



(21) 申请号 201780052750.5

(22) 申请日 2017.08.17

(65) 同一申请的已公布的文献号
申请公布号 CN 109643345 A

(43) 申请公布日 2019.04.16

(30) 优先权数据
15/276895 2016.09.27 US

(85) PCT国际申请进入国家阶段日
2019.02.27

(86) PCT国际申请的申请数据
PCT/US2017/047363 2017.08.17

(87) PCT国际申请的公布数据
W02018/063562 EN 2018.04.05

(73) 专利权人 英特尔公司
地址 美国加利福尼亚州

(72) 发明人 M.E.库纳维斯 D.M.德拉姆
R.L.萨希塔 K.S.格雷沃尔

(74) 专利代理机构 中国专利代理(香港)有限公
司 72001

专利代理师 高苇娟 申屠伟进

(51) Int.Cl.

G06F 21/54 (2006.01)

G06F 21/62 (2006.01)

G06F 21/64 (2006.01)

H04L 9/06 (2006.01)

(56) 对比文件

US 2013312102 A1, 2013.11.21

CN 101473300 A, 2009.07.01

US 8782435 B1, 2014.07.15

US 2004025032 A1, 2004.02.05

CN 105790931 A, 2016.07.20

CN 102341806 A, 2012.02.01

US 2013061058 A1, 2013.03.07

US 2014082327 A1, 2014.03.20

王明华等. 二进制代码块: 面向二进制程序的
细粒度控制流完整性校验方法.《信息安全学
报》.2016, (第02期),

马巧梅等. 基于完整性验证的软件防篡改方
案.《计算机应用与软件》.2016, (第08期),

审查员 王春圆

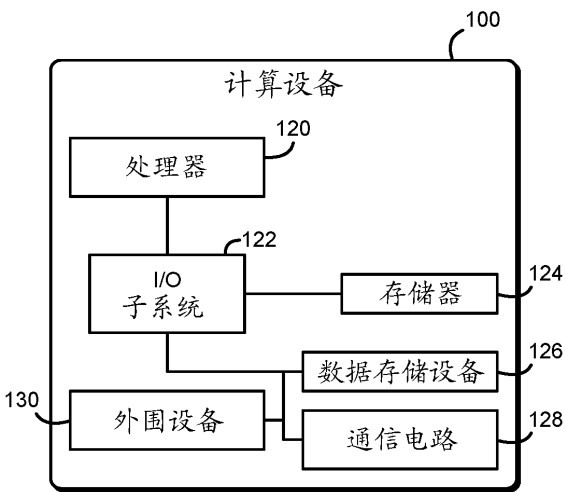
权利要求书7页 说明书23页 附图6页

(54) 发明名称

用于确定性代码流完整性保护的技术

(57) 摘要

用于代码流完整性保护的技术包括静态分
析器,其标识受保护代码的原子代码路径中的潜
在小工具。在潜在小工具之后插入标记指令,所
述标记指令具有与该标记指令的地址对应的参
数,在原子代码路径的出口点之后插入散列评估
器指令,所述散列评估器指令具有与该标记指令
的地址对应的参数,并且在散列评估器指令之后
插入比较评估器指令和散列校验指令。目标计算
设备执行受保护代码并根据标记指令的参数更
新路径散列,根据散列评估器指令的参数确定预
期散列值,并且如果路径散列和预期散列值不匹
配则生成异常。描述并要求保护其他实施例。



1. 一种用于控制流完整性保护的计算设备,所述计算设备包括:

静态分析器,其要(i)标识受保护代码的原子代码路径,其中所述原子代码路径包括入口点和出口点,以及(ii)标识所述原子代码路径中的潜在小工具,其中所述潜在小工具为面向返回编程小工具、面向跳转编程小工具、或外部动态链接库(DLL)调用;以及

检测管理器,其要(i)在所述原子代码路径中在所述潜在小工具附近插入第一标记指令,所述第一标记指令具有与第一标记指令的地址对应的参数,其中第一标记指令在被执行时使目标计算设备的处理器根据第一标记指令的地址更新所述目标计算设备的路径散列,以及(ii)在所述受保护代码中在所述原子代码路径的出口点之后插入一个或多个控制流完整性评估指令,其中所述控制流完整性评估指令在被执行时使所述目标计算设备的处理器将所述路径散列与预期散列值进行比较。

2. 根据权利要求1所述的计算设备,其中所述原子代码路径包括不具有子例程调用、子例程返回、循环入口或者除了if语句跳转或case语句跳转之外的跳转的代码序列。

3. 根据权利要求1所述的计算设备,其中插入所述一个或多个控制流完整性评估指令包括:

在所述受保护代码中在所述原子代码路径的出口点之后插入第一散列评估器指令,所述第一散列评估器指令具有与第一标记指令的地址对应的参数;

在所述受保护代码中在第一散列评估器指令之后插入第一比较评估器指令;以及

在所述受保护代码中在第一比较评估器指令之后插入散列校验指令。

4. 根据权利要求3所述的计算设备,其中:

所述静态分析器进一步要标识所述原子代码路径中的第二潜在小工具,其中第二潜在小工具发生在所述原子代码路径中的第一潜在小工具之后;

所述检测管理器进一步要(i)在所述原子代码路径中在第二潜在小工具附近插入第二标记指令,所述第二标记指令具有与第二标记指令的地址对应的参数,以及(ii)在所述受保护代码中在第一散列评估器指令之后插入第二散列评估器指令,所述第二散列评估器指令具有与第二标记指令的地址对应的参数;并且

插入第一比较评估器指令进一步包括在第二散列评估器指令之后插入第一比较评估器指令。

5. 根据权利要求3所述的计算设备,其中:

所述静态分析器进一步要:(i)标识所述受保护代码的第二原子代码路径,其中第二原子代码路径包括入口点和出口点,并且其中所述原子代码路径的出口点包括第二原子代码路径的出口点,以及(ii)标识第二原子代码路径中的第二潜在小工具;

所述检测管理器进一步要(i)在第二原子代码路径中在第二潜在小工具附近插入第二标记指令,所述第二标记指令具有与第二标记指令的地址对应的参数,(ii)在所述受保护代码中在第一比较评估器指令之后插入第二散列评估器指令,所述第二散列评估器指令具有与第二标记指令的地址对应的参数,以及(iii)在所述受保护代码中在第二散列评估器指令之后插入第二比较评估器指令;并且

插入所述散列校验指令进一步包括在第二比较评估器指令之后插入所述散列校验指令。

6. 根据权利要求3所述的计算设备,其中所述检测管理器进一步要:

在所述原子代码路径中在所述原子代码路径的入口点处插入第二标记指令,所述第二标记指令具有与第二标记指令的地址对应的参数;以及

在所述受保护代码中在所述出口点之后并且在第一散列评估器指令之前插入第二散列评估器指令,所述第二散列评估器指令具有与第二标记指令的地址对应的参数。

7. 根据权利要求3所述的计算设备,其中:

所述静态分析器进一步要标识跳转到所述原子代码路径的入口点的所述受保护代码的非返回跳转;并且

所述检测管理器进一步要(i)在所述受保护代码中在所述非返回跳转之前插入第二标记指令,所述第二标记指令具有与第二标记指令的地址对应的参数,以及(ii)在所述受保护代码中在所述出口点之后并且在第一散列评估器指令之前插入第二散列评估器指令,所述第二散列评估器指令具有与第二标记指令的地址对应的参数。

8. 一种用于控制流完整性保护的方法,所述方法包括:

通过计算设备标识受保护代码的原子代码路径,其中所述原子代码路径包括入口点和出口点;

通过所述计算设备标识所述原子代码路径中的潜在小工具,其中所述潜在小工具为面向返回编程小工具、面向跳转编程小工具、或外部动态链接库(DLL)调用;

通过所述计算设备在所述原子代码路径中在所述潜在小工具附近插入第一标记指令,所述第一标记指令具有与第一标记指令的地址对应的参数,其中第一标记指令在被执行时使目标计算设备的处理器根据第一标记指令的地址更新所述目标计算设备的路径散列;以及

通过所述计算设备在所述受保护代码中在所述原子代码路径的出口点之后插入一个或多个控制流完整性评估指令,其中所述控制流完整性评估指令在被执行时使所述目标计算设备的处理器将所述路径散列与预期散列值进行比较。

9. 根据权利要求8所述的方法,其中所述原子代码路径包括不具有子例程调用、子例程返回、循环入口或者除了if语句跳转或case语句跳转之外的跳转的代码序列。

10. 根据权利要求8所述的方法,其中插入所述一个或多个控制流完整性评估指令包括:

在所述受保护代码中在所述原子代码路径的出口点之后插入第一散列评估器指令,所述第一散列评估器指令具有与第一标记指令的地址对应的参数;

在所述受保护代码中在第一散列评估器指令之后插入第一比较评估器指令;以及

在所述受保护代码中在第一比较评估器指令之后插入散列校验指令。

11. 根据权利要求10所述的方法,进一步包括:

通过所述计算设备标识所述原子代码路径中的第二潜在小工具,其中第二潜在小工具发生在所述原子代码路径中的第一潜在小工具之后;

通过所述计算设备在所述原子代码路径中在第二潜在小工具附近插入第二标记指令,所述第二标记指令具有与第二标记指令的地址对应的参数;以及

通过所述计算设备在所述受保护代码中在第一散列评估器指令之后插入第二散列评估器指令,所述第二散列评估器指令具有与第二标记指令的地址对应的参数;

其中插入第一比较评估器指令进一步包括在第二散列评估器指令之后插入第一比较

评估器指令。

12. 一种用于控制流完整性保护的计算设备, 所述计算设备包括:

路径散列更新器, 其要 (i) 执行受保护代码的原子代码路径, 其中所述原子代码路径包括入口点、出口点和一个或多个标记指令, (ii) 响应于所述原子代码路径的执行, 根据所述一个或多个标记指令中的第一标记指令的地址更新路径散列, 以及 (iii) 响应于所述原子代码路径的执行, 到达所述原子代码路径的出口点;

散列评估器, 其要 (i) 响应于到达所述出口点, 根据一个或多个地址生成预期散列值, 其中所述一个或多个地址中的每个对应于所述一个或多个标记指令中的标记指令, 以及 (ii) 确定所述路径散列是否与所述预期散列值匹配; 以及

代码流完整性校验器, 其要响应于所述路径散列与所述预期散列值不匹配的确定而生成代码流完整性异常。

13. 根据权利要求12所述的计算设备, 其中:

所述计算设备进一步包括处理器;

更新所述路径散列包括通过所述处理器执行所述原子代码路径的第一标记指令, 所述第一标记指令具有与第一标记指令的地址对应的参数;

生成所述预期散列值包括通过所述处理器执行一个或多个散列评估器指令的组, 其中一个或多个散列评估器指令的所述组的每个散列评估器指令对应于所述一个或多个标记指令中的标记指令;

确定所述路径散列是否与所述预期散列值匹配包括通过所述处理器执行比较评估器指令; 并且

响应于所述路径散列与所述预期散列不匹配的确定而生成所述代码流完整性异常包括通过所述处理器执行散列校验指令。

14. 根据权利要求13所述的计算设备, 其中:

根据第一标记指令的地址更新所述路径散列包括根据所述受保护代码中的第一标记指令的偏移更新所述路径散列; 并且

根据所述一个或多个地址生成所述预期散列值包括根据一个或多个偏移生成所述预期散列值, 其中所述一个或多个偏移中的每个对应于所述一个或多个标记指令中的标记指令的偏移。

15. 根据权利要求12所述的计算设备, 其中:

所述计算设备进一步包括路径散列寄存器和评估器散列寄存器;

更新所述路径散列包括将所述路径散列存储在所述路径散列寄存器中;

生成所述预期散列值包括将所述预期散列值存储在所述评估器散列寄存器中; 并且

确定所述路径散列是否与所述预期散列值匹配包括比较所述路径散列寄存器和所述评估器散列寄存器。

16. 根据权利要求12所述的计算设备, 其中:

所述计算设备进一步包括完整性校验位; 并且

确定所述路径散列是否与所述预期散列值匹配包括响应于所述路径散列与所述预期散列匹配的确定而设置所述完整性校验位。

17. 根据权利要求12所述的计算设备, 其中:

所述受保护代码包括第二原子代码路径,所述第二原子代码路径包括所述出口点和一个或多个标记指令的第二集合;

所述散列评估器进一步要(i)响应于到达所述出口点,根据一个或多个地址的第二集合生成第二预期散列值,其中一个或多个地址的第二集合的每个地址对应于第二原子代码路径的一个或多个标记指令的第二集合的标记指令,以及(ii)确定所述路径散列是否与第二预期散列值匹配;并且

生成所述代码流完整性异常包括响应于所述路径散列与所述预期散列值或第二预期散列值不匹配的确定而生成所述代码流完整性异常。

18. 根据权利要求17所述的计算设备,其中:

所述计算设备进一步包括处理器;

生成所述预期散列值包括通过所述处理器执行一个或多个散列评估器指令的第一组,其中第一组的每个散列评估器指令对应于所述原子代码路径的所述一个或多个标记指令中的标记指令;

确定所述路径散列是否与所述预期散列值匹配包括通过所述处理器执行比较评估器指令;

生成第二预期散列值包括通过所述处理器执行一个或多个散列评估器指令的第二组,其中第二组的每个散列评估器指令对应于第二原子代码路径的一个或多个标记指令的第二集合的标记指令;

确定所述路径散列是否与第二预期散列匹配包括通过所述处理器执行第二比较评估器指令;并且

生成所述代码流完整性异常包括通过所述处理器执行散列校验指令。

19. 根据权利要求12所述的计算设备,进一步包括:

状态管理器,其要(i)将先前路径散列保存在所述计算设备的安全存储位置中,以及(ii)响应于所述路径散列与所述预期散列值匹配的确定,从所述安全存储位置恢复所述先前路径散列;

其中执行所述原子代码路径包括响应于所述先前路径散列的保存而执行所述原子代码路径。

20. 根据权利要求12所述的计算设备,进一步包括存储器管理器,其要(i)响应于所述路径散列的更新,根据所述路径散列生成关键域选择器,以及(ii)利用所述关键域选择器执行加密的存储器访问。

21. 一种用于控制流完整性保护的方法,所述方法包括:

通过计算设备执行受保护代码的原子代码路径,其中所述原子代码路径包括入口点、出口点和一个或多个标记指令;

响应于执行所述原子代码路径,通过所述计算设备根据所述一个或多个标记指令中的第一标记指令的地址更新路径散列;

响应于执行所述原子代码路径,通过所述计算设备到达所述原子代码路径的出口点;

响应于到达所述出口点,通过所述计算设备根据一个或多个地址生成预期散列值,其中所述一个或多个地址中的每个对应于所述一个或多个标记指令中的标记指令;

通过所述计算设备确定所述路径散列是否与所述预期散列值匹配;以及

响应于确定所述路径散列与所述预期散列值不匹配,通过所述计算设备生成代码流完整性异常。

22. 根据权利要求21所述的方法,其中:

更新所述路径散列包括通过所述计算设备的处理器执行所述原子代码路径的第一标记指令,所述第一标记指令具有与第一标记指令的地址对应的参数;

生成所述预期散列值包括通过所述处理器执行一个或多个散列评估器指令的组,其中一个或多个散列评估器指令的所述组的每个散列评估器指令对应于所述一个或多个标记指令中的标记指令;

