(19)中华人民共和国国家知识产权局

(12)发明专利申请

(10)申请公布号 CN 112149152 A
(43)申请公布日 2020.12.29

(21)申请号 202010229272.4

(22)申请日 2020.03.27

(30)优先权数据
62/868884 2019.06.29 US
16/723468 2019.12.20 US
16/724059 2019.12.20 US
16/722707 2019.12.20 US
16/722342 2019.12.20 US
16/723927 2019.12.20 US
16/724105 2019.12.20 US
16/723871 2019.12.20 US
16/723977 2019.12.20 US
16/724026 2019.12.20 US
16/740359 2020.01.10 US

(71)申请人 英特尔公司
地址 美国加利福尼亚州

(72)发明人 D.M.德伦 M.勒梅 R.J.马斯蒂 G.奈格尔 J.W.布兰德特

(74)专利代理机构 中国专利代理(香港)有限公司 72001
代理人 李啸 姜冰

(51)Int.Cl.
G06F 21/60(2013.01)
G06F 21/78(2013.01)

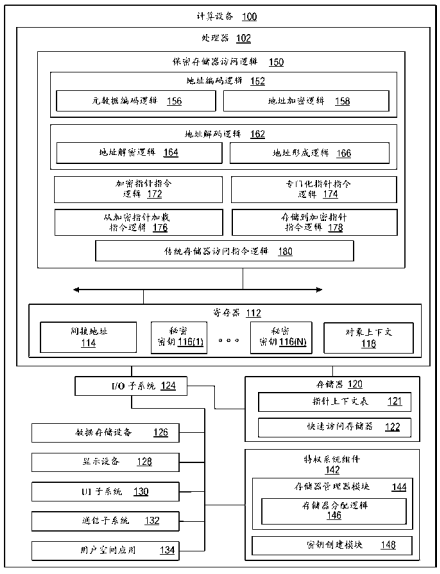
权利要求书4页 说明书83页 附图42页

(54)发明名称

使用加密的基地址和用于多租户环境的以密码方式的计算

(57)摘要

本文公开的技术在多租户环境中提供具有以密码方式编码的指针的以密码方式的计算。一种示例方法包括由可信运行时执行第一指令,以便为存储器中的私有存储器区域生成第一地址密钥,并且生成指向存储器中的私有存储器区域的第一以密码方式编码的指针。生成第一以密码方式编码的指针包括:将与私有存储器区域相关联的第一上下文信息存储在第一以密码方式编码的指针的第一位中,并且至少部分地基于第一地址密钥和第一调整对私有存储器区域的第一线性地址的片执行以密码方式的算法,第一调整包括第一上下文信息。该方法还包括允许多租户环境中的第一租户访问第一地址密钥和指向私有存储器区域的第一以密码方式编码的指针。



CN 112149152 A

1.一种处理器,包括:

存储器,用来存储可信运行时的第一指令;以及

核,包括用来在多租户环境中执行所述可信运行时的所述第一指令的电路,以:

为所述存储器中的私有存储器区域生成第一地址密钥;

生成指向所述存储器中的所述私有存储器区域的第一以密码方式编码的指针,包括:

将与所述私有存储器区域相关联的第一上下文信息存储在所述第一以密码方式编码的指针的第一位中;

至少部分地基于所述第一地址密钥和第一调整,对所述私有存储器区域的第一线性地址的片执行以密码方式的算法,所述第一调整包括所述第一上下文信息;以及

允许所述多租户环境中的第一租户访问所述第一地址密钥和指向所述私有存储器区域的所述第一以密码方式编码的指针。

2.根据权利要求1所述的处理器,其中,所述核要执行所述可信运行时的所述第一指令,以进一步:

基于确定所述第一租户被授权访问第二租户,允许所述第一租户访问第二地址密钥、第二数据密钥和指向所述第二租户的授权入口点的第二以密码方式编码的指针。

3.根据权利要求1所述的处理器,其中,所述核要执行所述可信运行时的所述第一指令,以进一步:

从所述第一租户接收访问第二租户的请求;以及

响应于确定所述第一租户被授权访问所述第二租户:

将所述第二租户的第二数据密钥存储在数据密钥寄存器中;

将所述第二租户的第二地址密钥存储在地址密钥寄存器中;以及

将控制转移给所述第二租户。

4.根据权利要求3所述的处理器,其中将控制转移给所述第二租户包括在指令指针寄存器中存储指向为所述第二租户分配的第二存储器区域中的授权入口点的第二以密码方式编码的指针。

5.根据权利要求1所述的处理器,其中所述核要:

执行所述第一租户的第一指令以:

将第二租户的第二代码密钥存储在代码密钥寄存器中;以及

使用目的地地址来访问为所述第二租户分配的第二存储器区域中的授权入口点;以及

执行所述第二租户的一个或多个第三指令以:

将所述第二租户的第二地址密钥存储在地址密钥寄存器中;以及

将所述第二租户的第二数据密钥存储在数据密钥寄存器中。

6.根据权利要求5所述的处理器,其中,响应于确定所述授权入口点包含授权指令并且所述目的地地址与存储器页边界对齐,所述目的地地址用于访问所述第二存储器区域中的所述授权入口点。

7.根据权利要求1所述的处理器,其中,所述核要执行所述第一租户的第二指令,以:

从第二租户的安全句柄中检索第二地址密钥、第二数据密钥和第二代码密钥;

将所述第二地址密钥加载到地址密钥寄存器中;

将所述第二数据密钥加载到数据密钥寄存器中;

将所述第二代码密钥加载到代码密钥寄存器中;以及

使用所述第二租户的所述安全句柄中的授权入口点指针来访问与所述第二租户相关联的第二存储器区域中的授权入口点。

8.根据权利要求7所述的处理器,其中所述第二租户的所述安全句柄存储在与所述第一租户相关联的所述私有存储器区域中。

9.根据权利要求1所述的处理器,其中所述核要:

执行所述第一租户的第三指令以将控制转移给第二租户,包括:

使用存储在所述第二指令的操作数中的索引来标识所述存储器中的表中的描述符,其中所述描述符包括至少第三以密码方式编码的指针和所述第二租户的第二代码密钥,所述第三以密码方式编码的指针指向为所述第二租户分配的第二私有存储器区域中的授权入口点;以及

将所述第二租户的所述第二代码密钥存储在代码密钥寄存器中;

将所述第三以密码方式编码的指针存储在指令指针寄存器中;以及

使用所述指令指针寄存器在所述授权入口点获取下一条指令。

10.根据权利要求1-9中的任一项所述的处理器,其中所述核包括:

第一寄存器,用来存储指向所述存储器中的共享存储器区域的共享的以密码方式编码的指针,其中所述核要执行所述可信运行时的所述第一指令,以进一步:

生成指向所述共享存储器区域的所述共享的以密码方式编码的指针,包括:

将与所述共享存储器区域相关联的共享上下文信息存储在所述共享的以密码方式编码的指针的第一位中;以及

至少部分地基于共享地址密钥和所述共享上下文信息,对所述共享存储器区域的第二线性地址的片执行所述以密码方式的算法;以及

允许所述第一租户和第三租户访问所述共享地址密钥、共享数据密钥和指向所述共享存储器区域的所述共享的以密码方式编码的指针。

11.根据权利要求10所述的处理器,其中所述核要执行所述第一租户的第四指令以解码所述共享的以密码方式编码的指针,包括:

至少部分地基于所述共享地址密钥和所述共享上下文信息,对所述第二线性地址的加密片执行所述以密码方式的算法,以计算所述第二线性地址的解密片;

至少部分地基于所述第二线性地址的所述解密片来生成所述第二线性地址;以及

访问所述共享存储器区域中的所述第二线性地址处的数据。

12.根据权利要求11所述的处理器,其中所述核要执行所述第四指令以进一步:

对所述共享存储器区域中的所述第二线性地址处的数据执行第二以密码方式的算法,以至少部分地基于所述共享地址密钥和至少部分地从所述共享的以密码方式编码的指针导出的调整来计算解密数据。

13.一种方法,包括:

由多租户环境中的可信运行时执行第一指令,以:

为存储器中的私有存储器区域生成第一地址密钥;以及

生成指向所述存储器中的所述私有存储器区域的第一以密码方式编码的指针,包括:

将与所述私有存储器区域相关联的第一上下文信息存储在所述第一以密码方式编码

的指针的第一位中;以及

至少部分地基于所述第一地址密钥和第一调整,对所述私有存储器区域的第一线性地址的片执行以密码方式的算法,所述第一调整包括所述第一上下文信息;以及

允许所述多租户环境中的第一租户访问所述第一地址密钥和指向所述私有存储器区域的所述第一以密码方式编码的指针。

14.根据权利要求13所述的方法,其中执行所述第一指令进一步包括:

基于确定所述第一租户被授权访问第二租户,允许所述第一租户访问第二地址密钥、第二数据密钥和指向所述第二租户的授权入口点的第二以密码方式编码的指针。

15.根据权利要求13所述的方法,其中执行所述第一指令进一步包括:

从所述第一租户接收访问第二租户的请求;以及

响应于确定所述第一租户被授权访问所述第二租户:

将所述第二租户的第二数据密钥存储在数据密钥寄存器中;

将所述第二租户的第二地址密钥存储在地址密钥寄存器中;以及

将控制转移给所述第二租户。

16.根据权利要求15所述的方法,其中将控制转移给所述第二租户包括:

在指令指针寄存器中存储指向为所述第二租户分配的第二存储器区域中的授权入口点的第二以密码方式编码的指针。

17.根据权利要求13所述的方法,进一步包括:

执行所述第一租户的第一指令以:

将第二租户的第二代码密钥存储在代码密钥寄存器中;以及

使用目的地地址来访问为所述第二租户分配的第二存储器区域中的授权入口点;以及

执行所述第二租户的一个或多个第三指令以:

将所述第二租户的第二地址密钥存储在地址密钥寄存器中;以及

将所述第二租户的第二数据密钥存储在数据密钥寄存器中。

18.根据权利要求17所述的方法,其中,响应于确定所述授权入口点包含授权指令并且所述目的地地址与存储器页边界对齐,所述目的地地址用于访问所述第二存储器区域中的所述授权入口点。

19.根据权利要求13所述的方法,进一步包括:

执行所述第一租户的第二指令,以:

从第二租户的安全句柄中检索第二地址密钥、第二数据密钥和第二代码密钥;

将所述第二地址密钥加载到地址密钥寄存器中;

将所述第二数据密钥加载到数据密钥寄存器中;

将所述第二代码密钥加载到代码密钥寄存器中;以及

使用所述第二租户的所述安全句柄中的授权入口点指针来访问与所述第二租户相关联的第二存储器区域中的授权入口点。

20.根据权利要求19所述的方法,其中所述第二租户的所述安全句柄存储在与所述第一租户相关联的所述私有存储器区域中。

21.根据权利要求13所述的方法,进一步包括:

