收时,将使用适当的序列数据解密密钥162来对数据进行解密。序列数据解密密钥162可以是以置乱方式被存储的客户端库代码的一部分,或者其可以被包含在智能卡(安全处理器、安全存储器等)中,或者其可以利用来自服务器的单播消息或任何其他方式来被传送。可以存在一个以上的序列数据解密密钥162,序列数据(60和160)加密和解密可以使用一个或多个加密算法,其中所有序列数据(60和160)或者仅仅是序列数据(60和160)的一部分被加密算法中的任意一个利用一个或多个

解密密钥来处理。

[0066] 在图2和图3中，可以利用一个或多个加密/解密算法并使用一个或多个加密/解密密钥来对这两个图中所示的功能(F1151到Fn159) 进行加密。在图2中，在组件64和66中示出了两个功能加密密钥，这些密钥中的每一个都可以在对功能加密时使用。可以使用方法和密钥的任何置换或组合来对实际功能进行加密和解密。例如，可以存在以两个不同密钥使用的两个不同的加密算法，或者以一个密钥使用的一个算法，或者以一个密钥使用的两个算法，或者以三个密钥使用的三个算法。图2中的服务器侧处理和图3中的客户端代码库将知道如何应用不同的功能加密方法和密钥。每一侧知道如何对功能进行加密/解密的方式可以以由对这些组件的代码进行编程的编程人员所确定的任何方式来定义。存在无限数量的可以使得可以执行这种编程，几个例子包括以下：

[0067] 1.在使用两个算法来对功能进行加密/解密时，使用算法1 和算法2来交替地对功能进行加密。

[0068] 2.将算法1和密钥1用于前10个功能，将算法2和密钥2用于接下来的10个功能，将算法1和密钥3用于对接下来的10 个功能进行加密，以此类推。

[0069] 3.在使用单个密钥时，使用密钥1对所有奇数功能进行加密 /解密，使用密钥2对所有偶数功能进行加密/解密。

[0070] 4.使用密钥1和算法1对功能1、4、8、12、…进行加密/解密，使用密钥2和算法2对功能2、5、9、13、…进行加密 /解密，使用密钥3和算法3对功能3、6、10、14、…进行加密/解密，等等。

[0071] 还可以使用公钥/私钥加密(公钥基础设施PKI)来对功能进行加密。这会需要硬件支持或者可以快速执行PKI解密的快速CPU，或者库可以在被初始化时或在运行时被解密。

[0072] 当使用上述方法中的一种方法来对功能进行加密时，将会存在由客户端库添加的功能解密步骤，其可以发生在运行时执行时或在库初始化时，或者在库执行期间的某些时刻时(诸如就在功能执行之前)。添加功能加密/解密的一个方法是使用定序器指令加密块或模块40来使服务器对功能进行加密，以使用适当的密钥和算法来对每个功能进行加密。加密后的功能将需要被解密，并且客户端代码库100(图3) 将使定序器指令解密模块140来利用适当的定序器数据解密密钥和算法对功能进行解密。

[0073] 下面，图4示出了一些功能的示例以及它们如何被存储在服务器 10和客户端12的存储器中。例如，在用于功能1的存储地址中，可能存在用于执行某些类型的位移位的移位指令序列(310)。功能2可以是执行异或功能的异或功能(312)。功能3是显示为AES.n(314)的AES 算法的部分实现。功能4是包括所有轮回的完整AES(316)算法。功能5是把输入数据映射到输出数据的映射算法318。功能6是执行任何类型的转换、移位、混洗等的转换算法320。图7是执行密码或普通散列计算的散列函数322。为了描述简洁，图4仅示出了9个功能。然而，在库中可以包括数百个或者甚至数千个功能。

[0074] 图4中所示以及在本专利申请其他地方描述的功能可以被进一步分解成较小单元，所执行的处理将执行对于正确地处理数据所必需的适当的较小功能块。例如，AES算法316可以被分解成30、40、50或更多个较小功能。因此，每个功能块(诸如F1到Fn)可以是包含完整算法或者算法的一块或一部分(诸如多源码行算法中源码的单个行或部分)的功能处理块。

[0075] 可选地，本发明的另一方面包括“伪(Bogus)”功能(324)，其不被使用或者可能被用来标记黑客行为尝试，以为包括客户端库的功能增加附加的复杂度。当用来标记或指示黑客行为尝试时，伪功能将设置指示符，表明由于黑客行为尝试很可能执行不应被执行的功能。

[0076] 图4还示出了被添加以使代码更难理解的伪装(Camouflage)功能326。实际上，伪装功能可以是比实际上调用的功能更可识别的功能。许多伪装功能之一可以是在不属于DRM处理一部分的数据上运行的整个AES算法的文本簿(text book)软件示例，而在DRM处理中使用的实际AES看上去不像文本簿算法并且可以被分解成许多子功能块。除了众多伪功能和伪装功能之外，还可以存在许多看起来类似的或相同的功能(在不同数据上运行)被包含在客户端库中。这意味着，可以存在许多(例如12个)非常类似的代码块，其中每个代码块执行功能的一部分或全部，并且在每个不同版本如何在数据上运行或者如何对数据进行访问方面存在微小差别，并且这些不同(例如12个)不同版本被分布到客户端库中。因此在客户端库中可以存在许多看起来类似的功能，所以不会存在对于某个功能的代码进行定位的一个位置点。如果黑客寻找代码来定位AES解密，则他可能找到看上去可能是AES 解密代码的12个或24个或36个不同的代码版本。可以存在许多不同的散列函数，不同的伪功能，不同的伪装功能，以及许多用于所使用的每个功能的看起来类似的功能。

[0077] 图5示出了可由图1的服务器10执行的DRM处理在保护数据时所执行的处理步骤。在该示例中，输入数据是用于一条内容的主内容加密密钥405。主内容加密密钥405随后由一系列功能F1411到Fn419处理。这些功能执行密码功能、转换、映射等来保护数据，如在本专利申请的其他部分所述的那样。可以执行任意数量的功能，并且这些功能选自上面结合图2、图3和图4所述的功能以及其他功能。在图5中，使用命名F1411，F2412，F3413，F4414。然而，在实际实现中，号码将不是顺序的，而是更随机的。

[0078] 对用于一条内容的主内容加密密钥405进行加密的DRM处理可被用来保护内容密钥免受黑客窃取内容的尝试。主内容加密密钥405 具有在其上执行的一系列功能，以产生主内容加密密钥405的DRM保护版本，并且被称为DRM保护的主内容加密密钥495。尽管图5示出了 DRM处理被应用于主机内容加密密钥405，但是图4中所示的步骤可以处理任何数据，并且处理并不仅限于主内容加密密钥。例如，系统中所使用的任何类型的数据可以在步骤405被输入、被处理并且在步骤 495被输出。

[0079] 在图5中，步骤被标注为功能步骤F1、F2、F3…Fn。功能指示符 F1、F2、F3等被用来指示被应用于数据的一系列功能步骤。实际上，功能F1、F2、F3到Fn将是由DRM序列密钥确定的功能，并且因为功能的实际序列可以由DRM序列密钥确定，所以处理步骤的序列是可编程的。

[0080] 每个功能模块可以被进一步分解成较小功能块，并且可以在没有附加数据(密钥，变量，转换数据，常量等)被应用于功能处理块的的情况下或者在有附加数据(密钥，变量，转换数据，常量等)被应用于功能处理块的的情况下执行功能块处理。

[0081] 这些实施例允许图2的服务器把独特的DRM处理应用到正在服务器与客户端之间发送的每条DRM保护的数据。通过改变序列数据密钥(图2中的60)，应用于正被保护的数据的DRM处理将发生改变。这意味着，通过改变序列数据60，实际的DRM处理功能以及可选地还有所使用的密钥也将发生改变。执行如图5所示的处理步骤的各个客户端可以使用各自的

或公共的密钥，以及甚至各自的虚拟机或加密后的状态机处理方法或指令集。每个单独的虚拟机或加密后的状态机可以利用对于客户端独特的密钥或对于所有客户端公共的密钥来分别加密。

[0082] 下表展示了映射到DRM处理功能代码(与地址或状态或虚拟机指令类似)的DRM序列密钥的逻辑分解，其中使用DRM序列密钥数据来指示被应用作为DRM处理一部分的功能序列。

[0083] 处理所需要的功能块分解

[0084] F1＝具有密钥位3、4的XOR数据字节

[0085] F2＝向右移位3位

[0086] F3＝使用映射功能8的映射

[0087] F4＝使用转换功能17的转换

[0088] F5＝完整的128位AES加密算法

[0089] F1＝代码块33

[0090] F2＝代码块137

[0091] F3＝代码块8

[0092] F4＝代码块246

[0093] F5＝代码块227

[0094] 如图6所示，DRM序列密钥(图2和图3中的序列数据60和160) 可以例如由服务器加密并发送到客户端。加密可以是对于客户端独特的或者对于一组客户端或全部客户端公共的。可替代地，可以使用SSL 解码或加密地将DRM序列密钥发送到客户端。对DRM序列密钥的保护也可以使用在此描述的处理。可以使用例如用来保护用于主内容加密密钥的DRM序列密钥的另一DRM序列密钥来对DRM序列密钥进行加密。这意味着，系统中每个密钥或每条数据可以由DRM序列密钥数据所定义的对于客户端独特的DRM处理来保护。

[0095] 发送被用来指示应当用来处理加密数据的步骤序列的DRM序列密钥的方法可以本身由附加的DRM序列密钥数据来保护。客户端与服务器之间的协议可以使用协议层DRM序列密钥来产生对于客户端独特的加密协议层。当然，可以使用多种实施例用于多种类型的加密/ 解码应用，并且可以在相同的库中重新使用这些技术来保护库的不同部分。

