(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 106462677 B
(45)授权公告日 2020.01.10

(21)申请号 201580028800.7

(22)申请日 2015.03.31

(65)同一申请的已公布的文献号
申请公布号 CN 106462677 A

(43)申请公布日 2017.02.22

(30)优先权数据
1405755.8 2014.03.31 GB

(85)PCT国际申请进入国家阶段日
2016.11.30

(86)PCT国际申请的申请数据
PCT/EP2015/057009 2015.03.31

(87)PCT国际申请的公布数据
WO2015/150376 EN 2015.10.08

(73)专利权人 爱迪德技术有限公司
地址 荷兰霍夫多普

(72)发明人 Y.古 H.约翰逊 Y.埃夫特卡里 B.西斯塔尼 R.杜兰德

(74)专利代理机构 中国专利代理(香港)有限公司 72001
代理人 张凌苗 张涛

(51)Int.Cl.
G06F 21/12(2013.01)
G06F 21/14(2013.01)

(56)对比文件
US 2008/0288921 A1,2008.11.20,
CN 102947835 A,2013.02.27,
CN 1592875 A,2005.03.09,
Bin Zeng.Strato: A Retargetable Framework for Low-Level.《22nd USENIX Security Symposium》.2013,
Bin Zeng.Strato: A Retargetable Framework for Low-Level.《22nd USENIX Security Symposium》.2013,

审查员 李婧雯

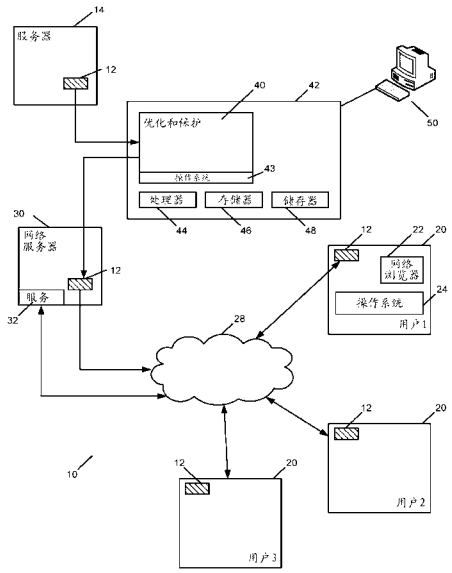
权利要求书3页 说明书28页 附图13页

(54)发明名称

一种保护软件项目的方法和装置

(57)摘要

一种方法包括:在第一中间表示中执行软件项目的优化;在不同于第一中间表示的第二中间表示中执行软件项目的保护。



CN 106462677 B

1.一种保护软件项目的方法,包括:

在第一类中间表示中执行在所述第一类中间表示中所表现的软件项目的优化;

在第二类中间表示中执行在所述第二类中间表示中所表现的软件项目的保护,其中所述第二类中间表示不同于所述第一类中间表示;以及

在执行所述优化和所述保护之后,以(i)源代码表示或者(ii)适合于供网络浏览器使用的脚本表示来输出所述软件项目。

2.如权利要求1所述的方法,其中执行优化包括在在第二类中间表示中执行保护之前和之后在第一类中间表示中执行优化。

3.如权利要求2所述的方法,进一步包括在执行优化之后且在随后执行保护之前将软件项目从第一类中间表示转换成第二类中间表示,以及在执行保护之后且在随后执行优化之前将软件项目从第二类中间表示转换成第一类中间表示。

4.如权利要求1所述的方法,其中执行保护包括在第一类中间表示中执行优化之前和之后在第二类中间表示中执行保护。

5.如权利要求1所述的方法,其中第一类中间表示是LLVM中间表示,即LLVM IR。

6.如权利要求1至5中的任一个所述的方法,其中优化包括针对软件项目的大小、运行时速度和运行时存储器要求以及核和GPU处理器的使用中的一个或多个的优化。

7.如权利要求1至5中的任一个所述的方法,其中在第二类中间表示中执行软件项目的保护包括对软件项目应用一个或多个保护技术。

8.如权利要求6所述的方法,其中一个或多个保护技术包括白盒保护技术、节点锁定技术、数据流混淆、控制流混淆和变换、同态数据变换、密钥隐藏、程序互锁以及边界混合中的一个或多个。

9.如权利要求1至5中的任一个所述的方法,其中保护组件是伪装引擎。

10.如权利要求1至5中的任一个所述的方法,进一步包括提供以输入表示的软件项目,并且在执行优化和保护之前将以输入表示的软件项目转换成第一类中间表示。

11.如权利要求10所述的方法,其中在执行优化和保护之前将以输入表示的软件项目转换成第一类中间表示包括将软件项目从输入表示转换成第二类中间表示,然后将软件项目从第二类中间表示转换成第一类中间表示。

12.如权利要求10所述的方法,其中输入表示是源代码表示。

13.如权利要求11所述的方法,其中源代码表示是C、C++、Objective-C、Java、JavaScript、C#、Ada、Fortran、ActionScript、GLSL、Haskell、Julia、Python、Ruby和Rust中的一个。

14.如权利要求10所述的方法,其中输入表示是二进制代码形式。

15.如权利要求1至5中的任一个所述的方法,其中所述输出包括在执行所述优化和保护之后将软件项目转换成输出表示。

16.如权利要求1至5中的任一个所述的方法,进一步包括在编译和链接输出的软件项目之后对所述输出的软件项目应用二进制保护。

17.如权利要求1至5中的任一个所述的方法,其中将软件项目转换成输出表示包括将软件项目从第一类中间表示转换成第二类中间表示,然后将软件项目从第二类中间表示转换成源代码表示。

18.如权利要求15所述的方法,其中输出表示是JavaScript表示。

19.如权利要求15所述的方法,其中输出表示是JavaScript的子集。

20.如权利要求1至5中的任一个所述的方法,包括将软件项目从第一类中间表示转换成脚本表示。

21.如权利要求1至5中的任一个所述的方法,其中软件项目是用于在用户设备上执行的应用程序。

22.如权利要求1至5中的任一个所述的方法,其中软件项目是库、模块和代理中的一个或多个。

23.如权利要求1至5中的任一个所述的方法,其中软件项目是安全软件项目。

24.如权利要求1至5中的任一个所述的方法,进一步包括将软件项目递送到用户设备以用于执行。

25.权利要求1至5中的任一个所述的方法,进一步包括还在第一类中间表示中执行软件项目的保护和/或在第二类中间表示中执行软件项目的优化和/或在不同于第一和第二类中间表示的另一中间表示中执行软件项目的保护和/或在不同于第一和第二类中间表示的另一中间表示中执行软件项目的优化。

26.一种保护软件项目的方法,包括对两个软件项目执行任何前述权利要求的方法,以及从软件项目中的一个调用另一软件项目。

27.一种保护软件项目的方法,包括:

在第一类中间表示中执行在所述第一类中间表示中所表现的软件项目的保护;

在第二类中间表示中执行在所述第二类中间表示中所表现的软件项目的进一步保护,其中所述第二类中间表示不同于所述第一类中间表示;以及

在执行所述保护和所述进一步保护之后,以(i)源代码表示或者(ii)适合于供网络浏览器使用的脚本表示来输出所述软件项目。

28.如权利要求27所述的方法,进一步包括在以下中的至少一个中执行软件项目的优化:第一类中间表示;第二类中间表示;以及不同于第一和第二类中间表示的另一中间表示。

29.一种被布置成实施权利要求1至28中的任一个所述的方法的装置。

30.一种或多种计算机可读介质,包括计算机程序代码,其被布置成当在适当的计算机装置上执行时实施权利要求1至28中的任一个的方法。

31.一种或多种计算机可读介质,包括根据权利要求1至28中的任一个所述的方法保护和优化的软件项目。

32.一种用于保护软件项目的计算机装置,包括:

优化器组件,其被布置成在第一类中间表示中执行在所述第一类中间表示中所表现的软件项目的优化;以及

保护器组件,其被布置成在第二类中间表示中执行在所述第二类中间表示中所表现的软件项目的保护,其中所述第二类中间表示不同于所述第一类中间表示;

其中,所述计算机装置被布置成在执行所述优化和所述保护之后,以(i)源代码表示或者(ii)适合于供网络浏览器使用的脚本表示来输出所述软件项目。

33.如权利要求32所述的装置,其中装置被布置成使得优化器组件在保护器组件在第

二类中间表示中执行软件项目的保护之前和之后在第一类中间表示中执行软件项目的优化。

34.如权利要求32所述的装置,其中装置被布置成使得保护器组件在优化器组件在第一类中间表示中执行软件项目的保护之前和之后两者在第二类中间表示中执行软件项目的保护。

35.如权利要求32至34中的任一个所述的装置,其中第一类中间表示是LLVM IR。

36.如权利要求32至34中的任一个所述的装置,其中优化组件包括一个或多个LLVM优化工具。

37.如权利要求32至34中的任一个所述的装置,其中保护组件被配置成对软件项目应用一个或多个保护技术,包括白盒保护技术、节点锁定技术、数据流混淆、控制流混淆和变换、同态数据变换、密钥隐藏、程序互锁和边界混合中的一个或多个。

38.如权利要求32至34中的任一个所述的装置,进一步包括输入转换器,其被布置成将软件项目从输入表示转换成LLVM IR。

39.如权利要求38的装置,其中输入表示是二进制表示以及源代码表示中的一个。

40.如权利要求32至34中的任一个所述的装置,进一步包括二进制重写保护工具,其被布置成在所输出的软件项目的编译和链接之后,对所输出的软件项目应用二进制重写保护。

41.如权利要求32至34中的任一个所述的装置,其中保护器组件还被布置成在第一类中间表示中执行软件项目的保护。

42.如权利要求32至34中的任一个所述的装置,包括进一步的保护器组件,被布置成在第一类中间表示中执行软件项目的保护。

43.一种统一伪装工具集,包括:

保护组件;

优化器组件;

由保护组件和优化器组件使用的中间表示之间的转换器;以及

输出,被布置成在所述优化器组件和保护器组件的使用之后,提供以(i)源代码表示或者(ii)适合于供网络浏览器使用的脚本表示的输出。

44.如权利要求43所述的统一伪装工具集,其中优化器组件包括一个或多个LLVM优化器工具,并且统一伪装工具集包括用于从输入表示转换成LLVM中间表示的一个或多个LLVM前端工具。

45.如权利要求43所述的统一伪装工具集,其中保护组件实现以下技术中的一个或多个:白盒保护技术、节点锁定技术、数据流混淆、控制流混淆、同态数据变换、控制流变换、密钥锁定、程序互锁以及边界混合。

46.如权利要求43至45中的任一个所述的统一伪装工具集,进一步包括输出转换器,其用于转换成是JavaScript的子集的输出表示。

47.如权利要求43至45中的任一个所述的统一伪装工具集,被布置成使用保护组件和优化组件对软件项目应用保护和优化的多个替换步骤。

48.一种包括权利要求43至45中的任一个所述的统一伪装工具集的计算机装置。

一种保护软件项目的方法和装置

技术领域

[0001] 本发明涉及用于提供软件的安全保护和性能优化的方法和装置。

背景技术

[0002] 近年来存在程序设计员为其提供软件的最终用户计算机设备的数目中的大幅增加，此增加中的大部分是在(包括智能电话、平板计算机等的)用于移动电话和移动计算的设备的领域中，而且在更加传统样式的台式计算机以及被嵌入诸如汽车、电视等的其它制成品中的计算机的领域中。被提供给此类设备的软件中的大部分以通常称为“app”的应用程序的形式，并且此软件可能通常被以本机(native)代码、诸如JavaScript之类的脚本语言以及诸如Java之类的其它语言的形式提供。

[0003] 如果未使用各种软件保护技术来适当地保护软件，则此类软件以及使用软件传递给用户的数据或内容常常处于危及的风险中。例如，此类技术可用来使得攻击者很难提取可以用来获得对诸如视频、音频或其它数据类型之类的内容的未授权访问的加密密钥，并且可用来使得攻击者很难复制软件以用于在其它设备上的未授权使用。

[0004] 然而，此类软件保护技术的使用可能导致软件性能的降低，例如降低执行速度、增加将软件存储在用户设备上所需的存储器的量或者增加执行所要求的存储器。此类软件保护技术还可能难以跨大范围的不同软件类型而应用，例如用不同源代码语言编写或者以特定本机代码格式存在的预先存在的软件。

[0005] 能够为软件项目提供针对攻击的保护并跨诸如不同源代码语言和本机代码类型之类的一定范围的软件表示提供此类保护、同时还保持最终用户设备上的软件的良好性能水平将是期望的。递送以该方式适当地保护的软件供用在多个不同的平台类型上也将是期望的。

发明内容

[0006] 本发明提供了一种统一安全框架，其中将被用于表示之间的转化、用于优化、编译等的第一集合中的软件工具的优点与被用于软件保护的第二集合中的软件工具的优点组合。在一个示例中，第一集合中的软件工具可以是LLVM项目的工具，其一般地使用LLVM中间表示进行操作。然而，可使用来自使用其它中间表示进行操作的其它集合的工具，例如来自Microsoft公共语言基础设施的工具，其通常使用公共中间语言CIL。下面，由第一集合中的软件工具使用的中间表示将被表示为第一中间表示。请注意，第一集合中的软件工具还可包括用于软件保护的工具，诸如二进制重写保护工具。

[0007] 中间表示是既不是最初意图用于在最终用户设备上执行、也不是最初意图被软件工程师在构造原始源代码时使用的软件表示，但是此类活动原则上当然是可能的。在下面描述的本发明的示例中，在中间表示中既没有铸造(cast)到统一安全框架的原始软件输入也没有铸造供在最终用户设备上使用的已变换软件输出。

[0008] 第二工具集合中的软件工具使用不同的中间表示，其通常更适合于或者最初意图

供对通过统一安全框架处理的软件项目应用安全保护变换的软件工具使用。此中间表示在下面一般地表示为第二中间表示，并且不同于第一中间表示。可以这样的方式设计第二中间表示：使得以诸如C和C++之类的语言的源代码可以被适当的转换工具容易地转化成第二中间表示，并且可以从其容易地重构用相同或类似语言的源代码。

[0009] 更一般地，本发明提供了一种统一安全框架，其中提供了用于对软件项目应用安全变换的软件工具使得可以在多个不同的中间表示中例如连续地对软件项目执行多个安全变换步骤。该统一安全框架还可提供用于对软件项目应用优化变换使得可以在多个不同中间表示中例如连续地对软件项目执行多个优化变换步骤的软件工具。

[0010] 本发明可用来接受以任何输入语言或本机代码/二进制表示的输入软件项目以用于优化和保护，并以各种形式输出受保护并优化的软件项目，所述各种形式包括任何期望的本机代码/二进制表示、JavaScript或JavaScript的子集等。在某些实施例中，例如特定二进制代码的输入表示可与输出表示相同，从而对现有二进制代码软件项目执行优化和保护。

[0011] 为了此目的，本发明提供了一种方法，包括在第一中间表示中执行软件项目的优化，并且在不同于第一中间表示的第二中间表示中执行软件项目的保护。

[0012] 可在执行第二中间表示中的保护之前和之后两者执行第一中间表示中的优化，并且本方法因此可包括在第一次执行优化之后且在随后执行保护之前将软件项目从第一中间表示转换成第二中间表示，并且在执行保护之后和随后第二次执行优化之前将软件项目从第二中间表示转换成第一中间表示。

[0013] 类似地，可在执行第一中间表示中的优化之前和之后两者执行第二中间表示中的保护，并且本方法因此可包括在第一次执行保护之后且在随后执行优化之前将软件项目从第二中间表示转换成第一中间表示，并且在执行优化之后且在随后第二次执行保护之前将软件项目从第一中间表示转换成第二中间表示。

[0014] 可以交替地执行相关中间表示中的保护和优化的步骤任何次数，以保护或优化开始，并且以交替方式继续进行一个或多个其它步骤。

[0015] 如上面所提及的那样，第一中间表示可以是LLVM中间表示LLVM IR，但是可以使用其它中间表示，诸如Microsoft CIL。

[0016] 更一般地，本发明可提供使用在一个或多个中间表示中执行的优化步骤来执行软件项目的优化，并且使用其某些或全部可与被用于执行优化的中间表示相同或不同的一个或多个中间表示中的保护步骤来执行软件项目的保护。

[0017] 软件项目的优化可包括各种类型的优化，例如针对软件项目的大小、运行时速度和运行时存储器要求中的一个或多个的优化。用以实现此类优化的技术可包括矢量化、空闲时间、恒定传播、无用赋值消除、内联展开、可达性分析、保护突破正常(protection break normal)及其它优化。

[0018] 第二中间表示中的软件项目的保护包括对软件项目应用一个或多个保护技术，特别是保护软件的程序和/或数据方面免于攻击的安全保护技术。此类技术可包括例如白盒保护技术、节点锁定技术、数据流混淆、控制流混淆和变换、同态数据变换、密钥隐藏、程序互锁、边界混合及其它。可以以各种方式将所使用的技术组合在一起以形成一个或多个工具，例如作为被实现为优化和保护工具集的部分的伪装(cloaking)引擎。

[0019] 以通常不同于第一和第二中间表示两者的输入表示提供软件项目。本方法因此可涉及到在执行优化之前且通常也在执行上面提及的保护之前将软件项目从输入表示转换成第一中间表示。在某些实施例中，输入表示中的软件项目被转换成第二中间表示，并且然后在第一优化之前且可选地也在执行保护之前被从第二中间表示转换。

[0020] 输入表示可以是源代码表示，诸如C、C++、Objective-C、Java、JavaScript、C#、Ada、Fortran、ActionScript、GLSL、Haskell、Julia、Python、Ruby和Rust。然而，输入表示可以替换地是本机代码表示，例如用于诸如x86、x86-64、ARM、SPARC、PowerPC、MIPS以及m68k处理器族中的任何之类的特定处理器族的本机代码(即二进制代码)表示。输入表示还可以是硬件描述语言(HDL)。如众所周知的，HDL是可用来对电子电路的结构、设计和操作进行编程的计算机程序语言。HDL可以例如是VHDL或Verilog，但是将领会的是许多其它HDL存在并且可以替换地在本发明的实施例中使用。由于HDL(及其使用和实现)是众所周知的，在本文中将不对其进行更详细的描述，然而，例如在http://en.wikipedia.org/wiki/Hardware_description_language处可以找到更多细节，其全部公开被通过引用结合到本文中。

[0021] 当已执行了上面的优化和保护过程时，可将软件项目转换成输出表示。此处理阶段还可包括进一步优化和/或保护阶段。在某些实施例中，将软件项目转换成输出表示包括将软件项目编译(并且通常还链接)成输出表示，例如本机代码表示。然后还可以在编译和链接之后对软件项目应用进一步二进制保护技术。