执行所述第一租户的第三指令以将控制转移给第二租户,包括:使用存储在所述第三

指令的操作数中的索引来标识所述存储器中的表中的描述符，其中所述描述符包括至少第三以密码方式编码的指针和所述第二租户的第二代码密钥，所述第三以密码方式编码的指针指向为所述第二租户分配的第二私有存储器区域中的授权入口点；以及

将所述第二租户的所述第二代码密钥存储在代码密钥寄存器中；

将所述第三以密码方式编码的指针存储在指令指针寄存器中；以及

使用所述指令指针寄存器在所述授权入口点获取下一条指令。

22.根据权利要求13-21中的任一项所述的方法，进一步包括：

在第一寄存器中存储指向所述存储器中的共享存储器区域的共享的以密码方式编码的指针；

执行所述可信运行时的所述第一指令，以进一步：

生成指向所述共享存储器区域的所述共享的以密码方式编码的指针，包括：

将与所述共享存储器区域相关联的共享上下文信息存储在所述共享的以密码方式编码的指针的第一位中；以及

至少部分地基于共享地址密钥和所述共享上下文信息，对所述共享存储器区域的第二线性地址的片执行所述以密码方式的算法；以及

允许所述第一租户和第三租户访问所述共享地址密钥、共享数据密钥和指向所述共享存储器区域的所述共享的以密码方式编码的指针。

23.根据权利要求22所述的方法，进一步包括：

解码所述共享的以密码方式编码的指针，包括：

至少部分地基于所述共享地址密钥和所述共享上下文信息，对所述第二线性地址的加密片执行所述以密码方式的算法，以计算所述第二线性地址的解密片；

至少部分地基于所述第二线性地址的所述解密片来生成所述第二线性地址；以及

访问所述共享存储器区域中的所述第二线性地址处的数据。

24.根据权利要求23所述的方法，进一步包括：

对所述共享存储器区域中的所述第二线性地址处的数据执行第二以密码方式的算法，以至少部分地基于所述共享地址密钥和至少部分地从所述共享的以密码方式编码的指针导出的调整来计算解密数据。

25.一种包括指令的机器可读介质，当所述指令被执行时，使处理器执行如任一前述权利要求所述的方法。

使用加密的基地址和用于多租户环境的以密码方式的计算

[0001] 相关申请的交叉引用

本申请是提交于2019年12月20日的美国专利申请序列号16/724059以及提交于2019年12月20日的美国专利申请序列号16/723468的部分继续申请(并要求其权益和优先权),所述两个专利申请都通过引用并入到本文中。本申请也是提交于2019年12月20日的美国专利申请序列号16/724105、提交于2019年12月20日的美国专利申请序列号16/724026、提交于2019年12月20日的美国专利申请序列号16/723977、提交于2019年12月20日的美国专利申请序列号16/723927、提交于2019年12月20日的美国专利申请序列号16/723871、提交于2019年12月20日的美国专利申请序列号16/722707、以及提交于2019年12月20日的美国专利申请序列号16/722342的部分继续申请(并要求其权益和优先权),所有的所述七个专利申请都要求提交于2019年6月29日的美国临时申请号64/868884的优先权,并且所有的所述七个专利申请都通过引用并入到本文中。本申请还要求提交于2019年6月29日的美国临时申请号62/868884的权益和优先权,其通过引用并入到本文中。

技术领域

[0002] 本公开一般上涉及计算机系统的领域,并且更特别地,涉及使用加密的基地址的以密码方式的计算以及在多租户环境中使用的以密码方式的计算。

背景技术

[0003] 保护计算机系统中的存储器免于软件缺陷和保密性弱点是显著关切之事。当程序将数据写入缓冲器时,发生缓冲器溢出(其可能影响存储器安全),并且缓冲器溢出超越了缓冲器的边界,使得相邻的存储器位置被覆写。类似地,超过缓冲器的末端而进入另一页的读取可能触发访问违例或故障。另一存储器安全性违例被称为悬摆指针。悬摆指针是未解析到有效目的地的引用。这可能发生在存储器被解除分配(deallocate)而没有修改指向解除分配的(或释放的)存储器的现有指针的值的情况下。如果系统重新分配了释放的存储器,并且悬摆指针被用于访问重新分配的存储器,则可能会出现不可预测的行为(包括系统故障)。当前的计算技术已使用了架构和元数据来提供数据保护。例如,在先前的解决方案中,处理器将使用查找表来编码关于针对所有权、存储器大小、位置、类型、版本等的数据的策略或数据。然而,这种元数据要求附加的存储(存储器开销),并负面地影响性能,特别是对于利用细粒度元数据的实现而言。因此,需要不同的方法来为计算系统提供存储器安全。

附图说明

[0004] 为了提供对本公开及其特征和优点的更全面理解,结合附图对下面的描述进行参考,附图中相似的附图标记表示相似的部分,在附图中:

图1是根据本公开的至少一个实施例的配置有保密存储器访问逻辑的示例计算设备的简化框图;

图2是示出根据本公开的至少一个实施例的图1的保密存储器访问逻辑的应用的简化

环境图；

图3是根据至少一个实施例的计算设备中的硬件、软件单元和数据流的可能示例细节的简化框图；

图4是如本文中公开的为间接地址提供保密性的过程的至少一个实施例的简化流程图，所述过程可以由图1的计算设备执行；

图5是如本文中公开的用于验证先前保密的间接地址的过程的至少一个实施例的简化流程图，所述过程可以由图1的计算设备执行；

图6是根据本公开的至少一个实施例的以密码方式编码的示例指针的图；

图7是根据本公开的至少一个实施例的可用于生成图6的指针的可能上下文信息的图；

图8是根据至少一个实施例的用于解密以密码方式编码的指针以产生线性地址的环境的图；

图9是示出根据至少一个实施例的将上下文信息嵌入到编译的代码中的编译器的简化框图；

图10是根据至少一个实施例的过程的至少一个实施例的简化流程图，所述过程用于编译代码以使用具有加密的基地址的以密码方式编码的指针；

图11是用于生成具有加密的基地址的以密码方式编码的指针的过程的至少一个实施例的简化流程图，所述过程可以由图1的计算设备执行；

图12是根据至少一个实施例的生成专用指针的示例过程的简化流程图；

图13是根据至少一个实施例的与生成线性地址相关联的示例过程的流程图，所述线性地址从具有加密的基地址的以密码方式编码的指针中生成；

图14是根据至少一个实施例，当生成具有加密的基地址的以密码方式编码的指针时检测表冲突的示例过程的流程图；

图15是根据至少一个实施例的基于具有加密的基地址的以密码方式编码的指针在存储器访问期间检测故障的示例过程的流程图；

图16A是根据至少一个实施例的使用具有加密的基地址的以密码方式编码的指针的软件代码的示例；

图16B是根据至少一个实施例的从图16A的软件代码中输出的汇编语言的示例；

图17是根据至少一个实施例的尝试使用以密码方式编码的指针来访问存储器的示例对手的图；

图18是根据至少一个实施例的试图使用以密码方式编码的指针来访问存储器的另一示例对手的图；

图19是根据至少一个实施例的可用于加密和解密的具有密文窃取(XTS)块密码(block cipher)的基于XEX的调整的码本模式的框图；

图20是根据至少一个实施例的可用于加密和解密的高级加密标准(AES)计数器模式块密码的框图；

图21A-21B是示出根据至少一个实施例的将具有加密的基地址的以密码方式编码的指针绑定到由该指针所引用的数据的加密的示例过程的流程图；

图22是根据至少一个实施例的与绑定到数据加密的以密码方式编码的指针相关联的示例过程的流程图；

图23是示出根据至少一个实施例的将以密码方式编码的指针的另一实施例绑定到由该指针所引用的数据的加密的示例过程的流程图；

图24是示出根据至少一个实施例的将编码的指针的又一实施例绑定到由该指针所引用的数据的加密的示例过程的流程图；

图25是示出根据至少一个实施例的支持多租户的示例单地址空间的多租户环境的简化框图，在所述多租户环境中使用具有加密的基地址的以密码方式编码的指针来实施隔离；

图26是一个或多个实施例中的多租户环境中可与租户相关联的可能的密钥和信息的示例；

图27是示出根据至少一个实施例的用于从一个租户跳转(jump)到另一租户的指令的示例操作的框图；

图28是根据至少一个实施例的可用于为多租户环境生成以密码方式编码的指针的可能上下文信息的图；

图29是示出根据至少一个实施例的在多租户环境中使用密码术上下文索引的简化框图；

图30是示出根据至少一个实施例的与使用编码的指针在多租户环境中加载租户相关联的示例过程的流程图；

图31是示出根据至少一个实施例的与在多租户环境中使用编码的指针将控制从一个租户转移给另一租户相关联的另一示例过程的流程图；

图32A是示出根据至少一个实施例的与在多租户环境中使用编码的指针将控制从一个租户转移给另一租户相关联的另一示例过程的流程图；

图32B是示出根据至少一个实施例的与在多租户环境中使用编码的指针将控制从一个租户转移给另一租户相关联的另一示例过程的流程图；

图33是示出根据至少一个实施例的与在多租户环境中使用指针将控制从一个租户转移给另一租户相关联的示例过程的流程图；

图34是示出根据至少一个实施例的与使用以密码方式编码的指针来编译租户代码以供访问存储器和代码相关联的示例过程的流程图；

图35是示出根据至少一个实施例的示例以密码方式的计算环境的框图；

图36是示出根据至少一个实施例的示例处理器的框图；

图37A是示出根据某些实施例的示例性有序流水线和示例性寄存器重命名、无序发布/执行流水线两者的框图；

图37B是示出根据某些实施例的要被包括在处理器中的有序架构核和示例性寄存器重命名、无序发布/执行架构核两者的示例性实施例的框图；

图38是根据至少一个实施例的示例计算机架构的框图；以及

图39是根据本公开的实施例的对照使用软件指令转换器将源指令集中的二进制指令转换成目标指令集中的二进制指令的框图。

具体实施方式

[0005] 下面的公开提供了用于实现以密码方式的计算的各种可能的实施例或示例。以密

码方式的计算是计算行业中的重要趋势，其中计算本身的完全基础从根本上变成了使用密码。以密码方式的计算代表了巨变，对系统保密性的根本反思，对行业有广泛的影响。

[0006] 本申请中公开的实施例涉及基地址加密，其中指向对数据的存储器位置的指针用标签和/或其它元数据来编码，并且可被用于导出数据/代码以密码方式的(例如，加密和解密)算法和地址以密码方式的算法的调整输入的至少一部分。因此，在以密码方式的寻址层与数据/代码加密和解密之间创建了以密码方式的绑定。这隐式地强制了边界，因为偏离超过了对象(例如，数据)的末端的指针很可能对该相邻对象使用不正确的标签值。此外，实施例还允许使用存储器分配大小对边界进行显式检查，这将在本文中进一步描述。在一个或多个实施例中，用元数据(例如，随机生成的位的标签)和指向存储器位置的线性地址(本文中也称为“存储器地址”)对指针进行编码。在一个或多个实施例中，线性地址是存储器中的对象(或对象内部的某物)的基地址。基地址的片(slice)或段(segment)包括被加密并嵌入到指针中的多个位。基于秘密地址密钥和包括上下文信息(例如，存储器分配大小、对象的类型、许可等)和/或指针中编码的元数据的调整，对基地址片进行加密(和解密)。当访问指针时，供应相同的上下文信息(和编码的元数据)，以便正确地解密基地址片并生成线性地址。绑定数据加密和指针可通过使用基于指针的调整和秘密数据密钥在存储器位置处加密数据(或代码)来实现。用于加密(和解密)数据的基于指针的调整可从编码的指针和潜在的附加上下文信息中导出。特别地，可至少部分地基于基地址的解密片以及编码的指针中的可能的元数据来创建用于数据的基于指针的调整。在至少一些实施例中，与指针分开存储的上下文信息也可被包括在调整中。