[0096] 图7A、图7B和图7C通过示出由每个客户端应用于输入数据的功能序列来表示三个不同客户端应用DRM处理的方式。在图7A中，客户端1应用功能33、137、8和246来保护数据，序列流部分或完全由序列数据(图3中的序列数据162)确定。在图7B中，客户端2应用功能137、8、8、104，而客户端3通过执行功能61、4、307和175来执行DRM 处理。图7A、图7B和图7C中所标识的功能中的每一个都可以具有与公知的安全加密算法相同的最小加密强度。对数据进行解密(和进行黑客行为)所需要的实际处理对于每个客户端来说将是独特的。在图7A、图7B和图7C中，独特地执行特定于客户端的DRM处理，因为序列数据是独特的，或者客户端库的存储器布局(包括功能的存储器布局)是独特的，或者这两者都是独特的。

[0097] 在使用智能卡的系统中，可以将客户端使用的定序数据、密钥数据或状态机数据中的一些或全部添加到智能卡。可替代地，基于智能卡的系统设计可以使得需要智能卡数据以及智能卡外部的其他数据。

[0098] 图8示出了可以由服务器执行的用于对来自图7的处理流程进行加密的处理的流

程图。在图8中，标记为“ENC”(625、635、645) 的组件可以是状态机、状态表或者用于导出处理流程的数据、虚拟指令、或者可以在设备CPU上或设备智能卡(如何有的话)中执行并且可以对于设备固件而言是本地的或者被下载到设备中的代码。“ENC” (625、635、645)在被存储到存储器中时可以被加密，并且只有在处理器执行或处理例如625所示的步骤时被解密。图8所示的处理(组件 645、635和625)通过图9的流程图中组件845、835、825所表示的客户端处理步骤而被进行逆向处理。

[0099] 现在参考图10，图10是用于对可以由服务器实现的主内容加密密钥1310(或其他数据)进行保护的处理的表示。DRM处理密钥控制块 1320包括代码、数据(状态机或序列或密钥控制或其他)或者用来定义密钥序列的虚拟指令以及被应用以保护主内容加密密钥1310或任何数据的数据处理(1330)。DRM处理密钥控制块1320和/或DRM处理指令状态机、虚拟机指令处理器或统称或分别称为DRM处理1330的定序器也可以任何方式被加密以进一步置乱客户端库处理。用来定义 DRM处理的DRM处理密钥控制数据1320也可以利用在此定义的方法来被加密。通过修改DRM处理密钥控制数据1320，数据处理步骤将发生改变，因为DRM处理密钥控制数据1320定义由DRM处理1330所使用的处理步骤序列。

[0100] 在一个实施例中，每个客户端库在其执行被应用来保护输入数据的处理(DRM处理(图10中的组件1330))的方式方面是独特的。DRM 处理密钥控制(图10中的组件1320)对于将在此所述的任何技术应用到DRM处理密钥控制(1320)和DRM处理(1330)两者的每个客户端来说也可以是独特的。与在只有DRM处理(1330)使用这些技术时相比，这么做就提供了更置乱的执行环境。因此，DRM处理密钥控制 (1320)可以具有其自己的密钥和处理流程，并且除了用来保护数据 1310的DRM处理是独特的之外，这还使密钥处理和客户端处理是独特的。黑客将被迫对DRM处理密钥控制数据1320和DRM处理1330块两者都进行黑客行为以便对DRM进行黑客行为。每个块(1320、1330) 对于客户端来说可以是独特的，其中独特的加密被应用于用于每个客户端或每个内容下载或任何粒度的运行时代码。这意味着，DRM处理密钥控制1320还可以使用附加密钥控制数据(未示出)而被密钥控制，其中该附加密钥控制数据在对需要被执行以应用或去除DRM保护的步骤进行定序时使用。类似地，DRM处理1330执行其DRM处理的方式是以DRM序列数据(或处理密钥控制1320数据)的形式使用序列数据，并且DRM处理1330内部地实际处理DRM处理密钥控制1320的方式也可以被密钥控制(未示出)。在此描述的技术可以被递归地应用，以保护密钥数据和序列数据以及其他任何数据，并且对使用在此所述的处理进行DRM保护的每个数据项的内部处理进行置乱。

[0101] 在一个实施例中，VM指令集对于每个客户端来说是可密钥控制的，就像用来定义应用于密钥或客户端的DRM的VM指令序列那样。可密钥控制的VM指令集是这样一种指令集，即其中使用密钥来加密指令集或者VM执行的运行序列由密钥控制，或者这二者，或者加密密钥用于加密指令集而另一密钥用于控制指令定序执行流。DRM处理序列与VM指令集一起是可密钥控制的，其中这二者对于每个具体客户端设备或者一条内容或者对这两者来说是可独特密钥控制的。

[0102] DRM序列密钥数据可以由服务器加密，并且在运行时在客户端中被解密。用于DRM序列密钥数据解密的加密和解密方法可以随时间改变。这迫使黑客重新开始对客户端库的黑客攻击。DRM序列密钥处理、DRM处理和客户端库都可以是对于每个库是独特的，或者项中的任何一个可以是对于每个客户端来说是独特的。

[0103] 被加密的DRM序列密钥数据可以使用PKI密钥或Diffie Hellmen 序列产生的密钥、或者PKI和/或Diffie Hellmen和/或特定于客户端的密钥产生代码的组合来被加密。

[0104] 通过将DRM密钥控制处理应用于多阶段设计中DRM处理的每个阶段，DRM设计者可以将不同的虚拟机与相同或不同的密钥控制方法一起使用于处理的每个阶段，从而迫使黑客不得不对多个加密的VM 进行逆向工程。每条DRM处理可以使用单独的加密以增强安全性。

[0105] 在此描述的系统和方法也可以被用来以特定于客户端的方式保护存储在客户端上的数据。例如，用来对可以在没有网络连接的情况下观看的电影文件进行保护的主内容密钥(离线DRM)可以具有为了对使用对于客户端独特的处理步骤存储的电影进行解密所需要的特定于内容的密钥数据。这通过使用DRM定序并且使序列数据对于由客户端执行的每次电影下载是独特的来实现。同样地，用于任一客户端上每条内容的解密密钥对于每条内容来说将是独特的，另外，用于每个客户端设备的客户端库可以是独特的。在这样的配置中，黑客攻击的结果可以被隔离到一个客户端的一条内容，从而使得不可能出现全面的DRM黑客攻击。

[0106] 这些系统和方法也可以应用于智能卡，其中在智能卡中具有VM 指令集密钥控制算法。使用智能卡(用于任何类型的数据，包括 ECM/EMM数据)的客户端将使用智能卡内部的序列数据来防止对智能卡的静态分析，以及防止对接口到智能卡的客户端代码的静态分析，因为解密密钥将是必须的，并且解密密钥将不会是本地库的一部分，而是DRM序列数据或从服务器发送的其他数据的一部分。

[0107] 在此描述的系统和方法的设计中，还可以包括代码侦探(code spy)。代码侦探是多条代码或功能，其或者收集用于随后报告给服务器的数据，或者它们收集在客户端中运行时处理期间所使用的数据以建立将被用来检测黑客攻击试图的情形。这些情形将在怀疑受到黑客攻击时被报告回内容分发系统(或其他适合的接收方)的前端，或者将创建对于客户端软件的正确运行所必需的数据。例如，代码侦探可以获取例如包括序列号的硬件寄存器值，随后序列号被散列处理从而其是不明显的，随后，散列处理后的序列号在某一时刻或者在某个事件之后被存储在存储器中。随后，或者与侦探数据被收集(读取和经过散列处理的序列号)同时，或者优选在稍晚时候，另一代码功能将使用侦探数据，或者将侦探数据报告回服务器，从而允许服务器进行服务器侧克隆检测。代码条也可以具有对于对代码条进行个性化处理所必需的用于数据的占位符，以进行一些新的并且黑客不能期望的事情，并且代码条的激活在随机的时刻发生或者在每次客户端运行时发生。

[0108] 代码块占位符的一个示例工作如下。对于客户端库操作的前十天，黑客看到读取存储地址的一条引诱代码(decoy code)，并且该存储地址处的数据在随后的客户端库处理中以不重要的(引诱)方式被使用。引诱代码是由DRM序列密钥数据确定的默认处理路径。以后， DRM序列密钥或数据将发生变化，并且该变化调用探测客户端的新代码(侦探代码)以检测篡改。例如，在DRM处理的前十天期间，引诱例程收集看起来无害的数据(不读取特定于芯片的或特定于客户端的数据)，并且看起来似乎执行静态数据读取和写入，但是，在操作的第十一天或者在DRM序列数据发生变化时，不执行引诱存储器存取代码，并且被称为侦探代码的新的一条代码被运行。侦探代码将读取与客户端硬件相关的标识数据或序列号或MAR地址或量子数据，其随后被报告回服务器或前端，或者被用来检测克隆(诸如与在系统

处理的其他部分中所使用的序列号不匹配的客户端设备序列号)，这是因为当黑客对库进行黑客攻击时，在观测前十天期间库观测(library observation)时，其被黑客欺骗。将使用服务器侧或客户端侧上的数据来检查侦探数据以查明其是否正确以及是否还没有被篡改。存在不受限制的方式来将侦探代码添加到客户端库，并且侦探代码处理可以保持休眠直到发生触发事件(诸如DRM处理数据的变化)或客户端库内部事件。引诱代码功能和块是预先计划的客户端库或DRM处理步骤或可以被用在处理DRM步骤中或者被用来收集客户端或服务器使用的数据(侦探数据)以检测客户端库黑客攻击和克隆的功能。