确定所述路径散列是否与所述预期散列值匹配包括通过所述处理器执行比较评估器指令;并且

响应于确定所述路径散列与所述预期散列不匹配而生成所述代码流完整性异常包括通过所述处理器执行散列校验指令。

23. 根据权利要求21所述的方法,其中:

更新所述路径散列包括将所述路径散列存储在所述计算设备的路径散列寄存器中;

生成所述预期散列值包括将所述预期散列值存储在所述计算设备的评估器散列寄存器中;并且

确定所述路径散列是否与所述预期散列值匹配包括比较所述路径散列寄存器和所述评估器散列寄存器。

24. 根据权利要求21所述的方法,其中确定所述路径散列是否与所述预期散列值匹配包括响应于确定所述路径散列与所述预期散列匹配而设置所述计算设备的完整性校验位。

25. 根据权利要求21所述的方法,进一步包括:

响应于更新所述路径散列,通过所述计算设备根据所述路径散列生成关键域选择器;以及

通过所述计算设备使用所述关键域选择器执行加密的存储器访问。

26. 一种机器可读介质,其上存储有计算机程序代码,当所述代码被执行时促使所述机器执行如权利要求8-11,21-25中任一项所述方法。

27. 一种用于控制流完整性保护的计算设备,所述计算设备包括:

用于通过计算设备标识受保护代码的原子代码路径的部件,其中所述原子代码路径包括入口点和出口点;

用于通过所述计算设备标识所述原子代码路径中的潜在小工具的部件,其中所述潜在小工具为面向返回编程小工具、面向跳转编程小工具、或外部动态链接库(DLL)调用;

用于通过所述计算设备在所述原子代码路径中在所述潜在小工具附近插入第一标记指令的部件,所述第一标记指令具有与第一标记指令的地址对应的参数,其中第一标记指令在被执行时使目标计算设备的处理器根据第一标记指令的地址更新所述目标计算设备的路径散列;以及

用于通过所述计算设备在所述受保护代码中在所述原子代码路径的出口点之后插入一个或多个控制流完整性评估指令的部件,其中所述控制流完整性评估指令在被执行时使所述目标计算设备的处理器将所述路径散列与预期散列值进行比较。

28. 根据权利要求27所述的计算设备,其中所述原子代码路径包括不具有子例程调用、

子例程返回、循环入口或者除了if语句跳转或case语句跳转之外的跳转的代码序列。

29. 根据权利要求27所述的计算设备,其中插入所述一个或多个控制流完整性评估指令包括:

在所述受保护代码中在所述原子代码路径的出口点之后插入第一散列评估器指令,所述第一散列评估器指令具有与第一标记指令的地址对应的参数;

在所述受保护代码中在第一散列评估器指令之后插入第一比较评估器指令;以及

在所述受保护代码中在第一比较评估器指令之后插入散列校验指令。

30. 根据权利要求29所述的计算设备,进一步包括:

用于通过所述计算设备标识所述原子代码路径中的第二潜在小工具的部件,其中第二潜在小工具发生在所述原子代码路径中的第一潜在小工具之后;

用于通过所述计算设备在所述原子代码路径中在第二潜在小工具附近插入第二标记指令的部件,所述第二标记指令具有与第二标记指令的地址对应的参数;以及

用于通过所述计算设备在所述受保护代码中在第一散列评估器指令之后插入第二散列评估器指令的部件,所述第二散列评估器指令具有与第二标记指令的地址对应的参数;

其中插入第一比较评估器指令进一步包括在第二散列评估器指令之后插入第一比较评估器指令。

31. 一种用于控制流完整性保护的计算设备,所述计算设备包括:

用于通过计算设备执行受保护代码的原子代码路径,其中所述原子代码路径包括入口点、出口点和一个或多个标记指令的部件;

用于响应于执行所述原子代码路径,通过所述计算设备根据所述一个或多个标记指令中的第一标记指令的地址更新路径散列的部件;

用于响应于执行所述原子代码路径,通过所述计算设备到达所述原子代码路径的出口点的部件;

用于响应于到达所述出口点,通过所述计算设备根据一个或多个地址生成预期散列值的部件,其中所述一个或多个地址中的每个对应于所述一个或多个标记指令中的标记指令;

用于通过所述计算设备确定所述路径散列是否与所述预期散列值匹配的部件;以及

用于响应于确定所述路径散列与所述预期散列值不匹配,通过所述计算设备生成代码流完整性异常的部件。

32. 根据权利要求31所述的计算设备,进一步包括:

用于更新所述路径散列包括通过所述计算设备的处理器执行所述原子代码路径的第一标记指令的部件,所述第一标记指令具有与第一标记指令的地址对应的参数;

用于生成所述预期散列值包括通过所述处理器执行一个或多个散列评估器指令的组的部件,其中一个或多个散列评估器指令的所述组的每个散列评估器指令对应于所述一个或多个标记指令中的标记指令;

用于确定所述路径散列是否与所述预期散列值匹配包括通过所述处理器执行比较评估器指令的部件;并且

响应于确定所述路径散列与所述预期散列不匹配而生成所述代码流完整性异常包括通过所述处理器执行散列校验指令。

33. 根据权利要求31所述的计算设备,进一步包括:

用于更新所述路径散列包括将所述路径散列存储在所述计算设备的路径散列寄存器中的部件;

用于生成所述预期散列值包括将所述预期散列值存储在所述计算设备的评估器散列寄存器中的部件;并且

用于确定所述路径散列是否与所述预期散列值匹配包括比较所述路径散列寄存器和所述评估器散列寄存器的部件。

34. 根据权利要求31所述的计算设备,其中确定所述路径散列是否与所述预期散列值匹配包括响应于确定所述路径散列与所述预期散列匹配而设置所述计算设备的完整性校验位。

35. 根据权利要求31所述的计算设备,进一步包括:

用于响应于更新所述路径散列,通过所述计算设备根据所述路径散列生成关键域选择器的部件;以及

用于通过所述计算设备使用所述关键域选择器执行加密的存储器访问的部件。

用于确定性代码流完整性保护的技术

[0001] 对相关美国专利申请的交叉引用

[0002] 本申请要求2016年9月27日提交的题为“TECHNOLOGIES FOR DETERMINISTIC CODE FLOW INTEGRITY PROTECTION”的美国实用专利申请序列号15/276,895的优先权。

背景技术

[0003] 面向返回编程(ROP)漏洞利用是可以规避将存储器的位置标记为不可执行的某些防御的恶意的软件(恶意软件)的越来越常见的形式。ROP漏洞利用通过将每个以“返回”指令结束的大量现有的可执行代码段(称为小工具(gadget))串起来而工作。每个ROP小工具通常短,并且通常不对应于现有过程或甚至可执行代码中的现有指令边界。攻击者构造恶意堆栈,所述恶意堆栈包括一系列指向所期望的小工具序列的返回地址。通过使计算机的处理器使用恶意堆栈而不是合法系统堆栈执行软件来实行ROP漏洞利用。例如,可以通过粉碎堆栈、使用缓冲区溢出漏洞利用、转向新堆栈或以其他方式破坏系统堆栈来引入恶意堆栈。面向跳转编程(JOP)和面向调用编程(COP)漏洞利用是类似的,但目标小工具分别以间接跳转指令或调用指令而不是返回指令开始或结束。

[0004] 可以通过维持与普通系统堆栈(也称为“遗留堆栈”)并行的“影子堆栈”来防止某些ROP漏洞利用。影子堆栈维持普通软件不可访问的存储器中的遗留堆栈的副本,并且可用于确定遗留堆栈是否已被恶意软件篡改。可用于帮助防止ROP、JOP或其他控制流漏洞利用的其他措施可以包括将ENDBRANCH指令插入到二进制文件中以专门标记针对间接分支或跳转的被允许目标。

附图说明

[0005] 在附图中通过示例的方式而非通过限制的方式图示了本文中描述的概念。为了图示的简单和清楚,各图中图示的元件不一定按比例绘制。在认为适当的情况下,参考标签在各图之间重复以指示对应或类似的元件。

[0006] 图1是用于代码流完整性保护的计算设备的至少一个实施例的简化框图;

[0007] 图2是可以由图1的一个或多个计算设备建立的各种环境的简化框图;

[0008] 图3是用于静态地分析和检测(instrument)可以由图1-2的计算设备执行的代码的方法的至少一个实施例的简化流程图;

[0009] 图4是为了代码流完整性保护所检测的原子代码路径的至少一个实施例的简化流程图;

[0010] 图5A和5B是可以由图1-2的计算设备执行的用于代码流完整性保护的方法的至少一个实施例的简化流程图;以及

[0011] 图6是可以由图1-2的计算设备的处理器执行的用于代码流完整性保护的方法的至少一个实施例的简化流程图。

具体实施方式

[0012] 虽然本公开的概念对各种修改和替代形式敏感,但是其特定实施例已被在附图中通过示例的方式示出并且将在本文中详细地描述。然而,应当理解,不意图将本公开的概念限制于所公开的特定形式,而正相反,意图是涵盖与本公开和所附权利要求一致的所有修改、等同物和替代物。

[0013] 说明书中对“一个实施例”、“实施例”、“说明性实施例”等的引用指示所描述的实施例可以包括特定特征、结构或特性,但是每个实施例可能或可能不一定包括该特定特征、结构或特性。此外,这样的短语不一定是指同一实施例。进一步地,当结合实施例描述特定特征、结构或特性时,主张:无论是否被明确地描述,结合其他实施例实现这样的特征、结构或特性在本领域技术人员的知识内。另外,应当领会,以“A、B和C中的至少一个”的形式包括在列表中的项目可以意味着(A);(B);(C);(A和B);(A和C);(B和C);或(A、B和C)。类似地,以“A、B或C中的至少一个”形式列出的项目可以意味着(A);(B);(C);(A和B);(A和C);(B和C);或(A、B和C)。

[0014] 在一些情况下,所公开的实施例可以以硬件、固件、软件或其任何组合来实现。所公开的实施例还可以被实现为由一个或多个暂时性或非暂时性机器可读(例如,计算机可读)存储介质承载或存储在之上的指令,其可以由一个或多个处理器读取和执行。机器可读存储介质可以体现为用于存储或传输以机器可读形式的信息的任何存储设备、机构或其他物理结构(例如,易失性或非易失性存储器、媒体盘或其他媒体设备)。

[0015] 在附图中,可能以特定布置和/或排序示出了一些结构特征或方法特征。然而,应当领会,可能不需要这样的特定布置和/或排序。而是,在一些实施例中,可以以与说明性图中所示不同的方式和/或顺序来布置这样的特征。另外,在特定图中包括结构特征或方法特征不有意暗示在所有实施例中都需要这样的特征,并且在一些实施例中,可以不包括这样的特征或者可以将其与其他特征组合。

[0016] 现在参考图1,在说明性实施例中,计算设备100可以用于代码流完整性保护。在使用中,如下面进一步描述的,编译器计算设备100a可以静态地分析和检测受保护代码,并且目标计算设备100b可以执行所检测到的代码。编译器计算设备100a标识受保护代码内的一个或多个原子代码路径,并在原子代码路径中的每个潜在小工具之后插入标记指令。编译器计算设备100a在来自原子代码路径的每个出口点处插入用来核实是否执行了有效原子代码路径的每个标记指令的代码。在使用中,目标计算设备100b执行包括标记指令的原子代码路径。每个标记指令使目标计算设备100b基于当前标记指令的偏移来更新路径散列寄存器。在退出原子代码路径时,目标计算设备100b确定当前路径散列寄存器是否与针对该出口点的任何原子代码路径的预期散列值匹配。因此,计算设备100可以允许对抗代码流漏洞利用(诸如面向调度编程、面向返回编程或面向跳转编程攻击)的确定性保护。甚至针对跨不同的静态地链接的包(例如,DLL)做出的调用,也提供这样的保护。另外,计算设备100可以在性能和/或所需处理器硬件方面的合理成本提供控制流完整性保护。尽管被描述为利用编译器计算设备100a编译和检测受保护代码并且利用目标计算设备100b执行受保护代码,但是应当理解,在一些实施例中,同一计算设备100可以分析、检测和执行受保护代码。

[0017] 计算设备100可以体现为能够执行本文中描述的功能的任何类型的计算或计算机

设备,包括但不限于计算机、台式计算机、工作站、服务器、膝上型计算机、笔记本计算机、平板计算机、移动计算设备、可穿戴计算设备、网络器具、web器具、分布式计算系统、基于处理器的系统和/或消费电子设备。如图1中所示,计算设备100说明性地包括处理器120、输入/输出子系统122、存储器124、数据存储设备126和通信子系统128。当然,在其他实施例中,计算设备100可以包括其他或附加组件,诸如在台式计算机中通常发现的那些(例如,各种输入/输出设备)。另外,在一些实施例中,说明性组件中的一个或多个可以被合并并在另一组件中,或者以其他方式形成另一组件的一部分。例如,在一些实施例中,存储器124或其部分可以被合并并在处理器120中。

[0018] 处理器120可以体现为能够执行本文中描述的功能的任何类型的处理器。处理器120可以体现为(一个或多个)单核或多核处理器、数字信号处理器、微控制器、或者其他处理器或处理/控制电路。类似地,存储器124可以体现为能够执行本文中描述的功能的任何类型的易失性或非易失性存储器或数据存储器。在操作中,存储器124可以存储在计算设备100的操作期间使用的各种数据和软件,诸如操作系统、应用程序、程序、库和驱动程序。

[0019] 存储器124经由I/O子系统122通信地耦合到处理器120,所述I/O子系统122可以体现为用来促进关于处理器120、存储器124和计算设备100的其他组件的输入/输出操作的电路和/或组件。例如,I/O子系统122可以体现为或以其他方式包括存储器控制器中枢、输入/输出控制中枢、平台控制器中枢、集成控制电路、固件设备、通信链路(即,点对点链路、总线链路、电线、电缆、光导、印刷电路板迹线等)和/或其他组件和子系统以促进输入/输出操作。在一些实施例中,I/O子系统122可以形成片上系统(SoC)的一部分,并且被连同处理器120、存储器124和计算设备100的其他组件一起合并并在单个集成电路芯片上。

[0020] 数据存储设备126可以体现为被配置用于短期或长期数据存储的任何类型的一个或多个设备,诸如例如存储器设备和电路、存储器卡、硬盘驱动器、固态驱动器或其他数据存储设备。计算设备100的通信子系统128可以体现为能够通过网络使能够实现计算设备100和其他远程设备之间的通信的任何通信电路、设备或其集合。通信子系统128可以被配置成使用任何一个或多个通信技术(例如,有线或无线通信)和相关联的协议(例如,以太网、蓝牙®、Wi-Fi®、WiMAX等)来实现这样的通信。

[0021] 如所示,计算设备100还可以包括一个或多个外围设备130。外围设备130可以包括任何数目的附加输入/输出设备、接口设备和/或其他外围设备。例如,在一些实施例中,外围设备130可以包括显示器、触摸屏、图形电路、键盘、鼠标、扬声器系统、麦克风、网络接口和/或其他输入/输出设备、接口设备和/或外围设备。