[0022] 在编译之后，可首先将软件项目从第一中间表示转换成第二中间表示并转换到被传递至编译器的源代码表示上，或者可以直接地将软件项目以第一中间表示传递至编译器。在第一情况下，可以使用对源代码表示进行操作的编译器，诸如C/C++编译器。在第二情况下，如果第一中间表示是LLVM IR，则可以使用LLVM编译器。在任何情况下，编译器可以是优化编译器，以便向受保护软件项目提供进一步水平的优化。

[0023] 将软件项目转换成输出表示还可包括在编译之前对以第一中间表示的软件项目应用二进制重写保护工具，和/或可在过程中的其它时间应用此类工具。

[0024] 代替将软件项目编译成本机代码表示，可替换地将软件项目转换成脚本表示且尤其转换成可以在最终用户设备上执行的脚本表示。方便地，可出于此目的使用JavaScript表示，因为此类脚本可以由最终用户设备上的网络(web)浏览器直接地执行。更特别地，可使用是JavaScript的子集的asm.js表示，因为asm.js适合于在最终用户设备上的特别高效的执行。例如，如果第一中间表示是LLVM IR，则可使用Emscripten工具来将软件项目从第一中间表示转换成asm.js表示。

[0025] 如果输入表示是硬件描述语言，则输出表示通常可以以能够在更加面向硬件的级别处(诸如在网表中)描述电子电路的相应表示。在本文中描述诸如编译和链接之类的处理方面的情况下，技术人员将领会的是当将本发明与HDL输入表示一起使用时，可使用诸如使用适当工具的合成之类的等同步骤，并且可将适用于HDL工作的适当软件工具用于本发明的保护和优化方面。输出软件项目然后是应用适当混淆/保护和优化步骤的情况下的电子系统的描述。

[0026] 软件项目可以是多种软件项目中的任何，诸如用于在用户设备、库、模块、代理等上执行的应用程序。特别地，软件项目可以是诸如库、模块或代理之类的安全软件项目，其包含用于实现诸如加密/解密和数字权限管理功能之类的安全功能的软件。可将本方法应

用于两个此类软件项目，并且这些软件项目中的一个可例如通过过程调用或其它引用而使用另一个中的功能。类似地，根据本发明优化和保护的软件项目可利用或调用诸如系统层或硬件层之类的较低层中的安全相关或受保护的功能。类似地，软件项目可描述电子系统，并且被以HDL提供用于输入到本发明的实施例。

[0027] 本发明还提供了一种保护软件项目的方法，包括对软件项目应用一个或多个保护技术，并且使用一个或多个LLVM工具来优化软件项目，并且可将本发明的此方面与在本文中的别处提及的各种选项组合。例如，可使用被布置成使用不同于LLVM中间表示的中间表示进行操作的保护组件来对软件项目应用一个或多个保护技术，并且本方法还可包括使用LLVM工具将软件项目在一个或多个表示与LLVM中间表示之间转换。本方法可用来以asm.js或本机代码表示中的一个输出受保护且优化的软件项目。

[0028] 在如上面所讨论的软件项目的处理之后，可将软件项目递送到一个或多个用户设备以用于执行。可以以各种方式(诸如通过有线、光学或无线网络、使用计算机可读介质以及以其它方式)将软件项目递送到用户设备。

[0029] 可通过网络或以其它方式在一个或多个计算机可读介质上提供用于提供所讨论方法和装置的软件，用于与适当的输入和输出设施相组合地在适当的计算机装置(例如包括存储器和一个或多个处理器的计算机设备或多个此类设备)上执行，以使得操作员能够控制诸如键盘、鼠标和屏幕之类的装置连同用于存储计算机程序代码以便使本发明在装置上付诸实践的持久性储存器。

[0030] 本发明因此还可提供用于保护软件项目的计算机装置，包括被布置成在诸如LLVM IR之类的第一中间表示中执行软件项目的优化的优化器组件以及被布置成在第二中间表示中执行软件项目的保护的保护器组件。

[0031] 装置可被布置成使得优化器组件在保护器组件在第二中间表示中执行软件项目的保护之前和之后两者在第一中间表示中执行软件项目的优化。

[0032] 优化组件可包括一个或多个LLVM优化工具。

[0033] 保护组件可被布置成对软件项目应用一个或多个保护技术，包括白盒保护技术、节点锁定技术、数据流混淆、控制流混淆和变换、同态数据变换、密钥隐藏、程序互锁和边界混合中的一个或多个。

[0034] 装置可以进一步包括被布置成将软件项目从输入表示转换成LLVM IR的输入转换器，并且该输入表示可以是二进制或本机代码表示、字节代码表示以及源代码表示中的一个。该装置可以进一步包括被布置成将已优化且保护的软件项目输出为二进制代码的编译器和链接器以及被布置成将已优化且保护的软件项目输出为asm.js代码的输出转换器。

[0035] 本发明还提供了一种统一伪装工具集，其包括保护组件、优化器组件以及用于在由保护组件和优化器组件使用的中间表示之间转换的一个或多个转换器。优化器组件可包括一个或多个LLVM优化器工具，并且统一伪装工具集可包括用于从输入表示转换成LLVM中间表示的一个或多个LLVM前端工具。在统一伪装工具集的某些实施例中，可提供保护组件和/或优化器组件以在多于一个中间表示中对软件项目应用变换。

[0036] 统一伪装工具集还可例如用实现以下技术中的一个或多个的保护组件实现如在本文中阐述的所述实施例的各种其它方面：白盒保护技术、节点锁定技术、数据流混淆、控制流混淆和变换、同态数据变换、密钥隐藏、程序互锁以及边界混合；该统一伪装工具集进

一步包括被布置成编译并链接成本机代码表示的编译器和链接器;并且该统一伪装工具集进一步包括用于转换成是JavaScript的子集的输出表示的输出转换器。

[0037] 本发明还提供了已使用所述方法和/或装置优化和保护的一个或多个软件项目,并且可在计算机存储器中、在计算机可读介质上、通过电信或计算机网络以及以其它方式提供、存储或传输此类软件项目。

附图说明

[0038] 现在将参考附图仅通过示例的方式描述本发明的实施例,在所述附图中:

[0039] 图1示意性地图示了根据本发明的包括优化和保护工具集40的计算机系统的示例;

[0040] 图2更详细地图示了图1的优化和保护工具集40的实施例;

[0041] 图3提供了本发明的方法实施例的流程图;

[0042] 图4图示了可以由图2的优化和保护工具集40实现的工作流程;

[0043] 图5图示了与图4的工作流程类似的工作流程,但在其内,以源代码表示的输入软件项目被使用LLVM前端工具转换成LLVM IR;

[0044] 图6类似于图5,但具有以二进制或本机代码表示的输入软件项目;

[0045] 图7图示了与图4至6的工作流程类似的工作流程,但在其内,使用LLVM编译器中间层工具来实现以第一中间表示的软件项目的二进制重写保护;

[0046] 图8示出了可使用图2的优化和保护工具集来实现的工作流程,其中输出表示是asm.js或其它可执行脚本表示;

[0047] 图9用某些进一步变化和细节示意性地示出了图2的优化和保护工具集;

[0048] 图10示出了可以如何将图2的布置扩展至使用更大数目的中间表示,并在这些中间表示中的不同的一些中应用优化和/或保护;

[0049] 图11图示了通过优化和保护工具集对诸如安全库、模块和代理之类的软件项目的处理;

[0050] 图12是示意性地图示了根据本发明的实施例的结构保护方法的流程图;

[0051] 图13示意性地图示了示例字典树(trie);以及

[0052] 图14示意性地图示了字典树形式的受保护结构。

具体实施方式

[0053] 在随后的描述中和附图中,描述了本发明的某些实施例。然而,将领会的是本发明不限于所描述的实施例,并且某些实施例可不包括下面描述的所有特征。然而,将明显的是可以在不脱离如在所附权利要求书中阐述的本发明的更宽泛精神和范围的情况下在本文中做出各种修改和改变。

[0054] 现在参考图1,示出了在其内可使本发明付诸实践的示例性计算机系统10。软件项目12例如由软件项目12先前已被存储在其处的服务器14提供。软件项目12可意图用于各种不同的目的,但是在图1的系统中,其是意图用于在多个用户计算机20中的一个或多个上执行和使用的应用程序(有时称为app,取决于诸如应用程序被如何递送和其在用户设备和更宽泛的操作环境的背景下如何被使用的各方面)。用户计算机20可以是个人计算机、智能电

话、平板计算机或任何其它适当的用户设备。通常，此类用户设备20将包括向在用户设备上运行的其它软件实体(诸如网络浏览器22)提供服务的操作系统24。可以以各种形式向用户设备递送软件项目12，但是通常其可以是本机可执行代码、诸如Java字节代码之类的一般底层代码或诸如Java脚本之类的脚本语言的形式。通常，将在网络浏览器22内或在其直接控制下执行一般底层代码或脚本语言软件项目12。以本机可执行代码的软件项目12更有可能在操作系统24的直接控制下执行，但是诸如Google NaCI和PNaCI之类的某些类型的本机代码是在网络浏览器环境内执行的。

[0055] 图1的软件项目12通常可由远程网络服务器30通过数据网络28(诸如因特网)被递送到一个或多个用户设备，但是可使用其它递送和安装布置。所图示的网络服务器或一个或多个其它服务器还可向用户设备20且特别是向在用户设备20上执行的软件项目12提供数据、支持、数字权限管理和/或其它服务32。

[0056] 软件项目12可能在用户设备20上易受以各种方式的攻击和损害，无论是在那些设备20上的执行之前、期间还是之后。例如，软件项目可实现攻击者可尝试例如通过提取算法的加密密钥或细节(其可使得能够实现针对该特定软件项目、针对特定数字内容等的数字权限管理技术的未来规避)来损害的数字权限管理技术。

[0057] 系统10因此还提供了优化和保护工具集40，其被用来在软件项目12递送到用户设备20之前优化和保护软件项目12。在图1中，优化和保护工具集40在软件项目12被递送到网络服务器20之前作用于软件项目12，但是其可以在服务器14、网络服务器30中、在开发环境(未示出)中或者在别处实现。图1中的优化和保护工具集40被示为在操作系统43的控制下在适当的计算机装置42上执行。计算机装置42通常将包括一个或多个处理器44，其在通过输入/输出设施50进行的用户的控制下使用存储器46来执行优化和保护工具集40的软件代码。可以将计算机装置42及优化和保护工具集40的功能跨被适当的数据网络连接所连接的多个计算机单元分布。用来提供优化和保护工具集40的软件的部分或全部可被存储在非易失性储存器48中和/或一个或多个计算机可读介质中和/或可通过数据网络传输到计算机装置42。

[0058] 请注意，要通过本发明的各方面来优化和保护的软件项目12还可以是供与诸如应用程序之类的另一软件项目一起或被其使用的组件。为此，软件项目12可以是例如库、模块、代理或类似物。

[0059] 在图2中示意性地示出了优化和保护工具集40的示例性实施方式。优化和保护工具集40包括优化器组件100和保护器组件110。优化器组件100适合于在软件项目12上实现优化技术。优化器组件100被配置成在第一中间表示IR1中实现此类技术，使得需要在优化器组件100执行软件项目的优化之前将软件项目12呈现成第一中间表示IR1。保护器组件110适合于在软件项目12上实现保护技术。保护组件被配置成在第二中间表示IR2中实现此类技术，使得需要在保护器组件110执行软件项目12的保护之前将软件项目12呈现成该第二中间表示。第一和第二中间表示是相互不同的表示。通常，保护器组件110不能在软件项目处于第一中间表示中时对软件项目进行操作，并且优化器组件不能在软件项目处于第二中间表示中时对软件项目进行操作。

[0060] 优化器组件100和保护器组件110中的每一个被实现为优化和保护工具集40中的多个子组件102、112。特定组件的子组件可提供相对于彼此而言不同和/或复制的功能，例

如使得可将组件的总体作用以各种方式分布在优化和保护工具集40的软件内。

[0061] 优化和保护工具集40还提供了多个转换器，其适合于将软件项目12从一个表示转换成另一个。这些转换器包括被布置成将软件项目从优化器组件100所使用的第一中间表示IR1转换成保护器组件110所使用的第二中间表示IR2的第一转换器组件120以及被布置成将软件项目从保护器组件110所使用的第二中间表示IR2转换成优化器组件100所使用的第一中间表示IR1的第二转换器组件122。当然，可将第一和第二转换器组件120、122组合在单个功能软件单元(诸如单模块、可执行或面向对象方法)中，如果期望的话。

[0062] 软件项目12被以输入表示Ri提供给优化和保护工具集40。此输入表示可以是任何多个不同表示中的一个，例如或者第一和第二中间表示IR1、IR2或者诸如源代码表示、二进制代码表示等的另一表示。

[0063] 类似地，软件项目12被以输出表示Ro从优化和保护工具集40输出。该输出表示也可以是任何数目的不同表示中的一个，例如第一和第二中间表示IR1、IR2或者诸如源代码表示、二进制代码表示等的另一表示中的任一个。

[0064] 优化和保护工具集40还可包括一个或多个其它组件，每个被布置成对在特定表示中的软件项目12进行操作。此类组件可例如包括提供被布置成对在二进制表示Rb中的软件项目12进行操作的二进制保护工具的二进制保护组件130、提供被布置成对在二进制表示或者诸如第一中间表示等的某个其它表示中的软件项目12进行操作的二进制重写保护工具的二进制重写保护组件135。

[0065] 除第一转换器组件120和第二转换器组件122之外，优化和保护工具集40因此还提供有在图2中还示为X3……Xn的其它转换器组件124，其被用于根据期望将软件项目12在各种表示之间转换。举例来说，一个此类转换器组件124、126可从C/C++源代码表示转换至第二中间表示IR2，并且另一此类转换器组件可从第二中间表示IR2转换回到C/C++源代码表示。

[0066] 图2还示出了作为优化和保护工具集40的部分的一个或多个编译器或编译器和链接器组件140，其可以用来例如对软件项目12进行编译和链接以将软件项目12通常转换成本机或二进制代码表示或另一适当的目标表示。

[0067] 可以用于输入表示Ri以及优化和保护工具集40内的其它表示的源代码表示的示例包括C、C++、Objective-C、C#、Java、JavaScript、Ada、Fortran、ActionScript、GLSL、Haskell、Julia、Python、Rubu以及Rust，但技术人员将知道许多其它的。输入表示Ri可以替换地是本机或二进制代码、字节代码等或者可能是第一和第二中间表示中的一个。

[0068] 可以用于输出表示Ro的表示的示例包括用于在用户设备上直接执行的本机代码表示，其包括适合用于在网络浏览器的控制下执行的诸如PNaCl和NaCl之类的本机代码表示，诸如Java字节代码之类的字节代码表示，诸如Java源代码之类的适合用于解释执行或运行时编译的表示，诸如JavaScript和诸如asm.js之类的JavaScript的子集之类的脚本表示并且可能包括第一或第二中间表示。

[0069] 第一中间表示IR1通常可被选作方便用于、适合用于或者另外被选用于执行优化技术的中间表示。特别地，第一中间表示可以是LLVM IR(LLVM中间表示)。为技术人员所知且例如在LLVM网站“http://llvm.org”处讨论的LLVM项目提供许多模块化且可再使用的编译器和工具链技术，其：

[0070] (i)引入支持语言无关指令集和类型系统的良好地指定的通用中间表示(LLVM IR)；

[0071] (ii)提供了完整编译器系统和基础设施的中间层，其获取在LLVM IR中的软件项目并发射在LLVM IR中的软件项目12的高度优化版本，其准备好用于用大范围的源代码表示编写的程序的编译时、链接时、运行时和“空闲时”优化；

[0072] (iii)支持用于源代码及其它表示的丰富LLVM前端工具，所述其它表示不仅包括C和C++，而且包括其它流行的编程语言，诸如上面提及的源代码语言以及Java字节代码等；

[0073] (iv)用一组LLVM后端工具，目前支持许多其它流行平台和系统，并且不久的将来将支持更多移动平台；以及

[0074] (v)用OpenGL及低端和高端GPU工作。

[0075] 适于用作第一中间表示的其它表示包括Microsoft公共中间语言(CIL)。

[0076] 第二中间表示IR2通常可被选作方便用于、适合用于或者另外被选用于执行保护技术的中间表示。第二中间表示可例如以此类方式来设计和实现，即用例如C和C++的特定语言的源代码可以被容易地转换成第二中间表示，并且使得用相同或类似语言的源代码可以容易地由第二中间表示构成。

[0077] 由优化器执行的优化技术可包括用以增加软件项目12的执行速度、减少执行空闲时间、减少软件项目12的存储和/或执行所需的存储器、增加核或GPU的使用等的技术。这些及其它优化功能方便地由LLVM项目提供。用以实现此类优化的技术可包括矢量化、空闲时间、恒定传播、无用分配消除、内联展开、可达性分析、正常保护突破及其它优化。

[0078] 保护器组件110的目的是保护软件项目12的功能或数据处理和/或保护软件项目12所使用或处理的数据。这可以通过应用伪装技术来实现，所述伪装技术诸如同态数据变换、控制流变换、白盒密码术、密钥隐藏、程序互锁和边界混合。

[0079] 特别地，被保护器组件110处理之后的软件项目12将提供与在此类处理之前相同的功能或数据处理——然而，此功能或数据处理通常是以如下这样的方式在受保护软件项目12中实现的，所述方式即使得用户设备20的操作员不能以非计划或未授权方式从软件项目12访问或使用此功能或数据处理(而如果用户设备20提供有未受保护形式的软件项目12，则用户设备20的操作员可能能够以非计划或未授权方式访问或使用该功能或数据处理)。类似地，软件项目12在被保护器组件110处理之后可以以受保护或被混淆的方式存储秘密信息(诸如密码密钥)以从而使得攻击者更加难以(如果不是不可能的话)推断或访问该秘密信息(而如果用户设备20提供有未受保护形式的软件项目12，则用户设备20的操作员可能能够推断或访问该秘密信息)。

[0080] 例如：

[0081] ·软件项目12可包括至少部分地基于软件项目12要处理的一个或多个数据项目的判定(例如判定块或分支点)。如果软件项目12被以未受保护形式提供给用户设备20，则攻击者可能能够迫使软件项目12执行，使得在处理判定之后遵循执行路径，即使该执行路径并不意图被遵循。例如，判定可包括测试程序变量B为真还是假，并且软件项目12可被布置成使得如果判定标识到B为真，则遵循/执行执行路径P_T，而如果判定标识到B为假，则遵循/执行执行路径P_F。在该情况下，攻击者可以(例如通过使用调试程序)在判定标识到B为真的情况下迫使软件项目12遵循路径P_F和/或在判定标识到B为假的情况下迫使软件项目12遵