[0109] 图11是对于特定客户端独特地进行加密的特定于客户端的虚拟机或序列数据的简化框图。客户端将使用VM数据密钥1240来对虚拟机或序列数据流或者这两者进行解密。可以采用到每个客户端的单播传递的方式，或者使用到一个或多个客户端的多播传递广播，或者使用从客户端中的智能卡所获得、并且在智能卡初始化期间被预先传递的数据，或者使用任何其他数据交换机制，来把该VM数据密钥1240 传送到客户端。VM密钥数据可以是一个或多个密钥。

[0110] 在另一实施例中，可以存在多个密钥用于代码块、虚拟机、运行时完整性检查代码和运行时安全性扩展。可以对于系统中所使用的每个密钥或数据项使用不同加密算法来对密钥或数据进行保护。例如，用于一个组件的密钥块可以使用多个加密后和置乱后的密钥来对客户端库的代码块进行加密。因此，该组件的加密将由一个或多个密钥组成，并且例如，由这些密钥保护的代码块每一个都可以使用不同的加密/解密算法。作为示例，使用多密钥序列的密钥1的处理可以使用 AES-ECB模式加密，而使用密钥2的处理的第二部分可以使用不同的算法，如三重DES。密钥3将使用Blowfish，密钥4使用AES-CTR模式等。如上所述为任何一个处理块或任何数据使用多个密钥就提供了更多的安全层，其中为了成功地黑客攻击(或理解)在这些块中所执行的或为了数据保护而执行的处理，这些安全层必须被黑客攻击。

[0111] 为了说明的目的，假定序列流数据的长度是88字节。DRM定序器流数据的加密不必仅使用一个密钥或甚至一种加密算法，相反，定序器流数据可以使用加密算法的组合，诸如对于DRM定序器流数据的前256位(32字节)使用AES，对于DRM定序器数据接下来的192位(24字节)使用三重DES，并且对于接下来的256位(32字节)使用Blowfish 等。对于用于保护该示例中88字节DRM定序器流数据的三个加密算法中的每一个，可以使用单个密钥，或者，可以使用多个密钥(2或3个或更多个)来保护88字节的序列，其中在DRM定序器流数据的一部分上操作的每个算法都具有其自己的密钥。

[0112] 在有些设计中，图11中的客户端密钥不必随着客户端的不同而变化，并且可以是对于客户端组或对于全部客户端公共的密钥。然而，该系统和方法也支持以下设计，即图11中的客户端密钥1242可以随着客户端的不同而变化、或者根据区域而变化、或者根据前端(或服务器)而变化、或者根据国家而变化、或根据DRM软件设计者所期望的任何其他粒度而变化。实施例也允许不同的客户端密钥1242和DRM处理方法对于每个客户端或区域或所有不必改变客户端库软件设计的任何情况来说是独特的。同样地，利用在此所述的技术，从一个软件代码基得出多个不同的DRM。

[0113] 当期望改变DRM处理方法时，VM密钥控制数据可以不但被用来对VM进行密钥控制，而且可以在多个VM和解密算法被内置到单个客户端库代码图像中时选择N个VM或定序器中

的一个。这会允许单个客户端被分发到客户端设备，其中该单个客户端包含可以使用不同密钥控制和解密算法的多个DRM处理方法。在客户端库发布之后，使用第一一DRM处理方法和相关的密钥控制数据和解密密钥，而客户端库中的其他DRM处理方法和VM保持休眠直到注意到正被使用的第一DRM 处理方法已经被危及安全。同样地，单个客户端库也可以包含也处于休眠的不同虚拟机或定序器，并且在DRM密钥控制变化时，当与在使用第一DRM处理和密钥控制方法时的客户端库的操作相比时，激活非常不同的运行时执行环境。通过改变以下一项或多项得到该非常不同的运行时执行环境：DRM密钥控制；或所选择的VM或序列；或DRM 处理密钥控制。这允许构建可以在第一次所使用的DRM处理方法中发生黑客攻击或安全破坏(security breech)事件时发生变化，而不必下载新的客户端库DRM的客户端库。因为客户端库将使用客户端库外部的密钥控制数据，所以黑客在休眠的DRM处理方法被服务器激活之前不能对休眠的DRM处理方法进行黑客攻击。

[0114] 现在将简要描述开发环境的一个实施例。

[0115] -创建50到100个可被用来实现DRM处理的软件功能(图2中的 F1-151到Fn-159)或代码块。

[0116] -在预处理器中，把50到100个代码块改变(例如，变换或分解) 为数量更大的(比方说1000或更多个)代码块或功能或功能块。可替代地，简单地写用于几十个到几百个代码块的代码，其中只需要对将被DRM处理需要的代码块进行测试。不属于DRM处理一部分而仅作为伪装的代码块不必作为DRM处理的一部分而被测试，并且可以被测试以仅仅为了确保它们不将系统挂起。可以使伪装功能看起来比DRM 处理中使用的实际功能更真实。例如，伪装AES算法可以使用经过授权的开发源代码的“文本簿”版本的AES算法，在没有容易被识别的“文本簿”算法编码技术的情况下，DRM处理所使用的实际AES算法可能看起来完全不同于文本簿版本。另外，在主处理代码检测到黑客正在篡改代码时，可以激活伪装功能。客户端库将运行伪装代码，提供不属于真实DRM处理代码的代码执行路径并且简单地将错误的处理路径发送给黑客。

[0117] -建立可以通过代码块定序的一个或多个定序器或虚拟机，以利用DRM序列数据所定义的处理序列来执行DRM处理序列。

[0118] -测试(多个)定序器和DRM处理。

[0119] -建立对上述定序器、或序列控制信息或定序器数据或代码块进行加密的一个或多个定序器加密器，其中解密密钥和/或执行是“能够密钥控制的”并且可以在运行时被下载到设备中并且不被静态地嵌入到代码固件图像中。

[0120] -可选地，包括可以由来自前端的序列密钥控制数据激活的一个或多个代码块侦探。可以使用代码块侦探来检测受黑客攻击的软件，并且可选地报告客户端黑客攻击和可用来检测篡改的相关数据，并且向服务器报告克隆或在客户端使用这样的数据或这二者。侦探代码还可以去激活正确的DRM序列密钥处理路径，并且执行包括在客户端库中的伪装代码。

[0121] -1000个代码块中的很多都可以处于休眠，直到DRM序列密钥控制数据指示它应当运行。另外，用于DRM处理的执行路径1可以使用一组代码块，随后在一段时间之后，DRM处理将使用库中10个代码块或100个代码块或1000个代码块中的不同代码块。这实际使得一个库代码图像可以包含不同版本的DRM处理并且允许DRM处理在没有代码库更新的情况下发

生改变。如果进一步对代码库进行加密，则包含在代码库内的各种不同的DRM处理序列可以使用不同的加密密钥和处理。当DRM处理改变时，它改变成在发生改变之前黑客不能得到(或看到)的一组库功能和解密密钥。

[0122] -系统的增强版本基于或者嵌入到代码库中的DRM密钥控制(或定序)数据，或者在运行时或当发送触发以激活或改变VM指令定序或定序保护/置乱时从服务器动态发送的DRM密钥控制(或定序)数据，来改变VM指令执行保护加密/解密。

[0123] -库设计可以包括激活VM中的新解密路径或新VM指令或新VM 寻址或数据存取或上述任何一个的激活触发(或控制位变化)。这表明，库可以基于输入数据或者客户端或服务器或这两者所检测到的事件改变执行代码图像解密的方式。例如，对于操作的前60天来说，代码图像解密路径使用包含用于解密的128位AES的路径，并且在第61 天，代码库将执行使用库中不同代码的三重DES解密的不同代码。因为对用于三重DES加密代码的代码库进行解密所必需的密钥不被包含在客户端库中，所以黑客不可能已经检测到存在三重DES加密代码和在某些时刻激活的三重DES代码解密库。黑客不能暴力攻击三重代码图像，因为其首先被压缩并且随后被加密，并且对代码图像的压缩处理极大地增加了暴力攻击的难度。黑客不知道代码何时将被启用以及代码将进行什么操作以及库中包含什么侦探数据(如果有的话)。其实，用两个不同密钥加密或者以两种不同方式加密的两个或多个代码图像的结合在客户端库中布置“定时炸弹(ticking time-bomb)”，客户端库将包含黑客还不能攻击的代码。

[0124] -在一个实施例中，上述情况被扩展为包括在对库加密之前执行的二进制代码压缩处理或其他置乱处理，使得分布式客户端库在加密之前包含压缩的二进制代码。这使得基于指令集知识按查字典攻击和暴力(brute-force)攻击变得更难，因为加密后的数据被压缩或被置乱。

[0125] -另外，可以添加可如上面在侦探代码部分中所述的那样被使用的侦探功能。侦探数据可以采用许多不同的形式，并且一些例子包括观看了多少电影、某些频道变化之间的时间间隔、客户端首次开机的时间等，并且可使用其他类型的量子数据来执行克隆检测。可以使用具有使每个代码图像不同的独特密码的代码图像来产生量子数据。对于某个客户，可以不同于其他客户，在于：如上所述，因为客户端库没有被嵌入在其中的密钥来对客户端库的其他部分进行解密，所以黑客只能对库的一部分进行反向工程。传送客户端库的一种方法是在客户端库中甚至不包括用于代码的第一操作部分的客户端代码解密密钥，并且需要在客户端运行时所获得的客户端库或功能加密密钥(图2 中的64、66和其他未示出的)，从而迫使黑客不得不在客户端运行期间从客户端提取代码图像解密密钥。而且，因为可能存在用来建立库的两个或更多个不同功能加密密钥，所以将迫使黑客在所使用的功能解密密钥发生改变时重新黑客攻击客户端库(在非静态环境中重复运行时库黑客攻击)。