[0022] 现在参考图2,在说明性实施例中,编译器计算设备100a在操作期间建立环境200。说明性环境200包括静态分析器204和检测管理器206。环境200的各种组件可以体现为硬件、固件、软件或其组合。照此,在一些实施例中,环境200的组件中的一个或多个可以体现为电路或电气设备的集合(例如,静态分析器电路204和/或检测管理器电路206)。应当领会,在这样的实施例中,静态分析器电路204和/或检测管理器电路206中的一个或多个可以形成处理器120、I/O子系统122和/或编译器计算设备100a的其他组件中的一个或多个的一部分。另外,在一些实施例中,说明性组件中的一个或多个可以形成另一组件的一部分和/或说明性组件中的一个或多个可以彼此独立。

[0023] 静态分析器204被配置成标识受保护代码202的原子代码路径。受保护代码202可

以体现为任何二进制代码、字节代码、中间表示或可以由目标计算设备100b的处理器120执行或被进一步编译以由目标计算设备100b的处理器120执行的其他代码。原子代码路径可以体现为例如不包含子例程调用和返回,不包含除了if-then-else和case语句的跳转之外的跳转,并且不包含包括for、while和do-while循环的循环的代码路径。每个原子代码路径包括入口点和出口点,并且可以体现为不具有子例程调用、子例程返回、循环入口或除了if语句跳转或case语句跳转之外的跳转的代码序列。静态分析器204被进一步配置成标识每个原子代码路径中的潜在小工具。潜在小工具可能包括面向返回编程小工具、面向跳转编程小工具或外部动态链接库(DLL)调用。在一些实施例中,静态分析器204可以被配置成标识跳转到原子代码路径的入口点的受保护代码202的非返回跳转。

[0024] 检测管理器206检测受保护代码202,从而生成所检测到的受保护代码202'。检测管理器206被配置成在每个原子代码路径中在每个潜在小工具附近(例如,紧接在每个小工具之前或紧接在每个小工具之后)插入标记指令,其中每个标记指令具有与标记指令的地址对应的参数。在一些实施例中,检测管理器206被进一步配置成在受保护代码202中在非返回跳转之前插入标记指令,所述标记指令具有与标记指令的地址对应的参数。检测管理器206被进一步配置成在受保护代码202中在原子代码路径的出口点之后插入一个或多个控制流完整性评估指令。控制流完整性评估指令可以包括散列评估器(evaluator)指令、比较评估器指令和散列校验(check)指令。检测管理器206可以被配置成在受保护代码202中在原子代码路径的出口点之后插入散列评估器指令,其中每个散列评估器指令具有与对应标记指令的地址对应的参数,以在受保护代码202中在散列评估器指令的每组之后插入比较评估器指令,并在受保护代码202中在针对每个出口点的最后插入的比较评估器指令之后插入散列校验指令。检测管理器206可以被配置成在受保护代码202中在原子代码路径的每个入口点处插入进栈指令,并且在受保护代码202中在针对每个出口点的散列校验指令之后插入退栈指令。

[0025] 仍然参考图2,在说明性实施例中,目标计算设备100b在操作期间建立环境220。说明性环境220包括路径散列更新器222、散列评估器226、代码流完整性校验器232、状态管理器234,并且在一些实施例中包括存储器管理器236。环境220的各种组件可以体现为硬件、固件、软件或其组合。照此,在一些实施例中,环境220的组件中的一个或多个可以体现为电路或电气设备的集合(例如,路径散列更新器电路222、散列评估器电路226、代码流完整性校验器电路232、状态管理器电路234和/或存储器管理器电路236)。应当领会,在这样的实施例中,路径散列更新器电路222、散列评估器电路226、代码流完整性校验器电路232、状态管理器电路234和/或存储器管理器电路236中的一个或多个可以形成处理器120、I/O子系统122和/或目标计算设备100b的其他组件中的一个或多个的一部分。另外,在一些实施例中,说明性组件中的一个或多个可以形成另一组件的一部分和/或说明性组件中的一个或多个可以彼此独立。

[0026] 路径散列更新器222被配置成执行受保护代码202'的原子代码路径。每个原子代码路径包括入口点、出口点和一个或多个标记指令。路径散列更新器222被配置成响应于执行原子代码路径而根据标记指令的地址来更新路径散列。如所示,计算设备100b进一步包括路径散列寄存器224,并且更新路径散列可以包括将路径散列存储在路径散列寄存器224中。路径散列寄存器224可以体现为处理器120的内部寄存器,并且可以被保存和恢复为过

程的过程控制块的一部分。可以通过利用处理器120执行原子代码路径的标记指令来更新路径散列,其中每个标记指令具有与该标记指令的地址对应的参数。当执行原子代码路径时,路径散列更新器222可以到达原子代码路径的出口点。

[0027] 散列评估器226被配置成响应于到达出口点而根据与在出口点处结束的有效原子代码路径的标记指令对应的一个或多个地址生成预期散列值。如所示,计算设备100b进一步包括评估器散列寄存器228,并且生成预期散列值可以包括将预期散列值存储在评估器散列寄存器228中。评估器散列寄存器228可以体现为处理器120的内部寄存器,并且可以被保存和恢复为过程的过程控制块的一部分。可以通过由处理器120执行一个或多个散列评估器指令的组来生成每个预期散列值,其中该组的每个散列评估器指令对应于在出口点处结束的有效原子代码路径的标记指令。散列评估器226被进一步配置成确定路径散列是否与预期散列值匹配。如所示,计算设备100b进一步包括完整性校验位230。完整性校验位230可以体现为处理器120的内部寄存器,并且可以被保存和恢复为过程的过程控制块的一部分。确定路径散列是否与预期散列值匹配可以包括响应于确定路径散列与预期散列匹配而设置完整性校验位230。确定路径散列是否与预期散列值匹配可以包括由处理器120执行比较评估器指令。

[0028] 代码流完整性校验器232被配置成响应于确定路径散列与预期散列值不匹配而生成代码流完整性异常。生成代码流完整性异常可以包括由处理器120执行散列校验指令。

[0029] 状态管理器234被配置成将先前路径散列保存在计算设备100b的安全存储位置中,并且如果该路径散列与预期散列值匹配,则从安全存储位置恢复先前路径散列。可以在执行原子代码路径之前保存先前路径。保存先前路径散列可以包括由处理器120执行进栈指令以将先前路径散列存储在安全堆栈中,并且恢复先前路径散列可以包括由处理器120执行退栈指令以从安全堆栈取回(retrieve)先前路径散列。

[0030] 存储器管理器236被配置成在原子代码路径的执行期间将路径散列用作参数来实行加密的存储器访问。换言之,响应于各种标记指令的执行而更新的路径散列的值可以用于实行加密的存储器访问。

[0031] 尽管被图示为由分离的编译器计算设备100a和目标计算设备100b建立,但是应当理解,在一些实施例中,环境200、220中的一些或全部可以由同一计算设备100建立。例如。在一些实施例中,受保护代码202可以由同一计算设备100检测和执行。

[0032] 现在参考图3,在使用中,编译器计算设备100a可以执行用于静态地分析和检测代码的方法300。应当领会,在一些实施例中,方法300的操作可以由如图2中所示的编译器计算设备100a的环境200的一个或多个模块执行。另外或替代地,在一些实施例中,方法300可以体现为存储在计算机可读介质上的各种指令,其可以由处理器120和/或计算设备100的其他组件执行以使计算设备100实行方法300。计算机可读介质可以体现为能够由计算设备100读取的任何类型的介质,包括但不限于存储器124、数据存储设备126、计算设备100的其他存储器或数据存储设备、计算设备100的外围设备可读的便携式介质,和/或其他介质。

[0033] 方法300开始于框302,其中编译器计算设备100a静态地分析受保护代码202。受保护代码202可以体现为源代码、目标代码、可执行代码、字节代码或可以由目标计算设备100b执行的任何其他计算机代码。如上面所描述的,静态分析可以由编译器、链接器和/或独立的静态分析工具执行。编译器计算设备100a可以在静态分析期间分析和检测受保护代

码202,如下面进一步描述的。当编译器计算设备100a检测受保护代码202时,可以将一个或多个指令插入到所检测到的受保护代码202'中。

[0034] 在框304中,编译器计算设备100a标识受保护代码202内的一个或多个原子代码路径。原子代码路径可以体现为不包括任何子例程调用或返回、不包括除了if-then-else语句或case语句的跳转之外的任何跳转、并且不包括任何循环(例如,for循环、while循环、do-while循环等)的连续代码段。如下面进一步描述的,原子代码路径的运行处理可被子例程调用、某些跳转或循环中断,并且原子代码的处理可在返回时从中断恢复。每个原子代码路径包括入口点和出口点。某些原子代码路径可以分支到多个出口点,并且每个出口点可以从多个原子代码路径到达。在一些实施例中,在框306中,编译器计算设备100a可以标识多个相关原子代码路径的组。组内的原子代码路径可以重叠执行,共享入口点或出口点,或者以其他方式相关。

[0035] 现在参考图4,示意图400图示了原子代码路径组402的一个潜在实施例。如所示,原子代码路径组402包括单个入口点404和两个出口点406、408。原子代码路径组402进一步包括多个代码路径段410、412、414、416、418。如所示,入口点404通向代码路径段410,其分支到代码段412、414、416。代码路径段412通向出口点406。代码路径段414、416连接到代码路径段418,其通向出口点408。因此,原子代码路径组402包括至少三个原子代码路径:包括代码路径段410、412的路径,包括代码路径段410、414、418的路径,以及包括代码路径段410、416、418的路径。

[0036] 向后参考图3,在框304中标识原子代码路径之后,在框308中编译器计算设备100a在原子代码路径的入口点处插入入口代码。如下面结合图5A和5B进一步描述的,在运行时,目标计算设备100b执行入口代码以为控制流完整性校验作准备。在框310中,编译器计算设备100a插入用来将目标计算设备100b的路径散列寄存器224进栈到安全堆栈的一个或多个进栈指令。安全堆栈可以体现为目标计算设备100b的被保护免受未经授权访问和/或修改的任何安全存储器位置。例如,安全堆栈可以体现为加密的存储器页面或体现为处理器120的寄存器文件。在框312中,编译器计算设备100a插入用来初始化目标计算设备100b的路径散列寄存器224的一个或多个指令。所述指令可以将路径散列寄存器224清零(clear),将路径散列寄存器224设置成诸如零之类的默认值,或者以其他方式准备路径散列寄存器224以执行控制流保护。在框314中,编译器计算设备100a插入用来初始化目标计算设备100b的完整性校验位230的一个或多个指令。所述指令可以将完整性校验位230清零,将完整性校验位230设置成诸如零之类的默认值,或者以其他方式准备完整性校验位230以执行控制流保护。在框316中,编译器计算设备100a在原子代码路径的入口点处插入标记指令。可以将标记指令插入在入口点或入口点附近的地址处,以使得在进入原子代码路径时执行标记指令。编译器计算设备100a可以包括标记指令的偏移,作为标记指令的参数。如下面进一步描述的,在执行期间,运行时的标记指令基于所供应的标记指令的偏移来更新路径散列寄存器224。

[0037] 在一些实施例中,在框318中,编译器计算设备100a可以在跳转到原子代码路径的入口点的非返回跳转指令之前插入标记指令。类似地,编译器计算设备100a可以包括标记指令的偏移,作为标记指令的参数。在非返回跳转之前执行标记指令有效地包括原子代码路径的控制流程中的跳转,并且因此允许校验非返回跳转和原子代码路径的代码流完整

性。

[0038] 在框320中,编译器计算设备100a标识原子代码路径中的一个或多个潜在小工具。每个小工具可以体现为以可以用在代码流漏洞利用中的二进制字符串结束的指令序列。例如,以返回指令或跳转指令的等同物结束的小工具可以分别用在面向返回编程(ROP)或面向跳转编程(JOP)漏洞利用中。对于某些处理器架构,诸如Intel® 64和IA-32架构,指令可能具有可变长度,并且因此小工具不需要与原子代码路径的指令边界对齐。

[0039] 在框322中,编译器计算设备100a在原子代码路径中的每个潜在小工具附近插入标记指令。在一个实施例中,标记指令可以放置在原子代码路径中的潜在小工具之前。在另一实施例中,标记指令可以放置在原子代码路径中的潜在小工具之后。标记指令可以被在紧接在小工具之前或之后的指令边界处插入在所检测到的受保护代码202'中。因此,如果小工具被作为漏洞利用的一部分而执行,则可能不执行标记指令。另外或替代地,如果小工具被作为漏洞利用的一部分而执行,则可能以错误的顺序执行标记指令。编译器计算设备100a包括标记指令的偏移,作为标记指令的参数。如下面进一步描述的,在执行期间,标记指令基于所供应的标记指令的偏移来更新路径散列寄存器224。

[0040] 在框324中,编译器计算设备100a在原子代码路径的出口点处插入出口代码。如下面结合图5A和5B进一步描述的,在运行时,出口代码由目标计算设备100b执行,以确定是否已经执行到出口点的有效原子代码路径并且从而校验代码流完整性。出口代码可以被紧接在出口点之后插入在所检测到的受保护代码202'中。

[0041] 在框326中,编译器计算设备100a针对在出口点处结束的原子代码路径的每个标记指令插入散列评估器指令。每个散列评估器指令包括等于对应标记指令的偏移的参数。以与对应标记指令相同的顺序插入散列评估器指令。在运行时,执行每个散列评估器指令使目标计算设备100b更新评估器散列寄存器228。因为散列评估器指令具有与有效原子代码路径的标记指令相同的参数和排序,所以评估器散列寄存器228的最终值与针对该有效原子代码路径的路径散列寄存器224的期望值匹配。由于静态地标识了有效原子代码路径和潜在小工具,因此散列评估器指令和相关联的参数也可以被静态地包括在所检测到的受保护代码202'中。

[0042] 在框328中,编译器计算设备100a在(一个或多个)散列评估器指令之后插入比较评估器指令。在运行时,执行比较评估器指令使目标计算设备100b比较路径散列寄存器224和评估器散列寄存器228的值。如果那些值匹配,则目标计算设备100b可以例如使用逻辑或运算来设置完整性校验位230。如果值不匹配,则目标计算设备100b可以不改变完整性校验位230的值。如果路径散列寄存器224与评估器散列寄存器228匹配,则这指示目标计算设备100b已经以正确的顺序执行了原子代码路径的所有标记指令,从而指示恰当地执行了原子代码路径。

[0043] 在框330中,编译器计算设备100a确定针对当前出口点是否存在其他原子代码路径。例如,编译器计算设备100a可以检查相关原子代码路径的组以找到在当前出口点处结束的附加原子代码路径。在框332中,编译器计算设备100a检查是否仍有附加原子代码路径。如果这样,则方法300循环回到框324,其中编译器计算设备100a针对所述附加原子代码路径插入散列评估器指令和比较评估器指令。如果针对当前出口点不存在更多原子代码路径,则方法300前进到框334。

[0044] 在框334中,编译器计算设备100a插入散列校验指令。散列校验指令使目标计算设备100b确定任何比较评估器指令是否在路径散列寄存器224和评估器散列寄存器228之间找到匹配,并且如果否,则生成代码流完整性错误、异常、中断,或以其他方式指示已发生代码流完整性违规。例如,散列校验指令可以确定是否设置完整性校验位230,并且如果否,则生成代码流完整性异常。

[0045] 在框336中,编译器计算设备100a插入用来使目标计算设备100b从安全堆栈恢复路径散列寄存器224的被保存值的退栈指令。如上面所描述的,当进入原子代码路径时,路径散列寄存器224值可能已被进栈到安全堆栈。恢复路径散列寄存器224可以允许目标计算设备100b继续校验被例如由于子例程调用或循环所中断的原子代码路径的控制流完整性。