循路径P_T。因此，在某些实施例中，保护器组件110旨在通过对软件项目12内的判定应用一个或多个软件保护技术来防止攻击者(或者至少使得攻击者更加难以)这样做。

[0082] ·软件项目12可包括安全相关功能、访问控制功能、密码功能以及权限管理功能中的一个或多个。此类功能常常涉及到秘密数据的使用，秘密数据诸如一个或多个密码密钥。处理可涉及到使用一个或多个密码密钥和/或对或用一个或多个密码密钥进行操作。如果攻击者能够标识或确定秘密数据，则已发生安全漏洞，并且可规避被秘密数据保护的数据(诸如音频和/或视频内容)的控制或管理。因此，在某些实施例中，保护器组件110旨在通过对软件项目12内的此类功能应用一个或多个软件保护技术来防止攻击者(或者至少使得攻击者更加难以)标识或确定一个或多个秘密数据片。

[0083] “白盒”环境是用于软件项目的执行环境，其中，假设软件项目的攻击者具有被操作的数据(包括中间值)、存储器内容和软件项目的执行/过程流程的完全访问以及可见性。此外，在白盒环境中，假设攻击者能够例如通过使用调试程序来修改被操作的数据、存储器内容和软件项目的执行/过程流程——这样，攻击者可以对软件项目进行实验并尝试操纵其操作，旨在规避最初计划的功能和/或标识秘密信息和/或为了其它目的。

[0084] 事实上，甚至可假设攻击者知道由软件项目执行的底层算法。然而，软件项目可能需要使用秘密信息(例如一个或多个密码密钥)，其中，此信息需要保持向攻击者隐藏。类似地，将期望防止攻击者修改软件项目的执行/控制流程，例如防止攻击者迫使软件项目在判定块之后采取一个执行路径而不是合法的执行路径。存在在本文中称为“白盒混淆技术”的许多技术，其用于对软件项目12进行变换，使得其对白盒攻击有抵抗力。在S. Chow等人在Selected Areas in Cryptography, 9th Annual International Workshop, SAC 2002, Lecture Notes in Computer Science 2595 (2003), p250-270中的“White-Box Cryptography and an AES Implementation”中和S. Chow等人在Digital Rights Management, ACM CCS-9 Workshop, DRM 2002, Lecture Notes in Computer Science 2696 (2003), p1 -15中的“A White-box DES Implementation for DRM Applications”中可以找到此类白盒混淆技术的示例，其全部公开被通过引用结合到本文中。在US 61 / 055,694和WO 2009/140774中可以找到附加示例，其全部公开被通过引用结合到本文中。某些白盒混淆技术实现了数据流混淆——参见例如US7,350,085、US7,397,916、US6,594,761和US6,842,862，其全部公开被通过引用结合到本文中。某些白盒混淆技术实现了控制流混淆——参见例如US6,779,114、US6,594,761和US6,842,862，其全部公开被通过引用结合到本文中。然而，将领会的是存在其它白盒混淆技术，并且本发明的实施例可使用任何白盒混淆技术。

[0085] 作为另一示例，可能的是软件项目12可意图提供(或分发)给特定用户设备20(或一组特定的用户设备20)并被其使用，并且因此期望将软件项目12“锁定”到该特定用户设备20，即防止软件项目12在另一用户设备20上执行。因此，存在在本文中称为“节点锁定”保护技术的许多技术，其用于对软件项目12进行变换，使得受保护的软件项目12可以在一个或多个预定/特定用户设备20上执行(或被其执行)，但是将不会在其它用户设备上执行。在WO2012/126077中可以找到此类节点锁定技术的示例，其全部公开被通过引用结合到本文中。然而，将领会的是存在其它节点锁定技术，并且本发明的实施例可使用任何节点锁定技术。

[0086] 数字水印是众所周知的技术。特别地，数字水印涉及到修改初始数字对象以产生加水印数字对象。进行修改从而将特定数据(称为有效负荷数据)嵌入或隐藏到初始数字对象中。有效负荷数据可例如包括标识数字对象的所有权或其它权限信息的数据。有效负荷数据可标识加水印数字对象的(计划)接收者，在该情况下，将有效负荷数据称为数字指纹——此类数字水印可以用来帮助追踪数字对象的未授权拷贝的起源。可以对软件项目应用数字水印。在US7,395,433中可以找到此类软件水印技术的示例，其全部公开被通过引用结合到本文中。然而，应领会的是存在其它软件水印技术，并且本发明的实施例可使用任何软件水印技术。

[0087] 向不同的用户设备20提供软件项目12的不同版本可能是期望的。软件项目12的不同版本为不同的用户设备20提供相同的功能——然而，受保护软件项目12的不同版本被不同地编程或实现。这帮助限制成功地攻击受保护软件项目12的攻击者的影响。特别地，如果攻击者成功地攻击了其受保护软件项目12的版本，则该攻击(或通过该攻击发现或访问的数据，诸如密码密钥)可能不适合于与受保护软件项目12的不同版本一起使用。因此，存在用于对软件项目12进行变换、使得生成软件项目12的不同的受保护版本(即使得引入“分集”)的许多技术，在本文中称为“分集”技术。在WO2011/120123中可以找到此类分集技术的示例，其全部公开被通过引用结合到本文中。然而，应领会的是存在其它分集技术，并且本发明的实施例可使用任何分集技术。

[0088] 上面提及的白盒混淆技术、节点锁定技术、软件水印技术和分集技术是软件保护技术的示例。将领会的是存在对软件项目12应用保护的其它方法。因此，应将如本文所使用的术语“软件保护技术”理解成意指对软件项目12应用保护的任何方法(旨在挫败攻击者的攻击，或者至少使得攻击者更加难以成功进行其攻击)，诸如上面提及的白盒混淆技术中的任一个和/或上面提及的节点锁定技术中的任一个和/或上面提及的软件水印技术中的任一个和/或上面提及的分集技术中的任一个。

[0089] 存在保护器组件110可用来在软件项目260内实现上面提及的软件保护技术的许多方式。例如，为了保护软件项目12，保护器模块110可修改软件项目12内的一个或多个代码部分和/或可向软件项目220中添加或引入一个或多个新的代码部分。用来进行这些修改的实际方式或者用来编写新代码部分的实际方式当然可以改变——毕竟，存在编写软件以实现相同功能的许多方式。

[0090] 二进制保护组件130用于接受在由编译器和链接器140编译之后以本机或二进制代码或字节代码的形式软件项目12，并且应用二进制保护技术，诸如完整性验证、反调试、代码加密、安全加载以及安全存储。二进制保护组件然后通常将软件项目12重新包装成完全受保护的二进制，所述二进制具有在其在用户设备20上的加载和执行期间可以被访问和使用的必需安全数据。

[0091] 因此，对于其中开发者可以访问所有源代码的软件项目12而言，可以使用优化和保护工具集40来使用保护组件112首先在第二中间表示中对应用程序的源代码应用源代码保护工具，并且然后对已通过使用源代码保护技术来保护的二进制应用二进制保护。在源代码和二进制代码域两者中对软件项目12应用此类保护导致更加有效地受到保护的软件项目12。

[0092] 图3图示了可使用优化和保护工具集40来实现的工作流程200中的某些。软件项目

12被提供以输入表示Ri给工具集。此表示通常可能是如上面所讨论的源代码或二进制代码表示。在步骤205处，软件项目被转换成第一中间表示。这可能涉及到使用单个转换器组件120-128或者两个或更多转换器组件。通常，软件项目可能被从输入表示Ri直接地转换成第一中间表示或者从输入表示Ri经由诸如第二中间表示之类的另一表示转换成第一中间表示。

[0093] 然后在步骤210处使用图2的优化器组件100来优化在第一中间表示IR1中的软件项目12，并且然后在步骤S215处使用图2的第一转换器120转换成第二中间表示IR2。然后在步骤220处使用图2的保护器组件110来保护在第二中间表示IR2中的软件项目12，并且然后在步骤225处使用图2的第二转换器122来转换回到第一中间表示IR1。

[0094] 然后在步骤230处再次地使用图2的优化器组件100来优化第一中间表示IR1中的软件项目12。然后其可在被以输出表示Ro输出之前在步骤235中经受进一步处理的各种方面。进一步处理的各方面可包括编译和链接、二进制保护、转换到其它表示等中的一个或多个。

[0095] 图中的虚线流程箭头指示在第二优化步骤230之后，工作流程200可返回到用于转换回到第二中间表示的步骤215以及保护和优化的一个或多个其它步骤。

[0096] 可以以不同的方式改变图3的工作流程200。例如，可在保护步骤220之前或之后优化软件项目12仅一次，并且进一步处理的步骤235可省略或者包括多个步骤。保护或者优化可在另一者之前执行，并且可执行优化和保护的任何数目的其它步骤。从输入表示Ri到被用于优化的表示IR1的转换可包括多个转换步骤，例如从Ri至IR2的转换，后面是从IR2至IR1的转换。另一处理步骤235可包括其它优化和/或保护步骤，例如二进制重写保护步骤。

[0097] 现在将描述如何可实现图2的优化和保护工具集40及诸如图3的那些之类的工作流程的更特定示例。在这些特定示例中，第一中间表示通常是上面所讨论的LLVM IR。这使得本发明能够扩展本机应用程序保护的范围以为了更好的性能和安全性，并且还为优化和保护工具集40的大得多的操作范围开拓了新的安全可能性。

[0098] 对于发明人而言已变得显而易见的是在使软件项目12准备好分发到多个用户设备20时存在安全与性能之间的冲突问题。一般地，受保护软件引入将降低受保护且尤其是被伪装形式的软件的性能的所需冗余和开销。应用于软件项目的保护技术越多，对性能的影响越显著。因此，需要平衡性能和安全。

[0099] 典型的保护技术可将静态程序相关性变换成部分静态且部分动态的相关性。这完全防止了通常比动态攻击更容易执行的静态攻击。然而，其还引入了这些保护技术可能破坏依赖于静态相关性性质的分析的某些优化能力的限制。由于此限制，保护和优化策略需要在较少的安全/保护但例如在执行速度方面较好的优化和/或较小的程序大小与较多的安全/保护但较少的优化之间做出选择。

[0100] 图4图示了可以使用优化和保护工具集40来实现的工作流程。以输入表示Ri(其为C/C++源代码表示Rc)向优化和保护工具集40提供软件项目12。这被传递至工具集组件群组300，其由从表示Rc至第二中间表示IR2的转换器X3、保护器组件110以及从第二中间表示IR2返回至源代码表示Rc的转换器X4组成。如果第一中间表示中的LLVM优化将不会发生，则可以使软件项目12在被传递到编译器、优化器和链接器140之前连续地通过这些功能中的每一个以保护软件项目12，并且然后到二进制保护组件130上以以输出表示(其为本机/二

进制代码表示Rb)输出软件项目12。还提供了一组安全库和代理145供在编译/链接软件项目12时使用以及如果需要的话供二进制保护组件130使用。

[0101] 通过优化器组件100来补充工具集组件群组300，所述优化器组件100在这里出于阐明的目的被示为实现一个或多个LLVM优化工具的单个子组件102，但可以使用多个子组件102，例如在优化的每个阶段处使用不同的子组件、多个子组件或子组件的不同组合。然后使用图2的X1和X2转换器将软件项目12从使用X3转换器124形成和/或如由工具集组件群组300中的保护器组件110输出的第二中间表示转换成供LLVM优化工具使用的第一中间表示，并且在被LLVM优化工具优化之后对软件项目12进行转换以便由保护器组件110保护和/或由X4转换器转换回到Rc表示。

[0102] 在图4中使用虚线图示了某些替换工作流程路径。例如，在由保护器组件110进行的处理和到IR1表示的转换之后，可以将软件项目12直接地发送到编译器、优化器和链接器140而没有由优化器组件100进行的第二处理步骤。类似地，在由优化器组件100进行的第二处理步骤之后，可以将软件项目12直接地发送到编译器、优化器和链接器140而没有由X1和X4转换器进行的转换，如果编译器、优化器和链接器140能够处理以第一中间表示的输入的话。

[0103] X1和X2转换器因此提供在第二中间表示中由保护器组件提供的保护技术的域与在第一中间表示中由LLVM优化工具提供的优化技术的域之间的桥，从而将优化和保护工具集40的这两个操作区域整合。该方法还帮助解决上面所讨论的保护与优化之间的冲突，因为优化和保护工具集40可以利用可用LLVM优化工具和技术的能力，以在由保护器组件110应用保护技术之前和之后都提供优化。通过在多个级别处启用优化，可能去除安全与性能之间的限制，使得针对同一软件项目12可以实现更好的安全性和改善的性能两者。

[0104] 图5图示了可以使用优化和保护工具集40来实现的另一工作流程。在此图中，软件项目12被以作为源代码表示Rs的输入表示提供给优化和保护工具集40。源代码表示Rs可以是例如Objective-C、Java、JavaScript、C#、Ada、Fortran、ActionScript、GLSL、Haskell、Julia、Python、Ruby或Rust。软件项目12被传递至转换器X5，其将源代码表示Rs转换成第一中间表示。可将转换器X5提供为提供从多种源代码表示到LLVM IR的转换的一组LLVM前端工具320的部分。现在在LLVM IR中的软件项目12可以被传递至优化器组件100以用于由LLVM优化器工具进行的第一优化步骤，或者直接地到X1转换器(如虚线所示)以便在传递至保护器组件110之前转换成第二中间表示。图5的其余部分对应于图4。请注意，图5的工具集组件群组300未被示为包括X3转换器，因为其在图5的工作流程中是不必要的，但是尽管如此其可以被包括在此群组中，如果期望的话。

[0105] 由于一组非常丰富的可用LLVM前端工具320可以将许多不同的语言转换成LLVM IR，并且从而利用LLVM编译设施来获得尖端的分析和更好的性能，所以如图5中所示，这些LLVM前端工具可以用来扩展优化和保护工具集40的前端能力以经由其中可以应用保护器组件110的保护技术的第一中间表示将一大组编程语言中的程序源代码转换成第二中间表示。

[0106] 图6图示了可以使用优化和保护工具集40来实现的另一工作流程。在此图中，软件项目12被以作为本机/二进制表示Rb的输入表示Ri提供给优化和保护工具集40，以便在用户设备20的特定平台或类别上执行。二进制表示Rb可以是例如x86、x86-64、ARM、SPARC、

PowerPC、MIPS以及m68k二进制表示中的任何。软件项目12被传递至转换器X6，其将二进制表示Rb转换成第一中间表示。可将转换器X6提供为提供从多种二进制表示到LLVM IR的转换的一组LLVM二进制工具330的部分。图6的其余部分对应于图4和5。

[0107] 通过以该方式使用LLVM二进制工具，以本机/二进制代码中的软件项目12可以在被在第二中间表示中转换以便输入到保护器组件300以便应用诸如伪装之类的保护技术之前被转换成LLVM IR形式。如果输出表示Ro是用于与输入表示二进制代码不同的目标平台的二进制代码，则可以容易地使用优化和保护工具集40来用编译器、优化器和链接器140的适当配置与应用所需保护技术同时地针对不同的目标平台达到此输出目标。

[0108] LLVM编译器中间层工具包括尖端程序分析能力，诸如更精确的混叠分析、指针溢出分析以及相关性分析，其可以提供可以用来出于不同的目的对程序进行变换的丰富的程序性质和相关性。图2中所示的二进制重写保护组件135提供一个或多个二进制重写保护工具，其接受以LLVM IR形式的软件项目12，通过利用LLVM的程序分许功能来进行混淆变换，并且导致LLVM IR中的软件项目12的更安全版本。

[0109] 二进制重写保护组件135可以以许多不同的方式增强软件项目12的保护，包括独立二进制重写保护、利用二进制保护工具的二进制重写保护以及利用源伪装工具和二进制保护工具两者的二进制重写保护：

[0110] 独立二进制重写保护——一般地，二进制保护以二进制形式来保护二进制代码，并且某些此类保护技术需要对二进制表示进行工作，例如完整性验证、安全加载以及动态代码加密。并且，如果所需程序信息变得可用，则二进制保护可以应用某些种类的变换。

[0111] 然而，现有二进制保护工具往往具有对分析能力的有限支持，使得直接地以二进制形式可以完成非常有限的二进制变换。替换地，二进制重写保护工具可以适合于作用于在诸如LLVM IR之类的中间表示中的软件项目12，其中，可以利用更加尖端的程序分析支持，从而应用不能容易地直接应用于在二进制表示中的软件的许多变换技术。

[0112] 在独立模式下，使用一个或多个LLVM二进制工具330将不受保护的二进制代码表示中的软件项目12转换成LLVM IR，并且然后使用二进制重写保护组件135通过与LLVM程序分析工具相交互而对软件项目12应用某些程序变换。然后通过使用LLVM IR至二进制转换器、编译器、优化器和链接器或以其它方式将LLVM IR中的重写的软件项目12转换成受保护的二进制代码表示。

[0113] 可以通过使用二进制重写保护组件135将用二进制保护工具进行的二进制重写保护——在此模式下以二进制代码表示提供给优化和保护工具集40的软件项目12——混淆成受保护的二进制表示。然后可以通过使用诸如由图2的二进制保护组件130提供的一般二进制保护工具来保护软件项目12。

[0114] 以该方式通过使用二进制重写保护和二进制保护两者将不同的保护层组合在一起导致更加安全的软件项目12。

[0115] 用源级别保护和二进制保护两者——一般地保护诸如上面所讨论的第二中间表示之类的源代码类型表示的处理——进行的二进制重写保护可以提供更加广泛且深入的数据流和控制流保护。图7使用类似于图6的工作流程将其图示了，其中，使用LLVM二进制工具来将以二进制表示提供给优化和保护工具集40的软件项目12转换成第一中间表示。另外在图7中，在保护器组件112的作用之后从优化器组件100或者替换地直接地从转换器X2输

出的软件项目12被指引到二进制重写保护工具135。在二进制重写保护工具135的操作之后，软件项目12然后被传递到编译器、优化器和连接器140上，如前所述。二进制重写保护工具135是可以在此布置中使用的LLVM编译器中间层工具345的示例。如图7中的虚线所示，软件项目12在第一优化之后可替换地在没有保护器组件112的处理或第二优化阶段的情况下被径直指引到二进制重写保护工具，或者可以省略第一或第二优化步骤的方式被处理。

[0116] 网络应用程序是使用网络浏览器作为客户端环境的应用程序。网络应用程序通常被以浏览器支持的编程语言(诸如JavaScript)编码，与诸如HTML之类的浏览器呈现标记语言组合，并且依赖于其主机网络浏览器以使得其可执行，“asm.js”是例如在网站http://asmjs.org处所讨论的JavaScript的有限子集。“asm.js”支持类似于C的计算，但是由于其是JavaScript的子集，所以其在任何支持JavaScript的网络浏览器中正确地运行而不要求任何进一步的特殊支持。asm.js所使用的子集使得容易使用类型推理的平常方法来标识低级操作，“asm.js”依赖于支持WebGL(缓冲器和类型阵列，诸如UInt32、INt 16等)所需的扩展以便支持低级结构、阵列等，但这些通常在主机网络浏览器中可用。可以使用“use asm”指令在JavaScript文件中标记JavaScript程序符合“asm.js”表示。主机网络浏览器然后在不存在用于“asm.js”的明确支持的情况下可以忽视此指令，或者如果支持可用的话，可以针对与“asm.js”表示的符合性检查程序。如果支持在网络浏览器中可用，则asm.js代码可以以与普通JavaScript相比大大增加的速度和效率运行，通常是通过asm.js代码到本机二进制代码表示的编译。