[0007] 在一个或多个实施例中，可能存在用于加密和解密要嵌入在指针中的基地址的片的不同调整的变体。例如，不同的和/或附加的上下文信息(诸如各种类型的元数据、以密码方式的上下文标识符、明文基地址的部分或其任何合适组合)可被用于调整中，所述调整被用于加密/解密指针中的存储器地址的片。类似地，也可能存在用于加密和解密由编码的指针所引用的数据的调整的变体。在其它实施例中，编码的指针的附加部分可用在基于指针的调整中，或者整个编码的指针可被用作基于指针的调整。此外，在至少一些实施例中，不同的和/或附加的上下文信息(诸如元数据、以密码方式的上下文标识符、明文地址的片或其任何合适组合)也可用于调整中，所述调整被用于加密/解密由编码的指针所引用的数据。

[0008] 应注意的是，虽然应理解的是数据和代码可以由间接地址(本文中也称为“指针”)所引用，但是为了便于说明，本文中的描述可以仅指由指针所引用并以密码方式绑定到该指针的数据(而不是代码)。然而，应理解的是，一般来说，关于这种数据的存储器访问和加密/解密的讨论也意在可适用于代码。此外，本文中使用的术语“对象”意在表示由指针所引用的存储器中的值或信息或一组值或信息。例如，对象可以是堆分配、局部或全局变量、函数、代码等。

[0009] 为了说明以密码方式的计算中的基地址加密的若干实施例，首先理解与数据保护和存储器安全相关联的操作和活动是重要的。因此，下面的基本信息可以被视为可以适当解释本公开的基础。

[0010] 当前的计算技术(例如，用于过程/内核分离的页表、虚拟机管理器、受管理运行时(Managed Runtime)等)已经使用了架构和元数据来提供数据保护。例如，在先前的解决方

案中，处理器将使用查找表来编码关于针对所有权、存储器大小、位置、类型、版本等数据的策略或数据。动态存储和加载元数据要求额外的存储(存储器开销)，并且影响性能，特别是对于细粒度元数据(诸如对于功能即服务(FaaS)工作负载或对于每个对象的边界信息)。

[0011] 以密码方式的计算可以减轻或解决许多以上提及的问题(以及更多的问题)。以密码方式的计算可能会用全新的细粒度保护模型来使过程分离、用户空间和内核的传统模式冗余。在以密码方式的计算中，保护是使用密码的，其中处理器和加速器同样利用秘密密钥和密码来以越来越精细的粒度提供访问控制和分离。此外，代替当前系统中的虚拟机和过程分离，利用以密码方式的计算，各个功能可以成为边界，允许借助于加密的指针来共享对象，其中加密的指针和密钥提供下到各个数据对象的受控访问。

[0012] 本文中公开的以密码方式的计算的实施例可以利用以密码方式的寻址层的概念，其中处理器基于上下文信息(例如，隐式和显式元数据、以密码方式的上下文标识符、指针中编码的元数据等)解密软件分配的存储器基地址(线性/虚拟地址空间，有时被称为“指针”)。如本文中所使用的，“调整”除了通常的明文或密文输入和密钥(例如，秘密密钥116(1))之外，还可以指对块密码的额外输入以及其它事物。调整包括代表值的一个或多个位。在一个或多个实施例中，调整可以组成用于块密码的初始化向量(IV)的全部或部分。当执行基地址片的解密时，如果用于创建调整的信息(例如，上下文信息)对应于由存储器分配器(例如，软件分配方法)对存储器地址的原始分配，则处理器可以正确地解密基地址片并生成完整的明文基地址。否则，随机地址结果可能会导致故障并被处理器所捕获。

[0013] 已经从以密码方式编码的指针生成的完整明文基地址可以被处理器用作对数据加密密码的调整的至少一部分，所述数据加密密码被用于加密/解密它们所引用的数据(由以密码方式编码的指针所引用的数据)，在以密码方式的寻址层与数据/代码加密之间创建密码绑定。在其它实施例中，以密码方式的地址(或地址片)可以被用作用于数据加密密码的调整的至少一部分。应注意的是，用作块密码的输入以加密/解密存储器地址的调整在本文中也被称为“地址调整”。类似地，用作块密码的输入以加密/解密数据的调整在本文中也被称为“数据调整”。

[0014] 通过将元数据以密码方式编码到地址及其引用的数据中，以密码方式的计算可以降低或消除对用来提供策略和上下文信息/元数据的额外的分开的存储器/存储设备的需要。由于仅元数据的降低，这可以在计算行业中节省高达数十亿美元(例如，动态随机存取存储器(DRAM)中的费用)。客户可以收获存储器成本中的这些节省，同时仍然利用以密码方式的计算获得他们想要的保密性、安全和无错误功能。通过允许安全推测，以密码方式的计算的基本以密码方式的分离策略可以允许处理器自由推测并提供提高的性能。

[0015] 在数据保密性基本上与以密码方式的存储器寻址链接的以密码方式的计算中，对数据的处理和细粒度以密码方式的访问控制是非常重要的。以密码方式的计算将所有计算向量从CPU转换到GPU，从加速器转换到FPGA等。利用以密码方式的计算，保护可以是以密码方式的，其中处理器和加速器同样利用秘密密钥和密码以越来越精细的粒度提供访问控制和分离。此外，代替虚拟机和过程分离，各个函数可以变成边界，地址空间被共享，而指针被加密，其中密钥提供下到各个数据对象的受控访问。因此，在以密码方式的操作中能力可能会相互交织，以提供对数据对象的有粒度的访问控制，同时在系统的每个级别防止缓冲器溢出、类型混淆和暂时(例如，释放后使用)弱点。以密码方式的代码可以原本地安全地执

行，并且不需要解释器或受管理运行时来提供存储器和类型安全。存储器可能会从隔离的域和容器中移到全局共享的存储器模型，在所述模型中数据基于以密码方式的访问控制机制是可访问的，并且不存在难以缩放的分布式许可、分页和相关联的控制结构。甚至文件也可以被安全地直接存储在存储器中(例如，存储在非易失性存储器模块中，诸如非易失性双列直插式存储器模块(NVDIMM))、被单独地加密、以密码方式改变大小、并且不受软件错误损坏。这可能会对功能安全性、可靠性和多租户产生牵连，潜在地允许为提高处理性能的更多推测。

[0016] 密码学继续变得更快和更轻。例如，高级加密标准(AES)数十年来一直是对数据加密的主流(使用128位块密码)。与此同时，今天的存储器寻址通常是64位。尽管本文中的实施例可以参考对64计算机的64位存储器寻址来说明和解释，但是所公开的实施例并不意在如此受限，并且可以容易地适用于适应32位、128位或用于指针的任何其它可用位大小。同样地，本文中的实施例还可以适用于适应块密码的各种大小(例如，64位、48位、32位、16位等使用Simon、Speck、可调整的K密码、PRINCE或任何其它块密码)。

[0017] 最近呈现了适合于指针加密的轻量级密码。例如，PRINCE密码可以在3个时钟中实现，在10nm工艺中只需要799μm^{2}的面积，在十分之一的硅面积中提供AES的时延的一半。以密码方式的计算可以利用这些新密码以及其它密码，引入新颖的计算机架构概念，包括但不限于：(i)以密码方式的寻址，即在处理器使用关于所引用数据(例如，嵌入在指针中的元数据和/或外部元数据)的上下文信息、地址的片本身或其任何合适组合作为调整来对数据指针进行加密；以及(ii)使用以密码方式编码的指针或其部分、不以密码方式编码的指针或其部分、与所引用的数据相关联的上下文信息或其任何合适组合作为用于数据加密的调整，在核处对数据本身加密。可调整的各种加密模式可被用于包括元数据的该目的(例如，计数器模式(CTR)和具有密文窃取(XTS)的基于异或-加密-异或(XEX)的调整的码本模式)。除了提供数据机密性的加密之外，其隐含的完整性可以允许处理器使用正确的密钥流和调整来确定数据是否被正确解密。在一些块密码加密模式中，块密码创建密钥流，然后将其与输入块组合(例如，使用异或操作)以产生加密或解密的块。在一些块密码中，密钥流被馈送到下一个块密码中，以执行下一个输入块的加密或解密。

[0018] “元数据墙”可能指的是附加获取关于存储器操作的元数据的问题，所述操作诸如访问控制、对象类型/大小以及版本。今天的计算机架构要求处理器查找元数据或关于数据的数据，以确定是否允许存储器访问。用于元数据的额外存储器访问会影响性能，要求用于元数据的附加存储设备，并且元数据本身需要被保护以提供保密性。一些当前的解决方案以边界表的形式添加元数据，硬件将使用所述边界表来检测缓冲器溢出，所述解决方案已被证明对某些工作负载具有高达4倍的性能影响，其中存储器开销高达400%。类似地，影子栈元数据允许控制流实现技术，并且存储器标记使用了元数据以进行版本化(versioning)，并且能力添加元数据以供验证数据类型。存储器标记不适用于减轻类型混淆和防止未初始化的使用变量。此外，虽然使用纠错码位可以降低存储器标记的开销，但它仍然可能要求附加的设备，这可能会增加成本。能力机器也可以使用粗指针以将保密性元数据内嵌嵌入到指针中，由于指针大小加倍，导致大量存储器开销(例如，在指针密集型应用中为25%)。