[0046] 在框338中,编译器计算设备100a确定受保护代码202中是否存在附加原子代码路径。如果这样,则方法300循环回到框308以插入入口代码、标记指令和出口代码,并且以其他方式生成所检测到的受保护代码202'。如果不存在另外的原子代码路径,则方法300循环回到框302,其中编译器计算设备100a可以处理附加的受保护代码202。

[0047] 再次参考图4,如上面所描述的,图400图示了可由编译器计算设备100a分析和检测的原子代码路径组402。图400还图示了可由编译器计算设备100a输出的所检测到的原子代码路径组402'。所检测到的原子代码路径组402'包括所检测到的代码路径段410'、412'、414'、416'、418',其被通过在每个潜在小工具之后插入标记指令而检测到。所检测到的代码路径段410'包括标记指令420,所检测到的代码路径段412'包括标记指令422、424,所检测到的代码路径段414'包括标记指令426,所检测到的代码路径段416'包括标记指令428,并且所检测到的代码路径段418'包括标记指令430。每个标记指令包括等于该标记指令的存储器中偏移的参数。例如,标记指令420在偏移A处,标记指令422在偏移B处,等等,如图4中所示。

[0048] 如图4中进一步所示,已经在出口点406、408之后插入出口代码。散列评估器指令432、434、436跟随出口点406。散列评估器指令432、434、436形成对应于原子代码路径410'、412'的组438。因此,散列评估器指令432的参数A匹配标记指令420的偏移A,散列评估器指令434的参数B匹配标记指令422的偏移B,并且散列评估器指令436的参数C匹配标记指令424的偏移C。比较评估器指令440跟随组438,并且散列校验指令442跟随比较评估器指令442。

[0049] 类似地,散列评估器指令444、446、448跟随出口点408。散列评估器指令444、446、448形成对应于原子代码路径410'、414'、418'的组450。因此,散列评估器指令444的参数A匹配标记指令420的偏移A,散列评估器指令446的参数D匹配标记指令426的偏移D,并且散列评估器指令448的参数E匹配标记指令430的偏移E。比较评估器指令452跟随组450。因为出口点408具有两个潜在有效的原子代码路径,所以散列评估器指令454、456、458跟随比较评估器指令452。散列评估器指令454、456、458形成对应于原子代码路径410'、416'、418'的组460。因此,散列评估器指令454的参数A匹配标记指令420的偏移A,散列评估器指令456的参数F匹配标记指令428的偏移F,并且散列评估器指令458的参数E匹配标记指令430的偏移E。比较评估器指令462跟随组460,并且散列校验指令464跟随比较评估器指令462。

[0050] 如图4中所示,可以在入口点404之前插入进栈指令466,并且可以分别在散列校验指令442、464中的每个之后插入退栈指令468、470。应当注意,对路径散列值进行操作的进

栈和退栈指令可以与对通用寄存器内容进行操作的常规处理器进栈和退栈指令不同。在一个实施例中,对路径散列值进行操作的进栈和退栈指令可以将路径散列值插入到受保护存储器区域中或从受保护存储器区域提取路径散列值。

[0051] 现在参考图5A和5B,在使用中,目标计算设备100b可以执行用于代码流完整性保护的方法500。应当领会,在一些实施例中,方法500的操作可以由如图2中所示的目标计算设备100b的环境220的一个或多个模块执行。另外或替代地,在一些实施例中,方法500可以体现为存储在计算机可读介质上的各种指令,其可以由处理器120和/或计算设备100的其他组件执行以使计算设备100实行方法500。计算机可读介质可以体现为能够由计算设备100读取的任何类型的介质,包括但不限于存储器124、数据存储设备126、计算设备100的其他存储器或数据存储设备、计算设备100的外围设备可读的便携式介质,和/或其他介质。

[0052] 可以在所检测到的受保护代码202'的执行期间由目标计算设备100b执行方法500。如上面结合图3的方法所描述的,编译器计算设备100a可以静态地分析受保护代码202以生成所检测到的受保护代码202'。目标计算设备100b可以经由网络连接接收所检测到的受保护代码202',或者可以以其他方式被提供所检测到的受保护代码202'。当然,如上面所描述的,在一些实施例中,可以由同一计算设备100检测和执行所检测到的受保护代码202'。

[0053] 方法500开始于框502,其中目标计算设备100b确定是否已到达原子代码路径的入口点。可能例如在开始执行原子代码路径时或者当中断另一原子代码路径以执行该原子代码路径时到达原子代码路径入口点。如上面结合图3所描述的,可以在受保护代码202的静态分析期间标识原子代码路径的入口点,并且因此可以在受保护代码202'中静态地编码是否已经到达入口点的确定。如果没有到达入口点,则方法500循环回到框502以继续监视入口点。如果已到达入口点,则方法500前进到框504。

[0054] 在框504中,目标计算设备100b将当前路径散列状态保存在安全存储位置中。例如,目标计算设备100b可以保存路径散列寄存器224的当前状态。保存当前路径散列状态可以允许目标计算设备100b中断针对当前原子代码路径的代码流完整性校验并执行针对新原子代码路径的代码流完整性校验。在框506中,目标计算设备100b执行进栈指令以将路径散列寄存器224的内容存储到安全堆栈。如上面所描述的,安全堆栈可以体现为被保护免受未经授权访问和/或修改的任何安全存储器位置,诸如加密的存储器页面或处理器120的寄存器文件。安全堆栈可以支持相对小的数目的条目(例如,16、32或128),并且因此可以使用处理器中的专用寄存器文件来实现。

[0055] 在框508中,目标计算设备100b初始化路径散列状态以为针对新原子代码路径的控制流完整性校验作准备。在框510中,目标计算设备100b初始化路径散列寄存器224和评估器散列寄存器228。目标计算设备100b可以例如将寄存器224、228清零,将寄存器224、228设置成诸如零之类的默认值,或以其他方式准备寄存器224、228以执行控制流保护。在框512中,目标计算设备100b(例如通过将完整性校验位230设置成零)将完整性校验位230清零。

[0056] 在框514中,目标计算设备100b基于入口点的地址更新路径散列寄存器224。基于入口点的地址更新路径散列寄存器224指示目标计算设备100b已执行了入口点处的代码。例如,目标计算设备100b可以利用入口点的偏移将路径散列寄存器224设置成路径散列寄

寄存器224的先前值的散列。在框516中,为了更新路径散列寄存器224,目标计算设备100b执行具有等于入口点的偏移的偏移参数的标记指令。当执行标记指令时,处理器120使用所供应的参数更新路径散列寄存器224。

[0057] 在框518中,目标计算设备100b执行原子代码路径的代码。目标计算设备100b可以执行受保护代码202的原始代码(例如,应用程序代码)以及被插入在所检测到的受保护代码202'中的检测。在框520中,目标计算设备100b基于原子代码路径中的各个位置处包括的标记指令的地址来更新路径散列寄存器224。例如,目标计算设备100b可以利用原子代码路径中的标记指令的偏移将路径散列寄存器224设置成路径散列寄存器224的先前值的散列。如上面结合图3所描述的,那些标记指令可以跟随在原子代码路径中找到的潜在小工具。基于跟随小工具的地址更新路径散列寄存器224指示目标计算设备100b已经执行了跟随小工具的代码,这继而指示小工具未被作为漏洞利用的一部分而执行。在框522中,为了更新路径散列寄存器224,目标计算设备100b执行具有偏移参数的标记指令,所述偏移参数等于标记指令的偏移。当执行标记指令时,处理器120使用所供应的参数更新路径散列寄存器224。

[0058] 在一些实施例中,在框524中,目标计算设备100b可以将路径散列寄存器224的内容用作用于加密的存储器访问的关键(key)域选择器。与有效代码路径相关联的散列值(即,路径散列寄存器224的内容)可以由存储器访问指令使用,以便将在线性地址空间和应用程序代码中可见的应用程序特定的存储器条目坐标(即,线性关键域选择器)转化成仅为指令集架构和处理器120所知的这些坐标的秘密物理映射(即,物理关键域选择器)。物理关键域选择器可以用作用于以安全方式访问加密的存储器区域的调整值。不遵循有效原子代码路径的代码(诸如执行小工具或其他漏洞利用代码的恶意软件)可能使用正确的线性关键域选择器访问敏感的存储器区域;然而,物理关键域选择器是基于路径散列寄存器224所确定的,并且因此将是不正确的。使用不正确的物理关键域选择器(即,路径散列寄存器224的错误值)访问加密的存储器区域可以使加密的存储器区域被解密成随机明文。

[0059] 在框526中,在一些实施例中,目标计算设备100b可以从原子代码路径内执行新的原子代码路径。例如,目标计算设备100b可以执行子例程调用或者可以进入循环体。响应于执行新的原子代码路径,目标计算设备100b可以执行方法500的另一实例,以框502开始。换言之,当原子代码路径被中断时,可以保存当前路径散列状态,并且可以执行新的原子代码路径。在新的原子代码路径例如通过从子例程返回或退出循环而完成之后,可以恢复先前原子代码路径的执行。在一些实施例中,在框528中,例如响应于使目标计算设备100b执行未对齐的返回或跳转指令的漏洞利用,目标计算设备100b可以执行返回或跳转指令。通过计算设备100b执行的返回或跳转指令检查是否设置了控制流完整性位。如果否,则目标计算设备100b可以生成代码流完整性异常,如下面结合图5B的框550所描述的那样。

[0060] 在框530中,目标计算设备100b确定是否已到达当前原子代码路径的出口点。例如,当从子例程返回、退出循环或以其他方式结束原子代码路径的执行时,可以到达原子代码路径出口点。如上面结合图3所描述的,可以在受保护代码202的静态分析期间标识原子代码路径的出口点,并且因此可以在所检测到的受保护代码202'中静态地编码是否已经到达出口点的确定。如果没有到达出口点,则方法500循环回到框518以继续执行原子代码路径。如果已到达出口点,则方法500前进到图5B中所示的框532。

[0061] 在框532中,目标计算设备100b针对在当前出口点处结束的原子代码路径确定预

期散列值。目标计算设备100b可以通过基于在当前出口点处结束的特定原子代码路径中包括的所有标记指令的偏移来更新评估器散列寄存器228而计算预期散列值。以与将在原子代码路径中执行标记指令相同的顺序更新评估器散列寄存器228。在框534中,为了生成预期散列值,目标计算设备100b针对原子代码路径的每个标记指令执行散列评估器指令。每个散列评估器指令被供应等于对应标记指令的偏移的参数。

[0062] 在框536中,目标计算设备100b将预期散列值与当前路径散列进行比较。目标计算设备100b可以比较路径散列寄存器224和评估器散列寄存器228的值以确定那些值是否匹配。路径散列寄存器224与评估器散列寄存器228匹配指示目标计算设备100b已经以正确的顺序执行了原子代码路径的所有标记指令,其指示原子代码路径被正确地执行。如果路径散列寄存器224和评估器散列寄存器228匹配,则目标计算设备100b可以设置完整性校验位230,并且如果那些值不匹配,则目标计算设备100b可以不改变完整性校验位230的值。例如,目标计算设备100b可以利用完整性校验位230的先前值和寄存器224、228之间的比较结果来执行逻辑或运算。在框538中,为了比较寄存器224、228,目标计算设备100b执行比较评估器指令。

[0063] 在框540中,目标计算设备100b确定是否存在针对当前出口点的附加原子代码路径。如上面结合图3所描述的,可以在受保护代码202中静态地标识所有原子代码路径和出口点。因此,也可以在所检测到的受保护代码202'中静态地编码是否仍有附加原子代码路径的确定。在框542中,目标计算设备100b检查是否仍有附加原子代码路径。如果这样,则方法500循环回到框532以计算和比较附加预期散列值。如果没有仍有附加原子代码路径,则方法500前进到框544。

[0064] 在框544中,目标计算设备100b确定预期散列值中的任一个是否与路径散列匹配;也就是说,目标计算设备100b确定任何比较评估器指令是否指示匹配。目标计算设备100b可以通过确定是否设置了比较完整性校验位230来确定是否找到了匹配。在框546中,为了确定预期散列值中的任一个是否与路径散列匹配,目标计算设备100b执行散列校验指令。在框548中,目标计算设备100b检查是否找到了匹配。如果这样,则方法500向前分支到下面描述的框552。如果否,则方法500前进到框550,其中目标计算设备100b生成代码流完整性异常。目标计算设备100b可以生成代码流完整性错误、异常、中断或以其他方式指示已经发生代码流完整性违规。在一些实施例中,目标计算设备100b可以基于代码流完整性异常执行一个或多个安全响应,诸如终止所检测到的受保护代码202'、报告违规或其他适当的安全响应。在生成代码流完整性异常之后,方法500可前进到框552。

[0065] 在框552中,目标计算设备100b从安全存储位置取回先前路径散列状态。例如,目标计算设备100b可以恢复路径散列寄存器224的先前状态。恢复先前路径散列状态可以允许目标计算设备100b恢复针对先前中断的原子代码路径的代码流完整性校验。在框554中,目标计算设备100b执行退栈指令以从安全堆栈取回先前路径散列值并将该值存储在路径散列寄存器224中。在恢复先前路径散列状态之后,方法500循环回到图5A中示出的框502,以继续针对附加原子代码路径的控制流完整性校验。

[0066] 现在参考图6,在使用中,目标计算设备100b可以执行用于代码流完整性保护的方法600。应当领会,在一些实施例中,方法600的操作可以由如图1中所示的计算设备100的处理器120执行。方法600开始于框602,其中处理器120确定是否执行进栈指令。如果否,则方

法600向前分支到下面描述的框606。如果执行进栈指令,则方法600分支到框604,其中处理器120将路径散列寄存器224的值进栈到安全堆栈上。如上面所描述的,安全堆栈可以体现为被保护免受未经授权访问和/或修改的任何安全存储器位置,诸如加密的存储器页面或处理器120的寄存器文件。在执行进栈指令之后,方法600前进到框606。

[0067] 在框606中,处理器120确定是否执行标记指令。如果否,则方法600向前分支到下面描述的框610。如果执行标记指令,则方法600分支到框608,其中处理器120使用标记指令的参数更新路径散列寄存器224的值。如上面所描述的,该参数可以体现为例如标记指令的偏移。该参数可以被静态地确定,并且例如可以被作为立即值供应给处理器120。处理器120可以使用散列函数来更新路径散列寄存器224,所述散列函数确保没有两组标记导致相同的散列值,但可能不需要高密码强度水平。例如,处理器120可以使用基于乘法器和伽罗瓦域逆变器构建块的散列函数,其可以仅使用少数目的组合逻辑水平来执行所供应的偏移值的位的强混合。作为另一示例,处理器120可以执行少数目(例如,1、3、5或7) AES轮,接着是基于乘法的合并。取决于所花费的芯片面积,可以在一个处理器时钟内完成这样的操作。在执行标记指令之后,方法600前进到框610。

[0068] 在框610中,处理器120确定是否执行评估器散列指令。如果否,则方法600向前分支到下面描述的框614。如果执行评估器散列指令,则方法600分支到框612,其中处理器120使用评估器散列指令的参数更新评估器散列寄存器228的值。如上面所描述的,该参数可以体现为例如对应标记指令的偏移。该参数可以被静态地确定,并且例如可以被作为立即值供应给处理器120。处理器120可以使用由标记指令使用的相同算法更新评估器散列寄存器228,以更新路径散列寄存器224。在执行评估器散列指令之后,方法600前进到框614。