[0117] 在现有技术中提供了用于将诸如C和C++之类的源代码表示转换成asm.js表示的工具。一个此类工具链将由将C和C++表示转换成LLVR IR的Clang工具(参见http://clang.llvm.org)以及将LLVM IR转换成asm.js表示的Emscripten工具(参见https://github.com/kripken/emscripten)组成。可以在Emscripten工具的应用之前应用LLVM优化工具作为此工具链的部分来实现优化。

[0118] 图8图示了如何可以使用优化和保护工具集40来优化和保护以C/C++源表示Rc提供的软件项目12，并以asm.js表示Ra而输出软件项目12。图8的工作流程遵循与图4至7的那些类似的方案。

[0119] 根据用粗虚线示出的第一工作流程路线，在C/C++表示Rc中输入的软件项目12被传递至工具集组件群组300，在那里，其被转换器X3转换成第二中间表示，然后被保护组件112保护，并且然后转换回到C/C++表示Rc。受保护软件项目12然后被传递至表示为X7的Clang组件350，其将C/++源代码表示Rc转换成第一中间表示IR1，通常为LLVM IR。此表示被传递至形成优化器组件102的部分的LLVM优化器310，并且然后至表示为X8的Emscriten组件360，其将第一中间表示转换成asm.js表示Ra以用于输出。

[0120] 根据一般地用实线示出的第二工作流程路线，在C/C++表示Rc中输入的软件项目12首先被传递至表示为X7的Clang组件350，其将C/++源代码表示Rc转换成第一中间表示IR1，通常为LLVM IR。此表示然后被传递至形成优化器组件102的部分的LLVM优化器310，并且然后至表示为X1的第一转换器122以便转换成第二中间表示一般传递至保护器组件112。在被保护器组件112处理之后，软件项目12被传递至表示为X2的第二转换器120以便转换回到第一中间表示且然后至优化器组件102以用于第二优化阶段。最后，软件项目12被传递至表示为X8的Emscripten组件360，其将第一中间表示转换成asm.js表示Ra以用于输出。此工

作流程内的某些替换是用细虚线示出的，由此可以省略第一或第二优化步骤。

[0121] 通过使用优化和保护工具集40来实现包括保护和优化的C/C++至asm.js转换，可以用C/C++来开发诸如网络应用程序之类的新的软件项目12以便用asm.js递送到用户设备，并且还将采取C/C++的现有软件项目12迁移至受保护且已优化的asm.js表示。由于asm.js使能浏览器可以比在使用一般JavaScript的情况下执行强得多的运行时优化，所以可以以高速度运行已优化且保护的asm.js软件项目12。事实上，由本发明进行的测试已显示用C/C++编写并使用如上面所讨论的优化和保护工具集40来处理以形成已优化且保护的asm.js代码的软件项目12可以比最初用本机代码编写的相应软件项目12更好地执行。这指示在优化和保护工具集40中使用的优化的优良性能。

[0122] 虽然图8示出了使用优化和保护工具集40来接受以C或C++输入的软件项目12，但是用如已讨论的优化和保护的后续步骤和到asm.js表示Ra的最终转换，可以通过使用不同的LLVM前端工具来代替图8中所示的Clang工具350而将诸如Object-C、Java、JavaScript、C#等其它源代码表示用于输入表示Ri。这打开了将采取除C/C++之外的语言的现有应用程序迁移到网络应用程序或者用这些语言来开发可以被使得可用于在浏览器环境中使用的新网络应用程序的许多新机会。

[0123] 类似地，可以通过用一个或多个LLVM二进制工具330(如与结合图7所讨论的)替换Clang工具350来改变图8中所示的工作流程以接受本地鸡/二进制表示Rb中的输入软件项目12。此类工作流程的显著优点是可以使采取本机代码表示的现有软件项目12迁移至网络应用程序以便在保持例如就执行速度而言的性能的同时以由保护组件112提供的增强安全性在浏览器环境(例如HTML5)中运行。

[0124] 图9再次地图示了图2中已示出的优化和保护工具集40，但是现在用某些其它特定细节和方面替换结合图3-8所讨论的工作流程。例如，图9中所示的优化和保护工具集40对使用LVM IR作为第一中间表示进行特定参考。采用诸如LLVM之类的技术框架可以帮助对用七天源代码表示、二进制代码表示等提供的软件项目12的保护应用面向C/C++源代码结构等或者最初针对C/C++源代码等编写的软件保护能力。

[0125] 图9因此显示用于输入到优化和保护工具集40的软件项目12可以采取C/C++源代码(表示Rc)、另一源代码(表示Rs)或本机/二进制代码(表示Rb)。如果输入软件项目12在C/C++源代码表示中，则可以使用X3转换器将其转换成被保护组件112使用的第二中间表示。可以使用LLVM前端/二进制工具320、330将输入软件项目12的所有不同表示转换成第一中间表示，其为LLVM IR。

[0126] 然后可以由统一工具集群组400的元件以各种方式处理输入软件项目12。这些组件包括在第二中间表示中对软件项目12进行操作的保护器组件110、在LLVM中间表示中对软件项目12进行操作的二进制重写保护组件135以及在LLVM中间表示中对软件项目12进行操作的优化组件102。统一工具集群组400还至少包括第一和第二X1、X2转换器122、120，其在LLVM中间表示与第二中间表示之间进行转换，使得统一工具集群组400的任何组件可以作用于软件项目12。

[0127] 在由统一工具集群组400的组件进行的处理之后，软件项目12可以被传递至各种组件以用于进一步处理以便在相关输出表示中形成软件项目12。如果在第二中间表示中从统一工具集群组400传递，则可以使用转换器X4 126将软件项目12转换回到C/C++源代码表

示Rc以便由C/C++编译器和链接器组件140-1进行编译和链接。如果在LLVM中间表示中从统一工具集群组400传递，则软件项目12可以被LLVM编译器和链接器140-2编译和链接。在两种情况下，来自优化和保护工具集40的输出然后是本机/二进制代码表示Rb中的软件项目12。替换地，可以将软件项目12在LLVM中间表示中从统一工具集群组400传递至由Emscripten工具360提供的转换器X8，使得来自优化和保护工具集40的输出然后是在asm.js表示Ra中的软件项目12。

[0128] 使用图9的优化和保护工具集40，可以使用相同的保护器组件110和可由该组件110实现的伪装及其它技术的工具集来保护诸如应用程序或软件模块或库之类的软件项目12，无论使用什么语言来实现该软件项目。如果用本机/二进制代码从优化和保护工具集40输出软件项目12，则这可以在本机执行环境(包括PNaCl)中运行，或者如果用JavaScript或asm.js输出，则其可以在网络浏览器环境中运行。这在图9的优化和保护工具40中是通过分两个不同的中间表示操作统一工具集群组400的组件而实现的，保护组件在第二中间表示中对软件项目12进行操作，并且至少优化组件100在LLVM中间表示中对软件项目12进行操作。

[0129] 图2-9中所示的布置大部分利用第一中间表示以便执行软件项目的优化并利用第二中间表示以便执行软件项目的优化。然而，参考图10，更一般地，本发明的实施例还可将第一表示用于执行软件项目的保护和/或将第二表示用于执行软件项目的优化。另外，虽然图2-9的布置利用两个中间表示，但本发明的实施例可利用三个或更多中间表示，每个中间表示被用于软件项目的优化和保护中的一者或两者。

[0130] 图10类似于图2，但示出了如何可以由优化和保护工具集40使用任意数目的中间表示IR1……IRN，每个中间表示被用于保护和优化中的一者或两者。例如，在图10的布置中，第一中间表示IR1被优化器组件100-1和保护器组件110-1两者使用，第二中间表示被优化器组件100-2使用但未被任何保护器组件使用，并且第三中间表示被保护器组件110-3使用但未被任何优化器组件使用。如针对图2一样，每个优化器组件可包括一个或多个优化器子组件(在图10中未示出)，并且每个保护器组件可包括一个或多个保护器子组件(在图10中也未示出)。这些子组件可可执行如上面所讨论的但在适当中间表示的范围内的优化和保护的任何功能。

[0131] 请注意，虽然图10示出了用于与每个不同中间表示一起使用的不同功能保护器和/或优化器组件，但还可以使保护器和/或优化器组件中的一个或多个在中间表示中的多个不同表示内工作。虽然图10中关于每个中间表示所示的组件是优化器和/或保护器组件，但可提供用于对软件项目执行其它任务和变换的组件，以便在中间表示中的一个或多个中使用。

[0132] 各种中间表示IR1……IRN可包括LLVM IR以及例如上面所讨论的各种其它表示。为了通常在使用工具集时的保护和/或优化的各种状态下将软件项目在各种中间表示IR1……IRN之间转换，提供了适当的转换器功能125。可将转换器功能125实现为例如单个库、类、工具或其它元件或者作为多个此类元件，每个此类元件执行所需转换类型中的一个或多个。并不是始终都需要提供各种中间表示之间的所有可能转换，并且类似地，可将某些转换提供为两个或更多其它转换的组合，例如通过诸如LLVM IR之类的更常使用的中间表示。

[0133] 在图10中示为优化和保护工具集40的部分的还有一个或多个二进制重写工具135、一个或多个二进制保护工具130以及一个或多个编译器和/或链接器工具140。根据工具集40的要求，这些中的每一个可使用中间表示IR1……IRN中的一个或多个或其它表示进行操作。

[0134] 在上面讨论并在图2、9和10中示出的优化和保护工具集40可以用来保护软件组件，诸如库、模块和代理以及应用程序，并且所有此类软件组件落在所述软件项目12的范围内。这在图11中图示了，其中，可以是安全库、模块、代理等的各种软件项目12被输入到优化和保护工具集40，其以受保护且已优化的形式输出这些软件项目12。根据要求可在本机/二进制代码表示Rb和/或asm.js表示Ra中输出任何此类软件项目12。将asm.js表示中的已优化且保护软件项目12中的一个或多个与本机/二进制代码表示中的已优化且保护软件项目12中的一个或多个且将这些中的每一个与底层系统层430和另一底层硬件层440相连的箭头420表示asm.js、本机和系统层中的每一个可以访问并使用分级结构中的每个底层别中的特征，诸如安全特征。

[0135] 一般地，诸如安全库、模块和代理之类的软件组件具有其自己的安全能力和特征，并且这些软件组件的稳健性和安全性在确保在内部使用该软件组件或者被用来对该软件组件进行参考或调用的应用程序的安全方面是关键的。

[0136] 本文所述的优化和保护工具集40及工作流程因此可以用来改善此类软件组件以及因此的在内部使用此类组件的应用程序的安全性。

[0137] 使用本发明的各方面，可以为用户设备20提供多个安全层，包括硬件级别安全特征、系统或操作系统级别安全特征、本机层安全特征和网络层安全特征。使用优化和保护工具集40保护的诸如库、模块和代理之类的软件组件可以提供对不应被使得可用于网络应用程序层的硬件和系统级别安全特征的访问。由于优化和保护工具集40可以用来用本机代码和JavaScript(包括asm.js)两者来创建受保护软件组件，所以其可以用来构造并支持调用从采取JavaScript/asm.js的受保护软件组件到采取本机代码的受保护软件组件的相关性。

[0138] 示例性保护技术

[0139] 下面描述的是用于对软件项目应用保护的一个示例性方法/技术(但如上面所讨论的，将领会的是许多不同的保护技术可用，并且可以与本发明的实施例一起使用)。该方法在本文中应称为“结构保护方法”。在本发明的某些实施例中，结构保护方法由上面提及的工具集40的保护器组件110(或其子组件112中的一个)实现/应用。

[0140] 然而，将领会的是本发明的某些实施例没有利用工具集40，并且因此本结构保护方法可由不同的软件保护系统实现/应用(由一个或多个数据处理装置的一个或多个处理器执行)。

[0141] 图12是示意性地图示了根据本发明的实施例的结构保护方法的流程图。

[0142] 本结构保护方法对源代码(即本结构保护方法进行修改以便应用保护的软件项目采取源代码格式)进行操作——当然，如上面所提及的那样，没有采取源代码格式的初始软件项目可被转换成源代码格式以便应用结构保护方法。本结构保护方法特别地被设想为对JavaScript代码进行操作，但将领会的是可将该保护方法实现成从而对用其它语言(诸如C/C++、源代码、Visual Basic源代码、Java源代码等)编写的软件进行操作。因此，一般地，

本结构保护方法涉及到接收源代码的输入项目，对该源代码输入项目应用保护技术(下面将描述)，并输出受保护的源代码项目。

[0143] 更特别地，本结构保护方法将保护源代码中的结构化数据项目作为目标，其中，结构化数据项目具有可独立修改的组件或字段。此类结构化数据项目的示例是对象或类或结构等(其可独立修改组件称为性质或元素)及阵列和列表(其可独立修改组件是阵列或列表的编索引元素)。在下文中，应将任何此类结构化数据项目简称为“结构”(但是当然不应将这理解为意指实施例仅限于保护诸如C/C++ struct之类的结构)，并且应将结构的可独立修改组件(或字段或元素或性质)简称为结构的“元素”。结构的元素可以是另一结构。

[0144] 如将变得显而易见的，通过修改如何表示结构(就其格式/布局而言)来保护结构。实际元素本身的表示(现在被存储在已修改结构内)也可被修改。也就是说，保留结构中的信息，但是修改其形式/布局和表示以使得其分析对于攻击者而言更有挑战性。

[0145] 请注意，可用多个不同的方式使用结构——例如可以使用作为具有两个元素的阵列的结构来(a)表示显示器上的点的x和y坐标或(b)表示用于针对变量/设置的范围的上界和下界。因此可能期望能够在正以不同的方式在软件项目中使用同一结构的情况下对同一结构应用不同的保护(或保护级别和/或类型)。

[0146] 类似地，结构在被用于同一目的时可根据结构位于该处且将被处理的软件项目或源代码内的位置而要求不同的保护(或保护级别和/或类型)。

[0147] 下面将参考以下示例性结构来描述结构保护方法(但是当然将领会的是本结构保护方法可应用于其它类型的结构，并且本发明的实施例不受此特定示例性结构的限制)。此示例性结构(下面用伪代码示出)表示用于关于公司的雇员的数据的记录：

[0148]
```
EmployeeRecord = {
  empName;          /* name of employee */
  empID;            /* employee identifier */
  hourlyRate;       /* hourly rate of pay */
  regHours;         /* regular work hours by week */
  ovtmHours;        /* overtime hours by week */
  managerID;        /* empID for this employee's manager */
  directReports;    /* managee empIDs (<= 40) */
  yearsAtCo;        /* no. years with company */
};
```

[0149] 保护系统的用户可标识要保护的源代码内的一个或多个结构。这可包括例如将EmployeeRecord(员工记录)标识为要保护的结构(因为EmployeeRecord的实例可能包含对于尝试的未授权访问或修改而言对攻击者有吸引力的数据)。这可涉及到例如用户检查源代码并确定/发现一个或多个此类结构，或者通知用户需要保护涉及或者表示或包含某些数据的任何结构等。

[0150] 已标识了要保护的源代码内的一个或多个结构，保护系统的用户生成保护描述信息。这可手动地执行或者可在已标识到要保护的一个或多个结构时完全或部分地以自动化方式执行。在下文中，应用两个文件或对象(称为KeyTemplates和DataTemplates(其可例如以JSON形式提供))来表示保护描述信息，但将领会的是可使用提供该信息的其它方式，使得本发明的实施例不限于使用此类KeyTemplates和DataTemplates对象/文件，并且本发明的实施例也不限于下面所讨论的KeyTemplates和DataTemplates对象/文件的特定格式。

[0151] 总而言之，DataTemplates指定要保护的(未受保护)结构的初始/实际结构/个会，潜在也还有什么类型和/或什么级别的保护将适用于那些结构的一个或多个元素，而KeyTemplates指定什么类型和/或什么级别的保护适用于保护在DataTemplates中定义的那些结构的结构/格式/布局。因此，保护描述信息指定或者包括标识/指示以下各项的数据：(a)要保护的(未受保护)结构的初始/实际结构/格式；(b)潜在地什么类型和/或什么级别的保护适用于那些结构的结构/格式/布局；以及(c)潜在地，还有什么类型和/或什么级别的保护适用于那些结构的一个或多个元素。

[0152] 在某些实施例中，保护描述信息可以可用于用户(其可能先前已被生成，或者可能由第三方提供等)，因此用户不需要经历标识结构并生成保护描述信息的上面提及的步骤——替换地，用户可仅仅向实现结构保护方法的保护系统/组件提供保护描述信息。

[0153] 因此，一般地，在步骤1200处，实现结构保护方法的系统/组件接收保护描述信息。

[0154] 首先转到KeyTemplates对象/文件(或规范)。结构的两个实例被类似地伪装/保护，如果其具有相同“密钥”的话。在这里，“密钥”可指定一系列或类型的保护或混淆(其可被串适当地表示为源代码语言中的标识符，诸如JavaScript标识符)。另外或替换地，该密钥还可指定保护级别。例如，密钥可被指定为串‘boundaryProtection5’，其指示保护系列的名称是‘boundaryProtection’且保护级别是级别5。换言之，密钥可标识或指定或指示(a)保护或混淆技术的类型或编码的种类(在上面提及的示例中，将类型称为‘boundaryProtection’)和/或(b)基于该特定保护或混淆技术或编码种类的保护级别或成都(在上面提及的示例中，保护级别是级别5)。例如，给定类型的编码的级别1可能为串中的字符提供线性有线串编码，而级别10可能将第三程度的多项式编码用于串中的字符，其操纵起来更慢，但使攻击者更加难以分析。技术人员将领会存在用于保护大量数据的许多不同类型的保护，并且可用各种程度的强度或复杂性来实现那些保护类型——因此，在本文中不应更详细描述此类保护类型和保护级别。将领会的是对于某些实施例而言保护类型可具有仅一个“级别”，在该情况下，密钥可仅指定保护类型而不是级别。类似地，某些实施例可仅使用一个类型的保护，对于其而言可以有多个级别可用，在该情况下，密钥可仅指定保护级别而不指定类型。在下文中，应将密钥表示为形式“<protection typexprotection level>”的串(诸如'boundaryProtection')，但是将领会的是表示密钥的其它方式是可能的。

[0155] KeyTemplates对象/文件具有一个或多个字段(或条目/性质)。在KeyTemplates对象/文件中，每个字段具有：

[0156] (a)值，其为如上面所阐述的密钥，或者

[0157] (b)具有两个分量(例如两个元素的阵列)的值，其中的一个标识KeyTemplates对象/文件中的另一字段/条目/性质，并且其中的另一个是如上面所阐述的密钥。