[0019] 相反，本文中描述的以密码方式的计算的实施例可以提供上下文信息(例如，指针

中编码的元数据、外部元数据)，将其代码化为对以密码方式的寻址和数据、以密码方式的寻址和代码或其组合的调整，从而移除了由于包含这种元数据而导致的潜在性能和存储器开销。特别地，以密码方式的计算中的基地址加密减轻了主要类别的存储器安全弱点，而同时使编译器能够将静态计算的上下文信息嵌入到程序代码中，以降低动态加载元数据中的时间和存储器开销，因为在过去仅依赖于动态元数据的方法中，这种开销已是相当大的。产生的加密的基地址片可能不需要除秘密密钥之外的附加保护，从而允许与数据相同的存储器的重用。变得越来越重要的功能安全标准要求使用存储器安全机制来应对弱点，诸如例如数据损坏、越界访问、控制流违例和访问许可违例。如本文中进一步讨论或指出的，使用加密的基地址的以密码方式的计算可以使用相同的统一机制，使用计算而非存储器来解决无数的这样的存储器安全弱点。

[0020] 转向图1，图1是根据本公开的至少一个实施例的配置有保密存储器访问逻辑的示例计算设备100的简化框图。在所示的示例中，计算设备100包括处理器102，所述处理器102具有一组保密存储器访问逻辑150和多个寄存器112。保密存储器访问逻辑150利用关于间接地址114的元数据，其被编码到间接地址114的未使用位(例如，64位地址的非规范位，或者例如由操作系统留出的地址范围，使得地址范围的对应高阶位可被用于存储元数据)，以便保密和/或提供对由间接地址114所指向的存储器位置的访问控制。例如，由保密存储器访问逻辑150提供的元数据编码和解码可以防止间接地址114被操纵而导致缓冲器溢出，和/或可以防止程序代码访问它不被许可访问的存储器。保密存储器访问逻辑150的地址编码逻辑152在存储器(例如，在堆中由操作系统)分配时被调用，并以多种不同方式中的任何一种被提供到执行程序，包括通过使用诸如malloc、alloc或new的函数;或者借助于加载器隐式地、或者由编译器静态分配存储器等。结果是，指向所分配的存储器的间接地址114用地址元数据来编码。

[0021] 地址元数据可以包括标签值或版本号。标签值可以包括为存储器分配生成的随机化位。随机化位可以被生成为对于存储器分配是唯一的。版本号可以是确定性的不同的值(诸如序列号)，所述序列号及时确定所引用的已分配存储器的当前所有权。每当为新分配的存储器创建间接地址时，序列号可以递增预定的量。标签/版本可被用作调整的一部分，以加密和解密间接地址中编码的基地址片。标签/版本也可用作调整的一部分，以加密和解密基地址引用的数据或代码。

[0022] 地址元数据还可以包括有效范围元数据。有效范围元数据允许执行程序以在有效范围内操纵间接地址114的值，但是如果使用超出有效范围的间接地址114来访问存储器，则可能潜在地损坏间接地址114。备选地或附加地，有效范围元数据可用于标识有效代码范围，例如程序代码被许可访问的存储器范围(例如，编码范围信息可用于在寄存器上设置显式范围)。可以在地址元数据中编码的其它信息包括对间接地址114的访问(或许可)约束(例如，间接地址114是否可用于写、执行或读所引用的存储器)。

[0023] 在本文将进一步描述的至少一些其它实施例中，其它元数据(或上下文信息)可以被编码在间接地址114的未使用位中，所述元数据诸如存储器分配大小(例如，由间接地址所引用的分配的存储器的字节)、数据或代码的类型(例如，由编程语言定义的数据或代码的类别)、和/或许可(例如，间接地址的读、写和执行许可)、数据或代码的位置(例如，与数据或代码的大小相组合的地址)、要存储指针本身的存储器位置、数据或代码的所有权、特

权级别(例如,用户或监管者)、以密码方式的上下文标识符(或密码术上下文ID)(例如,针对每个间接地址的随机化或确定性的唯一值)等。在其它实施例中,这样的上下文信息可以不被编码在间接地址中,而是当它被嵌入到代码流中时可以被静态地访问,或者借助于存储器中的查找表来动态地访问。在一些实施例中,地址元数据可以包括与间接地址相关联的随机化位的标签,以使该标签对于对手不可预测。对手可能试图猜测标签值,使得对手能够访问由指针所引用的存储器,并且随机化标签值与用于生成标签值的确定性方法相比,可能使对手将更不太可能成功地猜测该值。在一些实施例中,指针可以包括版本号,所述版本号及时确定被引用的分配数据的当前所有权,而不是随机化的标签值,也不是除了随机化的标签值之外还及时确定被引用的分配数据的当前所有权。即使对手能够猜测存储器的区域的当前标签值或版本号(例如,因为用于生成版本号的算法是可预测的),对手可能仍然不能正确地生成指针的对应加密部分,因为对手不具有对稍后将被用于解密指针的该部分的密钥的访问。

[0024] 地址解码逻辑162验证关于利用处理器指令(诸如MOV)的存储器读和写操作的编码的元数据,其中通用寄存器被用作存储器地址以从存储器读值(例如,加载)或向存储器写值(例如,存储),以及关于涉及"使用"存储器的其它操作(诸如,例如ADD等具有存储器操作数的算术指令,以及例如CALL/JMP等控制转移指令,等等)。这些被认为是存储器操作数,其可以指定处理器指令将访问以执行其操作的存储器中的位置。数据存储器操作数可以指定要操纵的数据在存储器中的位置,而控制转移存储器操作数可以指定在存储器中存储用于控制转移的目的地地址的位置。还可以调用地址解码逻辑162来验证用于加载由间接地址所引用的数据或代码的新指令以及用于存储由间接地址所引用的数据或代码的另一新指令的编码元数据。这些指令可以提供间接地址(或指针)作为参数以及上下文信息,所述参数以及上下文信息可以用作调整的一部分以用于解密嵌入在间接地址中的基地址片。

[0025] 示例保密存储器访问逻辑150被实施为处理器指令的一部分(例如,作为处理器指令集架构的一部分),或者微代码(例如,存储在只读存储器中并由处理器102直接执行的指令)。在其它实施例中,保密存储器访问逻辑150的部分可以被实施为硬件、固件、软件或其组合(例如,由计算设备100的特权系统组件142执行的编程代码)。例如,保密存储器访问逻辑150可以在软件中被实现为指令集仿真器(例如,诸如PIN工具的二进制探测工具),其利用如本文中公开的编码的地址来仿真指令逻辑。

[0026] 保密存储器访问逻辑150可由计算设备100执行,以例如在由计算设备100执行程序(诸如用户空间软件应用)期间"内嵌地(inline)"为间接地址提供保密性。如本文中所使用的,术语"间接地址"和"指针"每个除了其它事物以外还可以指地址(例如,虚拟地址或线性地址),诸如存储其它数据或指令的存储器位置的基地址。在示例中,存储的存储器位置(在该位置存储有数据或代码)的编码的存储器地址的寄存器可以充当指针。这样,间接地址114可以被实施为例如数据指针(其指数据的位置)、代码指针(其指可执行代码的位置)、指令指针或栈指针。因此,间接地址可以用其它术语来指代,诸如"指针"、"地址指针"或"指针地址"。如本文中所使用的,"元数据"除了其它事物以外还可以指关于或涉及间接地址114的信息,诸如有效数据范围、有效代码范围、指针访问许可、明文地址片的大小(例如,以位的幂来编码)、存储器分配大小、数据或代码的类型、数据或代码的位置、数据或代码的所有权、间接地址的版本、随机化位的标签、版本、软件的特权级别、以密码方式的上下文标识

符等。

[0027] 如本文中所使用的，“存储器访问指令”除了其它事物以外还可以指“MOV”或“LOAD”指令或使数据在一个存储位置(例如存储器)被读、拷贝或以其它方式被访问并被移动到另一存储位置(例如寄存器)的任何其它指令(其中“存储器”可以指主存储器或高速缓存，例如随机存取存储器的形式，“寄存器”可以指处理器寄存器，例如硬件)，或者访问或操纵存储器的任何指令。此外，如本文中使用的，“存储器存储指令”除了其它事物以外还可以指“MOV”或“STORE”指令或使数据在一个存储位置(例如寄存器)被读、拷贝或以其它方式被访问并被移动到另一存储位置(例如存储器)的任何其它指令，或者访问或操纵存储器的任何指令。在本文中的一个或多个实施例中，将在本文中进一步描述使用用加密基地址片编码的指针加载数据或代码的新指令(例如，“LdEP”指令)以及使用用加密基地址片编码的指针存储数据或代码的新指令(例如，“StEP”指令)。

[0028] 然而，本文中公开的间接地址编码/解码技术不限于MOV或加载/存储指令。例如，诸如调用和跳转指令的控制转移指令可以适用于以类似于本文中针对MOV指令所描述的方式来处理编码的间接地址，其中代码将在有效地址范围内执行。同样，给定由控制转移指令(例如，JMP/CALL)指定的编码的地址，指令指针(例如，寄存器)可以是范围限制的，这导致编码的地址被用于指令指针，从而将有效程序执行约束在有效地址范围内(实际上，程序计数器可以正确地递增，直到它到达编码的范围的末端)。此外，在一些架构中，任何数量的处理器指令可以具有采用间接地址形式的存储器操作数(例如，诸如加(ADD)、减(SUB)、乘(MUL)、与(AND)、或(OR)、异或(XOR)等算术操作可以具有采用间接地址形式的源/目的地存储器引用和/或它们可以具有源/目的地寄存器操作数)。然而，在其它架构中，存储器操作数的格式可以变化。例如，可以以某种方式(例如，通过添加)来组合寄存器以产生有效地址。此外，可以可选地包括其它参数(诸如换算因数)，所述换算因数将寄存器值中的一个(例如，索引)和/或嵌入在直接相加的指令中的恒定位移值相乘。此外，应当注意，虽然说明性实施例指的是“指令”，但是这种指令可以实施为例如处理器指令、操作系统例程或其它形式的计算机程序代码。

[0029] 示例保密存储器访问逻辑150包括地址编码逻辑152(其包括元数据编码逻辑156和地址加密逻辑158)和地址解码逻辑162(其包括地址解密逻辑164和地址形成逻辑166)。保密存储器访问逻辑150还包括加密指针指令逻辑172(“EncryptBaseAddr”指令)、专门化指针指令逻辑174(“SpecializePtr”指令)、从加密指针加载指令逻辑176中(“LdEP”指令)、存储到加密指针指令逻辑178(“StEP”指令)、和传统存储器访问指令逻辑180(例如，MOV指令)。地址编码逻辑152和地址解码逻辑162可以在处理器指令(例如，172、174、176、178、180)中实现，或者作为单独的指令或一系列的指令实现，或者作为由诸如操作系统内核或虚拟机监测器等特权系统组件执行的较高级代码实现，或者作为指令集仿真器实现。如以下更详细描述的，地址编码逻辑152和地址解码逻辑162每个使用元数据(例如，有效范围、许可元数据、存储器分配大小、类型、位置、所有权、版本、标签值、特权级别(例如，用户或监管者)、密码术上下文ID等中的一个或多个)和秘密密钥(例如，秘密密钥116(1))对间接地址114进行操作，以便在存储器分配/访问级别保密间接地址114。同样，如以下更详细描述的，数据加密逻辑(如图2中所示)和数据解密逻辑(如图2中所示)可以使用间接地址的至少一部分和秘密密钥(例如，秘密密钥116(2))对(由间接地址114所引用的)数据或代码进行