[0069] 在框614中,处理器120确定是否执行比较评估器指令。如果否,则方法600向前分支到下面描述的框622。如果执行比较评估器指令,则方法600分支到框616,其中处理器120将评估器散列寄存器228与路径散列寄存器224进行比较。在框618中,处理器120确定评估器散列寄存器228是否与路径散列寄存器224匹配。如果否,则方法600向前分支到下面描述的框622。如果评估器散列寄存器228与路径散列寄存器224匹配,则方法600前进到框620,其中处理器120设置完整性校验位230。在执行比较评估器指令之后,方法600前进到框622。

[0070] 在框622中,处理器120确定是否执行散列校验指令。如果否,则方法600向前分支到下面描述的框630。如果执行散列校验指令,则方法600分支到框624,其中处理器120确定是否设置完整性校验位230。在框626中,处理器120检查是否设置了完整性校验位230。如果设置,则方法600向前分支到下面描述的框630。如果未设置完整性校验位230,则方法600前进到框628,其中处理器120生成代码流完整性异常。在执行散列校验指令之后,方法600前进到框630。

[0071] 在框630中,处理器120确定是否执行退栈指令。如果否,则方法600循环回到框602以继续处理指令。如果执行退栈指令,则方法600分支到框632,其中处理器120将顶部值从安全堆栈退栈并将该值存储在路径散列寄存器224中。在执行退栈指令之后,方法600循环回到框602以继续处理指令。

[0072] 示例

[0073] 下面提供了本文中公开的技术的说明性示例。所述技术的实施例可以包括下面描述的示例中的任何一个或多个,以及其任何组合。

[0074] 示例1包括一种用于控制流完整性保护的计算设备,所述计算设备包括:静态分析器,其要(i)标识受保护代码的原子代码路径,其中所述原子代码路径包括入口点和出口点,以及(ii)标识所述原子代码路径中的潜在小工具;以及检测管理器,其要(i)在所述原子代码路径中在所述潜在小工具附近插入第一标记指令,所述第一标记指令具有与第一标记指令的地址对应的参数,其中第一标记指令在被执行时使目标计算设备的处理器根据第一标记指令的地址更新所述目标计算设备的路径散列,以及(ii)在所述受保护代码中在所述原子代码路径的出口点之后插入一个或多个控制流完整性评估指令,其中所述控制流完整性评估指令在被执行时使所述目标计算设备的处理器将所述路径散列与预期散列值进行比较。

[0075] 示例2包括示例1的主题,并且其中在所述原子代码路径中在所述潜在小工具附近插入第一标记指令包括在所述原子代码路径中紧接在所述潜在小工具之前或紧接在所述潜在小工具之后插入第一标记指令。

[0076] 示例3包括示例1和2中任一项的主题,并且其中所述原子代码路径包括不具有子例程调用、子例程返回、循环入口或者除了if语句跳转或case语句跳转之外的跳转的代码序列。

[0077] 示例4包括示例1-3中任一项的主题,并且其中标识所述潜在小工具包括标识可由控制流漏洞利用执行的一段原子代码路径。

[0078] 示例5包括示例1-4中任一项的主题,并且其中所述潜在小工具包括面向返回编程小工具或面向跳转编程小工具。

[0079] 示例6包括示例1-5中任一项的主题,并且其中所述潜在小工具包括外部动态链接库调用。

[0080] 示例7包括示例1-6中任一项的主题,并且其中:所述静态分析器进一步要静态地分析所述受保护代码;标识所述原子代码路径包括响应于静态地分析所述受保护代码而标识所述原子代码路径;并且标识所述原子代码路径中的所述潜在小工具包括响应于静态地分析所述受保护代码而标识所述潜在小工具。

[0081] 示例8包括示例1-7中任一项的主题,并且其中插入所述一个或多个控制流完整性评估指令包括:在所述受保护代码中在所述原子代码路径的出口点之后插入第一散列评估器指令,所述第一散列评估器指令具有与第一标记指令的地址对应的参数;在所述受保护代码中在第一散列评估器指令之后插入第一比较评估器指令;以及在所述受保护代码中在第一比较评估器指令之后插入散列校验指令。

[0082] 示例9包括示例1-8中任一项的主题,并且其中:所述静态分析器进一步要标识所述原子代码路径中的第二潜在小工具,其中第二潜在小工具发生在所述原子代码路径中的第一潜在小工具之后;所述检测管理器进一步要(i)在所述原子代码路径中在第二潜在小工具附近插入第二标记指令,所述第二标记指令具有与第二标记指令的地址对应的参数,以及(ii)在所述受保护代码中在第一散列评估器指令之后插入第二散列评估器指令,所述第二散列评估器指令具有与第二标记指令的地址对应的参数;并且插入第一比较评估器指令进一步包括在第二散列评估器指令之后插入第一比较评估器指令。

[0083] 示例10包括示例1-9中任一项的主题,并且其中:所述静态分析器进一步要:(i)标识所述受保护代码的第二原子代码路径,其中第二原子代码路径包括入口点和出口点,并

且其中所述原子代码路径的出口点包括第二原子代码路径的出口点,以及(ii)标识第二原子代码路径中的第二潜在小工具;所述检测管理器进一步要(i)在第二原子代码路径中在第二潜在小工具附近插入第二标记指令,所述第二标记指令具有与第二标记指令的地址对应的参数,(ii)在所述受保护代码中在第一比较评估器指令之后插入第二散列评估器指令,所述第二散列评估器指令具有与第二标记指令的地址对应的参数,以及(iii)在所述受保护代码中在第二散列评估器指令之后插入第二比较评估器指令;并且插入所述散列校验指令进一步包括在第二比较评估器指令之后插入所述散列校验指令。

[0084] 示例11包括示例1-10中任一项的主题,并且其中所述检测管理器进一步要:在所述原子代码路径中在所述原子代码路径的入口点处插入第二标记指令,所述第二标记指令具有与第二标记指令的地址对应的参数;以及在所述受保护代码中在所述出口点之后并且在第一散列评估器指令之前插入第二散列评估器指令,所述第二散列评估器指令具有与第二标记指令的地址对应的参数。

[0085] 示例12包括示例1-11中任一项的主题,并且其中:所述静态分析器进一步要标识跳转到所述原子代码路径的入口点的所述受保护代码的非返回跳转;并且所述检测管理器进一步要(i)在所述受保护代码中在所述非返回跳转之前插入第二标记指令,所述第二标记指令具有与第二标记指令的地址对应的参数,以及(ii)在所述受保护代码中在所述出口点之后并且在第一散列评估器指令之前插入第二散列评估器指令,所述第二散列评估器指令具有与第二标记指令的地址对应的参数。

[0086] 示例13包括示例1-12中任一项的主题,并且其中:插入第一散列评估器指令包括插入在被执行时使所述处理器根据第一标记指令的地址来更新所述目标计算设备的所述预期散列值的第一散列评估器指令;插入第一比较评估器指令包括插入在被执行时使所述处理器确定所述路径散列是否与所述预期散列值匹配的第一比较评估器指令;并且插入所述散列校验指令包括插入在被执行时使所述处理器响应于所述路径散列与所述预期散列值不匹配的确定而生成代码流完整性异常的散列校验指令。

[0087] 示例14包括示例1-13中任一项的主题,并且其中所述检测管理器进一步要:在所述受保护代码中在所述原子代码路径的入口点处插入进栈指令;并且在所述受保护代码中在所述散列校验指令之后插入退栈指令。

[0088] 示例15包括示例1-14中任一项的主题,并且其中:插入所述进栈指令包括插入在被执行时使目标计算设备的处理器将所述目标计算设备的路径散列保存到所述目标计算设备的安全存储位置的进栈指令;并且插入所述退栈指令包括插入在被执行时使所述处理器从所述安全存储位置恢复所述路径散列的退栈指令。

[0089] 示例16包括一种用于控制流完整性保护的计算设备,所述计算设备包括:路径散列更新器,其要(i)执行受保护代码的原子代码路径,其中所述原子代码路径包括入口点、出口点和一个或多个标记指令,(ii)响应于所述原子代码路径的执行,根据所述一个或多个标记指令中的第一标记指令的地址更新路径散列,以及(iii)响应于所述原子代码路径的执行,到达所述原子代码路径的出口点;散列评估器,其要(i)响应于到达所述出口点,根据一个或多个地址生成预期散列值,其中所述一个或多个地址中的每个对应于所述一个或多个标记指令中的标记指令,以及(ii)确定所述路径散列是否与所述预期散列值匹配;以及代码流完整性校验器,其要响应于所述路径散列与所述预期散列值不匹配的确定而生成

代码流完整性异常。

[0090] 示例17包括示例16的主题,并且其中:所述计算设备进一步包括处理器;更新所述路径散列包括通过所述处理器执行所述原子代码路径的第一标记指令,所述第一标记指令具有与第一标记指令的地址对应的参数;生成所述预期散列值包括通过所述处理器执行一个或多个散列评估器指令的组,其中一个或多个散列评估器指令的所述组的每个散列评估器指令对应于所述一个或多个标记指令中的标记指令;确定所述路径散列是否与所述预期散列值匹配包括通过所述处理器执行比较评估器指令;并且响应于所述路径散列与所述预期散列不匹配的确定而生成所述代码流完整性异常包括通过所述处理器执行散列校验指令。

[0091] 示例18包括示例16和17中任一项的主题,并且其中:根据第一标记指令的地址更新所述路径散列包括根据所述受保护代码中的第一标记指令的偏移更新所述路径散列;并且根据所述一个或多个地址生成所述预期散列值包括根据一个或多个偏移生成所述预期散列值,其中所述一个或多个偏移中的每个对应于所述一个或多个标记指令中的标记指令的偏移。

[0092] 示例19包括示例16-18中任一项的主题,并且其中所述原子代码路径包括不具有子例程调用、子例程返回、循环入口或者除了if语句跳转或case语句跳转之外的跳转的代码序列。

[0093] 示例20包括示例16-19中任一项的主题,并且其中:所述计算设备进一步包括路径散列寄存器和评估器散列寄存器;更新所述路径散列包括将所述路径散列存储在所述路径散列寄存器中;生成所述预期散列值包括将所述预期散列值存储在所述评估器散列寄存器中;并且确定所述路径散列是否与所述预期散列值匹配包括比较所述路径散列寄存器和所述评估器散列寄存器。

[0094] 示例21包括示例16-20中任一项的主题,并且其中:所述计算设备进一步包括完整性校验位;并且确定所述路径散列是否与所述预期散列值匹配包括响应于所述路径散列与所述预期散列匹配的确定而设置所述完整性校验位。

[0095] 示例22包括示例16-21中任一项的主题,并且其中:所述受保护代码包括第二原子代码路径,所述第二原子代码路径包括所述出口点和一个或多个标记指令的第二集合;所述散列评估器进一步要(i)响应于到达所述出口点,根据一个或多个地址的第二集合生成第二预期散列值,其中一个或多个地址的第二集合的每个地址对应于第二原子代码路径的一个或多个标记指令的第二集合的标记指令,以及(ii)确定所述路径散列是否与第二预期散列值匹配;并且生成所述代码流完整性异常包括响应于所述路径散列与所述预期散列值或第二预期散列值不匹配的确定而生成所述代码流完整性异常。

[0096] 示例23包括示例16-22中任一项的主题,并且其中:所述计算设备进一步包括处理器;生成所述预期散列值包括通过所述处理器执行一个或多个散列评估器指令的第一组,其中第一组的每个散列评估器指令对应于所述原子代码路径的所述一个或多个标记指令中的标记指令;确定所述路径散列是否与所述预期散列值匹配包括通过所述处理器执行比较评估器指令;生成第二预期散列值包括通过所述处理器执行一个或多个散列评估器指令的第二组,其中第二组的每个散列评估器指令对应于第二原子代码路径的一个或多个标记指令的第二集合的标记指令;确定所述路径散列是否与第二预期散列匹配包括通过所述处

理器执行第二比较评估器指令;并且生成所述代码流完整性异常包括通过所述处理器执行散列校验指令。

[0097] 示例24包括示例16-23中任一项的主题,并且进一步包括:状态管理器,其要(i)将先前路径散列保存在所述计算设备的安全存储位置中,以及(ii)响应于所述路径散列与所述预期散列值匹配的确定,从所述安全存储位置恢复所述先前路径散列;其中执行所述原子代码路径包括响应于所述先前路径散列的保存而执行所述原子代码路径。

[0098] 示例25包括示例16-24中任一项的主题,并且其中:所述计算设备进一步包括处理器;保存所述先前路径散列包括通过所述处理器执行进栈指令以将所述先前路径散列存储在安全堆栈中;并且恢复所述先前路径散列包括通过所述处理器执行退栈指令以从所述安全堆栈取回所述先前路径散列。

[0099] 示例26包括示例16-25中任一项的主题,并且其中所述安全存储位置包括加密的存储器页面。

[0100] 示例27包括示例16-26中任一项的主题,并且其中所述安全存储位置包括所述计算设备的处理器的安全寄存器文件。

[0101] 示例28包括示例16-27中任一项的主题,并且进一步包括存储器管理器,其要(i)响应于所述路径散列的更新,根据所述路径散列生成关键域选择器,以及(ii)利用所述关键域选择器执行加密的存储器访问。

[0102] 示例29包括一种用于控制流完整性保护的方法,所述方法包括:通过计算设备标识受保护代码的原子代码路径,其中所述原子代码路径包括入口点和出口点;通过所述计算设备标识所述原子代码路径中的潜在小工具;通过所述计算设备在所述原子代码路径中在所述潜在小工具附近插入第一标记指令,所述第一标记指令具有与第一标记指令的地址对应的参数,其中第一标记指令在被执行时使目标计算设备的处理器根据第一标记指令的地址更新所述目标计算设备的路径散列;以及通过所述计算设备在所述受保护代码中在所述原子代码路径的出口点之后插入一个或多个控制流完整性评估指令,其中所述控制流完整性评估指令在被执行时使所述目标计算设备的处理器将所述路径散列与预期散列值进行比较。

[0103] 示例30包括示例29的主题,并且其中在所述原子代码路径中在所述潜在小工具附近插入第一标记指令包括在所述原子代码路径中紧接在所述潜在小工具之前或紧接在所述潜在小工具之后插入第一标记指令。

[0104] 示例31包括示例29和30中任一项的主题,并且其中所述原子代码路径包括不具有子例程调用、子例程返回、循环入口或者除了if语句跳转或case语句跳转之外的跳转的代码序列。

[0105] 示例32包括示例29-31中任一项的主题,并且其中标识所述潜在小工具包括标识可由控制流漏洞利用执行的一段原子代码路径。

[0106] 示例33包括示例29-32中任一项的主题,并且其中所述潜在小工具包括面向返回编程小工具或面向跳转编程小工具。

[0107] 示例34包括示例29-33中任一项的主题,并且其中所述潜在小工具包括外部动态链接库调用。

[0108] 示例35包括示例29-34中任一项的主题,并且进一步包括:通过所述计算设备静态

地分析所述受保护代码;其中标识所述原子代码路径包括响应于静态地分析所述受保护代码而标识所述原子代码路径;并且其中标识所述原子代码路径中的潜在小工具包括响应于静态地分析所述受保护代码而标识所述潜在小工具。