[0158] 作为示例，KeyTemplate对象/文件可以使以下形式

[0159]
```
KeyTemplates = {
    EmployeeRecord = 'HRPriv10';
    EmployeeRecordHigh = ['EmployeeRecord', 'HRPriv15'];
    EmployeeRecordLow = ['EmployeeRecord', 'Basic3'];
};
```

[0160] 因此，用此特定KeyTemplates对象/文件：

[0161] -存在称为EmployeeRecord的KeyTemplates对象/文件的字段，其指示EmployeeRecord结构的实例可在级别10被对其应用HRPriv类型的保护或混淆或编码；

[0162] -存在称为EmployeeRecordHigh的KeyTemplates对象/文件的字段，其指示EmployeeRecord结构的实例可被对其应用HRPriv类型的保护或混淆或编码，但在级别15；

[0163] -存在称为EmployeeRecordLow的KeyTemplates对象/文件的字段，其指示EmployeeRecord结构的实例可在级别3被对其应用Basic类型的保护或混淆或编码。

[0164] 接下来转到DataTemplates对象/文件(或规范)，如所述，DataTemplates对象/文件的目的是指定用于要保护的结构的实际/初始(未受保护)结构或格式/布局。要保护的这些结构对应于对于其而言在KeyTemplates规范中指定了密钥(即保护类型和/或保护级别)的结构。DataTemplates对象/文件还可针对要保护的结构的元素中的一个或多个指定用于该元素的密钥，以便指定或标识要应用于该元素的保护的类型和/或级别。

[0165] DataTemplates对象/文件具有一个或多个字段(或条目/性质)。由于DataTemplates对象/文件指定结构/布局/格式，所以在DataTemplates对象/文件中存在对应于如上面所阐述的类型(a)的每个KeyTemplates字段的字段，即作为密钥的值。例如，针对如上面所阐述的类型(a)的每个KeyTemplates字段，在DataTemplates对象/文件中可存在与KeyTemplates字段具有相同名称的相应字段。请注意，如果KeyTemplates文件/对象具有如上面所阐述的类型(b)的字段(即，具有两个分量的值，其中的一个分量标识KeyTemplates对象/文件中的另一字段/条目，其中的另一个是密钥)，则在DataTemplates对象/文件中不需要存在相应字段，因为已经针对KeyTemplates对象/文件中的另一字段指定了此类字段将已指定的结构。这允许给定结构在KeyTemplates对象/文件中具有多个条目，使得根据其中使用该结构的上下文，可以使不同的保护级别和方案与该结构相关联。这还提供了DataTemplates和KeyTemplates对象/文件的高效存储/参考和高效更新/保持。

[0166] 因此，继续上面提及的示例，其中：

[0167]
```
KeyTemplates = {
   EmployeeRecord = 'HRPriv10';
   EmployeeRecordHigh = ['EmployeeRecord', 'HRPriv15'];
   EmployeeRecordLow = ['EmployeeRecord', 'Basic3'];
};
```

[0168] DataTemplates对象/文件还可以具有称为EmployeeRecord的字段，其可如下：

[0169]
```
DataTemplates = {
  EmployeeRecord = {
    empName = `CHRPriv10'; /* name of employee */
    empID = `CHRPriv10'; /* employee identifier */
    hourlyRate = `NHRPriv10'; /* hourly rate of pay */
    regHours = ['NHRPriv5',0,52]; /* regular work hours
    by week */
    ovtmHours = [`NHRPriv5',0,52]; /* overtime hours by
    week */
    managerID = `CBasic3'; /* empID for this employee's
    manager */
    directReports = [`CBasic3',0,40];  /* managee empIDs
    (<= 40) */
    yearsAtCo = `KHRPriv10'; /* no. years with company
    */
  }
};
```

[0170] 特别地,DataTemplates对象/文件中的每个字段的值是模板,其中,用于要定义/指定的结构的模板本身具有用以指定结构(即其布局/格式)的元素(对于这些元素中的一个或多个而言,可能以及用于该元素的密钥,以便指定或标识要应用于该元素的保护的类型和/或级别)的一个或多个字段。在上面提及的示例中,定义了EmployeeRecord结构的布局/格式。另外:

[0171] ·在DataTemplates对象/文件中用从'c'开始的串值描述了EmployeeRecord结构的元素empName、empID和managerID。这指示EmployeeRecord结构的元素empName、empID和managerID是串。元素empName和empID还具有相应密钥的指示,即HRPriv10。元素managerID具有不同相应密钥的指示,即Basic3。

[0172] ·在DataTemplates对象/文件中用从'N'开始的串值描述了EmployeeRecord结构的元素hourlyRate。

[0173] 这指示EmployeeRecord结构的元素hourlyRate是数。元素houriyRate还具有相应密钥的指示,即HRPrivio。作为数(如在DataTemplates对象/文件中用从'N'开始的串值指定的)的结构的元素在应用保护时可被转换成串,该串然后可被以针对作为串(如在DataTemplates对象/文件中用从'c'开始的串值指定的)的结构的元素相同的方式编码的。

[0174] 在DataTemplates对象/文件中用从'K'开始的串值描述了EmployeeRecord结构的元素yearsAtCo。这指示EmployeeRecord结构的元素yearsAtCo是整数(例如配合在2的补码形式中的32位中的一个)。元素yearsAtCo还具有相应密钥的指示,即HRPrivio。作为整数(如用从'K'开始的串值在DataTemplates对象/文件中指定的)的结构的元素在被应用保护时可被使用由用于该元素的密钥指定的无损同态编码来编码。

[0175] 在DataTemplates对象/文件中用表示元素阵列的[template,lowerSizeLimit,upperSizeLimit]形式的串值描述了EmployeeRecord结构的元素regHours、ovtmHours和directReports,其中,用模板来描述阵列的每个元素,其中阵列的大小位于lowerSizeLimit和upperSizeLimit元素之间,其中,lowerSizeLimit和upperSizeLimit是整数。如果lowerSizeLimit是0,则这指示阵列可以是空的;如果upperSizeLimit是0,则这指示阵列可以是任意大的。因此,例如在DataTemplates对象/文件中用串值[xNHRPriv5',

0,52]来描述regHours和ovtmHours两者,其指示它们是达到52个元素的(可能空)阵列,并且那些阵列的每个元素是xNHRPriv5'类型的,如上面所提及的那样,其指示它们是数且每个数将被以级别5的保护级别用HRPriv的保护类型进行保护。类似地,在DataTemplates对象/文件中用串值[CBasic3',0,40]来描述directReports,其指示它是达到40个元素的(可能空)阵列,并且该阵列的每个元素是CBasic3'类型的,如上面所提及的那样,其指示它们是串且每个串将被以级别3的保护级别用Basic的保护类型进行保护。

[0176] 可使用在DataTemplates中指定要保护的结构的元素的类型的其它方式,并且将领会的是可以使用其它元素类型(当然取决于在考虑中的源代码语言)。

[0177] 如上面所提及的那样,本发明的实施例可利用其它方式来指定要保护的每个结构的初始/未受保护形式(或格式/布局/结构或特定元素)以及应用于要保护的结构和元素的保护的级别和/或类型。

[0178] 我们接下来转到如何修改结构的格式/布局或者更准确地如何根据结构保护方法在不同的格式/布局中表示未受保护结构。

[0179] 称为字典树的数据结构是众所周知的——参见例如http://en.wikipedia.org/wiki/Trie。字典树是被用于基于可以被自然地划分成各部分(诸如数或单词)的键(在这里,术语"键"不同于上面所述的"密钥")或索引来快速地访问一组记录的数据结构。字典树可表示节点树,其中,根节点不包含内容,并且每个节点的后代包含或表示"选择"(即根据键从父节点移动至该父节点的子节点的选择)。节点还可指示其是否是最终的(即叶节点),在该情况下其指示(标识或表示或存储)由该键/索引选择的相应记录,或者其不是最终的,在该情况下其具有表示其它选择的一个或多个子节点。

[0180] 图13示意性地图示了示例性字典树。在此字典树中,键/索引是单词(在该情况下为单词:a、it、in、map、mat、me)。用选择替换方案的选择(在该情况下为字母)来标记链接(箭头)。用由从根节点开始进行的选择所指示的键/索引的累积部分来标记节点(圆圈)。数是由字典树存储的记录/数据(或者可以是用于单独存储的数据的地址或者到保持数据的单独存储记录阵列中的索引)。因此,此字典树所表示的映射是a→7、it→4、in→8、map→11、mat→17、me→5。

[0181] 某些(可选)优化可用于表示字典树:

[0182] ·节点中的当前标记可以去除,因为其提供的所有信息已由箭头上的标记提供。

[0183] ·然后可以将箭头上的标记移动至(现在未被占用)节点。箭头实际上不需要与箭头指向的节点上的标记分开的标记。

[0184] 这些变化使得节点和箭头的表示更加紧凑,同时对可以使用字典树来执行的查找没有影响。

[0185] 可以将要保护的结构的格式/布局表示为字典树。特别地,字典树的根节点可以表示结构本身,而字典树的其它节点可以表示结构的各元素(即,非根节点可以是或者可以表示各性质、字段、元素、阵列索引等)。

[0186] 已将要保护的结构表示为字典树,然后结构保护方法可通过调整/修改该字典树来保护结构。特别地,可重新标记字典树的节点(使得由攻击者进行的字典树的检查不会显示出节点表示什么的任何语义)。

[0187] 此外,可将字典树修改/调整成包括其它节点。例如,字典树内的节点的一个或多

个路径可具有包括的一个或多个其它节点(和潜在地来自那里的分支),使得可增加从根节点至表示结构元素的叶节点的链路的数目。另外或替换地,可调整节点的性质(例如,作为表示结构的性质节点的替代,可将节点改变成表示阵列索引)。例如,结构可具有称为"price"的元素,使得字典树然后具有用于选择元素price的一个或多个节点的路径——结构保护方法然后可涉及到调整字典树,使得为了选择元素price,节点路径涉及到根节点,后面是选择称为Q790A的表示性质的节点(其可对应于阵列),然后从具有索引7的阵列中选择表示元素的节点(其可对应于结构),然后选择称为fT9_x40k的表示该结构的性质的节点。

[0188] 因此,针对要保护的结构,可生成受保护/已混淆字典树(应领会不需要实际上生成表示未受保护结构的字典树且然后修改该字典树——即可直接地基于保护描述信息来生成"受保护"字典树)。特别地,给定根节点,然后针对要保护的结构的每个元素,结构保护方法可随机地选择步骤或节点以插入字典树中(其可形成字典树内的一个路径或者其可包括字典树内的一个或多个分支/路径)。要插入的节点的最小数目(随机地选择)是一,因为原始未受保护结构中的一个访问步骤不能小于已混淆字典树中的一个访问步骤。节点的数目的随机选择可至少部分地基于在KeyTemplates对象/文件中定义/指定的用于结构的密钥中所指示的保护级别——使得保护级别越高,越有可能随机选择的数目将较高,从而包括更多的步骤/节点,并且因此使得攻击者更加难以分析该混淆/保护——即可基于保护级别对随机数进行偏置。类似地,可随机地选择性质名称和阵列索引。在某些实施例中,不同阵列索引的数目由于效率原因而不应大大超过要进行的选择的数目(例如如果存在N个不同的元素要选择,则针对已混淆字典树指定不超过pN个不同的索引,其中,p是预定值,例如,p=2)。

[0189] 在某些实施例中,针对要保护的结构的每个元素,在字典树内独立于结构的其它元素而选择用于给定节点的访问路径(即,独立于存在于字典树中的节点/步骤而向字典树添加节点/步骤)。然而,这可导致与未受保护结构相比很大的字典树大小,这可能是不期望的。因此,在某些实施例中,字典树中的中间步骤/节点的随机选择有利于针对在已标记(未受保护)字典树中具有相同父节点的相邻元素所选择的步骤/节点。例如,如果(使用上面提及的示例)对于具有元素"price"的结构x而言我们将性质访问x.price扩展成y.Q790A[7].ft9_x40k,其中,x表示原始结构且y表示已混淆结构,则针对结构x中的另一元素"product",结构保护方法将趋向于选择用于x.product节点的部分共享混淆,诸如y.Q790A[7].gw1CWHn,或者具有较少的共享,可能是y.Q790A[3][8]。可随机地选择节点的"共享"的量和关于是否要共享的判定,再次地,随机判定基于/偏向于保护级别(例如,用针对结构的较高保护级别,可存在较少的共享,并且用针对结构的较低保护级别,可存在更多的共享)。

[0190] 在字典中包括节点/步骤的方式可改变——例如,某些方法可不包括用于阵列的索引(诸如上面所述用的[3]、[7]和[8]),某些方法可优选对字典树使用更平坦但更宽的形式,而其它的可优选对字典树使用更深入的形式等。这些选项可对应于如在KeyTemplates对象/文件中定义/指定的用于结构的密钥中所指示的保护类型。

[0191] 为了使得能够生成并在源代码内使用已混淆结构,可提供各种函数(或程序或例程)并使得其可用。结构保护方法因此可在步骤1202处包括基于接收到的保护描述信息将

这些函数中的一个或多个包括在源代码内，即修改源代码从而使用这些函数中的一个或多个。下面阐述这些函数的示例，但将领会的是可以提供其它函数，并不是所有这些函数都需要提供，并且可以替换地使用这些函数的其它公式化/表示。然后，在步骤1204处，可将函数转换成已混淆代码(基于保护描述信息)，如下面应描述的。示例性函数包括：

[0192] ·用以从未受保护结构创建已混淆/受保护结构的函数(在本文中称为templateEncode)，例如：

[0193] `obfStruct = templateEncode(struct,templateName,tag)`

[0194] 在这里，struct标识用于要保护的未受保护结构的实例的程序变量(在源代码中)；

[0195] obfStruct标识由函数templateEncode生成的用于受保护结构的实例的程序变量(在源代码中)；

[0196] templateName标识或指示KeyTemplates对象/文件中的条目(以从而标识/指定要应用于struct实例以获得obfStruct实例的编码/保护的类型和/或编码/保护的级别)；以及

[0197] tag是用来播种用于生成已混淆结构且用于对已混淆结构的单独元素执行保护的随机选择的值。

[0198] 因此，例如，如果x是用于EmployeeRecord结构的未受保护实例的源代码中的变量，则可以生成相应的受保护/已混淆版本，并且通过在软件项目中包括以下代码行而在源代码中用变量y来表示：

[0199] `y = templateEncode(x,'EmployeeRecord',t1);`

[0200] 如果期望较高的保护级别，则可以替换地包括以下代码行：

[0201] `y = templateEncode(x,'EmployeeRecordHigh',t1);`

[0202] templateEncode函数将使用上面提及的字典树方法生成已混淆/已编码结构实例。

[0203] templateEncode函数在步骤1204处将不需要转换。

[0204] ·用以从已混淆/受保护结构创建未受保护结构的函数(在本文中称为templateDecode)，例如：

[0205] `struct = templateDecode(obfStruct,templateName,tag)`

[0206] 在这里，obfstruct标识用于将不受保护/解码的受保护结构的实例的程序变量(在源代码中)；

[0207] struct标识由函数templateDecode生成的用于未受保护结构的实例的程序变量(在源代码中)；

[0208] templateName标识或指示KeyTemplates对象/文件中的条目(以从而标识/指定当最初生成受保护obfstruct结构时应用的编码/保护的类型和/或编码/保护的级别)；以及

[0209] tag是用来播种随机选择以生成已混淆结构且被用于对已混淆结构的单独元素执行保护的值。

[0210] 因此，例如，如果y是上面经由函数调用针对EmployeeRecord结构的受保护/已混淆实例生成的源代码中的变量

[0211] `y = templateEncode(x,'EmployeeRecord',t1);`

[0212] 则可以通过在软件项目中包括以下代码行来(重新)生成相应未受保护结构x：

[0213] `x = templateDecode(y,'EmployeeRecord',t1);`

[0214] templateDecode函数在步骤1204处将不需要转换。

[0215] ·用以访问受保护结构的元素的函数(在本文中称为templateAccess)，例如：

[0216]
```
z = templateAccess(obfStruct,templateName,tag,'path'
[,optIndexArry])
```

[0217] 在这里，obfstruct标识用于要访问的受保护结构的实例的程序变量(在源代码中)；

[0218] templateName标识或指示KeyTemplates对象/文件中的条目(以从而标识/指定当最初生成受保护obfstruct结构时应用的编码/保护的类型和/或编码/保护的级别)；以及

[0219] tag是用来播种随机选择以生成已混淆结构且被用于对已混淆结构的单独元素执行保护的值。提供对由纯串路径指定的元素的访问，将该路径中的步骤转换成由根据针对给定tag选择的特定随机选择而针对KeyTemplates中的templateName条目指定的编码类型和保护级别所选择的更复杂步骤。

[0220] 例如，如果期望对受保护EmployeeRecord结构y的yearsAtCo元素的访问，则可以在软件项目中包括以下代码行：

[0221]
```
z =
templateAccess(y,'EmployeeRecord',t1,`.yearsAtCo')
```

[0222] 返回的元素z可采取编码形式(如果已对受保护结构中的元素的值应用保护的话)；

[0223] 替换地，函数templateAccess可解除应用于该元素的保护并从而返回被访问的元素的未受保护值。

[0224] 简短地给出optindexArray的描述。

[0225] templateAccess函数在步骤1204处境需要转换。例如，原始源代码可能最初已具有以下代码：`z = x.price`。实例x可能已如上所述经由对函数templateEncode的调用而被转换成受保护结构实例y，使得用于元素price的数据作为`y.Q790A[7].ft9_x40k`而可访问。在步骤1202处，将用对函数templateAccess的调用来替换代码`z = x.price`，例如`z = templateAccess(y,templateName,tag,'.price')`。然后将在步骤1204处基于保护描述信息用代码将其替换，其中，如上面所提及的那样，字典树中的节点/步骤(Q790A,[7]和ft9_x40k)被基于由对应于templateName的密钥指定的保护的类型和/或级别且基于由tag执行的种子通过对templateEncode函数的调用而随机地插入并命名。

[0226] ·用以更新/设定受保护结构的元素的值的函数(在本文中称为templateupdate)，例如：

[0227]
```
templateUpdate(obfStruct,templateName,tag,'path'[,
optIndexArray],exp)
```

[0228] 在这里，obfstruct标识用于要访问/更新的受保护结构的实例的程序变量(在源

代码中)；

[0229] templateName标识或指示KeyTemplates对象/文件中的条目(以从而标识/指定当最初生成受保护obfstruct结构时应用的编码/保护的类型和/或编码/保护的级别)；以及

[0230] tag是用来播种随机选择以生成已混淆结构且被用于对已混淆结构的单独元素执行保护的值。提供对由纯串路径指定的元素的访问，将该路径中的步骤转换成由根据针对给定tag选择的特定随机选择而针对KeyTemplates中的templateName条目指定的编码类型和保护级别所选择的更复杂步骤——该元素将通过该元素分配exp而被更新。