[0126] 因为定序器或VM是黑客的主攻击点，所以在该区域中的软件设计可以使用软件保护的所谓“最佳实践”方法，包括：

[0127] -对每个代码图像进行数字签名；

[0128] -运行时代码完整性检查，其验证代码还没有被修改，调试器还没有被附加，代码断点还没有被插入；

[0129] -优选地使用多个加密密钥和多个定序密钥控制数据并且使用多个加密/解密算

法来对客户端库的不同部分进行保护；

[0130] -加密算法根据序列数据或事件数据而变化；

[0131] -使用特定于客户端的一次一密乱码本(one-time pads)进一步置乱发明的密钥和数据；

[0132] -密钥选择VM定序器中新的执行路径；

[0133] -休眠的序列路径根据密钥数据而变为激活，以允许未受黑客攻击的处理路径在某个时间事件之后变为有效。例如，在库被发布之后的90天。

[0134] -优选地，压缩后的软件代码图像被加密以增加暴力攻击的难度；

[0135] -使用多个加密/解密算法来用未被存储在客户端库中的解密密钥对软件代码图像进行解密；以及

[0136] -可以仅使用将在本专利申请的其他部分中描述的后编译时间处理来为客户端生成新的库。

[0137] 系统和方法的一个元素是把功能分解成代码块。这可以采用多种方式完成。图12是一种方式的示例，示出了用于被标记为F1、F2、F3 的3个功能(如来自图2的功能)的代码块序列映射。

[0138] 图12中的功能F1包括代码块(CP)80、31、608、422和96。当然，把功能分解成代码块可以包括附加的代码块。客户端操作的优选方法是不把黑客可利用的静态调用树暴露出来。因此，在客户端库软件中不存在从CP80到CP31，或者从CP31到CP608等的调用。相反，存在具有返回指令形式的代码块或者允许序列数据定义必须被执行以正确地去除DRM处理的代码块的实际序列的链接机制的集合。执行F1 的DRM序列数据将是指示/识别需要执行代码块80、31、608、422和 96的地址或索引或数据。类似地，用于功能F2的序列数据将指示代码块3033、1045、201、17和762。可选的设计元素是：可以将有些代码块配对以减小序列数据的大小，诸如对于功能F1，CP80具有对CP31 的调用，从而允许两条代码由指示CP80应当运行的序列数据执行。当完成CP31时，不存在对CP608的调用，因为这会给黑客提供将CP 80、31和608链接在一起作为黑客静态代码分析的一部分的能力。优选地，每个代码块都是其自己的岛(island)，排成一行，没有指示用于功能的代码块调用序列的链接信息。

[0139] 另外，因为功能被分解成代码块，所以可以包括代码块加扰器，其对用来执行DRM功能的代码块的次序进行加扰。服务器保持传送给客户端设备的被打乱的代码块代码图像上的数据，并且知道如何为特定客户端准备DRM序列密钥。图12所示的功能F1、F2、F3表示可以如何分解功能。例如，如果功能F1是AES加密算法的所有处理轮回，那么CP80、CP31、CP608、CP422、CP96将是实现AES算法的一个轮回或AES算法的所有轮回的代码块。当代码块仅表示算法的一部分时，比方说AES算法的一个轮回，那么DRM序列数据也可以包括指示执行代码块的次数的数据。例如，如果期望AES算法的32个轮回并且分解使得F1为一个轮回，那么序列数据可以包括指示应当处理32个轮回的F1的数据。另外，代码块可以包括处理以在功能分解针对算法的单个轮回时来对密钥和数据进行调度。例如，CP608可以包括数据检查，以检验哪个轮回正在被执行以及存取特定于轮回的数据。

[0140] 分解处理的一个感兴趣的因素在于：可以在通过对代码块进行加扰的单独的后软件开发处理进行软件编译和测试之后，对代码块或功能进行加扰。该处理类似于链接步骤，其不仅链接代码块的对象或二进制版本，而且其还通过以将允许服务器向DRM附加处理应

用适当的处理的方式布置代码块来对代码块进行“密钥控制”，DRM附加处理将由客户端中执行的DRM去除处理来正确地解密。客户端库中代码块的加扰次序将为服务器所知，并且服务器将使用能由客户端正确解密和处理的处理(密钥控制或定序)来应用加密。在一个示例中，还使用DRM序列密钥，并且DRM序列密钥将适合于代码块被加扰的方式。例如，图12中的功能F1被分解成代码块80、31、608、422和96。代码块的这种次序可以被传送到客户端1。然而，其他客户端可以在与其他库比较时接收加扰后的库。例如，通过使用后编译和软件测试，加扰处理可以对代码块进行加扰，使得每个客户端库的存储布局是独特的。例如，软件库(其中用于功能1的被编译和链接测试的版本的代码块可以被后编译加扰)如下面表所示：

[0141]

客户端F1号	用于功能F1的客户端中的代码段次序				
1	80	31	608	422	96
2	201	81	93	17	3
3	199	9	23	41	130
4	4	7	109	18	86

[0142] 在上面的表格中，针对使用相同的编译后和测试后的源代码所传送的四个不同客户端，示出了代码块存储地址寻址排序。在传送到客户端之前或者在软件安装过程期间或在其他任何时间，可以对代码块进行加扰。在此示例中，每个客户端将需要对于客户端独特的序列数据(DRM序列数据)来在每个客户端上正确地执行功能1。对于客户端来说独特的DRM序列数据将被用来指示将代码块拼起来的处理流。

[0143] 用于每个客户端的对于客户端而言独特的DRM序列数据如下所示：

[0144]

客户端号	在可选的加密或其他置乱之前的DRM序列数据
1	80,31,608,422,96
2	210,81,93,17,3
3	199,9,23,41,130
4	4,7,109,18,86

[0145] 如在上表中所反映出的那样，在为每个客户端独特地加扰客户端库代码图像时，DRM序列数据(DRM密钥控制数据)对于每个客户端来说可以独特的。另外，当应用适当的虚拟机指令集设计技术时，加扰功能可以在虚拟机指令执行环境中。可替代地，当为客户端库使用状态机或状态表格设计时，DRM序列数据将被合当地应用于该设计。值得注意的是，除了对DRM序列数据加密之外，可以利用对于客户端而言独特的转换表格以针对各个客户端对DRM序列数据处理进行置乱处理，从而服务器把对于客户端独特的转换应用于DRM序列数据。客户端将应用对于客户端独特的转换来创建如上所述的必需的序列数据。这种转换可以以任何期望的粒度被应用，诸如对每个代码块序列数据值应用，这意味着，每个客户端都可以包括由一次一密数据 (one-time pad data)组成的客户端一次一密乱码本，其中一次

一密数据被用来转换服务器和客户端侧二者上的实际DRM序列数据，其中两侧都应用一次一密。还可以应用进一步使DRM序列数据置乱并且映射到实际代码执行的其他方法。

[0146] 另一值得关注的实施例包括通过具有分别使用不同大小的密钥的多密钥处理代码块来为在系统中使用的密钥产生不同密钥长度。DRM序列数据选择不同的代码块来建立具有不同长度的DRM内容密钥。例如，在下面的表格中，存在处理表格中比特数的DRM处理功能块。DRM序列数据将指示DRM处理应当由具有128位密钥处理的功能以及具有不同位数的其他功能组成。使用这种可选步骤将提高黑客的失败，因为黑客无法知道系统的基本DRM内容密钥长度，并且将不得不分析代码来甚至确定密钥长度。每个客户端可以使用不同的密钥长度组合。

[0147]

功能号	1	2	3	4	5	6	7	8	9	10	11	12	13
密钥位长度	128	128	128	17	11	4	3	21	18	7	16	8	19

[0148] 上面的表格示出了每个功能所处理的密钥位数。当然，上面的每个功能都可以被分解成较小块，如在本发明其他地方所述。用于客户端的DRM处理将至少由128位密钥处理功能中的一个或多个以及更少位数的密钥处理功能的一个或多个组成。下面的表格示出了可以如何使用DRM序列数据来产生具有不同密钥长度的DRM处理。

[0149]

DRM客户端	DRM处理中所使用的功能	DRM密钥长度（以位为单位）
1	1,5,10	146
2	5,1	139
3	7,10,2	138
4	6,3,12	140

[0150] 对位进行处理的序列次序也可以改变。另外，可以使用多个处理链来产生更复杂的DRM处理步骤。例如，DRM客户端1可以使用功能 1、5和10以及具有相同密钥长度的其他功能，或者在第二轮(pass) 中重新使用密钥，诸如除了功能1、5、10的第一轮之外还添加使用功能5、10、1的第二轮。同样地，密钥将被重应用两次(每轮一次)。当然，可以为每轮发送附加密钥，并且在两轮示例中，在使用两轮146 位DRM处理时将发送292个位。多轮可以使用未在第一轮中使用的附加位(例如第一轮使用函1、5、10的DRM)，随后具有第二轮，第二轮使用功能1、5、10以及为第二轮的密钥长度增加8位的功能12。通过定义在DRM处理期间所应用的功能序列来创建和处理所有不同密钥长度，并且在该示例中，序列为功能1、5、和10的第一轮以及功能1、 5、10、12的第二轮。

[0151] 在图12中，右侧示出未加密的序列数据，其中每个框中的数与被运行来执行某一功能的代码块相对应。对于添加的安全性，可以在被发送到客户端之前使用利用一个或多个密钥的单个或不同的加密算法来对序列映射数据中所示的代码块进行加密。在该示例

中，可以对序列偶数值(即，元素0、2、4、6、8，…)使用AES，以及对序列的奇数元素(1、3、5、7、9…)使用DES，来加密组成功能“F1”的代码块。当然，本领域技术人员可以应用不同的算法组合和不同的模式来增加安全性。用来对代码块序列映射进行加密的密钥数据可以从特定于客户端的数据中得到，或者独特地被传送到每个客户端或者被嵌入在库内或者从特定于客户端硬件的数据或量子数据或其他一些方式得到。