操作，以便通过将数据/代码加密绑定到间接地址来保密由间接地址114所引用的存储器位置处的数据或代码。

[0030] 示例间接地址114被实施为寄存器112(例如，处理器102的通用寄存器)。示例秘密密钥116(1)-116(N)可由特权系统组件142的密钥创建模块148生成，并存储在寄存器112中的一个(例如，专用寄存器或机器专用寄存器(MSR))或由处理器102可读的另一存储器位置中。在一些实施例中，秘密密钥116(1)-116(N)可以被存储在仅由处理器可读的位置中。在其它实施例中，用于保密间接地址、数据和代码的秘密密钥116(1)-116(N)可以被存储在另一存储器位置中，诸如固件中、数据存储设备126或另一数据存储设备的保密部分中、或者适合于执行本文中描述的功能的另一种形式的存储器中。在一些实施例中，秘密密钥116(1)-116(N)可以跨保密通信信道来传输，并由执行者(诸如操作系统或虚拟机监测器，例如以下描述的特权系统组件142)来恢复。在虚拟机从一台机器迁移到另一台机器的虚拟化环境中，和/或在计算设备100上运行的虚拟机、过程或程序在间接地址和引用的数据和/或代码使用秘密密钥被保密之后开始睡眠/休眠模式并然后随后恢复的情况下，将需要重获和恢复秘密密钥。在这些情况下，秘密密钥可以在睡眠/休眠模式之前被存储或可能跨(保密)通信信道传输，并然后由执行者(诸如操作系统或虚拟机监测器，例如特权系统组件142)来检索/恢复。

[0031] 应当注意，本文中描述的实施例虑及了任何数量的秘密密钥要用于特定的程序。在一个示例中，相同的秘密密钥可以被用于程序中使用的所有间接地址。在另一示例中，不同的秘密密钥可以被用于与不同存储器分配相关联的每个间接地址，或者被用于与不同存储器分配相关联的存储器地址的每个预定义群。在仍有另外的实施例中，用于地址加密/解密的相同秘密密钥也可以用于加密被绑定到该地址的数据。在其它实施例中，一个秘密密钥可以用于地址加密/解密，而不同的秘密密钥可以用于被绑定到该地址的数据加密/解密。为了便于解释，本文中进一步描述的实施例提及“秘密地址密钥”或“地址密钥”指的是在存储器地址的加密和解密操作中使用秘密密钥，并且提及“秘密数据密钥”或“数据密钥”指的是在加密和解密数据的操作中使用秘密密钥。类似地，如本文中所使用的对“秘密代码密钥”或“代码密钥”的提及意在指的是在加密和解密代码的操作中使用秘密密钥。

[0032] 在存储器分配操作(例如，“malloc”)时(或在这期间)，存储器分配逻辑146为缓冲器分配存储器的范围，并返回间接地址114和元数据(例如，范围、许可元数据、存储器分配大小、类型、位置、所有权、版本、标签、特权级别、密码术上下文ID等中的一个或多个)。例如，存储器分配逻辑146可在间接地址114中(例如，在未使用/非规范位中)用随机化位或版本号来编码标签，或向指令供应元数据作为一个或多个分开的参数，其中(一个或多个)参数指定范围、代码许可信息、存储器分配大小、类型、位置、所有权、版本、标签、特权等级(例如，用户或监管者)、密码术上下文ID或一些其合适组合。说明性地，存储器分配逻辑146被实施在特权系统组件142的存储器管理器模块144中。存储器分配逻辑146启动地址编码逻辑152。地址编码逻辑152包括元数据编码逻辑156，其用诸如标签等元数据或用其它编码变体中的其它元数据(例如，范围、许可元数据、存储器分配大小、类型、位置、所有权、版本、标签值、特权级别、密码术上下文ID、其某种合适组合等)对间接地址114进行编码。地址编码逻辑152将元数据存储在间接地址114的未使用部分(例如，64位地址的非规范位)中。对于某些元数据或元数据的组合，间接地址114可以被编码在更大的地址空间(例如，128位地

址、256位地址)中,以适应元数据或元数据的组合的大小。

[0033] 在实施例中,地址编码逻辑152选择要在间接地址114中被加密和被编码的基地址的一部分(或片)。在其它实施例中,要被加密的基地址的片可以是先验已知的(例如,较低32位等)。地址加密逻辑158使用秘密地址密钥116(1)和地址调整来加密基地址的所选择片,如以下进一步描述的那样。在存储器访问操作(例如,读、写或执行操作)时,地址解码逻辑162解码先前编码的间接地址114。为此,地址解密逻辑164使用秘密密钥116(1)和地址调整来对间接地址114中编码的基地址的加密片进行解密,如以下进一步描述的那样。不作为要加密的片的一部分而被包括在内的基地址的较高地址位(UAB)可以在外部存储在表(例如,指针上下文表121)或寄存器(例如,对象上下文118)中。明文偏偏移的数量(例如25)被编码在间接地址114的较低位。在至少一个实施例中,它们可以被初始化为零。

[0034] 基于适当的操作(例如,地址解码逻辑162),间接地址114被返回到其原始(例如,规范)形式,以便恢复间接地址114的原始值(例如,真实的原始线性存储器地址)。为了在至少一个可能的实施例中做到这一点,地址解码逻辑162可以移除在间接地址114的未使用位中编码的地址元数据(例如,标签)(例如,将未使用位返回到它们的原始形式)。加密的片可以被解密,并与较高地址位串接。结果可以基于偏移来调节。如果间接地址114解码成功,则存储器访问操作成功完成。然而,如果编码的间接地址114已经被操纵(例如,被软件、无意地或被攻击者所操纵)使得其值落在由范围元数据指示的有效范围之外(例如,溢出缓冲器),则间接地址114将由于由地址解密逻辑164执行的解密过程而被损坏。损坏的间接地址将引发故障(例如,如果当从分页结构/页表中呈现时地址没有被映射,则引发一般保护故障或页故障)。可能导致生成故障的一个条件是稀疏地址空间。在这种情景下,损坏的地址很可能会到达未映射的页并产生页故障。以此方式,保密存储器访问逻辑150使得计算设备100能够提供针对缓冲器溢出攻击和类似受损的间接地址保密性。本文中公开的间接地址保密性技术的实施例也可被用于软件调试目的,或者用作访问控制机制,以防止软件访问软件不具有许可的存储器的区。此外,与其它缓冲器溢出减轻技术相比,所公开的间接地址保密性技术的实施例可以在没有任何二进制修改或者不需要重新编译传统代码的情况下操作。在一些情景下,本文中公开的间接地址指令可以在没有任何附加的存储器读/写或者没有任何附加的指令的情况下操作。此外,所公开的技术的实施例响应于能够读存储器和重写指针值的对手,以及能够创建/选择任意指针值的对手。此外,所公开的技术的实施例可以从非常小的存储器范围缩放到非常大的存储器范围,或者可以通过使用不同的编码指针在其它存储器范围内级联存储器范围。还有此外,所公开的技术的实施例对于动态存储器分配是有效的(例如,由于以编程方式内嵌创建范围编码指针的能力)。另外,所公开的技术的实施例可以被扩展以提供对数据的代码块(代码位置)访问控制。此外,所公开的技术的实施例与x86指令集的64位版本以及ARM、MIPS、PowerPC和其它处理器架构兼容,所述架构包括通过为包含地址的元数据保留地址范围的更小(例如,32位)架构和更宽(例如,大于64位)的地址位架构。

[0035] 如以下所述,所公开的技术的一些实施例利用地址解码逻辑的方面来支持传统代码兼容性。如本文中所使用的,“传统代码”可以指的是被设计成在较早的、或者现在已经过时的、或者不再被支持的计算机架构上工作的计算机代码的版本。例如,传统代码可包括最初为32位处理器开发但其现在运行在64位处理器上的软件。“传统代码”也指的是设计成不

使用或不适合使用专用指令来进行编码和加密间接地址的计算机代码版本，如本文所述。

[0036] 现在更详细地参考图1，计算设备100可以被实现为用于执行本文中描述的功能的任何类型的电子设备。例如，计算设备100可以被实施为但不限于，智能电话、平板计算机、可穿戴计算设备、膝上型计算机、上网本计算机、移动计算设备、蜂窝电话、手持电话、消息传送设备、车辆远程信息处理设备、服务器计算机、工作站、分布式计算系统、多处理器系统、消费电子设备和/或被配置为执行本文中描述的功能的任何其它计算设备。如图1中所示，示例计算设备100包括实施有保密存储器访问逻辑150的至少一个处理器102。

[0037] 计算设备100还包括存储器120、输入/输出子系统124、数据存储设备126、显示设备128、用户界面(UI)子系统130、通信子系统132、至少一个用户空间应用134和特权系统组件142(其示例性地包括存储器管理器模块144和密钥创建模块148)。在其它实施例中，计算设备100可以包括其它或附加组件，诸如通常在移动和/或固定计算机中找到的组件(例如，各种传感器和输入/输出设备)。此外，在一些实施例中，示例组件的一个或多个可以被结合到另一组件中，或者以其它方式形成另一组件的一部分。计算设备100的组件中的每个可以被实施为软件、固件、硬件或软件和硬件的组合。

[0038] 处理器102可以被实施为能够执行本文中描述的功能的任何类型的处理器。例如，处理器102可以被实施为多核处理器、其它多CPU处理器或处理/控制电路，或者多个相异的处理单元或电路(例如，CPU和GPU等)。处理器102具有多个寄存器112，其包括通用寄存器和专用寄存器。间接地址114和秘密密钥116(1)-116(N)被存储在寄存器112中。对象上下文118也可被存储在寄存器中。对象上下文118可以包括最高有效位或“较高地址位”(例如，57位间接地址中的较高22位)的片，其没有被编码在用于对象的间接地址(例如，114)中。如本文中将进一步描述的，在这种情景下，间接地址可以引用被存储在静态可寻址存储器区域(例如，快速访问存储器122)中的数据。对于数据被存储在静态可寻址存储器区域中的一些情景，可以从间接地址中生成数据的线性地址，而无需从例如动态可访问的表中动态地获得较高地址位或其它上下文信息。反而，可以从寄存器中获得较高地址位，并且可以在程序代码中静态地提供其它上下文信息。因此，存储在静态可寻址存储器区域中的数据可以比存储在需要动态查找上下文信息的其它存储器中的数据更快地被访问。然而，可能存在有某些实例，在其中即使数据存储在静态可寻址区域中，也可以动态检索其它上下文信息。当可期望的是将某些对象存储在静态可寻址区域中，但编译器不能为这些对象静态供应上下文信息时，这种情况可能会出现。