[0109] 示例36包括示例29-35中任一项的主题,并且其中插入所述一个或多个控制流完整性评估指令包括:在所述受保护代码中在所述原子代码路径的出口点之后插入第一散列评估器指令,所述第一散列评估器指令具有与第一标记指令的地址对应的参数;在所述受保护代码中在第一散列评估器指令之后插入第一比较评估器指令;以及在所述受保护代码中在第一比较评估器指令之后插入散列校验指令。

[0110] 示例37包括示例29-36中任一项的主题,并且进一步包括:通过所述计算设备标识所述原子代码路径中的第二潜在小工具,其中第二潜在小工具发生在所述原子代码路径中的第一潜在小工具之后;通过所述计算设备在所述原子代码路径中在第二潜在小工具附近插入第二标记指令,所述第二标记指令具有与第二标记指令的地址对应的参数;以及通过所述计算设备在所述受保护代码中在第一散列评估器指令之后插入第二散列评估器指令,所述第二散列评估器指令具有与第二标记指令的地址对应的参数;其中插入第一比较评估器指令进一步包括在第二散列评估器指令之后插入第一比较评估器指令。

[0111] 示例38包括示例29-36中任一项的主题,并且进一步包括:通过所述计算设备标识所述受保护代码的第二原子代码路径,其中第二原子代码路径包括入口点和出口点,并且其中所述原子代码路径的出口点包括第二原子代码路径的出口点;通过所述计算设备标识第二原子代码路径中的第二潜在小工具;通过所述计算设备在第二原子代码路径中在第二潜在小工具附近插入第二标记指令,所述第二标记指令具有与第二标记指令的地址对应的参数;通过所述计算设备在所述受保护代码中在第一比较评估器指令之后插入第二散列评估器指令,所述第二散列评估器指令具有与第二标记指令的地址对应的参数;以及通过所述计算设备在所述受保护代码中在第二散列评估器指令之后插入第二比较评估器指令;其中插入所述散列校验指令进一步包括在第二比较评估器指令之后插入所述散列校验指令。

[0112] 示例39包括示例29-37中任一项的主题,并且进一步包括:通过所述计算设备在所述原子代码路径中在所述原子代码路径的入口点处插入第二标记指令,所述第二标记指令具有与第二标记指令的地址对应的参数;以及通过所述计算设备在所述受保护代码中在所述出口点之后并且在第一散列评估器指令之前插入第二散列评估器指令,所述第二散列评估器指令具有与第二标记指令的地址对应的参数。

[0113] 示例40包括示例29-39中任一项的主题,并且进一步包括:通过所述计算设备标识跳转到所述原子代码路径的入口点的所述受保护代码的非返回跳转;通过所述计算设备在所述受保护代码中在所述非返回跳转之前插入第二标记指令,所述第二标记指令具有与第二标记指令的地址对应的参数;以及通过所述计算设备在所述受保护代码中在所述出口点之后并且在第一散列评估器指令之前插入第二散列评估器指令,所述第二散列评估器指令具有与第二标记指令的地址对应的参数。

[0114] 示例41包括示例29-40中任一项的主题,并且其中:插入第一散列评估器指令包括插入在被执行时使所述处理器根据第一标记指令的地址来更新所述目标计算设备的所述预期散列值的第一散列评估器指令;插入第一比较评估器指令包括插入在被执行时使所述处理器确定所述路径散列是否与所述预期散列值匹配的第一比较评估器指令;并且插入所

述散列校验指令包括插入在被执行时使所述处理器响应于所述路径散列与所述预期散列值不匹配的确定而生成代码流完整性异常的散列校验指令。

[0115] 示例42包括示例29-41中任一项的主题,并且进一步包括:通过所述计算设备在所述受保护代码中在所述原子代码路径的入口点处插入进栈指令;以及通过所述计算设备在所述受保护代码中在所述散列校验指令之后插入退栈指令。

[0116] 示例43包括示例29-42中任一项的主题,并且其中:插入所述进栈指令包括插入在被执行时使目标计算设备的处理器将所述目标计算设备的路径散列保存到所述目标计算设备的安全存储位置的进栈指令;并且插入所述退栈指令包括插入在被执行时使所述处理器从所述安全存储位置恢复所述路径散列的退栈指令。

[0117] 示例44包括一种用于控制流完整性保护的方法,所述方法包括:通过计算设备执行受保护代码的原子代码路径,其中所述原子代码路径包括入口点、出口点和一个或多个标记指令;响应于执行所述原子代码路径,通过所述计算设备根据所述一个或多个标记指令中的第一标记指令的地址更新路径散列;响应于执行所述原子代码路径,通过所述计算设备到达所述原子代码路径的出口点;响应于到达所述出口点,通过所述计算设备根据一个或多个地址生成预期散列值,其中所述一个或多个地址中的每个对应于所述一个或多个标记指令中的标记指令;通过所述计算设备确定所述路径散列是否与所述预期散列值匹配;以及响应于确定所述路径散列与所述预期散列值不匹配,通过所述计算设备生成代码流完整性异常。

[0118] 示例45包括示例44的主题,并且其中:更新所述路径散列包括通过所述计算设备的处理器执行所述原子代码路径的第一标记指令,所述第一标记指令具有与第一标记指令的地址对应的参数;生成所述预期散列值包括通过所述处理器执行一个或多个散列评估器指令的组,其中一个或多个散列评估器指令的所述组的每个散列评估器指令对应于所述一个或多个标记指令中的标记指令;确定所述路径散列是否与所述预期散列值匹配包括通过所述处理器执行比较评估器指令;并且响应于确定所述路径散列与所述预期散列不匹配而生成所述代码流完整性异常包括通过所述处理器执行散列校验指令。

[0119] 示例46包括示例44和45中任一项的主题,并且其中:根据第一标记指令的地址更新所述路径散列包括根据所述受保护代码中的第一标记指令的偏移更新所述路径散列;并且根据所述一个或多个地址生成所述预期散列值包括根据一个或多个偏移生成所述预期散列值,其中所述一个或多个偏移中的每个对应于所述一个或多个标记指令中的标记指令的偏移。

[0120] 示例47包括示例44-46中任一项的主题,并且其中所述原子代码路径包括不具有子例程调用、子例程返回、循环入口或者除了if语句跳转或case语句跳转之外的跳转的代码序列。

[0121] 示例48包括示例44-47中任一项的主题,并且其中:更新所述路径散列包括将所述路径散列存储在所述计算设备的路径散列寄存器中;生成所述预期散列值包括将所述预期散列值存储在所述计算设备的评估器散列寄存器中;并且确定所述路径散列是否与所述预期散列值匹配包括比较所述路径散列寄存器和所述评估器散列寄存器。

[0122] 示例49包括示例44-48中任一项的主题,并且其中确定所述路径散列是否与所述预期散列值匹配包括响应于确定所述路径散列与所述预期散列匹配而设置所述计算设备

的完整性校验位。

[0123] 示例50包括示例44-49中任一项的主题,并且其中所述受保护代码包括第二原子代码路径,所述第二原子代码路径包括所述出口点和一个或多个标记指令的第二集合;所述方法进一步包括:响应于到达所述出口点,通过所述计算设备根据一个或多个地址的第二集合生成第二预期散列值,其中一个或多个地址的第二集合的每个地址对应于第二原子代码路径的一个或多个标记指令的第二集合的标记指令;以及通过所述计算设备确定所述路径散列是否与第二预期散列值匹配;其中生成所述代码流完整性异常包括响应于确定所述路径散列与所述预期散列值或第二预期散列值不匹配而生成所述代码流完整性异常。

[0124] 示例51包括示例44-50中任一项的主题,并且其中:生成所述预期散列值包括通过所述计算设备的处理器执行一个或多个散列评估器指令的第一组,其中第一组的每个散列评估器指令对应于所述原子代码路径的所述一个或多个标记指令中的标记指令;确定所述路径散列是否与所述预期散列值匹配包括通过所述处理器执行比较评估器指令;生成第二预期散列值包括通过所述处理器执行一个或多个散列评估器指令的第二组,其中第二组的每个散列评估器指令对应于第二原子代码路径的一个或多个标记指令的第二集合的标记指令;确定所述路径散列是否与第二预期散列匹配包括通过所述处理器执行第二比较评估器指令;并且生成所述代码流完整性异常包括通过所述处理器执行散列校验指令。

[0125] 示例52包括示例44-51中任一项的主题,并且进一步包括:通过所述计算设备将先前路径散列保存在所述计算设备的安全存储位置中;以及响应于确定所述路径散列与所述预期散列值匹配,通过所述计算设备从所述安全存储位置恢复所述先前路径散列;其中执行所述原子代码路径包括响应于保存所述先前路径散列而执行所述原子代码路径。

[0126] 示例53包括示例44-52中任一项的主题,并且其中:保存所述先前路径散列包括通过所述计算设备的处理器执行进栈指令以将所述先前路径散列存储在安全堆栈中;并且恢复所述先前路径散列包括通过所述处理器执行退栈指令以从所述安全堆栈取回所述先前路径散列。

[0127] 示例54包括示例44-53中任一项的主题,并且其中所述安全存储位置包括加密的存储器页面。

[0128] 示例55包括示例44-54中任一项的主题,并且其中所述安全存储位置包括所述计算设备的处理器的安全寄存器文件。

[0129] 示例56包括示例44-55中任一项的主题,并且进一步包括:响应于更新所述路径散列,通过所述计算设备根据所述路径散列生成关键域选择器;以及通过所述计算设备使用所述关键域选择器来执行加密的存储器访问。

[0130] 示例57包括一种计算设备,所述计算设备包括:处理器;以及存储器,其具有存储在其中的多个指令,所述多个指令在被所述处理器执行时使所述计算设备实行示例29-56中任一项的方法。

[0131] 示例58包括一种或多种机器可读存储介质,其包括存储在其上的多个指令,所述多个指令响应于被执行而导致计算设备实行示例29-56中任一项的方法。

[0132] 示例59包括一种计算设备,其包括用于执行示例29-56中任一项的方法的部件。

[0133] 示例60包括一种用于控制流完整性保护的计算设备,所述计算设备包括:用于标识受保护代码的原子代码路径的部件,其中所述原子代码路径包括入口点和出口点;用于

标识所述原子代码路径中的潜在小工具的部件；用于在所述原子代码路径中在所述潜在小工具附近插入第一标记指令的部件，所述第一标记指令具有与第一标记指令的地址对应的参数，其中第一标记指令在被执行时使目标计算设备的处理器根据第一标记指令的地址更新所述目标计算设备的路径散列；以及用于在所述受保护代码中在所述原子代码路径的出口点之后插入一个或多个控制流完整性评估指令的部件，其中所述控制流完整性评估指令在被执行时使所述目标计算设备的处理器将所述路径散列与预期散列值进行比较。

[0134] 示例61包括示例60的主题，并且其中用于在所述原子代码路径中在所述潜在小工具附近插入第一标记指令的部件包括用于在所述原子代码路径中紧接在所述潜在小工具之前或紧接在所述潜在小工具之后插入第一标记指令的部件。

[0135] 示例62包括示例60和61中任一项的主题，并且其中所述原子代码路径包括不具有子例程调用、子例程返回、循环入口或者除了if语句跳转或case语句跳转之外的跳转的代码序列。

[0136] 示例63包括示例60-62中任一项的主题，并且其中用于标识所述潜在小工具的部件包括用于标识可由控制流漏洞利用执行的一段原子代码路径的部件。

[0137] 示例64包括示例60-63中任一项的主题，并且其中所述潜在小工具包括面向返回编程小工具或面向跳转编程小工具。

[0138] 示例65包括示例60-64中任一项的主题，并且其中所述潜在小工具包括外部动态链接库调用。

[0139] 示例66包括示例60-65中任一项的主题，并且进一步包括：用于静态地分析所述受保护代码的部件；其中用于标识所述原子代码路径的部件包括用于响应于静态地分析所述受保护代码而标识所述原子代码路径的部件；并且其中用于标识所述原子代码路径中的潜在小工具的部件包括用于响应于静态地分析所述受保护代码而标识所述潜在小工具的部件。

[0140] 示例67包括示例60-66中任一项的主题，并且其中用于插入所述一个或多个控制流完整性评估指令的部件包括：用于在所述受保护代码中在所述原子代码路径的出口点之后插入第一散列评估器指令的部件，所述第一散列评估器指令具有与第一标记指令的地址对应的参数；用于在所述受保护代码中在第一散列评估器指令之后插入第一比较评估器指令的部件；以及用于在所述受保护代码中在第一比较评估器指令之后插入散列校验指令的部件。

[0141] 示例68包括示例60-67中任一项的主题，并且进一步包括：用于标识所述原子代码路径中的第二潜在小工具的部件，其中第二潜在小工具发生在所述原子代码路径中的第一潜在小工具之后；用于在所述原子代码路径中在第二潜在小工具附近插入第二标记指令的部件，所述第二标记指令具有与第二标记指令的地址对应的参数；以及用于在所述受保护代码中在第一散列评估器指令之后插入第二散列评估器指令的部件，所述第二散列评估器指令具有与第二标记指令的地址对应的参数；其中用于插入第一比较评估器指令进一步的部件包括用于在第二散列评估器指令之后插入第一比较评估器指令的部件。

[0142] 示例69包括示例60-68中任一项的主题，并且进一步包括：用于标识所述受保护代码的第二原子代码路径的部件，其中第二原子代码路径包括入口点和出口点，并且其中所述原子代码路径的出口点包括第二原子代码路径的出口点；用于标识第二原子代码路径中

的第二潜在小工具的部件;用于在第二原子代码路径中在第二潜在小工具附近插入第二标记指令的部件,所述第二标记指令具有与第二标记指令的地址对应的参数;用于在所述受保护代码中在第一比较评估器指令之后插入第二散列评估器指令的部件,所述第二散列评估器指令具有与第二标记指令的地址对应的参数;以及用于在所述受保护代码中在第二散列评估器指令之后插入第二比较评估器指令的部件;其中用于插入所述散列校验指令进一步的部件包括用于在第二比较评估器指令之后插入所述散列校验指令的部件。

[0143] 示例70包括示例60-69中任一项的主题,并且进一步包括:用于在所述原子代码路径中在所述原子代码路径的入口点处插入第二标记指令的部件,所述第二标记指令具有与第二标记指令的地址对应的参数;以及用于在所述受保护代码中在所述出口点之后并且在第一散列评估器指令之前插入第二散列评估器指令的部件,所述第二散列评估器指令具有与第二标记指令的地址对应的参数。

[0144] 示例71包括示例60-70中任一项的主题,并且进一步包括:用于标识跳转到所述原子代码路径的入口点的所述受保护代码的非返回跳转的部件;用于在所述受保护代码中在所述非返回跳转之前插入第二标记指令的部件,所述第二标记指令具有与第二标记指令的地址对应的参数;以及用于在所述受保护代码中在所述出口点之后并且在第一散列评估器指令之前插入第二散列评估器指令的部件,所述第二散列评估器指令具有与第二标记指令的地址对应的参数。

[0145] 示例72包括示例60-71中任一项的主题,并且其中:用于插入第一散列评估器指令的部件包括用于插入在被执行时使所述处理器根据第一标记指令的地址来更新所述目标计算设备的所述预期散列值的第一散列评估器指令的部件;用于插入第一比较评估器指令的部件包括用于插入在被执行时使所述处理器确定所述路径散列是否与所述预期散列值匹配的第一比较评估器指令的部件;并且用于插入所述散列校验指令的部件包括用于插入在被执行时使所述处理器响应于所述路径散列与所述预期散列值不匹配的确定而生成代码流完整性异常的散列校验指令的部件。