[0152] 同样在图12中，每个功能F1、F2、F3例如可以具有它们自己的用于加密的密钥和它们自己的对每个功能独特的加密算法序列，如针对功能F1在之前段落所述。

[0153] 另一应用是为每个客户端设备产生独特的软件客户端库代码图像，其中独特的软件客户端在将客户端软件或固件图像传送到客户端时从共同的处理代码集和服务器得到，或者在离线处理中服务器使客户端软件图像是独特的。使客户端软件库代码图像是独特的技术也可以被应用于使用在本专利申请的其他部分所描述的DRM序列密钥方法的客户端库设计。当每个客户端库软件代码图像是独特的(或近似独特的)时，安全攻击或破坏被局限于单个客户端或者仅局限于使用同一客户端库软件图像的设备。存在许多方法来利用在此所述的技术来产生独特的客户端软件代码图像。下面提供一种方法来说明在用于准备独特的客户端软件库图像时系统和方法的应用。

[0154] 1.把客户端软件库设计分解成多个功能。这些功能是算法或者算法块或者代码序列。它们被统称为功能。

[0155] 2.标识每个功能和序列流的开始点(每个功能的调用次序)。例如，比方说功能8、4、2和1应当被调用以执行 DRM处理。标识调用次序8、4、2、1以及每个功能的开始点或者可以被用来标识功能的开始点的数据。

[0156] 3.分别测试每个功能以确保每个功能的软件处理是正确的，并且在客户端和服务器侧二者处正确应用DRM处理。

[0157] 4.开发标识每个代码块的开始和每个代码块的结束并且重新布置代码块的后编译和测试软件实用程序。保存每个客户端库的布置，从而在服务器与客户端通信时，服务器将了解如何利用DRM处理正确地处理数据，使得客户端可以对服务器加密数据进行解密。当每个客户端库包含软件来处理DRM序列密钥数据时，可以把DRM序列数据发送到客户端。可以使用在此所述的技术来为每个客户端独特地建立没有DRM序列密钥的库。而且，可以建立独特(加扰后)的客户端库，其中DRM序列密钥数据被并入客户端库处理中，以允许每个客户端的进一步的独特处理。没有DRM序列密钥数据的库可以在被发送到客户端时或者在由客户端安装时被加扰，并且优选地在被发送到客户端之前在服务器侧被加扰。

[0158] 5.当库中包含DRM序列密钥数据时，传送到客户端设备的代码库由多个功能组成，这些功能没有指示应当如何调用功能的信息。代码定序数据(DRM序列密钥)，或者可以被从服务器接收的数据重写的默认代码执行序列，以及执行序列变化可以被实现，而无需发送新的客户端库。这表明服务器提供的数据可以定义客户端库的代码执行序列。

[0159] 6.测试将由客户端库执行的每个可逆代码原语，以确保代码原语不破坏或损坏数据并且是真实可逆的。(也可以使用非可逆的功能，但是其在独特地随机化代码执行序列方面不太灵活，这表明，在可以如何定序方面，可以使可逆功能的执行序列比非可逆的更具随机性。)

[0160] 7.通过解译虚拟机指令集或者状态机序列或者其他定序处理来对客户端库的代

码结构进行定序。

[0161] 8.代码开发过程标识代码原语或者实现虚拟指令或可逆功能所必需的代码的地址。服务器客户端库构造器服务器处理知晓每个虚拟指令的长度。

[0162] 9.当客户端请求代码下载时，客户端库构造器服务器处理以随机次序打乱或布置虚拟指令。因此，在不对代码序列或客户端库重新编译的情况下，虚拟指令或代码原语的连续地址次序对于每个客户端来说是不同的。

[0163] 10.上述处理可以产生库，从而存在包括在代码库中的定序信息，并且客户端库构造器处理将适合的序列数据嵌入在随机化的代码库中。然而，其他实施例扩展上述客户端库存储器图像加扰处理，以建立客户端库，在客户端库中没有定序信息。在这样的实施例中，定义客户端库将如何执行数据的定序信息可以是在下载客户端库之后，在安装客户端库之后从服务器接收的数据，或者定序数据可以从受到保护的内容或者作为内容下载处理一部分或者在内容下载处理之后当消费者开始使用该内容 (播放、收听、运行、执行或访问内容)时被接收。

[0164] 11.当为客户端设备准备DRM数据时服务器会知道每个单独客户端库的布局，并且将使用对于每个客户端独特的密钥处理序列和指令序列数据来处理密钥数据。

[0165] 可以使用任何软件设计或代码结构来设计和处理在本专利申请中所述的任何数据或执行序列。可以以处理序列和其他数据来使用软件设计方法(诸如虚拟机指令集、状态机、分层状态机、流程图、定序器图、或功能点表(function point table)、或双间接功能指针表 (double indirection function pointer tables)(或多层指针间接)、或执行序列、状态或指令的链接列表)。

[0166] 另一方面在于定义被用来产生对于客户端独特的固件代码图像的处理，其中每个独特的代码图像包含公共的DRM处理步骤，或者每个代码图像可以包含对于客户端独特的DRM处理步骤。在一个实施例中，存在两个需要呈现的组件。这些组件是客户端固件库和定义被用来对从服务器接收的(未受保护的)数据进行加密的处理步骤(DRM 处理步骤)的信息。处理步骤被称为DRM处理步骤。

[0167] 对于客户端独特的DRM处理信息和对于客户端独特的或对于客户端公共的固件库可以被加载到客户端，并且以包括以下的多种方式中一种方式被执行：

[0168] 1.客户端固件或软件代码加载处理，其中每个客户端库是独特的(或近似独特的)。利用该方法，每个客户端接收独特的客户端库(代码图像或代码和图像数据)，并且优选地，在服务器所传送的软件代码图像中没有DRM 定序密钥。从服务器传送的每个库具有独特的代码和代码内固有的用于每个客户端的独特的DRM处理，并且不使用单独的DRM定序密钥。除了在使用DRM定序密钥的情况下客户端代码是独特的之外，实际的DRM处理对于每个客户端可以是独特的。

[0169] 2.添加有DRM定序密钥的以上处理。

[0170] 3.客户端固件或软件代码加载处理，其中每个客户端库是独特的，并且还需要服务器所传送的软件代码图像中的 DRM定序密钥，并且从服务器传送的每个库对于每个客户端来说是独特的，并且还使用独特的DRM定序密钥。除了客户端代码是独特的之外，实际的DRM处理对于每个客户端来说可以是独特的。此外，通过把对于客户端独特的DRM定序密钥应用于对于客户端独特的固件图像，可以使每个客户端的独特的DRM处理甚至更加独特。

[0171] 4.客户端固件或软件加载处理，其中每个客户端库是相同的，但是由作为代码库一部分发送或者作为单独步骤发送的DRM定序密钥独特地定义实际的DRM处理。

[0172] 5.两阶段或多阶段下载处理，其中一个或多个附加代码下载阶段支持公共的固件下载处理，从而每个附加下载阶段执行客户端完整性检查以确保客户端软件库还没有被修改、篡改或攻击。

[0173] 6.上面步骤中任一步骤的组合

[0174] 1.加载器-看起来像是当前DRM技术

[0175] 2.阶段2-对于客户端独特的DRM状态处理“密钥控制”数据使每个客户端是独特的。

[0176] 本领域技术人员将会进一步理解结合在此披露的实施例描述的各种示例性逻辑块、模块、电路和算法步骤通常可以实现为电子硬件、计算机软件、或两者的结合。为了明确地说明硬件和软件的这种可交换性，以上针对它们功能方面大致描述了各种示例性部件、功能块、模块、电路和步骤。这种功能是实现为硬件或者软件取决于特定应用和整个系统上施加的设计限制。本领域技术人员可以采用各种方法实现针对每个特定应用的所述功能，而这些实现方案不应当被解释为脱离本发明的范围。另外，在模块、功能块、电路或步骤内对功能的分组是为了便于描述。可以在不脱离本发明的情况下从一个模块、功能块或电路移动特定功能或步骤。

[0177] 结合在此披露的实施例描述的各种示例性逻辑块、模块和电路可以采用通用处理器、数字信号处理器(DSP)、专用集成电路(AISC)、现场可编程门阵列(FPGA)或其他可编程逻辑器件、分立式门或晶体管逻辑、分立式硬件部件，或者设计来执行在此所述功能的这些部件的任意组合来实现或执行。通用处理器可以是微处理器，在作为替换方式，处理器可以是任意处理器、控制器、微控制器或状态机。处理器还可以实现为计算装置的组合(例如DSP和微处理器的组合)、多个微处理器、与DSP内核结合的一个或多个微处理器、或其他任何这样的配置。

[0178] 结合在此披露的实施例描述的方法或算法的步骤可以直接在硬件、处理器执行的软件模块、或者在两者组合中实施。软件模块可以驻留在RAM存储器、闪速存储器、ROM存储器、EPROM存储器、 EEPROM存储器、寄存器、硬盘、可移动盘、CD-ROM、或其他任何形式的存储介质中。示范性存储介质可以耦接到处理器，使得该处理器可以从存储介质中读取信息并且向存储介质写入信息。作为替换方式，存储介质可以集成到处理器中。处理器和存储介质可以驻留在 ASIC中。

[0179] 对所披露的实施例的以上描述被提供使得本领域技术人员来构造或使用本发明。对这些实施例的各种修改对于本领域技术人员来说将会是明显的，并且在不脱离本发明的精神或范围的情况下，在此定义的一般原理可以应用于其他实施例。因此，本发明并非意在限于在此所示的实施例，而应赋予与在此披露的原理和新特征相一致的最宽的范围。