[0039] 计算设备100的存储器120可以被实施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设备。在操作中，存储器120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，以及操作系统、应用、程序、库和驱动程序。在一个或多个实施例中，存储器120包括指针上下文表121，其可以包含多个表条目。例如指针上下文表121的位置可以被指示为由诸如模型特定的寄存器(MSR)等寄存器定义的物理可寻址的表基。每个表条目可以包括与指向存储器中的对象的以密码方式编码的指针相关联的上下文信息。在一个实施例中，上下文信息可以包括大小元数据(例如，指示用于对象的存储器分配大小的值)、类型元数据(例如，对象的类型或类别)和许可元数据(例如，指向对象的指针的许可)。一个或多个表条目还可以包含与以密码方式编码的指针相关联的较高地址位(UAB)。例如，当较高地址位存储在寄存器(例如，118)中时，一个或多个其它表条目可以

对与以密码方式编码的指针相关联的较高地址位编码零。在至少一个实施例中，指针上下文表121可以由指向对象的以密码方式编码的指针的加密片来索引。每个表条目可以由相应以密码方式编码的指针的加密片来索引。然而，在其它实施例中，可以使用任何合适的技术来索引或以其它方式将表条目映射到其相应的指针，包括但不一定限于索引、其它指针、哈希表或表示在表条目与其相应指针之间的关系、连接、链接或关联的任何其它技术。此外，也可以使用其它索引。例如，足够唯一的以密码方式编码的指针的任何部分可以用于索引指针上下文表。例如，指针上下文表121可以由以密码方式编码的指针的至少一部分(诸如基地址的加密片、基地址的加密片加标签部分、整个以密码方式编码的指针等)来索引。然而，应注意的是，只要指针没有被软件修改，则整个以密码方式编码的指针可以适合作为索引。例如，如果软件更新指针中的偏移以指向对象内的字段，则指针将会改变。在这种情况下，通过整个指针对表进行索引可能会阻止找到表条目。如本文中进一步描述的，可以使用新的指令集架构(ISA)从用户空间中管理该表。然而，表中的冲突是可能的，因此操作系统负责处置这样的事件，这将在本文中进一步描述。

[0040] 存储器120的某些区域可以被定义为快速访问存储器122。快速访问存储器122代表寄存器(例如，112)可以指定区域的较高地址位的存储器的区域。例如，4GB的存储器区域可以被指派为快速访问存储器，其中4GB的数据(或代码)可以被存储，并且存储器访问可以由供应适当的上下文信息(例如，存储器分配大小、类型、许可)并且从寄存器(例如，对象上下文118)而不是存储器中的表(例如，指针上下文表121)中提出(pull)较高地址位的指令来执行。尽管如图所示，快速访问存储器区域可以是4GB，但是根据特定需要和实现，任何其它合适的存储器大小可以被指派为快速访问存储器。例如，如果对象与8字节边界对齐，则快速访问存储器区域可以是32GB。

[0041] 存储器120例如经由I/O子系统124以通信方式耦合到处理器102。I/O子系统124可以被实施为电路和/或组件，以便于对处理器102、存储器120和计算设备100的其它组件的输入/输出操作。例如，I/O子系统124可以被实施为或以其它方式包括存储器控制器集线器、输入/输出控制集线器、固件设备、通信链接(即，点对点链接、总线链接、电线、电缆、光导、印刷电路板迹线等)和/或其它组件和子系统，以便于输入/输出操作。在一些实施例中，I/O子系统124可以形成片上系统(SoC)的一部分，并且与处理器102、存储器120和/或计算设备100的其它组件一起结合在单个集成电路芯片上。

[0042] 数据存储设备126可以被实施为配置用于数据的短期或长期存储的任何类型的一个或多个物理设备，诸如例如存储器设备和电路、存储器卡、硬盘驱动、固态驱动、闪速存储器或其它只读存储器、作为只读存储器和随机存取存储器的组合的存储器设备或其它数据存储设备。

[0043] 显示设备128可以被实施为能够显示数字信息的任何类型的显示器，诸如液晶显示器(LCD)、发光二极管(LED)、等离子显示器、阴极射线管(CRT)或其它类型的显示设备。在一些实施例中，显示设备128可以耦合到触摸屏或其它人机界面设备，以允许用户与计算设备100交互。显示设备128可以是用户界面(UI)子系统130的一部分。用户界面子系统130可以包括多个附加设备，以促进用户与计算设备100的交互，所述设备包括物理或虚拟控制按钮或键、麦克风、扬声器、单向或双向静态和/或视频摄像机和/或其它设备。用户界面子系统130还可以包括诸如运动传感器、接近度传感器和眼睛跟踪设备的设备，其可以被配置为

检测、捕捉和处理涉及计算设备100的各种其它形式的人类交互。

[0044] 计算设备100还包括通信子系统132，所述通信子系统132可以被实施为能够实现计算设备100与其它电子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132可以被配置为使用任何一种或多种通信技术(例如，无线或有线通信)和相关联协议(例如，以太网、蓝牙™、Wi-Fi™、WiMAX、3G/LTE等)来实现这种通信。通信子系统132可以被实施为网络适配器，包括无线网络适配器。

[0045] 示例计算设备100还包括多个计算机程序组件，诸如用户空间应用134和特权系统组件142。用户空间应用134可以被实施为经由例如显示设备128或UI子系统130直接或间接与终端用户交互的任何计算机应用(例如，软件、固件、硬件或其组合)。用户空间应用134的一些示例包括文字处理程序、文档查看器/阅读器、网络浏览器、电子邮件程序、消息收发服务、计算机游戏、照相机和视频应用等。除了其它事物之外，特权系统组件142还促进了用户空间应用134与计算设备100的硬件组件之间的通信。特权系统组件142的部分可以被实施为能够执行本文中描述的功能的任何操作系统，诸如由微软公司的WINDOWS版本、由谷歌公司的ANDROID版本和/或其它版本。备选地或附加地，特权系统组件142的一部分可以被实施为能够执行本文中描述的功能的任何类型的虚拟机监测器(例如，类型1或类型2管理程序(hypervisor))。

[0046] 示例特权系统组件142包括多个计算机程序组件，诸如存储器管理器模块144和密钥创建模块148。特权系统组件142的组件中的每个可以被实施为软件、固件、硬件或软件和硬件的组合。例如，特权系统组件142的组件可以被实施为操作系统内核、虚拟机监测器或管理程序的模块。存储器管理器模块144将存储器120的部分分配到在计算设备100上运行的各种过程(例如，作为虚拟存储器地址的范围)。存储器管理器模块144被实施为例如加载器、存储器管理器服务或堆管理服务。密钥创建模块148创建秘密密钥116(1)-116(N)(例如，秘密地址密钥、秘密数据密钥、秘密代码密钥)，并将它们写到处理器102有权读取的一个或多个寄存器(例如，专用寄存器)。为了创建秘密密钥，密钥创建模块148可以执行例如随机数生成器或能够生成可执行本文中描述的功能的秘密密钥的另一算法。

[0047] 应注意的是，可以使用无数种方法来生成或获得本文中公开的实施例的密钥。例如，尽管密钥创建模块148被示为计算设备100的一部分，但是可以使用任何合适的认证过程从任何合适的外部源中获得一个或多个秘密密钥，以将密钥保密地传递给计算设备100，这可以包括生成作为那些过程的一部分的密钥。此外，特权系统组件142可以是可信执行环境(TEE)、虚拟机、处理器102、协处理器(未示出)或计算设备100中或保密地连接到计算设备100的任何其它合适的硬件、固件或软件的一部分。此外，密钥可以是“秘密的”，其意在表示所述密钥的值被隐藏、不可访问、模糊化或以其它方式对未授权的行动者(例如，软件、固件、机器、无关的硬件组件和人类)保密。

[0048] 图2是示出根据本公开的至少一个实施例的图1的保密存储器访问逻辑的应用的简化环境图。在一些实施例中，计算设备100可以在操作期间建立环境200(例如，原生和/或虚拟运行时或“执行”环境)。示例环境200中描绘的各种模块可以被实施为硬件、固件、软件或其组合。在环境200中，用户空间应用134(或特权系统组件142，例如，在加载用户空间应用134时)可以在计算设备100的操作期间时常发布存储器分配202。在被传到处理器102之前，通过特权系统组件142的存储器分配逻辑146可以根据需要翻译(例如，编译或解释)存

储器分配202。

[0049] 在处理器102中，响应于存储器分配202(例如，代替常规的“malloc”指令/函数调用)来执行地址编码逻辑152。在一个或多个实施例中，存储器管理器模块144(或存储器分配器)可以包括执行加密指针指令(EncryptBaseAddr)的堆分配器或栈分配器，所述指令执行地址编码逻辑152以及与对象相对于该对象被分配的区域的边界的大小相关的附加检查，以及用于在一些情景下存储上下文信息的表管理。此外，专门化指针指令(SpecializePtr)还可以执行地址编码逻辑152以及附加检查，以确保对象包含在由另一指针分配的存储器内，并且确保授予专用指针的许可不大于授予另一指针的许可。

[0050] 尽管常规的malloc指令简单地分配存储器并返回(不保密的)指针，但是地址编码逻辑152编码间接地址204，并返回编码的间接地址206，所述间接地址204包括元数据205，诸如随机化位的标签，或者在其它编码中可包括其它元数据(例如，范围许可信息、存储器分配大小、类型、位置、所有权、版本、特权级别、密码术上下文标识或密钥、或其任意组合等)。元数据可以以明文格式被嵌入在间接地址或指针(例如，标准64位寄存器或放大寄存器，诸如128位或256位，以适合更多的元数据)中，被嵌入在提供给指针加密/解密指令和数据访问指令的另一操作数内，被存储在存储器中的表中或控制寄存器中，或者经由它们的任意组合来提供。例如，标签值可以被嵌入在指针中，并且密码术上下文ID可以被存储在控制寄存器中。

[0051] 类似地，用户空间应用134或特权系统组件142可以时常发布存储器存储211，其可以由处理器102作为处理器指令来处置，所述处理器指令(例如，STORE、MOV指令)使用间接地址114从寄存器112(或其它存储单元)中读并写到存储器120或缓存。使用STORE指令作为示例，存储器存储指令逻辑170仅在以下动作之后才存储数据，成功执行地址解码逻辑162以对编码的间接地址206进行解码，并且还成功执行基于数据调整和秘密数据密钥116(2)的数据加密逻辑179以加密要存储在由间接地址204所指向的存储器位置的数据。地址解码逻辑162的成功执行是基于地址解密逻辑164的成功执行，所述地址解密逻辑164使用地址调整和秘密地址密钥116(1)来对编码的间接地址206的加密地址片进行解密。