[0146] 示例73包括示例60-72中任一项的主题,并且进一步包括:用于在所述受保护代码中在所述原子代码路径的入口点处插入进栈指令的部件;以及用于在所述受保护代码中在所述散列校验指令之后插入退栈指令的部件。

[0147] 示例74包括示例60-73中任一项的主题,并且其中:用于插入所述进栈指令的部件包括用于插入在被执行时使目标计算设备的处理器将所述目标计算设备的路径散列保存到所述目标计算设备的安全存储位置的进栈指令的部件;并且用于插入所述退栈指令的部件包括用于插入在被执行时使所述处理器从所述安全存储位置恢复所述路径散列的退栈指令的部件。

[0148] 示例75包括一种用于控制流完整性保护的计算设备,所述计算设备包括:用于执行受保护代码的原子代码路径的部件,其中所述原子代码路径包括入口点、出口点和一个或多个标记指令;用于响应于执行所述原子代码路径而根据所述一个或多个标记指令中的第一标记指令的地址更新路径散列的部件;用于响应于执行所述原子代码路径而到达所述原子代码路径的出口点的部件;用于响应于到达所述出口点而根据一个或多个地址生成预期散列值的部件,其中所述一个或多个地址中的每个对应于所述一个或多个标记指令中的标记指令;用于确定所述路径散列是否与所述预期散列值匹配的部件;以及用于响应于确

定所述路径散列与所述预期散列值不匹配而生成代码流完整性异常的部件。

[0149] 示例76包括示例75的主题,并且其中:用于更新所述路径散列的部件包括用于通过所述计算设备的处理器执行所述原子代码路径的第一标记指令的部件,所述第一标记指令具有与第一标记指令的地址对应的参数;用于生成所述预期散列值的部件包括用于通过所述处理器执行一个或多个散列评估器指令的组的部件,其中一个或多个散列评估器指令的所述组的每个散列评估器指令对应于所述一个或多个标记指令中的标记指令;用于确定所述路径散列是否与所述预期散列值匹配的部件包括用于通过所述处理器执行比较评估器指令的部件;并且用于响应于确定所述路径散列与所述预期散列不匹配而生成所述代码流完整性异常的部件包括用于通过所述处理器执行散列校验指令的部件。

[0150] 示例77包括示例75和76中任一项的主题,并且其中:用于根据第一标记指令的地址更新所述路径散列的部件包括用于根据所述受保护代码中的第一标记指令的偏移更新所述路径散列的部件;并且用于根据所述一个或多个地址生成所述预期散列值的部件包括用于根据一个或多个偏移生成所述预期散列值的部件,其中所述一个或多个偏移中的每个对应于所述一个或多个标记指令中的标记指令的偏移。

[0151] 示例78包括示例75-77中任一项的主题,并且其中所述原子代码路径包括不具有子例程调用、子例程返回、循环入口或者除了if语句跳转或case语句跳转之外的跳转的代码序列。

[0152] 示例79包括示例75-78中任一项的主题,并且其中:用于更新所述路径散列的部件包括用于将所述路径散列存储在所述计算设备的路径散列寄存器中的部件;用于生成所述预期散列值的部件包括用于将所述预期散列值存储在所述计算设备的评估器散列寄存器中的部件;并且用于确定所述路径散列是否与所述预期散列值匹配的部件包括用于比较所述路径散列寄存器和所述评估器散列寄存器的部件。

[0153] 示例80包括示例75-79中任一项的主题,并且其中用于确定所述路径散列是否与所述预期散列值匹配的部件包括用于响应于确定所述路径散列与所述预期散列匹配而设置所述计算设备的完整性校验位的部件。

[0154] 示例81包括示例75-80中任一项的主题,并且其中所述受保护代码包括第二原子代码路径,所述第二原子代码路径包括所述出口点和一个或多个标记指令的第二集合;所述计算设备进一步包括:用于响应于到达所述出口点而根据一个或多个地址的第二集合生成第二预期散列值的部件,其中一个或多个地址的第二集合的每个地址对应于第二原子代码路径的一个或多个标记指令的第二集合的标记指令;以及用于确定所述路径散列是否与第二预期散列值匹配的部件;其中用于生成所述代码流完整性异常的部件包括用于响应于确定所述路径散列与所述预期散列值或第二预期散列值不匹配而生成所述代码流完整性异常的部件。

[0155] 示例82包括示例75-81中任一项的主题,并且其中:用于生成所述预期散列值的部件包括用于通过所述计算设备的处理器执行一个或多个散列评估器指令的第一组的部件,其中第一组的每个散列评估器指令对应于所述原子代码路径的所述一个或多个标记指令中的标记指令;用于确定所述路径散列是否与所述预期散列值匹配的部件包括用于通过所述处理器执行比较评估器指令的部件;用于生成第二预期散列值的部件包括用于通过所述处理器执行一个或多个散列评估器指令的第二组的部件,其中第二组的每个散列评估器指

令对应于第二原子代码路径的一个或多个标记指令的第二集合的标记指令;用于确定所述路径散列是否与第二预期散列匹配的部件包括用于通过所述处理器执行第二比较评估器指令的部件;并且用于生成所述代码流完整性异常的部件包括用于通过所述处理器执行散列校验指令的部件。

[0156] 示例83包括示例75-82中任一项的主题,并且进一步包括:用于将先前路径散列保存在所述计算设备的安全存储位置中的部件;以及用于响应于确定所述路径散列与所述预期散列值匹配而从所述安全存储位置恢复所述先前路径散列的部件;其中用于执行所述原子代码路径的部件包括用于响应于保存所述先前路径散列而执行所述原子代码路径的部件。

[0157] 示例84包括示例75-83中任一项的主题,并且其中:用于保存所述先前路径散列的部件包括用于通过所述计算设备的处理器执行进栈指令以将所述先前路径散列存储在安全堆栈中的部件;并且用于恢复所述先前路径散列的部件包括用于通过所述处理器执行退栈指令以从所述安全堆栈取回所述先前路径散列的部件。

[0158] 示例85包括示例75-84中任一项的主题,并且其中所述安全存储位置包括加密的存储器页面。

[0159] 示例86包括示例75-85中任一项的主题,并且其中所述安全存储位置包括所述计算设备的处理器的安全寄存器文件。

[0160] 示例87包括示例75-86中任一项的主题,并且进一步包括:用于响应于更新所述路径散列而根据所述路径散列生成关键域选择器的部件;以及用于使用所述关键域选择器来执行加密的存储器访问的部件。