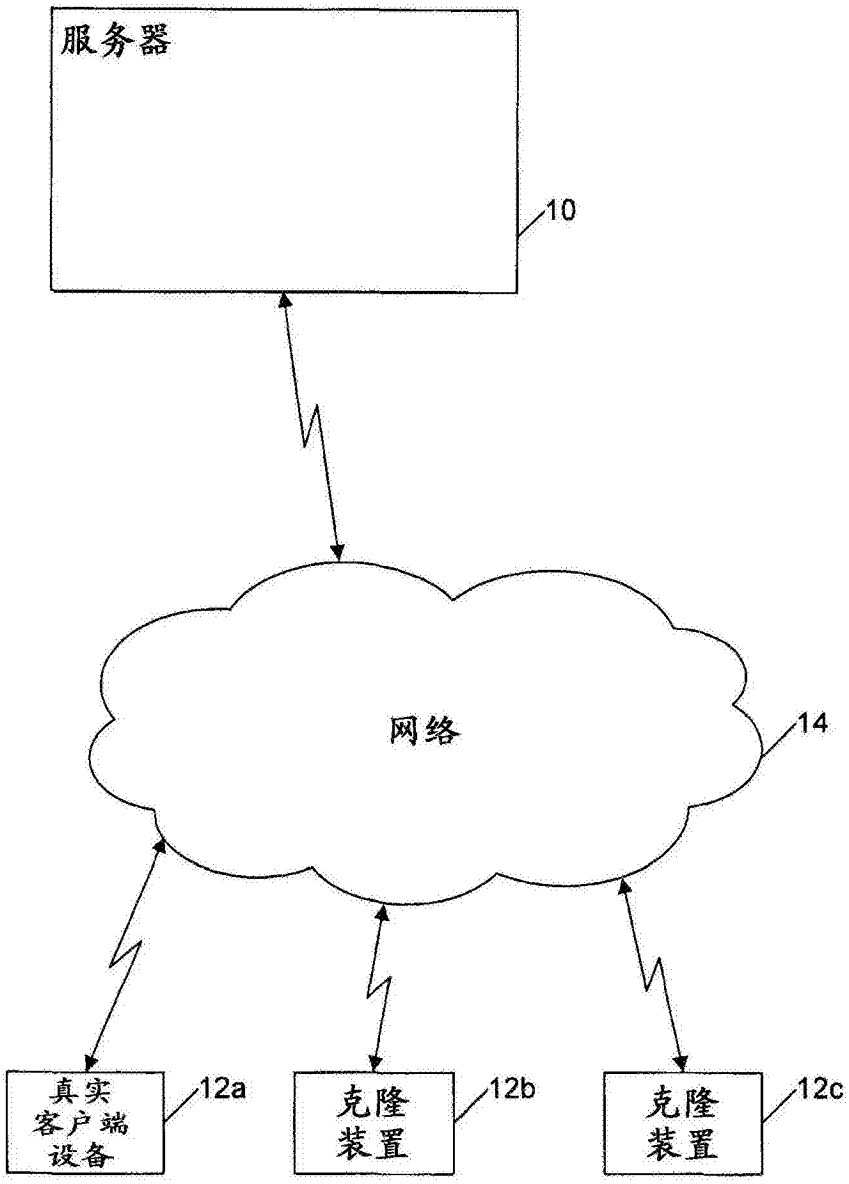


图1

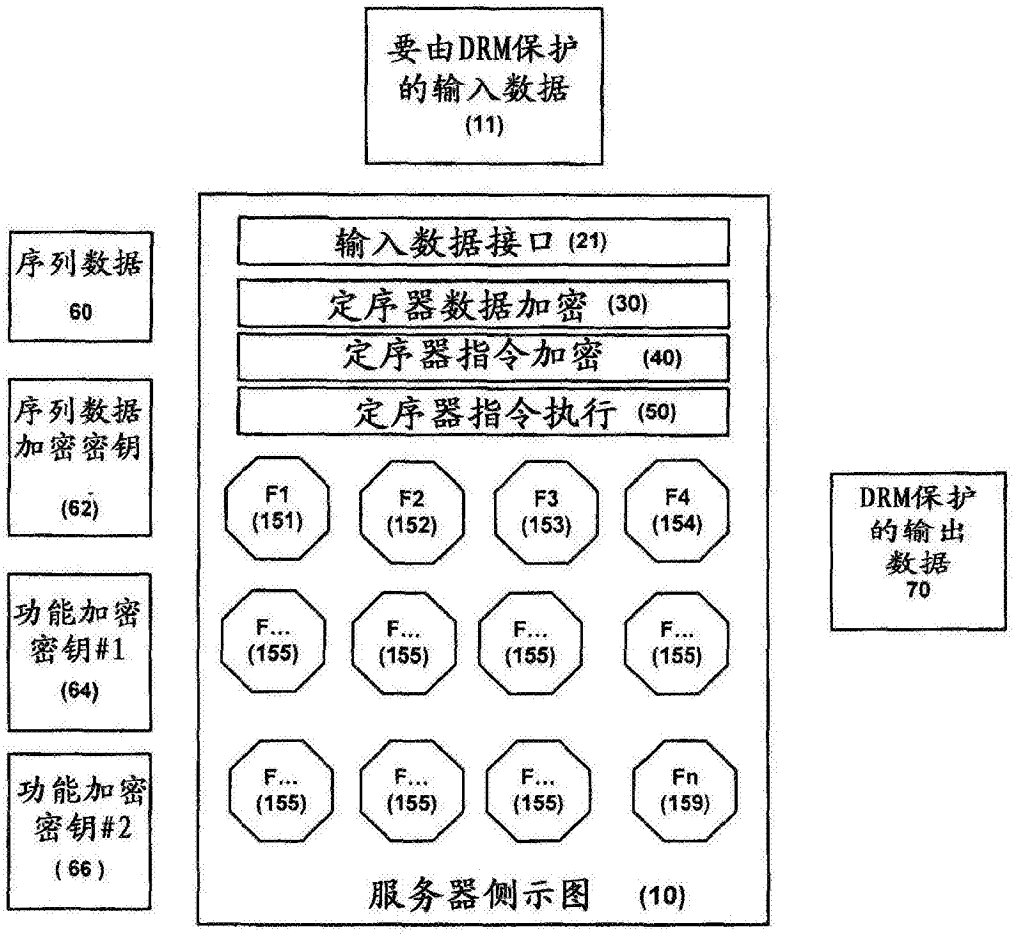

要由DRM保护的输入数据 (11)
序列数据 60
序列数据加密密钥 (62)
功能加密密钥#1 (64)
功能加密密钥#2 (66)
输入数据接口 (21)
定序器数据加密 (30)
定序器指令加密 (40)
定序器指令执行 (50)
F1 (151)
F2 (152)
F3 (153)
F4 (154)
F... (155)
Fn (159)
服务器侧示图 (10)
DRM保护的输出数据 70

图2

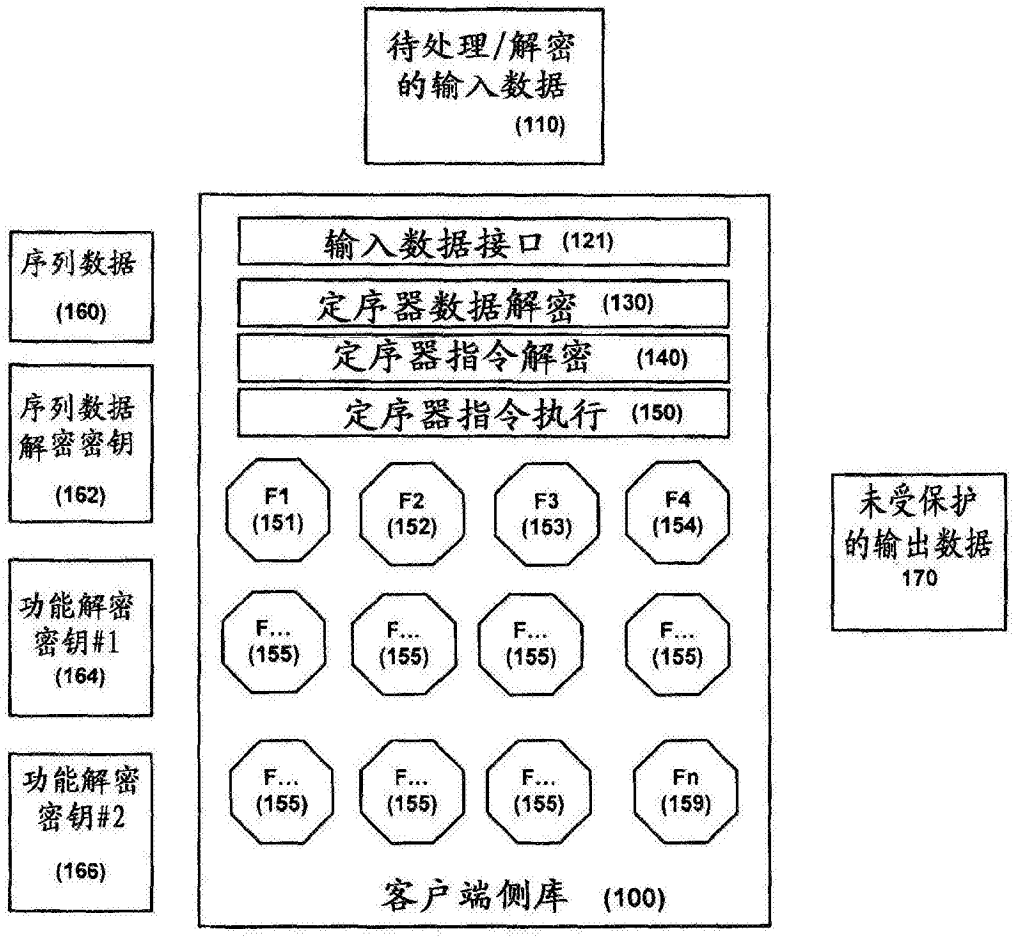

待处理/解密的输入数据 (110)
序列数据 (160)
序列数据解密密钥 (162)
功能解密密钥#1 (164)
功能解密密钥#2 (166)
输入数据接口 (121)
定序器数据解密 (130)
定序器指令解密 (140)
定序器指令执行 (150)
F1 (151)
F2 (152)
F3 (153)
F4 (154)
F... (155)
Fn (159)
客户端侧库 (100)
未受保护的输出数据 170