[0052] 类似地，用户空间应用134或特权系统组件142可以时常发布存储器负载220，所述存储器负载220可以由处理器102作为处理器指令来处置，所述处理器指令(例如，LdEP或MOV指令)使用间接地址114从存储器120中读并写到寄存器112。使用来自加密指针的LOAD(LdEP)指令作为示例，从加密指针加载指令逻辑176仅在成功执行地址解码逻辑162以对编码的间接地址206进行解码之后才执行存储器访问。地址解码逻辑162的成功执行基于地址解密逻辑164的成功执行，所述地址解密逻辑164使用地址调整和秘密地址密钥116(1)来对编码的间接地址206的加密地址片进行解密。一旦返回间接地址204并且访问存储器120以从由间接地址204所指向的存储器位置加载数据，就可以通过基于数据调整和秘密数据密钥116(2)执行数据解密逻辑177来解密所加载的数据。数据解密逻辑177的成功执行取决于用于创建数据调整以解密数据的间接地址的部分以及用于创建数据调整的附加元数据(如果有的话)是否对应于由间接地址所指向的存储器位置的原始分配。

[0053] 虽然在图2中地址解码逻辑162被示为从存储到加密指针指令逻辑178和从加密指针加载指令逻辑176的独立模块，但是应理解，地址解码逻辑162可以被并入到指令逻辑178和/或176中，或者可以被实施为独立的指令集。此外，应理解，地址解码逻辑162可被并入到

其它类型的指令中或被其它类型的指令所引用，所述指令备选地或附加地有LdEP、StEP和MOV指令(例如，具有存储器操作数的算术指令、调用、JMP等)。例如，诸如调用和JMP的控制转移指令可以将要执行的代码的编码的指针地址加载到处理器的程序计数器寄存器(例如，指令指针)(例如，RIP，其中RIP是64位代码中的指令指针寄存器)。然后，可以由处理器查询指令指针寄存器，并因此，当前程序计数器地址将是编码的形式(相对于当前程序计数器位置的偏移)。

[0054] 如果地址解码逻辑162成功对编码的间接地址206进行解码，(该成功解码包括地址解密逻辑164成功对编码的间接地址中的加密的地址片解密)，则原始间接地址204被返回到特权系统组件142，并且存储器访问完成，或者程序执行开始于新的程序计数器位置(在控制流改变的情况下)。如果编码的间接地址206没有被成功解码，则生成故障。基于存储器存储211的成功完成或故障，适当的验证或故障信号213被返回到用户空间应用134。类似地，基于存储器加载220的成功完成或故障，适当的验证或故障信号222被返回到用户空间应用134。

[0055] 采用加密基地址(EBA)格式的以密码方式编码的指针

图3是简化流程图300，其示出了与使用采用加密基地址(EBA)格式的以密码方式编码的指针的存储器分配和访问相关联的软件单元、硬件和数据流的附加细节。具有EBA格式的指针用与存储在基地址的数据相关联的元数据的至少一项和基地址的加密片来编码。在一个或多个实施例中，诸如计算设备100的计算设备包括软件单元310、硬件320和数据流330。软件单元310包括存储器分配器312、存储器分配器调用功能314和存储器访问功能316。硬件320包括指针密码操作和编码单元322、数据缓存单元324和数据密码操作单元326。数据流330包括未处理指针331、非类型化上下文332、加密的非类型化指针333、类型化上下文334、加密的类型化指针335、加密数据337和解密数据338。

[0056] 可以由存储器分配器312调用加密指针(EncryptBaseAddr)指令302。例如，存储器分配器调用功能314可以通过为对象请求存储器来调用存储器分配器312(例如，malloc指令)，并且存储器分配器312可以包括调用EncryptBaseAddr指令302的堆分配器或栈分配器。EncryptBaseAddr指令302使指针密码操作和编码单元322使用非类型化上下文332来加密未处理指针331的片。未处理指针331作为EncryptBaseAddr指令302的寄存器操作数被传递，并且代表对象要被存储在其中的所分配的存储器的明文基地址。要加密的基地址的片包括地址中的预定位(例如，位3至34)。基地址中的较高位可以被存储在外部(例如，存储在存储器的表中或寄存器中)。非类型化上下文332也作为寄存器操作数来传递，并且可以包括例如上下文信息，诸如对于对象的存储器分配大小和许可元数据。非类型化上下文332不包括对于对象的类型元数据。非类型化上下文332可以用作加密算法(例如，块密码)的调整输入，以加密未处理指针331(例如，明文基地址)的片。指针密码操作和编码单元322还可以生成随机化位的标签，其可以用作对于基地址片的调整的一部分。在至少一个实施例中，指针密码操作和编码单元322通过将标签、加密基地址片和指针333中的偏移串接起来，生成加密的、非类型化的指针333。偏移可以被初始化为零。

[0057] 存储器分配器调用功能314可以基于先前编码的指针(诸如加密的、非类型化的指针333)来调用专门化指针指令304(SpecializePtr)。在至少一个实施例中，加密的非类型化指针333、非类型化上下文332和类型化上下文334可以作为SpecializePtr指令304的寄

存器操作数来传递。SpecializePtr指令304使指针密码操作和编码单元322使用原始上下文(例如,非类型化上下文332)对加密的非类型化指针333的加密的基地址片进行解密。基地址的较高地址位可以被检索并与解密的基地址片串接。如果解密的基地址片表示原始基地址的位3到34(即,因为地址是8字节对齐的),则初始化为“0”的三个位被串接在解密的基地址片的末端以获得原始基地址。加密的非类型化指针333的偏移可以被加到原始基地址,以获得用于分配的存储器内的子区域(或用于分配的存储器的相同区域)的专用基地址,要为所述专用基地址生成专用指针。例如,如果偏移已经改变,则专用地址将指向分配的存储器中的位置。如果偏移没有改变,则专用地址将指向与原始基地址相同的位置。例如,当期望对相同对象进行受约束访问时,这可能是所期望的,因为可以降低对于专用地址的许可。一旦计算出专用地址,则新的上下文(例如,类型化上下文334)可被用作加密算法(例如,块密码)的调整输入,以加密专用地址的片。在至少一个实施例中,类型化上下文334包括存储器分配大小、许可和对象的类型。类型化上下文334中的存储器分配大小可以相同于或小于非类型化上下文332中的存储器分配大小。类型化上下文334中的许可元数据可以与非类型化上下文332中的许可元数据相同或更少。指针密码操作和编码单元322还可以生成随机化位的标签,其可以用作对于专用地址片的调整的一部分。用于访问相同存储器区域的指针的标签应当匹配,以确保正确的数据解密。在其它实施例中,可以改为使用在加密的非类型化指针333中编码的标签。在至少一个实施例中,指针密码操作和编码单元322通过串接标签(新生成的标签或在指针333中编码的原始标签)、加密的专用地址片和指针335中的偏移来生成加密的类型化指针335。偏移可以被初始化为零。

[0058] 在用基地址片或专用地址片、元数据(例如标签)和偏移对指针进行以密码方式编码之后,以密码方式编码的指针可被用于访问存储在由指针引用的存储器位置中的对象。因此,加密的非类型化的指针333和加密的类型化的指针335两者都可以用于基于其相应许可在其相应位置处访问存储器。出于说明的目的,图3进一步示出了诸如加密的类型化指针335的专用指针如何能够用于访问存储器。例如,存储器访问功能316可以基于诸如加密的类型化指针335的先前编码的指针调用从加密的指针加载指令306(LdEP)。在至少一个实施例中,加密的类型化指针335和类型化上下文334可以作为LdEP指令306的寄存器操作数来传递。LdEP指令306使指针密码操作和编码单元322使用类型化上下文334作为调整来对加密的类型化指针335的加密专用地址片进行解密,并生成未处理指针336。专用地址的较高地址位(其与基地址的较高地址位相同)可以被检索并与解密的专用地址片串接。如果解密的专用地址片表示原始专用地址的位3至34(即,因为专用地址是8字节对齐的),则初始化为“0”的三个位被串接在解密的专用地址片的末端,以获得原始专用地址。加密的类型化指针335的偏移可以被加到原始专用地址,以获得未处理指针336,其是要被访问的存储器中的位置。如果不存在偏移,则未处理指针336代表原始专用地址。

[0059] 一旦生成了未处理指针336,LdEP指令306就执行访问控制检查,诸如边界和许可。如果访问控制检查成功,则来自所需存储器位置的对象(或数据)被加载到目的地操作数中并被解密。未处理指针336被用于从存储器中加载数据,诸如将数据从数据高速缓存单元324加载到寄存器中。在至少一个实施例中,使用将加密的类型化指针335绑定到数据的调整来加密被访问的数据。例如,用于加密数据的调整可以包括未处理指针336和编码在加密的类型化指针335中的标签。在一个或多个实施例中,附加上下文还可以在调整中被使用,

所述附加上下文包括例如密码术上下文标识符(或密码术上下文ID)(例如,分配给指针的64位随机或确定性唯一值)和/或其它可变长度元数据(例如,范围、位置、所有权、版本、特权级别等)。LdEP指令306使数据密码操作单元326使用相同的调整来对加密的数据337进行解密,以生成解密的数据338。地址元数据可以包括与间接地址相关联的随机化位的标签,以使标签对于对手不可预测。对手可能试图猜测标签值,使得对手能够访问由指针引用的存储器,而随机化标签值与用于生成标签值的确定性方法相比,可以使得对手不太可能会成功猜测该值。在一些实施例中,取代随机标签值或除了随机标签值之外,间接地址还可以包括版本号(或其它确定性值)以及时确定所引用的分配数据的当前所有权。即使对手能够猜测对于存储器的区域的当前标签值或版本号(例如,因为用于生成版本号的算法是可预测的),对手可能仍然不能正确地生成指针的对应加密部分,因为对手不具有对稍后将用于解密指针的该部分的密钥的访问。

[0060] 现在参考图4,示出了用于保密间接地址的示例过程400。过程400的部分可以由计算设备100的硬件、固件和/或软件来执行(例如,由执行地址编码逻辑152的处理器102来执行)。过程400响应于(例如,通过存储器管理器模块的)存储器分配而开始。在框410中,计算设备100获得间接地址、要分配的存储器的大小、以及编码间接地址所需的其它输入(例如,代码块标识符、指令指针、许可元数据、可能的类型元数据和/或用于调整的其它可能的元数据,如本文中所述)。在框412中,计算设备100确定调用代码(例如,发起存储器分配的代码)是否被授权访问在框410中接收的间接地址(例如,间接地址204)。为此,计算设备100可以通过验证用于调用代码的指令指针或调用者特权级别信息(其可以从例如存储器管理器模块144的堆管理器中获得)来执行访问控制检查。如果计算设备100确定调用代码未被授权访问间接地址,则发生故障(414)。如果计算设备100确定调用代码被授权访问间接地址,则计算设备100前进到框416。在框416中,计算设备100确定间接地址的未使用(例如,非规范)地址位,以执行元数据编码(例如,标签值)。为此,计算设备100可以仅使用间接地址的较高(例如,最高有效)未使用/非规范位。应注意的是,编码的地址不需要在架构上是非规范的。相反,未使用/非规范地址可以仅是由例如特权系统组件142留出的存储器的范围,以实现如本文中公开的地址编码。