[0231] 例如，如果期望受保护EmployeeRecord结构y的yearsAtCo元素的更新(例如，将其值设置成13)，则可以在软件项目中包括以下代码行：

[0232] templateUpdate(y,'EmployeeRecord',t1,`.yearsAtCo',13))

[0233] 当已混淆元素被覆写时，必须用具有相同混淆的元素将其覆写，这意味着值exp的相应操纵是在将已混淆值exp存储在已混淆字典树中之前基于由DataTemplates指定的相应密钥而执行的。

[0234] 下面给出了optindexArray的描述。

[0235] templateupdate函数在步骤1204处境需要转换。例如，原始源代码可能最初已具有以下代码：x.price = z。实例x可能已如上所述经由对函数templateEncode的调用而被转换成受保护结构实例y，使得用于元素price的数据作为y.Q790A[7].ft9_x40k而可访问。在步骤1202处，将用对函数templateUpdate的调用来替换代码x.price = z，例如templateUpdate(y,templateName,tag,'.price',z)。

[0236] 然后将在步骤1204基于保护描述信息用代码y.Q790A[7].ft9_x40k = z来将其替换(在这里，为了简单起见，被存储的值不是受保护值，使得实际值z被存储在受保护结构y中)。

[0237] 可选optindexArray的目的是提供在运行时计算的索引。例如，诸如‘x [3] [15]’之类的路径不需要optindexArray，因为在该路径中提供了文字常数索引3和15。然而，如果需要在运行时计算索引，则这不再适用。如果源代码具有诸如‘x [i] [j]’之类的路径，其中，i和j的值当在源代码中编写路径时四未知的，则需要用以传递索引值的机制。此类符号索引被上面提及的操纵例程忽视，但是函数然后预期阵列自变量将遵循‘path’。例如，阵列自变量optindexArray可能包含[7*a+k,-2*b]，其将使得i和j的值分别地评估何时将进行函数调用，无论7*a+k和-2*b是什么。在本示例中，从左至右在路径中用阵列的元素代替符号索引。

[0238] 因此，作为示例，假设x是EmployeeRecord结构的未受保护实例。可以通过在源代码中包括以下代码行来对其进行保护，以形成受保护实例y：

[0239] y = templateEncode(x,'EmployeeRecordHigh',1352);

[0240] 在这里，正在使用高保护级别(由KeyTemplates中的EmployeeRecordHigh字段指定)，具有1352的标签/种子值。

[0241] 可用图14中所示的字典树来表示结果(即受保护结构y)。

[0242] 可以在步骤1202处通过在源代码中包括以下代码行来访问元素yearsAtCo：

[0243]
```
N =
templateAccess(y,'EmployeeRecordHigh',1352,`.yearsWithCo');
```

[0244] 步骤1204将使其转换成相当不同的某个东西，诸如：

[0245]
```
N = ((y.VqH0s.mHBa[1]*vs0198 & 0xFFFFFFFF)+ vs9410) &
0xFFFFFFFF;
```

[0246] 在这里，y.VqH0s.mHBa[1]访问用于元素yearsAtCo的受保护值(如图14中所示)。通过使用常数vs0i98和vs94io(其在本示例中是使用在整数模232的有限环内的线性映射针对值的编码而生成的常数)来解除已经应用于元素yearsAtCo的实际值的保护。

[0247] 可以在步骤1202处通过在源代码中包括以下代码行来将元素yearsAtCo更新成值w：

[0248]
```
N = templateUpdate(y,'EmployeeRecordHigh',1352,`.yearsWithCo',w);
```

[0249] 步骤1204将使其转换成相当不同的某个东西，诸如：

[0250]
```
y.VqH0s.mHBa[1] = ((w*vs4352 & 0xFFFFFFFF)+ vs3427) &
0xFFFFFFFF;
```

[0251] 在这里，y.VqH0s.mHBa[1]访问用于元素yearsAtCo的受保护值(如图14中所示)。通过使用常数vs4352和vs93427(其在本示例中是使用在整数模232的有限环内的线性映射针对值的编码而生成的常数)来对值w应用已经应用于元素yearsAtCo的实际值的保护。如在本技术领域中已知的的，常数vs4352和vs93427与常数vs0i98和vs94io相关(使得常数vs4352和vs93427被用于保护值且常数vs0i98和vs94io被用于使该受保护值不受保护)。

[0252] 虽然已描述了特定实施例，但技术人员将意识到仍在本发明的精神和范围内的对这些实施例的修改和变更。

[0253] 将领会的是所述的方法已被示为按照特定顺序执行的单独步骤。然而，技术人员将领会这些步骤可被组合或者按照不同的顺序执行，同时仍实现期望的结果。

[0254] 将领会的是可使用多种不同的信息处理系统来实现本发明的实施例。特别地，虽然附图及其讨论提供了示例性计算系统和方法，但其仅仅是为了在讨论本发明的各种方面时提供有用参考而提出的。可在任何适当的数据处理设备上执行本发明的实施例，诸如个人计算机、膝上型计算机、个人数字助理、移动电话、机顶盒、电视、服务器计算机等。当然，已出于讨论的目的简化了系统和方法的描述，并且其仅仅是可被用于本发明的实施例的许多不同类型的系统和方法中的一个。将领会的是逻辑块之间的边界仅仅是说明性的，并且替换实施例可将逻辑块或元素合并，或者可对各种逻辑块或元素施加功能的替换分解。

[0255] 将领会的是可将上面提及的功能实现为作为硬件和/或软件的一个或多个相应模块。例如，可将上面提及的功能实现为一个或多个软件组件以便由系统的处理器执行。替换地，可将上面提及的功能实现为硬件，诸如在一个或多个现场可编程门阵列(FPGA)和/或一个或多个专用集成电路(ASIC)和/或一个或多个数字信号处理器(DSP)和/或其它硬件布置上。在包含在其中的流程中或者如上所述地实现的方法步骤每个可由相应的各模块实现；在包含在其中的流程图中或者如上所述地实现的多个方法步骤可一起由单个模块实现。

[0256] 将领会的是在由计算机程序实现本发明的实施例的情况下，则存储或承载计算机

程序的一个或多个存储介质和/或一个或多个传输介质构成本发明的方面。计算机程序可具有一个或多个程序指令或程序代码，其在被一个或多个处理器(或一个或多个计算机)执行时执行本发明的实施例。如本文所使用的术语“程序”可以是针对在计算机系统上执行而设计的指令序列，并且可包括子例程、函数、程序、模块、对象方法、对象实现、可执行应用程序、小应用程序、小服务程序、源代码、目标代码、字节代码、共享库、动态链接库和/或针对在计算机系统上执行而设计的其它指令序列。存储介质可以是磁盘(诸如硬驱或软盘)、光盘(诸如CD-ROM、DVD-ROM或蓝光盘)或存储器(诸如ROM、RAM、EEPROM、EPROM、闪存或便携式/可移动存储器设备)等。传输介质可以是通信信号、数据广播、两个或更多计算机之间的通信链路等。

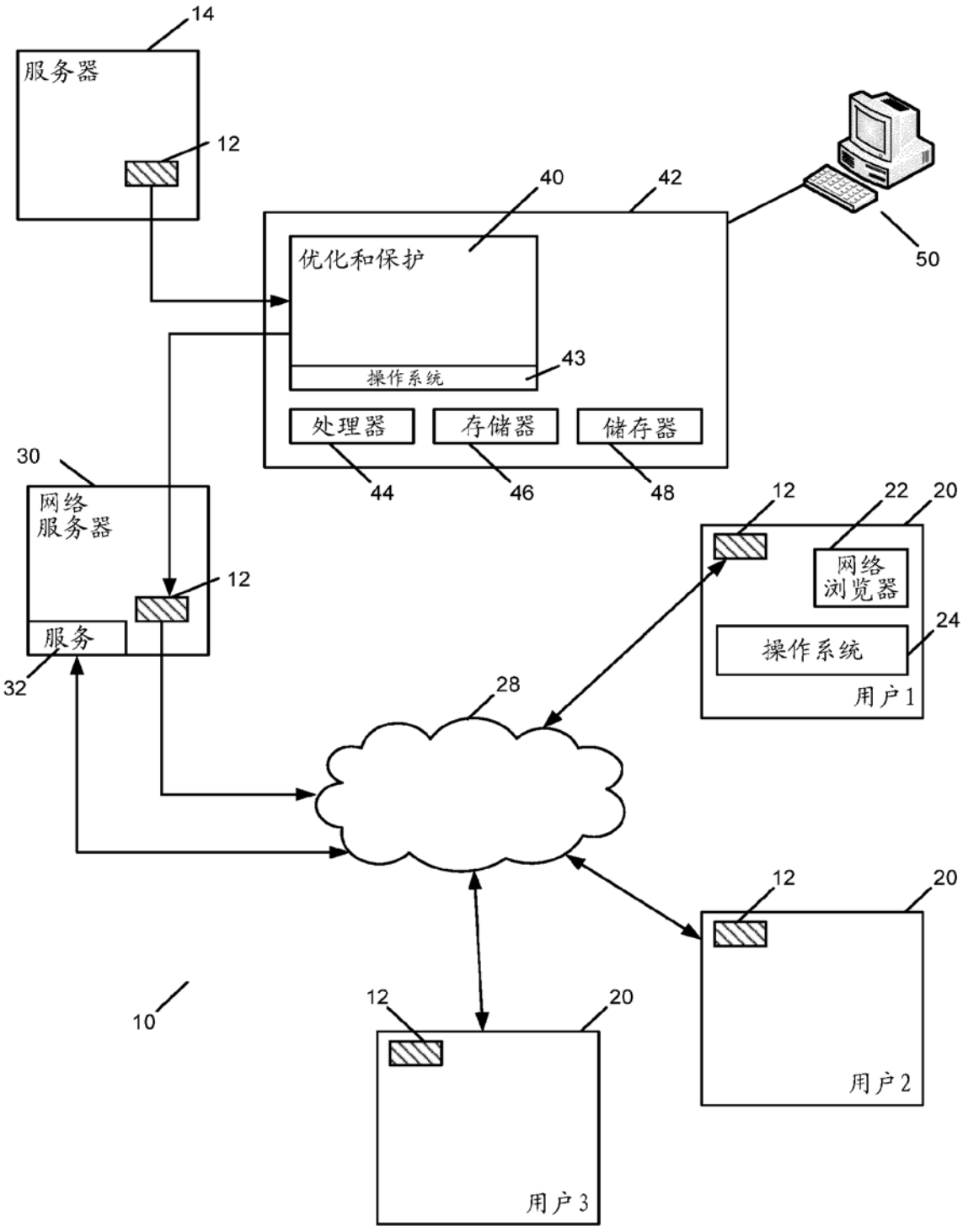

14
服务器
12
40
42
优化和保护
43
操作系统
处理器
存储器
储存器
44
46
48
50
30
网络
服务器
12
服务
32
28
12
22
20
网络
浏览器
操作系统
24
用户1
12
20
用户2
12
20
用户3
10

图 1

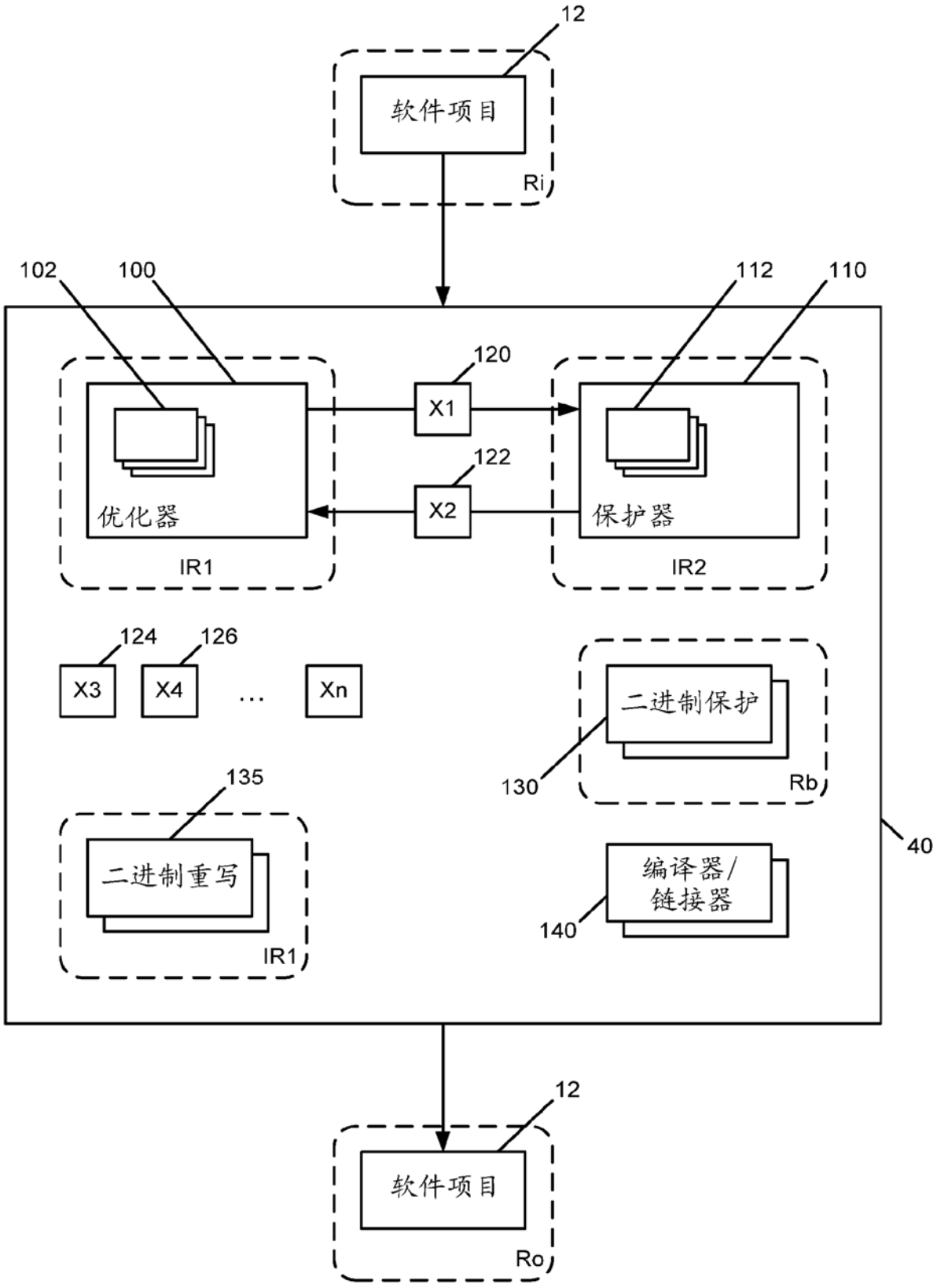

12
软件项目
Ri
102
100
112
110
120
X1
122
X2
优化器
IR1
保护器
IR2
124
126
X3
X4
...
Xn
二进制保护
130
Rb
135
二进制重写
IR1
编译器/
链接器
140
40
12
软件项目
Ro