图3

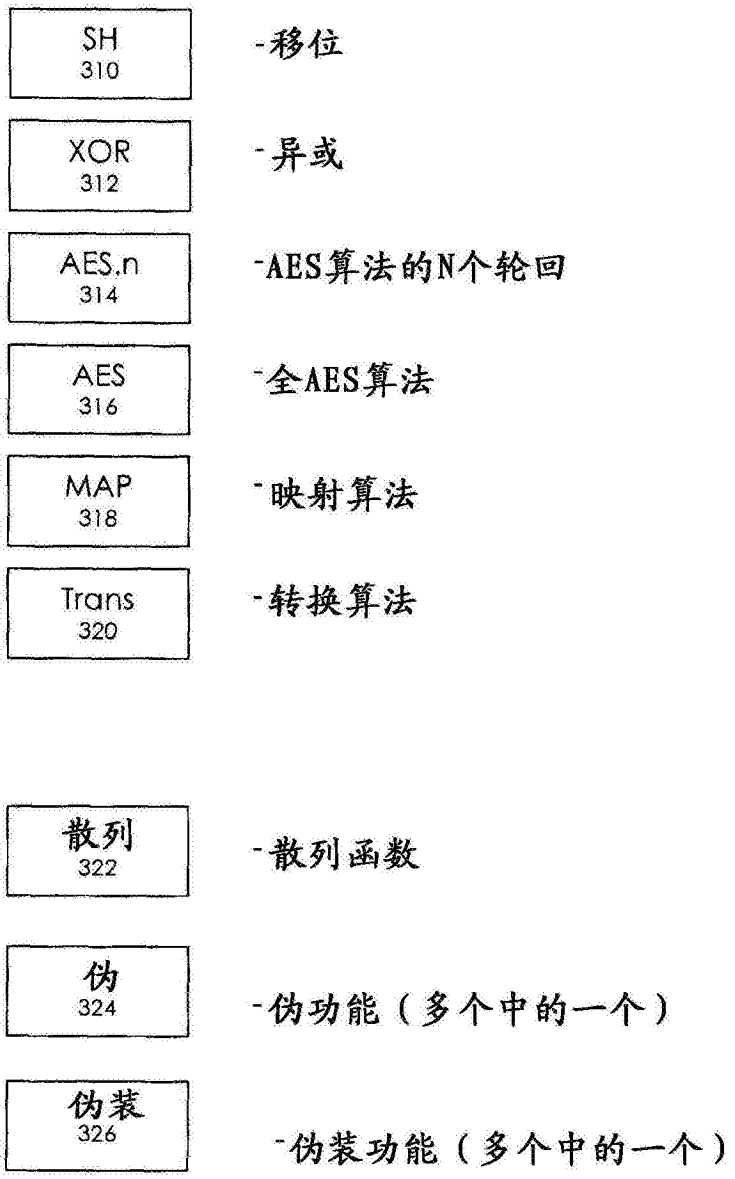

SH
310
-移位
XOR
312
-异或
AES,n
314
-AES算法的N个轮回
AES
316
-全AES算法
MAP
318
-映射算法
Trans
320
-转换算法
散列
322
-散列函数
伪
324
-伪功能（多个中的一个）
伪装
326
-伪装功能（多个中的一个）

图4

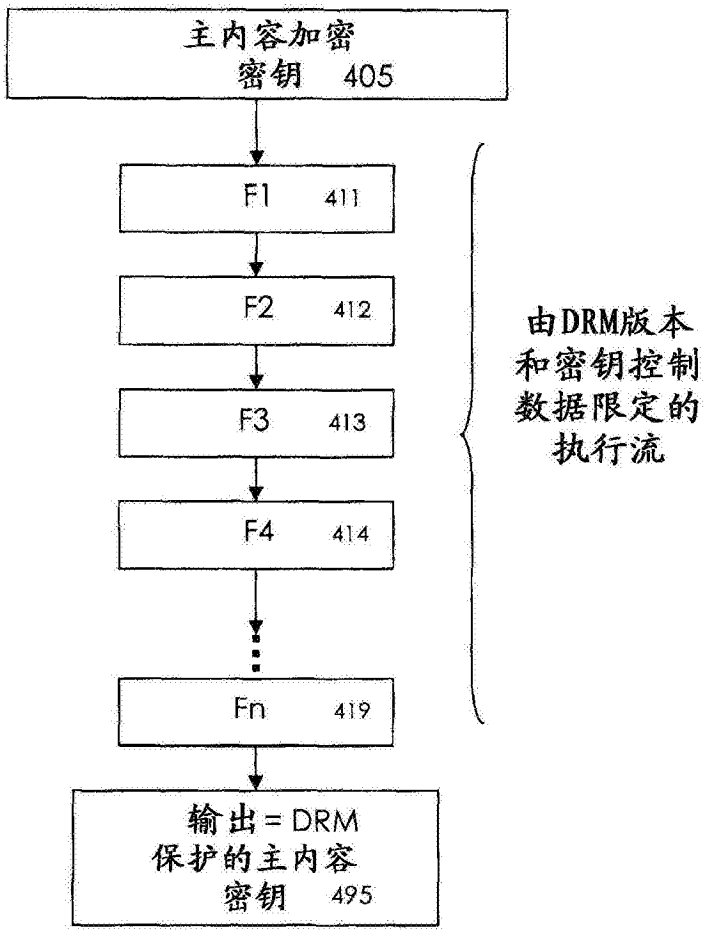

主内容加密
密钥 405
F1 411
F2 412
F3 413
F4 414
Fn 419
由DRM版本和密钥控制数据限定的执行流
输出 = DRM保护的主内容
密钥 495

图5

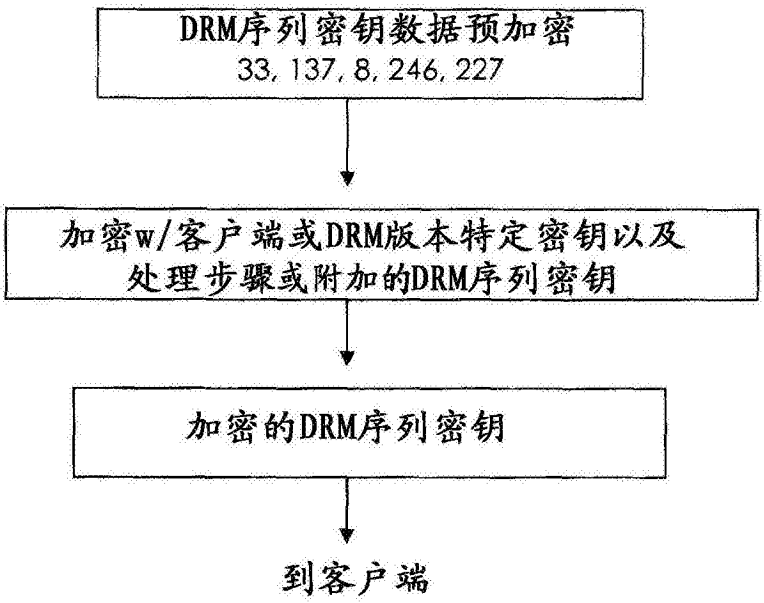

DRM序列密钥数据预加密
33, 137, 8, 246, 227
加密w/客户端或DRM版本特定密钥以及处理步骤或附加的DRM序列密钥
加密的DRM序列密钥
到客户端