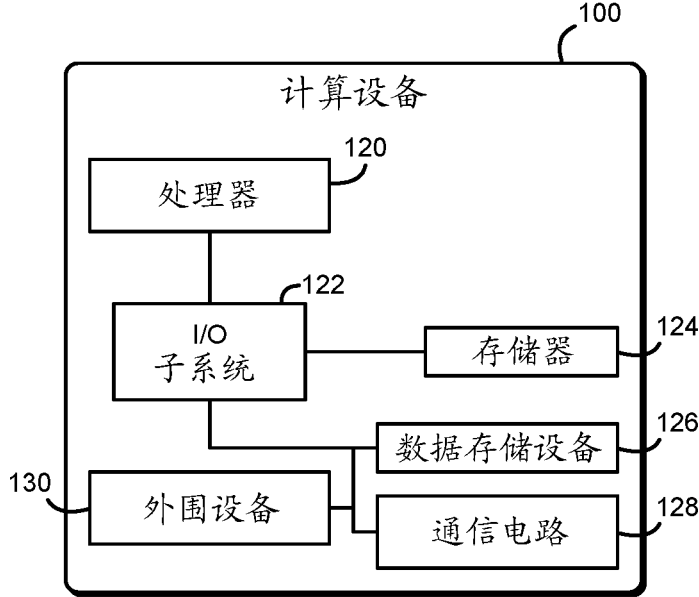


图 1

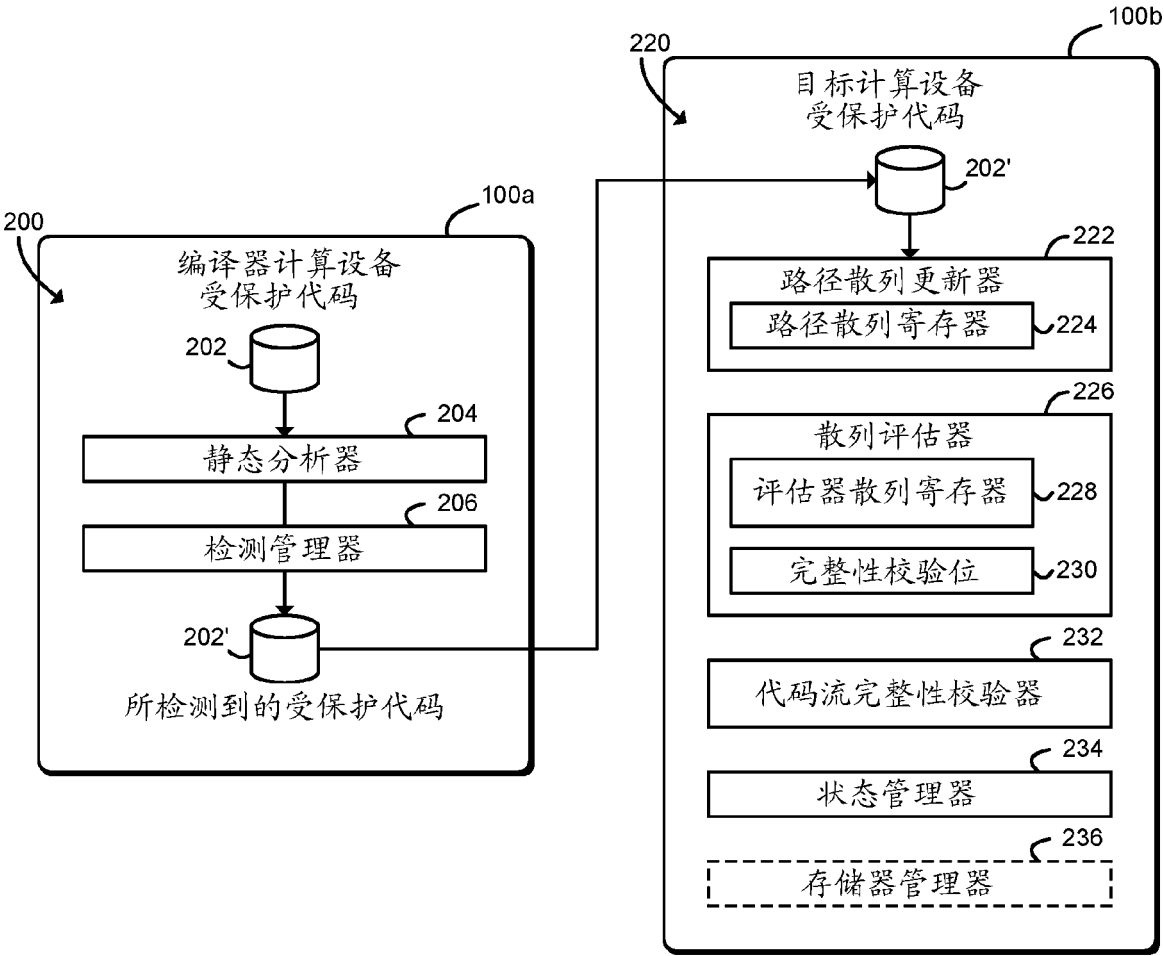


图 2

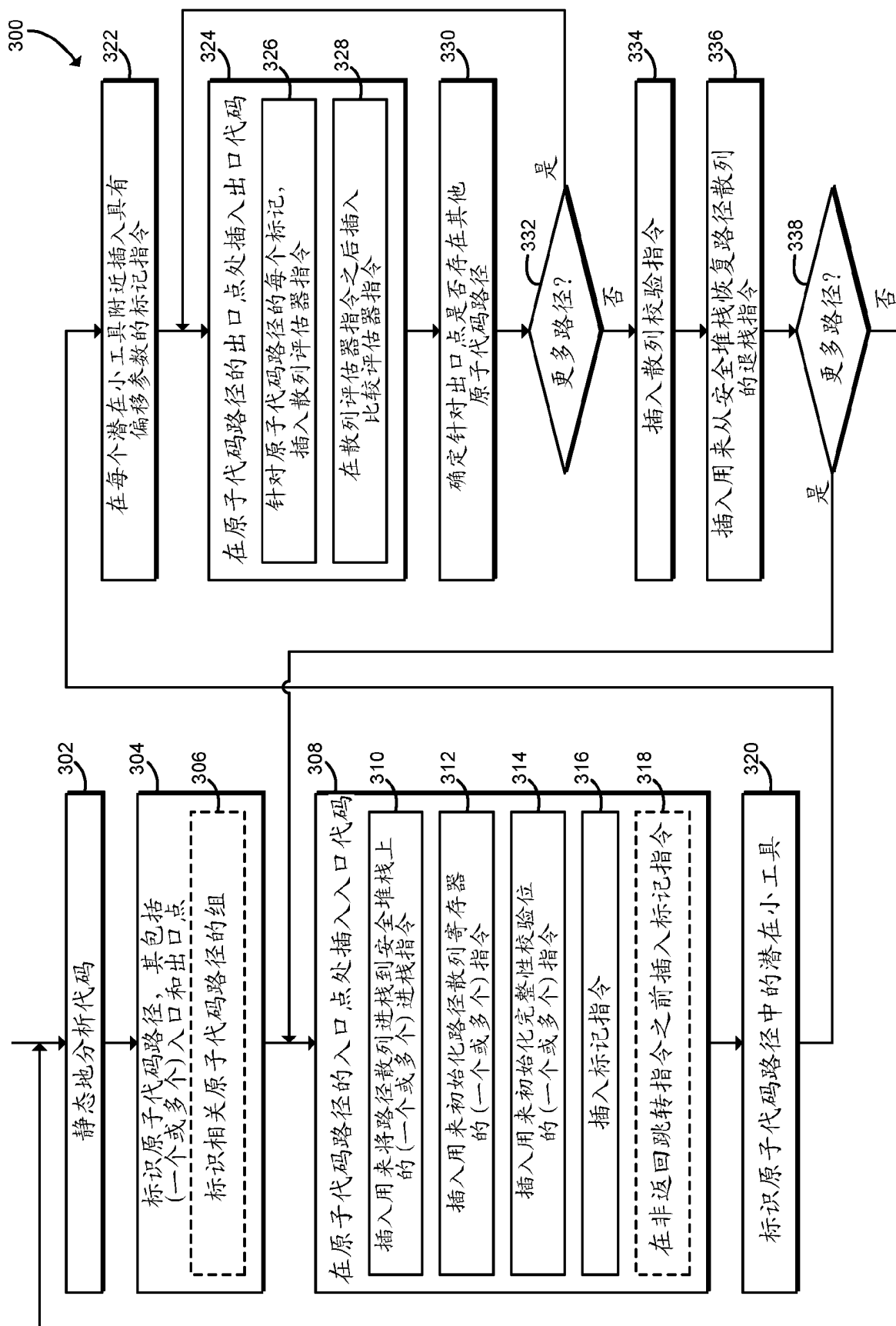


图 3

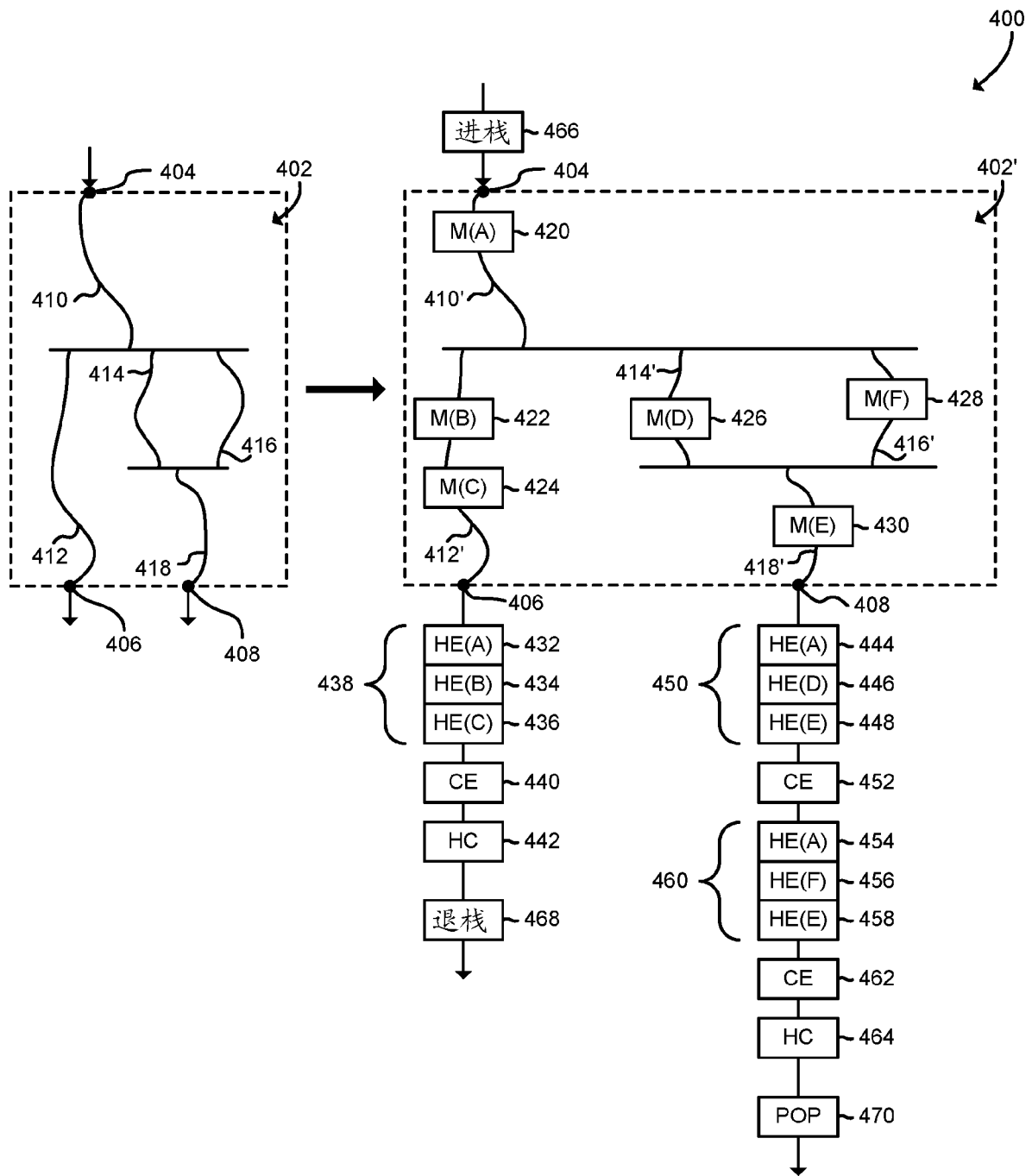


图 4

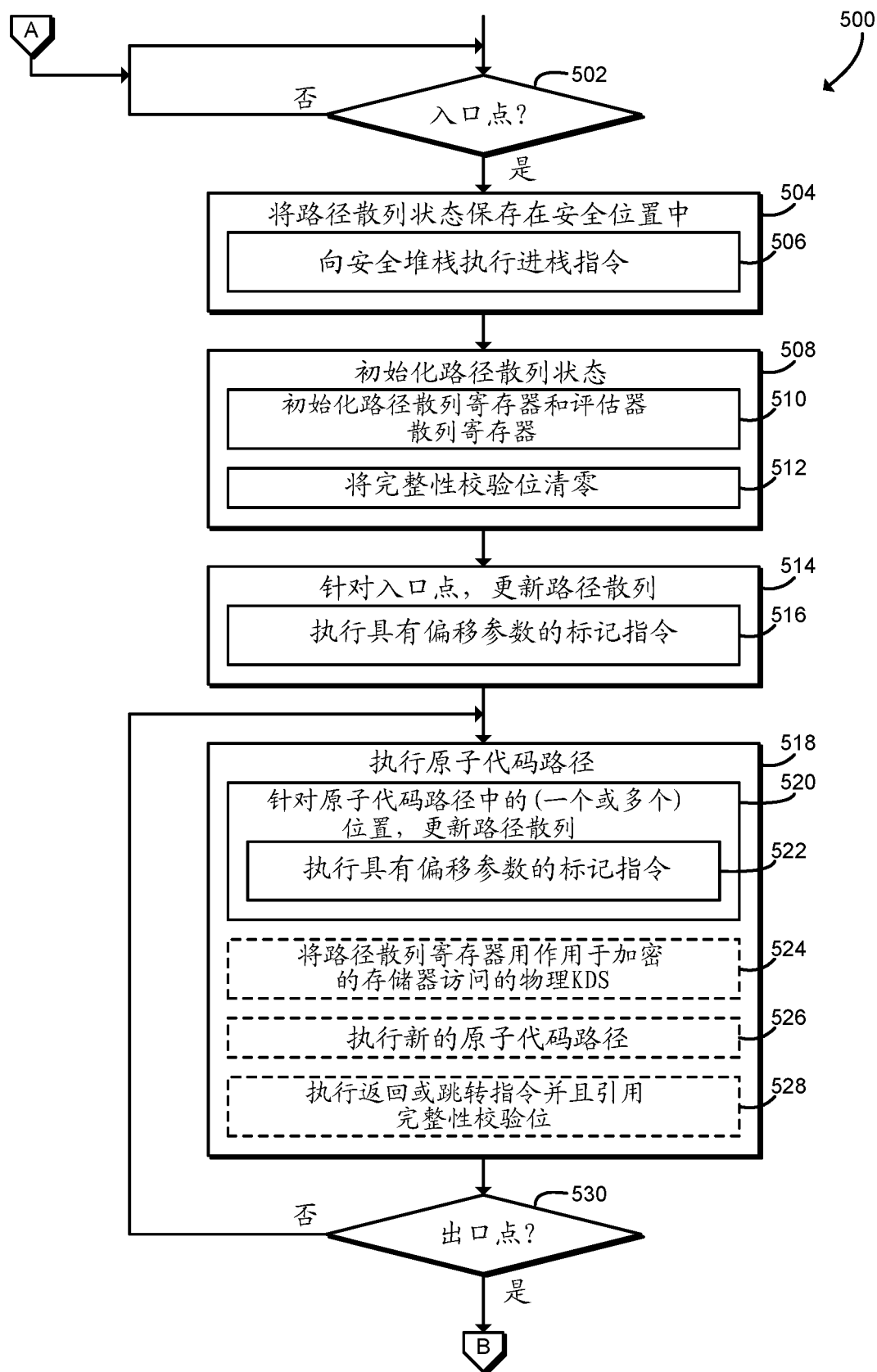


图 5A

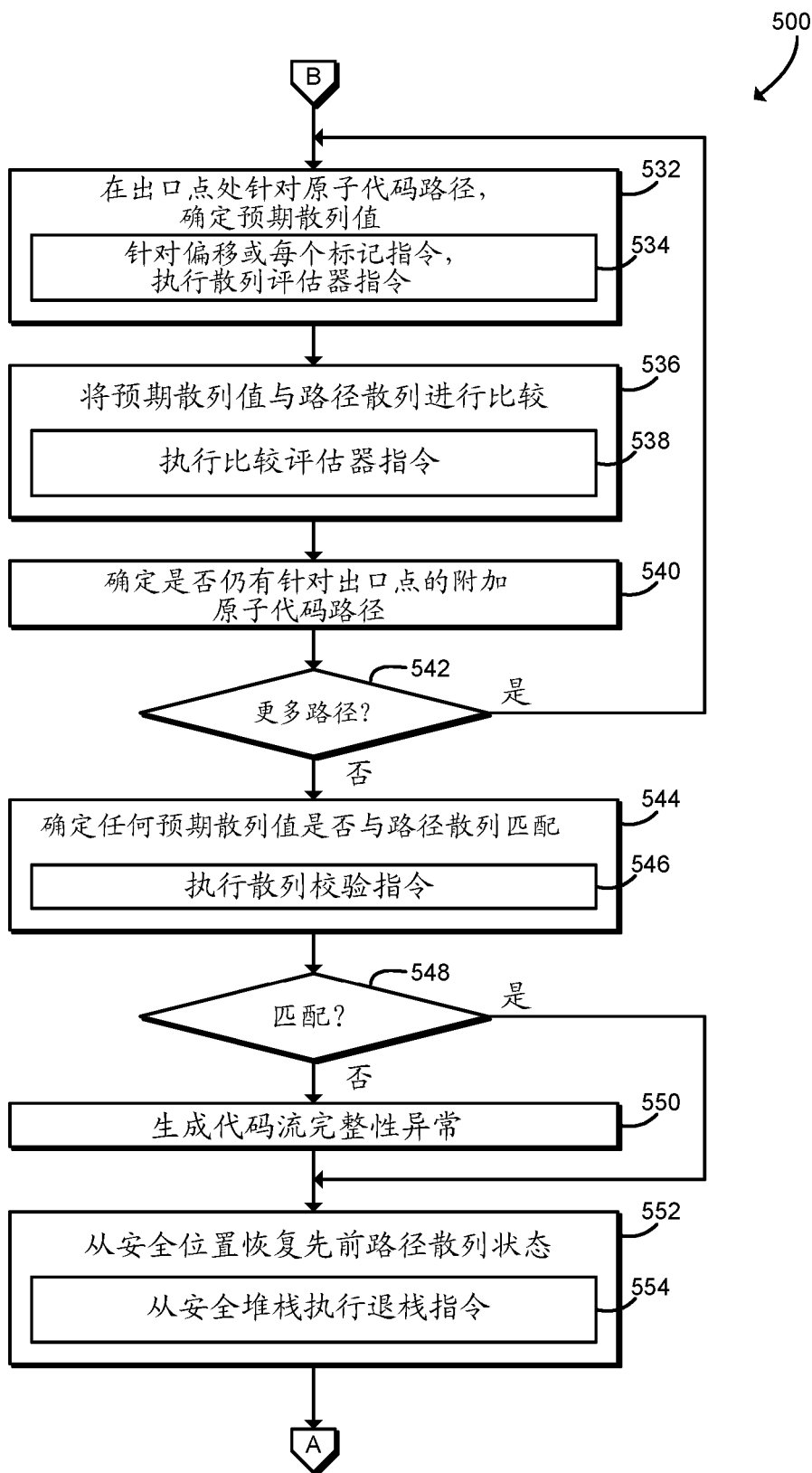


图 5B

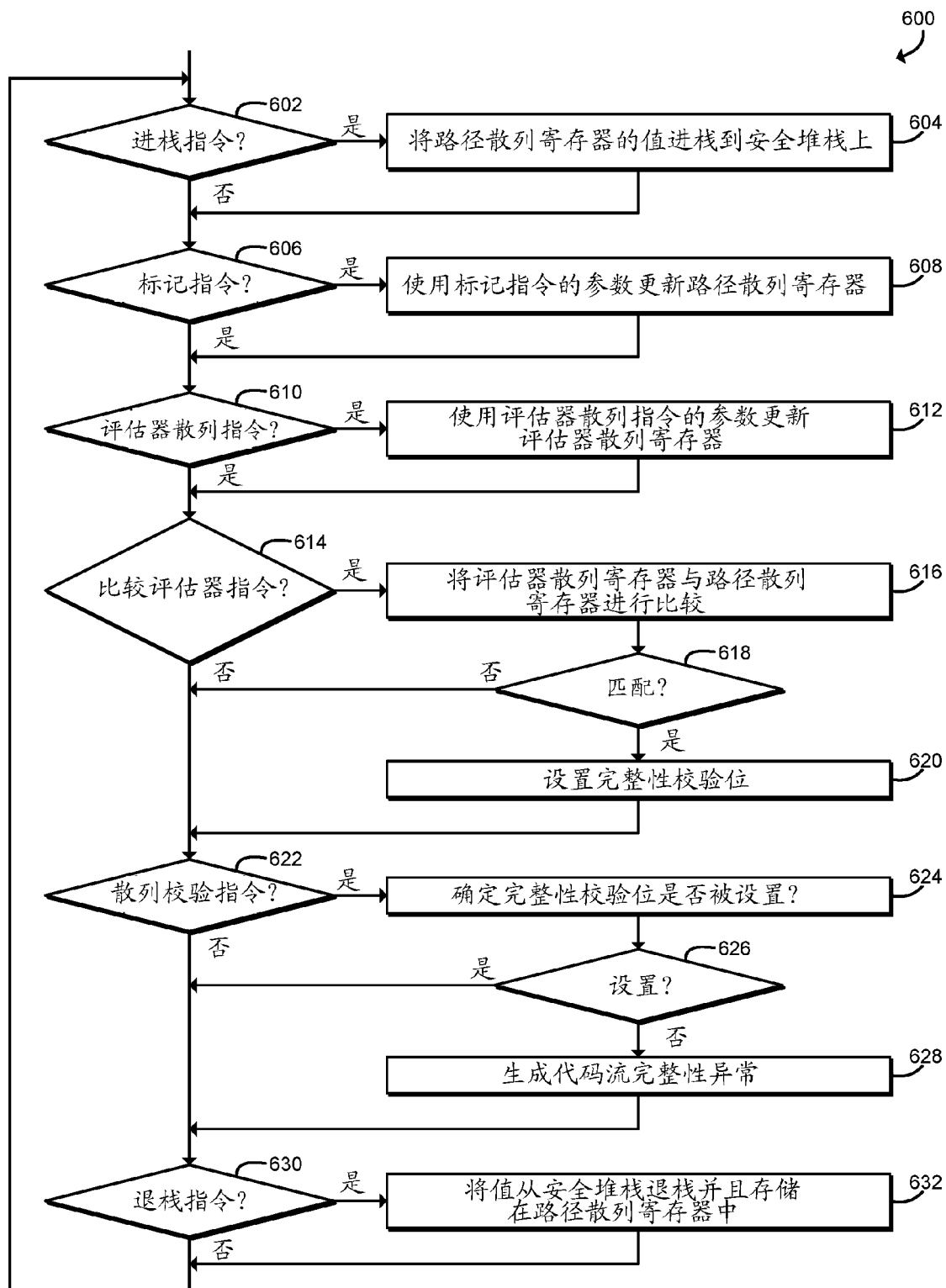


图 6