图 2

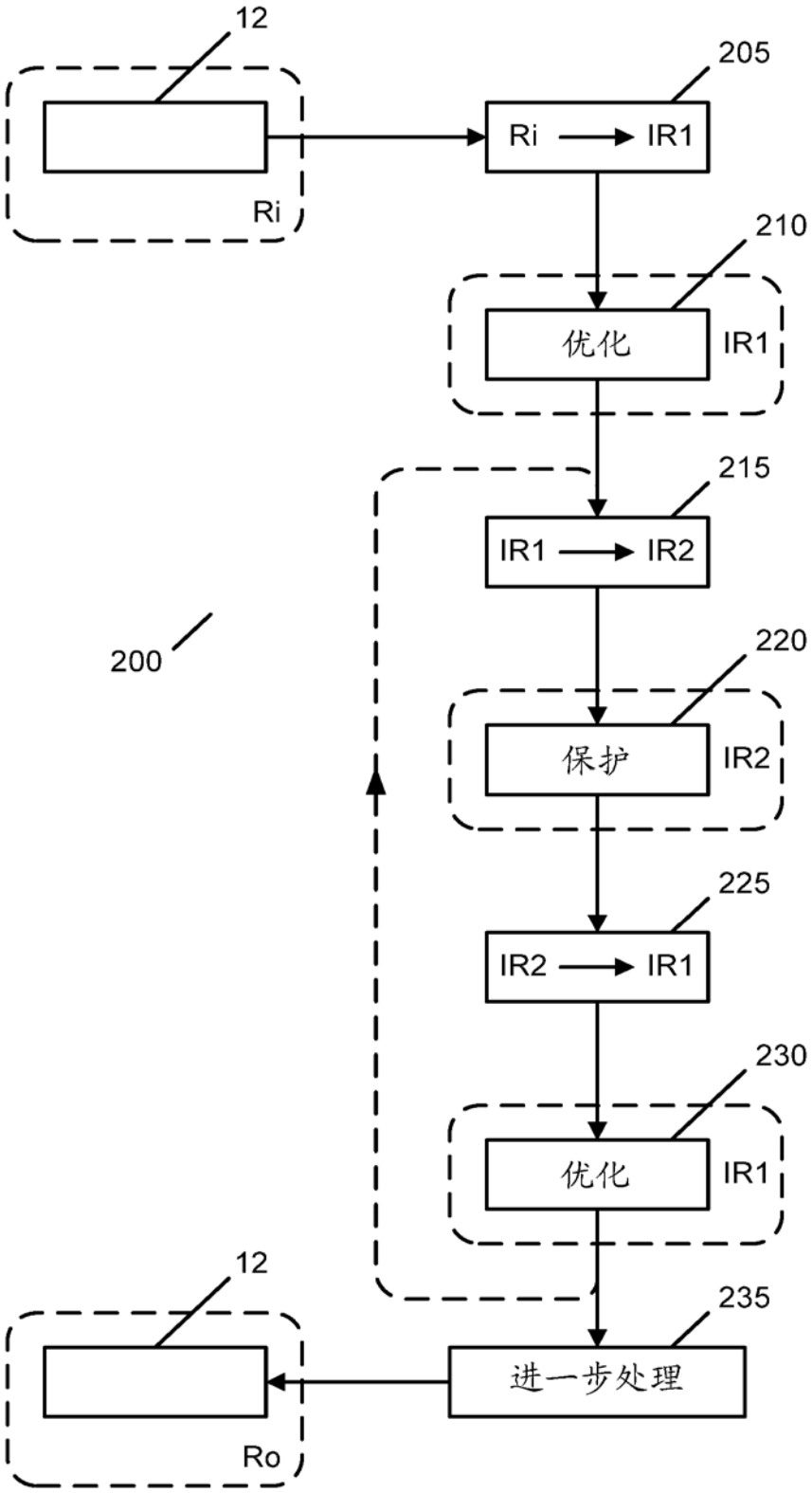

12
Ri
205
Ri
IR1
210
优化
IR1
215
IR1
IR2
200
220
保护
IR2
225
IR2
IR1
230
优化
IR1
12
235
进一步处理
Ro

图 3

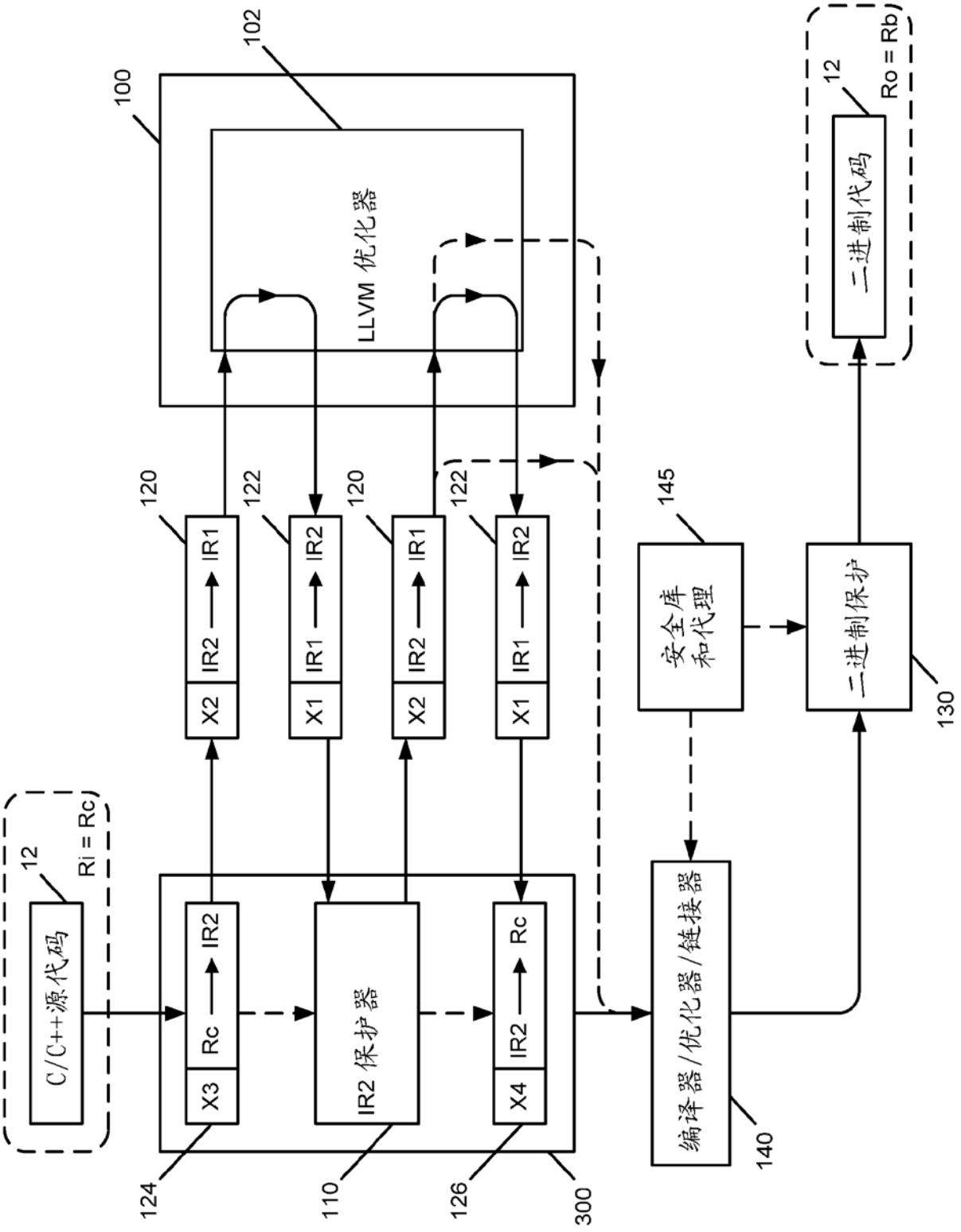

100
102
LLVM 优化器
12
Ri = Rc
C/C++源代码
120
122
IR1
IR2
X1
X2
X3
X4
Rc
IR2 保护器
124
110
126
300
145
安全库和代理
编译器/优化器/链接器
140
二进制保护
130
12
Ro = Rb
二进制代码

图 4

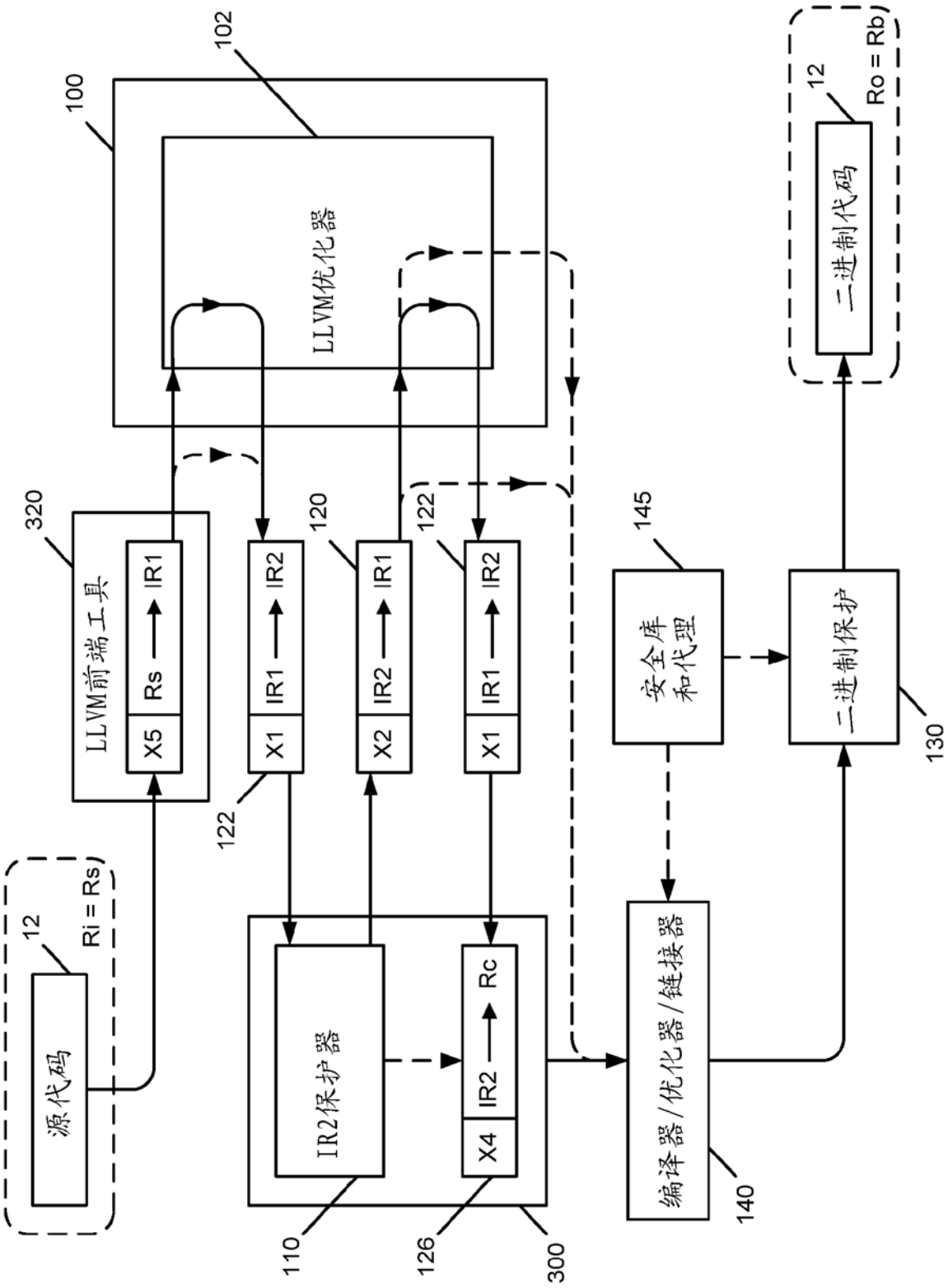

102
100
LLVM优化器
12
Ro = Rb
二进制代码
320
LLVM前端工具
X5
Rs
IR1
120
122
IR1
IR2
X1
X2
145
安全库
和代理
二进制保护
130
12
Ri = Rs
源代码
122
IR2保护器
X4
IR2
Rc
编译器/优化器/链接器
140
110
126
300

图 5

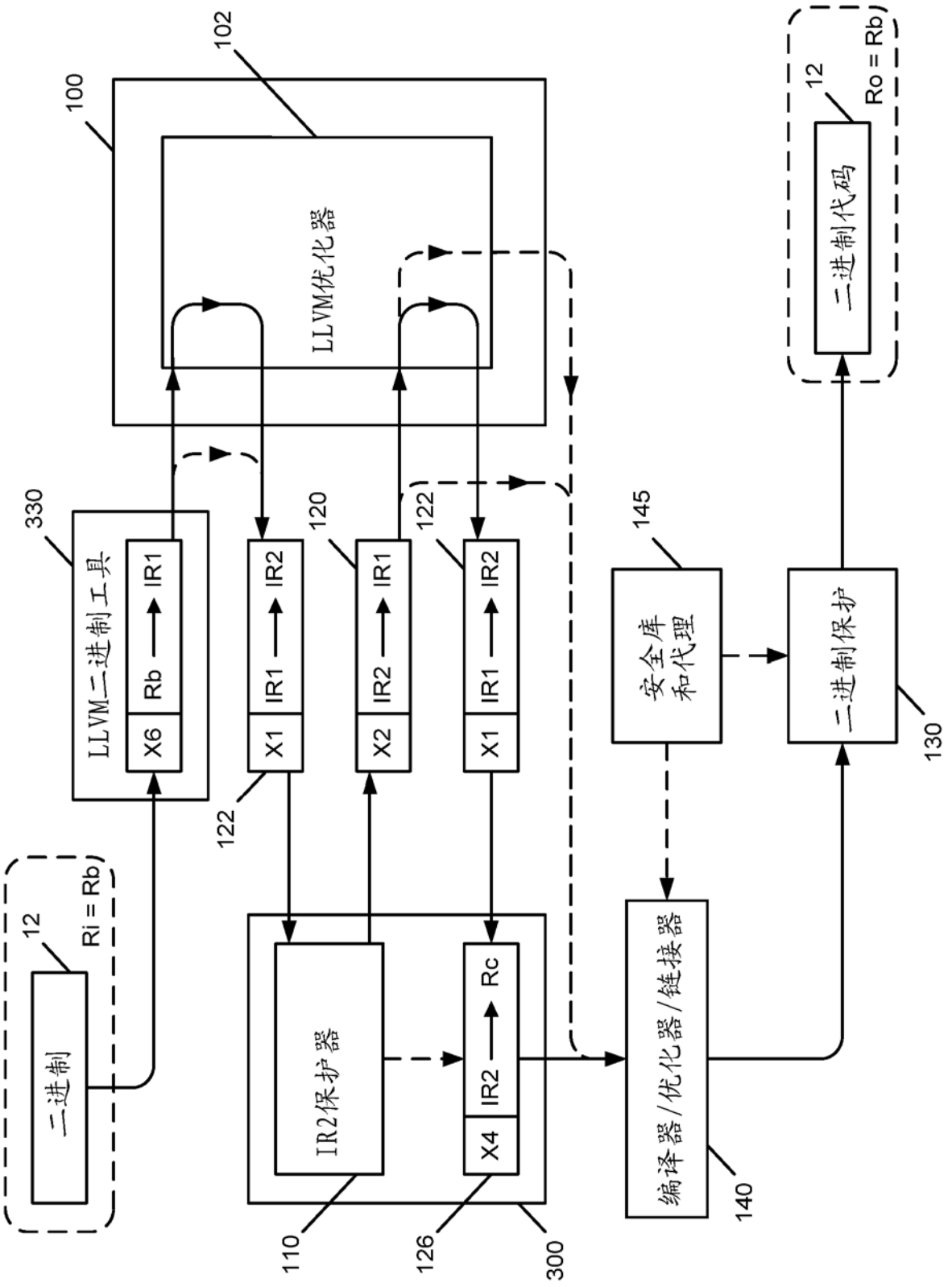

102
100
LLVM优化器
330
LLVM二进制工具
X6
Rb
IR1
120
X1
IR2
122
X2
IR1
122
X1
IR2
145
安全库和代理
二进制保护
130
12
Ro = Rb
二进制代码
12
Ri = Rb
二进制
IR2保护器
X4
IR2
Rc
编译器/优化器/链接器
140
110
126
300

图 6

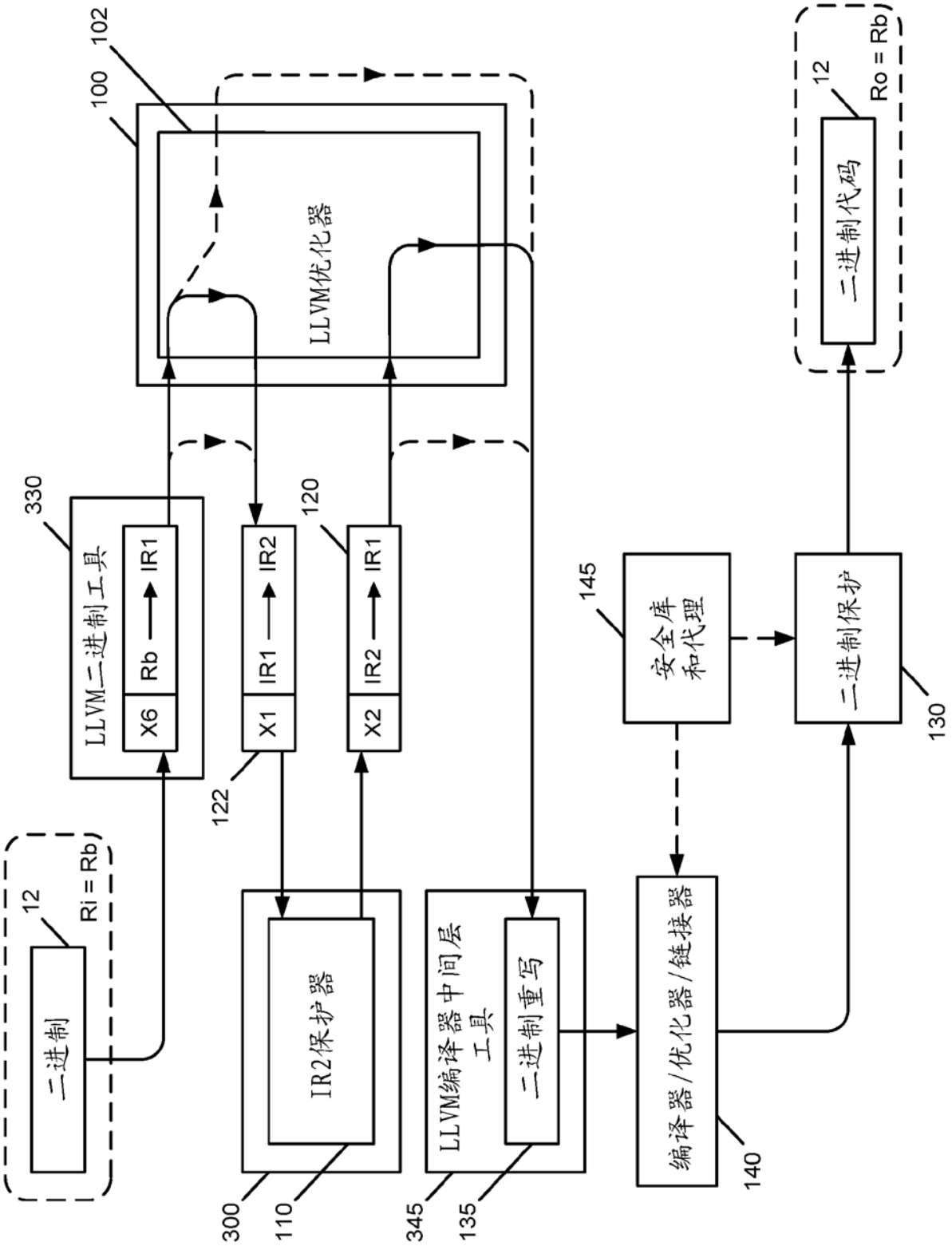

102
100
LLVM优化器
330
LLVM二进制工具
X6
Rb
IR1
120
X1
IR1
IR2
X2
IR2
IR1
122
12
Ri = Rb
二进制
IR2保护器
300
110
LLVM编译器中间层
工具
二进制重写
345
135
145
安全库
和代理
编译器/优化器/链接器
140
二进制保护
130
12
Ro = Rb
二进制代码