图6

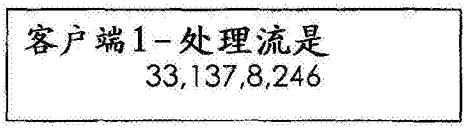

客户端1-处理流是
33,137,8,246

图7A

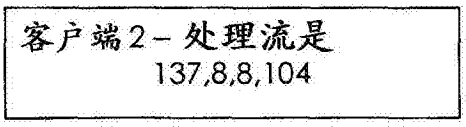

客户端2-处理流是
137,8,8,104

图7B


客户端3-处理流是
61,4,307,175

图7C

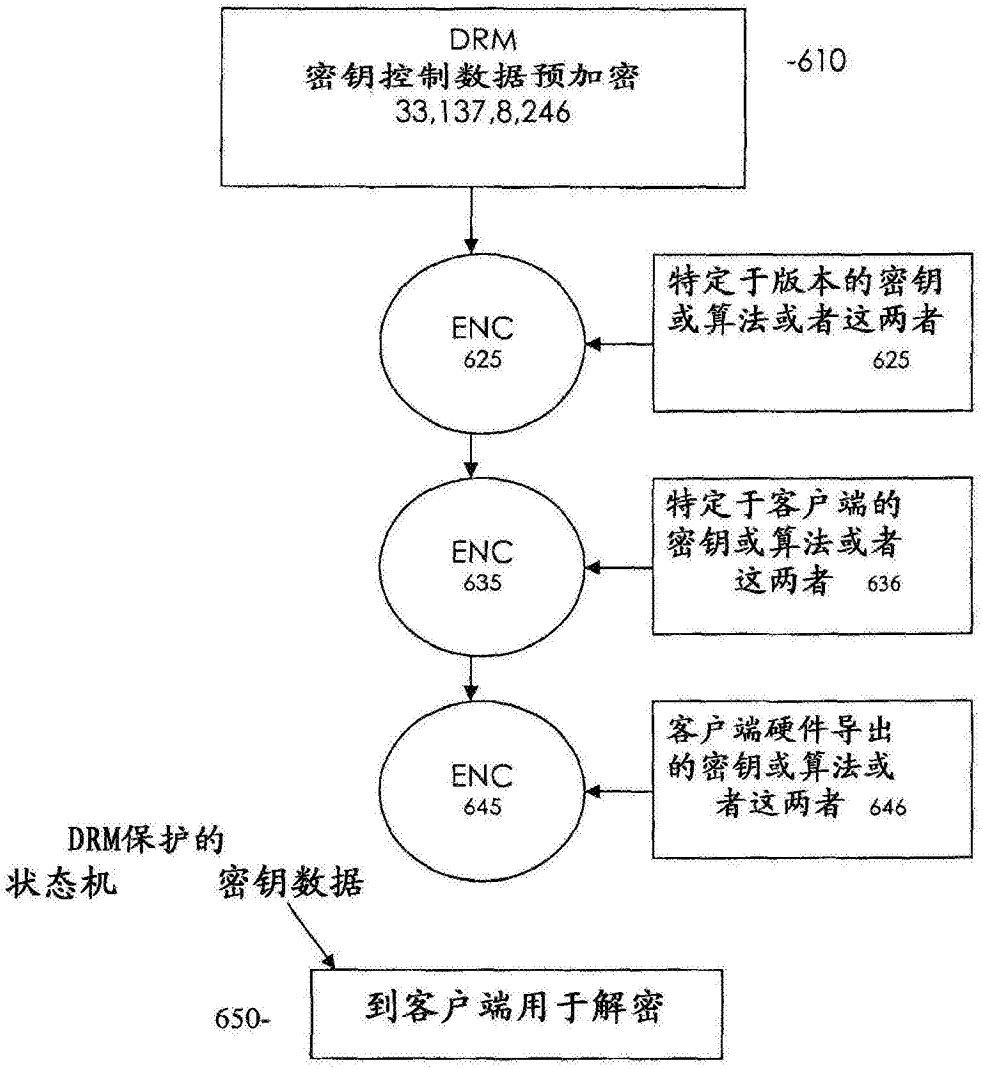

DRM
密钥控制数据预加密
33,137,8,246
-610
ENC
625
特定于版本的密钥
或算法或者这两者
625
ENC
635
特定于客户端的
密钥或算法或者
这两者 636
ENC
645
客户端硬件导出
的密钥或算法或
者这两者 646
DRM保护的
状态机
密钥数据
650-
到客户端用于解密
650-定义状态机或虚拟机处理功能/流

图8

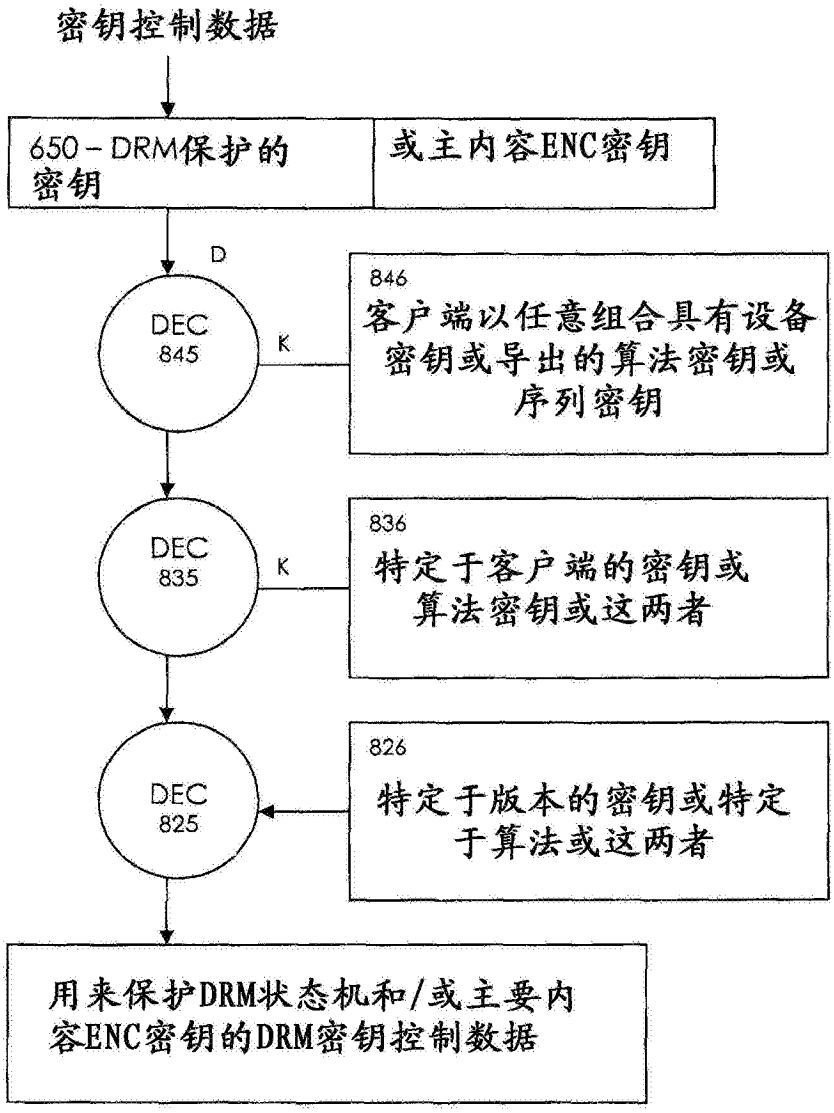

密钥控制数据
650 - DRM保护的密钥
或主内容ENC密钥
D
DEC 845
K
846
客户端以任意组合具有设备密钥或导出的算法密钥或序列密钥
DEC 835
K
836
特定于客户端的密钥或算法密钥或这两者
DEC 825
826
特定于版本的密钥或特定于算法或这两者
用来保护DRM状态机和/或主要内容ENC密钥的DRM密钥控制数据

图9

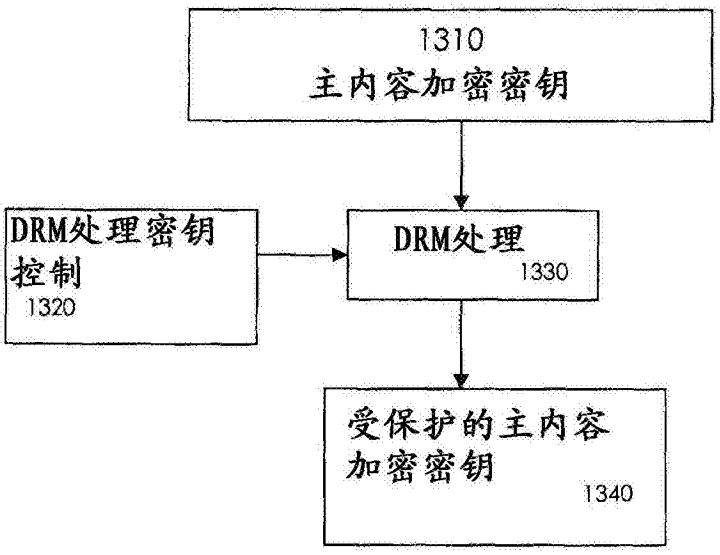


图10

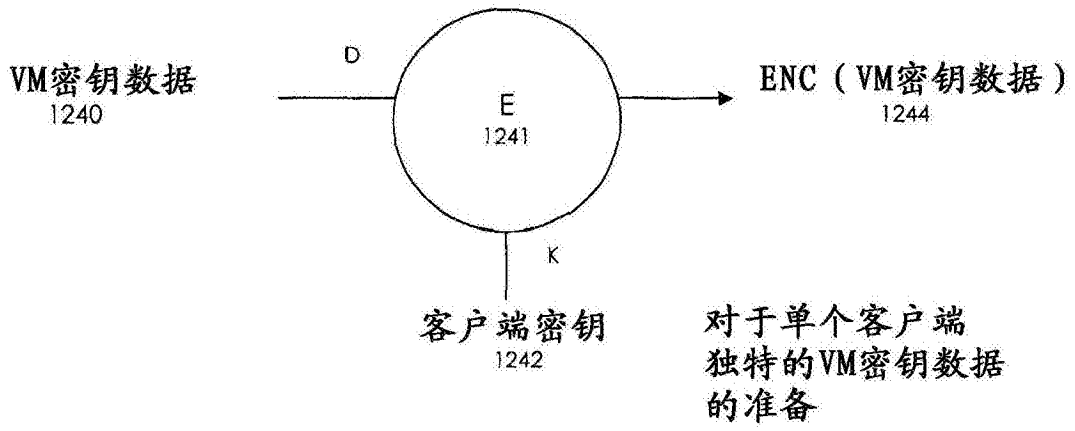
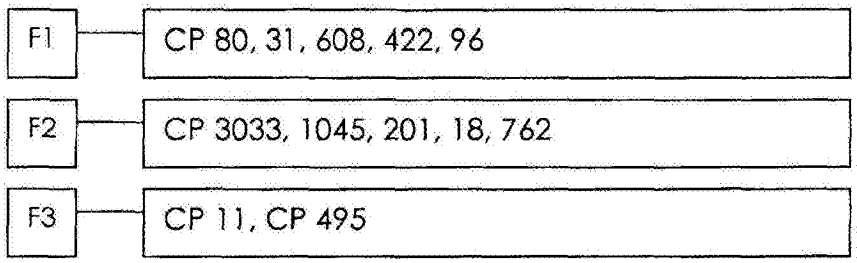


图11



序列映射

图12