图 7

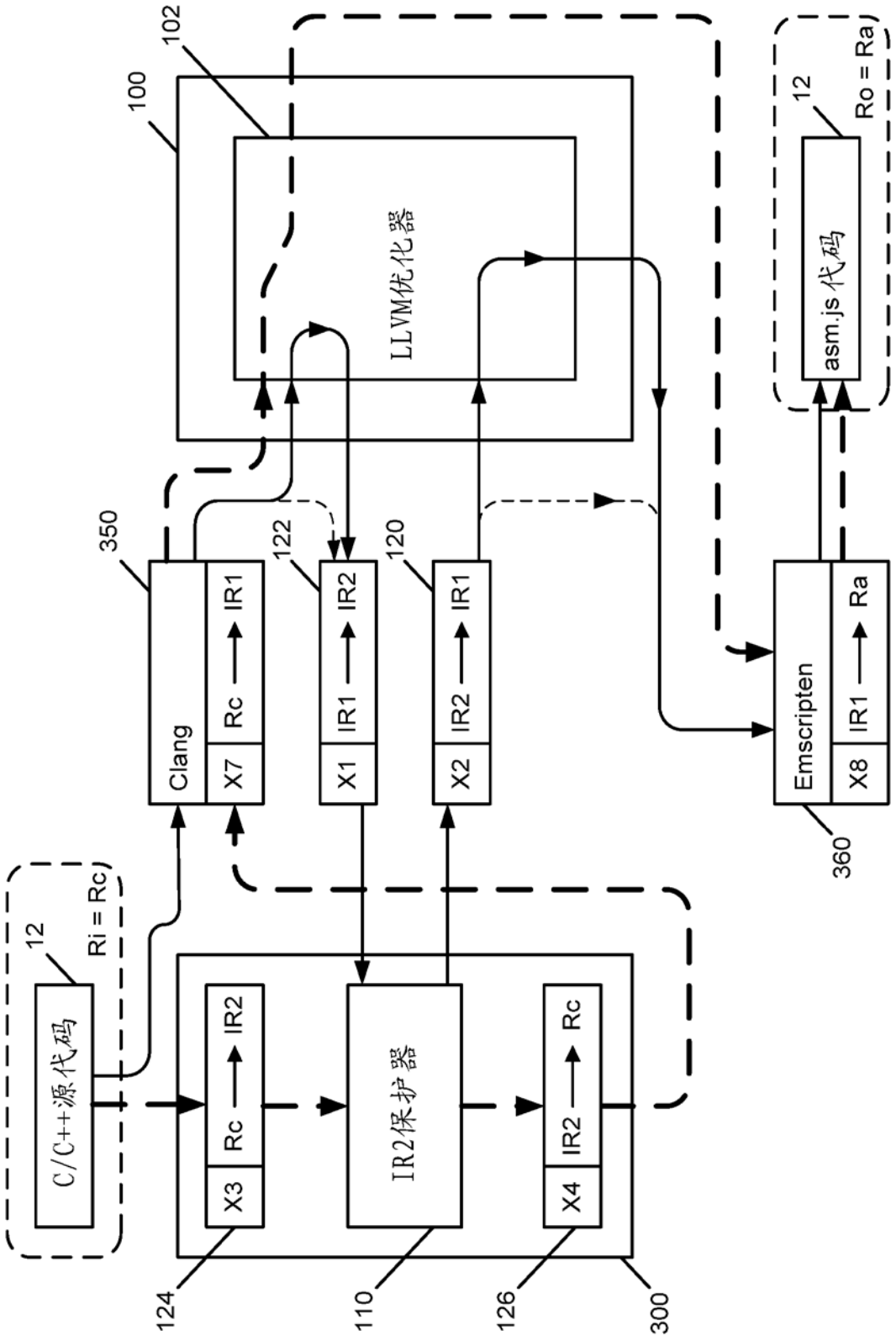

102
100
LLVM优化器
12
Ro = Ra
asm.js 代码
350
122
120
Clang
X7
IR1
Rc
IR2
X1
IR1
IR2
X2
IR1
Emscripten
X8
IR1
Ra
360
12
Ri = Rc
C/C++源代码
X3
Rc
IR2
IR2保护器
X4
IR2
Rc
124
110
126
300

图 8

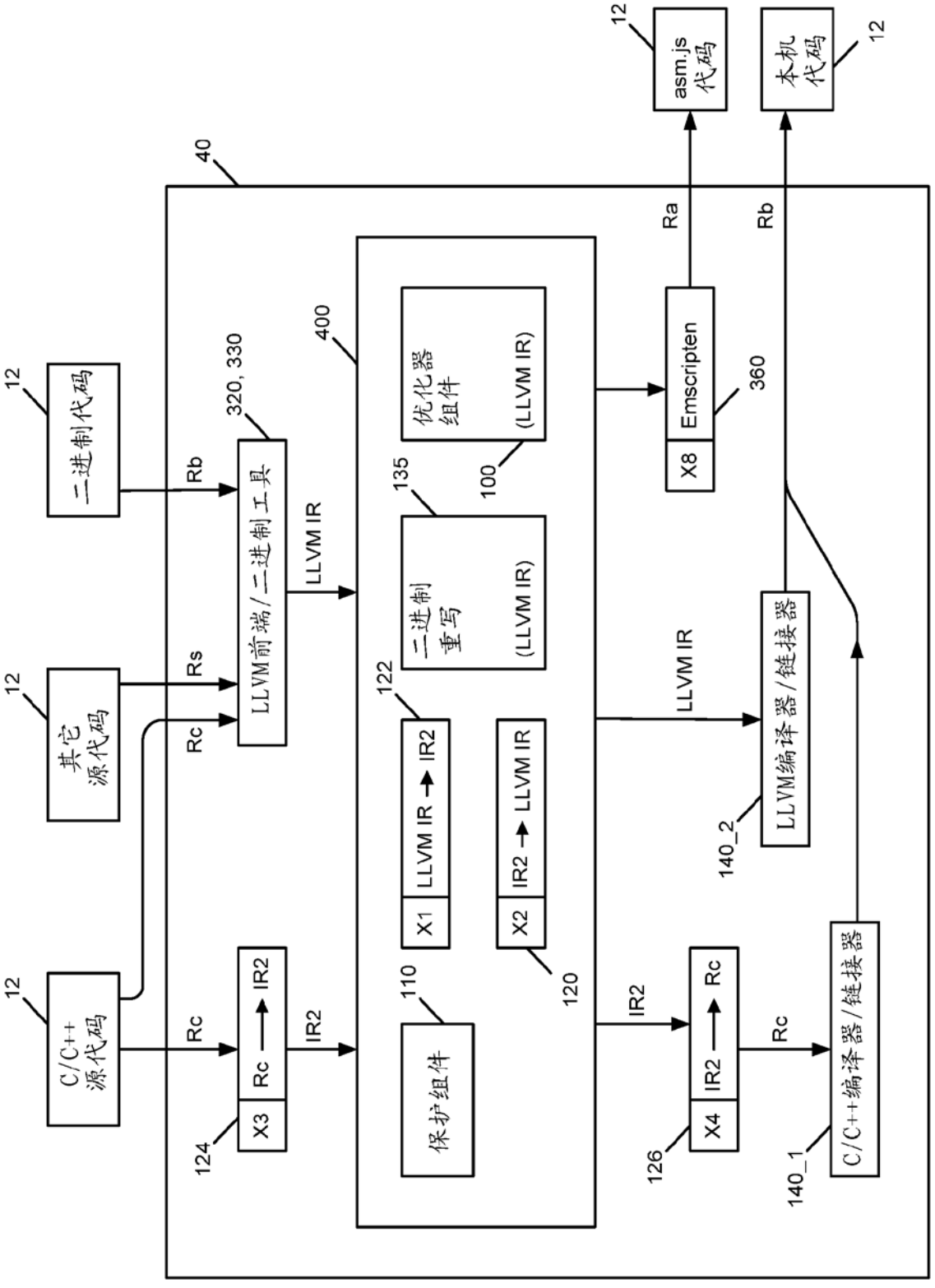

12
二进制代码
12
其它
源代码
12
C/C++
源代码
Rb
Rs
Rc
Rc
320, 330
LLVM前端/二进制工具
LLVM IR
IR2
Rc → IR2
X3
124
40
400
110
保护组件
122
LLVM IR → IR2
X1
135
二进制
重写
(LLVM IR)
100
优化器
组件
(LLVM IR)
120
IR2 → LLVM IR
X2
IR2
IR2 → Rc
X4
126
Rc
C/C++编译器/链接器
140_1
LLVM IR
LLVM编译器/链接器
140_2
Ra
Rb
Emscripten
X8
360
12
asm.js
代码
12
本机
代码

图 9

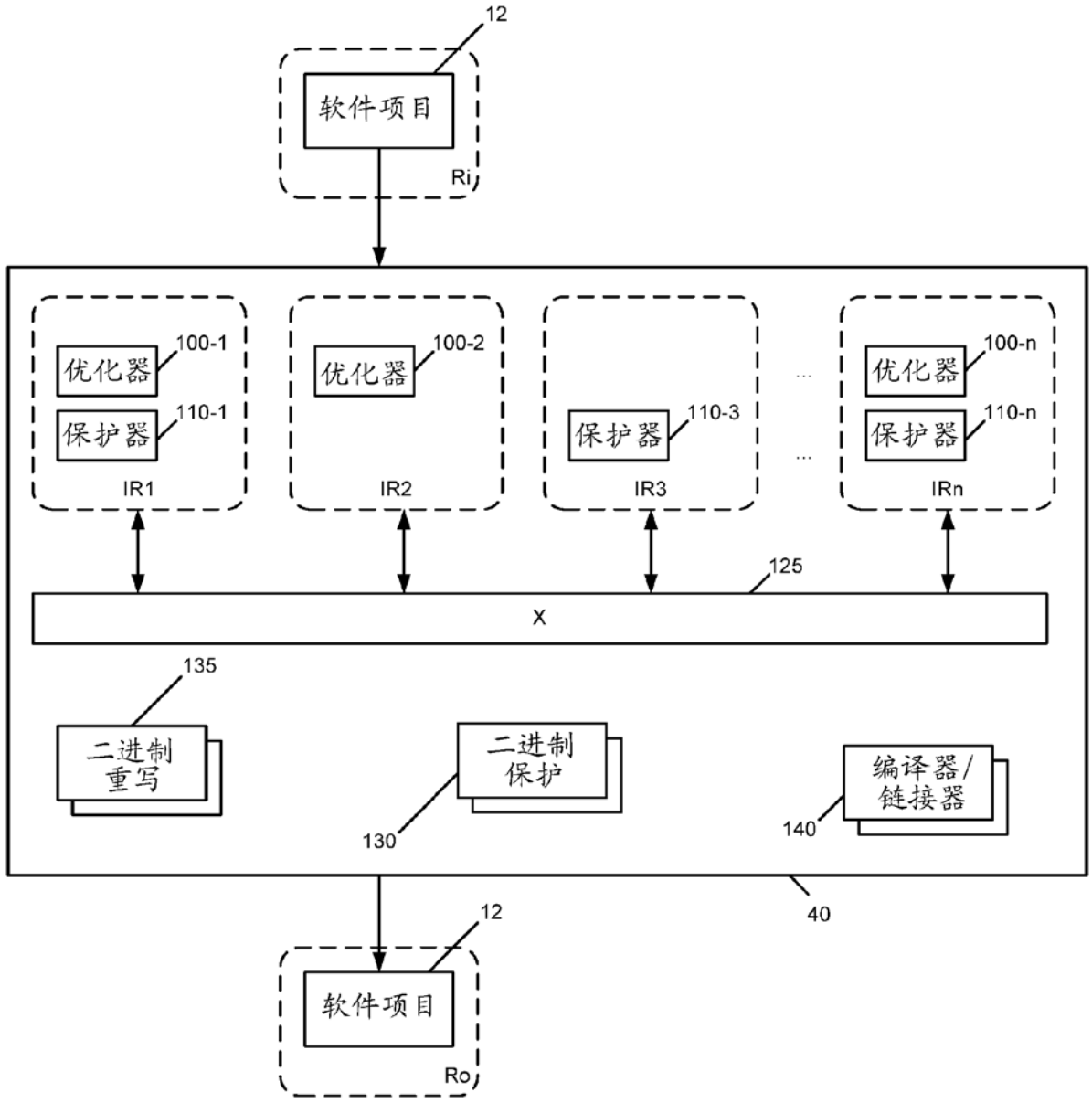

12
软件项目
Ri
优化器
100-1
保护器
110-1
IR1
优化器
100-2
IR2
保护器
110-3
IR3
...
优化器
100-n
保护器
110-n
IRn
125
X
135
二进制
重写
二进制
保护
130
编译器/
链接器
140
40
12
软件项目
Ro

图 10

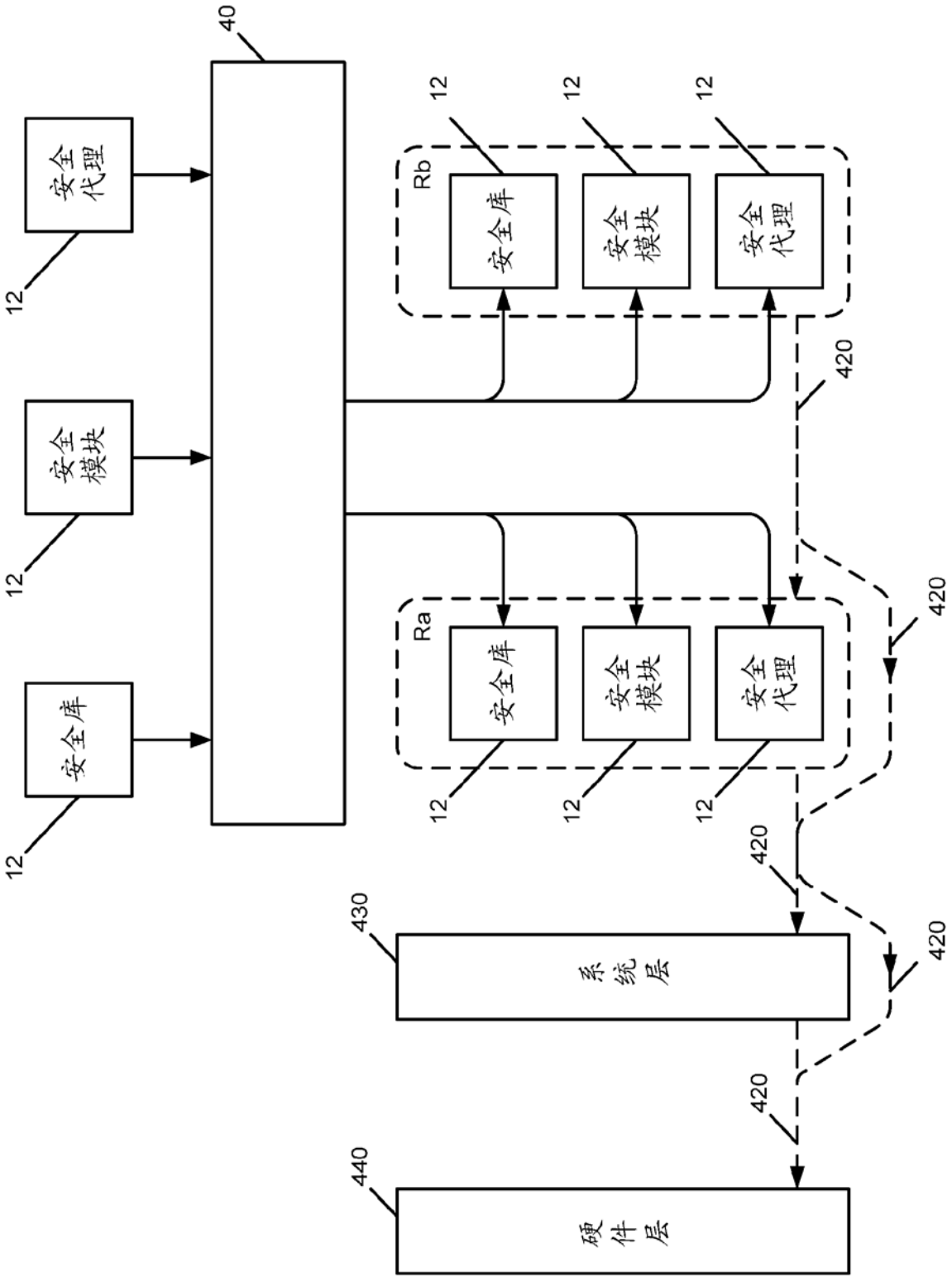

40
安全代理
安全模块
安全库
12
Rb
安全库
安全模块
安全代理
Ra
安全库
安全模块
安全代理
420
430
系统层
440
硬件层

图 11

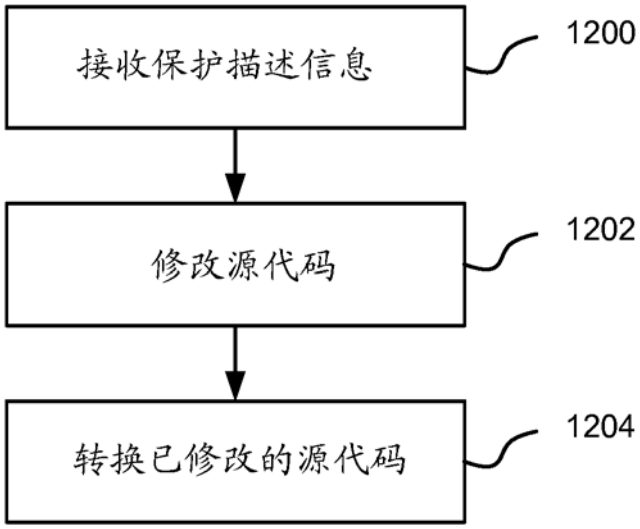

接收保护描述信息
1200
修改源代码
1202
转换已修改的源代码
1204

图 12

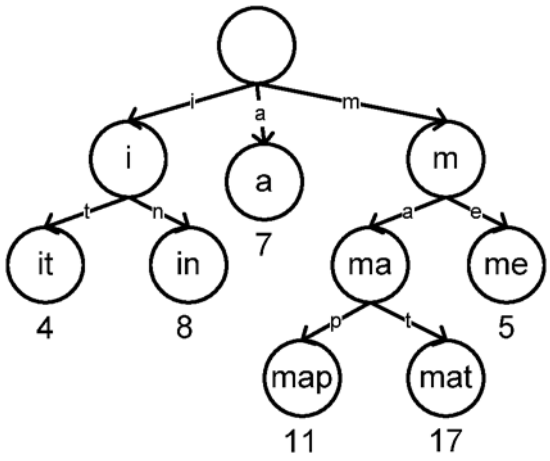

i
a
m
t
n
a
e
p
t
it
in
a
ma
me
map
mat
4
8
7
5
11
17

图 13

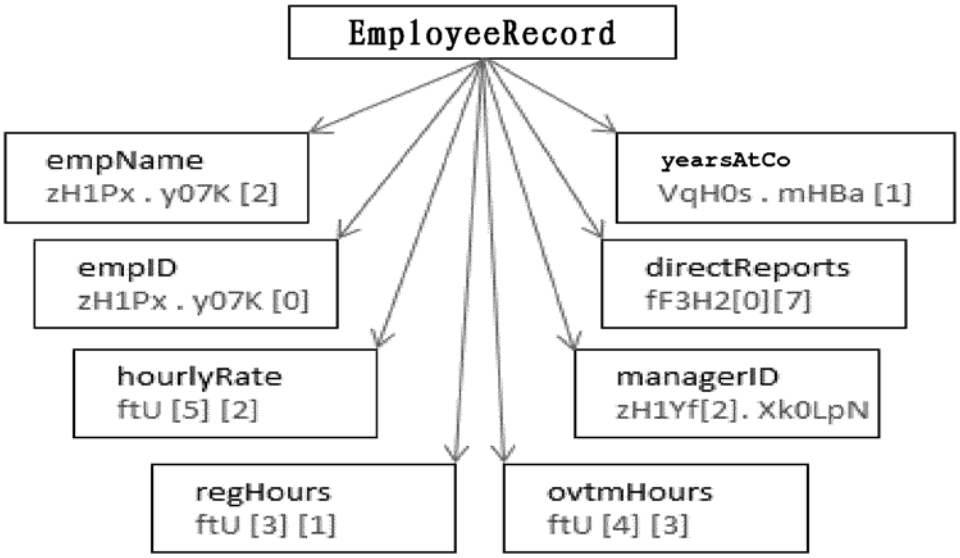

EmployeeRecord
empName
zH1Px . y07K [2]
empID
zH1Px . y07K [0]
hourlyRate
ftU [5] [2]
regHours
ftU [3] [1]
ovtmHours
ftU [4] [3]
managerID
zH1Yf[2]. Xk0LpN
directReports
fF3H2[0][7]
yearsAtCo
VqH0s . mHBa [1]

图 14