[0061] 在框418中,计算设备100创建元数据(例如,标签/版本值),并将元数据存储于在框416中选择的间接地址的未使用/非规范位中。在框422中,计算设备100加密基地址的一部分,其中要加密的基地址的部分是地址中预定数量的位。在一个示例中,57位地址可以存储在64位寄存器中。要加密的基地址的部分(或片)可以包括32位。如果基地址与8字节边界对齐,那么最低的三个有效位可以设置为零,并且要加密的部分(或片)可以包括位3-34。剩余的位35-56可以单独存储。为了节省指针中的空间,当基地址的加密片被解密时,最低有效的三个位可以被移除并添加回来。尽管这是为各种块密码加密选项提供合适大小的一种可能的实现方式,但是基于特定需要和实现细节,可以为间接地址中的加密和编码预确定任何其它合适数量的位。实际上,可以利用具有不同输入大小的其它块密码配置,并且可以相应地调节间接地址中的加密片的大小。在一些实施例中,使用标签/版本元数据和诸如存储器分配大小和许可元数据之类的其它上下文信息作为地址调整,用秘密地址密钥(例如,秘密地址密钥116(1))加密所使用的位/规范地址的预定部分的位。在所示的实施例中,在间接地址中编码的元数据(例如,标签/版本元数据)将不被加密,因为在可调整块密码的情

况下,处理器使用编码的元数据作为调整(从而影响加密的位)。通常,可用作调整的上下文信息包括但不一定限于:存储在间接地址的未使用位中的数据、缓冲器大小的上限、作为缓冲器大小的上限选择的二的幂边界的指数、代码块标识符、指令指针数据、元数据中编码的许可信息、版本号(在重新分配/撤销先前分配给程序的指针时有用,版本可以由处理器在寄存器中维护)和/或本文中描述的其它元数据(例如,明文地址片、大小、存储器分配大小、类型、位置、所有权、标签、特权级别、密码术上下文ID或其任何合适的组合)。

[0062] 如本文中所使用的,"调整"除了通常的明文或密文输入和密钥(例如,秘密密钥116(1)-116(N))之外,还尤其可以指对块密码的第二输入。在至少一些实施例中,调整可以组成用于块密码的初始化向量(IV)的全部或部分。对间接地址的片进行加密使得计算设备100能够检测间接地址何时被非法改变,因为加密算法将导致非法改变的位产生对对手不确定的位的随机序列,这可能在使用非法改变的间接地址时导致故障。

[0063] 使用密码模式加密算法(诸如可调整块密码)、使用在间接地址调整中编码的元数据(例如,标签/版本值)来对要加密的间接地址的部分进行加密,所述间接地址的部分可以在间接地址的较低部分中(例如,最低有效32位,对于8字节边界对齐的地址的位3-35等)。可调整块密码的一些示例包括:异或加密异或(XEX)、Liskov、Rivest和Wagner(LRW),以及具有密文窃取的基于XEX的调整的码本模式(XTS)。可以使用其它位扩散方法,在其中密文中的任何单个位的改变都会导致跨整个解密明文的改变。如果需要的话,备选实施例可以通过使用非密码操作方法来权衡性能的保密性,所述方法仍然实现类似于块密码的合理位扩散和混淆。

[0064] 为加密而选择的密码可以采用硬件实现,其使用具有位可选或以其它方式可变块大小的算法(例如,具有可以被构造成利用调整的适当块大小的任何块密码或类似扩散算法),或者允许具有使用剩余的未加密位(例如,固定块大小之外的额外位)的调整的固定块大小的算法。具有位可选块大小的密码可以接受位长度参数作为输入(在某些情况下,与要加密的明文一起),所述位长度参数指定要加密明文的多少位。在某些情况下,位长度参数指定的位数与明文中的位数相同,而在其它情况下,位长度参数指定的明文中的位数小于整个明文的长度。密码使用加密密钥对明文位进行加密(加密密钥的长度可以与位长度参数相同或不同)。在加密密钥比位长度参数长的情况下,可以在密码中使用等于位长度参数的密钥的位的子集。密码使用一系列的逻辑操作(包括至少两个逻辑与操作和两个逻辑异或操作)从明文中加密和由位长度参数指定的位一样多的位。操作中的每个都是在明文的位和密钥的位两者上执行的;也就是说,对至少一个明文位和至少一个密钥位执行操作中的每个。以这种方式,可以实现明文与密文之间的混淆和扩散两者。根据本公开的这种位长度可参数化密码可被称为K密码。

[0065] K密码也可以被配置成接收调整输入,使得加密(和解密)基于加密密钥和调整输入。例如,调整输入可以被加到由K密码生成的密钥调度中的特定数量的圆形密钥(round key)。调整输入可以具有与圆形密钥相同的长度,并且可以以任何合适的方式配置,包括本文中参考各种实施例描述的调整输入。

[0066] 在一些实施例中,密码具有足够的位扩散,使得对加密的地址位进行的任何位改变在解密时将同等地影响(通过级联)所有的位位置。这为给定任何改变或边界违例的损坏地址提供了基础。使用这种方法,如果对手试图篡改元数据(例如,标签/版本值),则得到的

解码地址将被损坏。在64位地址空间中，地址损坏将以高概率导致故障，从而允许由特权系统组件142(例如，操作系统/可执行/VMM/备选模式/调试跟踪/管理处理器/子系统等)捕捉地址损坏(和指针访问或边界违例)。

[0067] 密码调整可以被扩展为包括代码块标识符，以提供对哪些代码块(例如，调用代码的块)被许可使用间接地址/指针来访问存储器的访问控制。此外，指令指针(其可以被称为“程序计数器”)信息或范围可以被编码为指针加密调整(本文中也称为“地址调整”)的一部分。指令指针信息可被用来限制哪些代码可以访问哪些数据的范围。例如，所有代码都可以安排在64位地址空间内的存储器的固定块内。具有类似访问许可的代码可以在相同块或范围内被分组。地址调整可以包括存储器的块的标识符，指令从该块正在执行。以这种方式，代码和数据可以被相关联，并且受访问控制，使得来自不同代码块的对手将不能使用加密指针访问受保护块的数据，因为如果错误的代码块标识符被用作地址调整，则加密指针将不能适当地解码。此外，当代码块调用(例如malloc)以向自身分配存储器时，malloc可以使用调用代码的存储器块返回加密地址，以确保对所分配的存储器的私有访问(只要所分配的存储器没有被释放并然后被重新分配给另一代码块)。备选地，可以在地址调整中使用标识调用代码的其它方法，诸如保护密钥。仍有此外，由处理器102用来控制对存储器的访问的用于读/写/执行访问的元数据可以用作加密地址位的地址调整的一部分。此外，指令指针本身可以表示为编码的指针(例如，加密的基地址(EBA)格式)。在这种情况下，元数据和加密的基地址位可以用作标识访问数据指针或请求存储器分配/指派的代码块的“调整”的一部分。在424，编码的间接地址可以被输出，并且控制返回到存储器管理器144。

[0068] 现在参考图5，示出了用于解码间接地址的示例过程500。过程500的部分可以由计算设备100的硬件、固件和/或软件(例如，由执行保密mov逻辑地址解码逻辑162、从加密指针加载指令逻辑176、和/或存储到加密指针指令逻辑178的处理器102)来执行。过程500响应于诸如读、写或执行操作(例如传统MOV指令、新的LdEP指令、新的StEP指令)等存储器访问操作而开始。当然，不同的处理器架构可以用不同的指令的名称或不同的选项/参数来指代“MOV”、“LdEP”和“StEP”功能。因此，所公开的实施例适用于跨不同架构的所有类型的“MOV”、“LdEP”和“StEP”功能，而不考虑用于指代这些功能的术语。此外，MOV指令、LdEP指令和StEP指令是数个示例，并且可以请求对读/写数据进行存储器访问的任何指令都可以应用本文中公开的地址编码和解码方法。

[0069] 在框510中，计算设备100获得编码的间接地址(例如，编码的地址206，其可以从寄存器112中获得)。在框512中，计算设备100确定在框510中获得的编码的地址是否具有未使用的或非规范的位。如果计算设备100确定编码的地址不具有未使用的/非规范的位(例如，地址不落入非规范的或以其它方式保留的地址的范围内，无论该地址范围是32位、64位、128位还是备选架构可能要求的任何范围)，则生成故障(514)。如果计算设备100确定编码的地址具有未使用的/非规范的位(例如，地址落入非规范的或保留的地址范围内)，则计算设备100前进到框516。在框516中，计算设备100使用与图4的框422中使用的加密算法相对的解密算法，并使用与图4的框422中的加密算法所使用的相同的密钥和调整来对编码的地址的加密部分进行解密。本文中示出并描述了用于解码和解密具有加密基地址(EBA)格式的间接地址的示例过程。

[0070] 如果解密的地址包含未使用/非规范位，则在框520中，计算设备100通过例如移除

未使用/非规范位，将解密的间接地址返回到其原始(例如，规范)形式。此外，对于采用EBA格式的间接地址，可以从操作数中静态检索明文较高地址位，或者从存储器中动态检索明文较高地址位，并将其与基地址的解密片串接起来。此外，如果地址与某个字节边界对齐，则对应于该字节边界的适当数量的位可以被串接到基地址的解密片以形成最低有效位。例如，如果基地址与8字节边界对齐，则可以串接三个位。最后，可以将间接地址中的偏移添加到解密和解码的基地址中。

[0071] 在框522中，计算设备100使用由框520输出的解码地址作为“真”(例如，虚拟或线性)存储器地址(例如，作为指针)。在框524，计算设备100确定在框522用作存储器地址/指针的解码地址是否是损坏的地址。如果解码的地址被损坏，则引发故障(514)。如果解码的地址没有被破坏，则在框526中，计算设备100使用解码的地址作为存储器地址/指针，成功地完成存储器访问操作。

[0072] 即使没有检测到损坏，所得地址对对手来说也不会是确定性的(并因此是可用的)。除了上述缓冲器溢出减轻技术之外，本文中还公开了指针地址编码技术的其它应用。例如，处理器指令可以受特权级别或调用者位置授权(例如，指令指针块或堆管理器范围)的限制。在程序代码本身可以控制其自身的指针和范围的情况下，可以添加附加的指令。这些指令可以使用较大的存储器分配大小元数据作为输入，并且如果执行该指令的代码属于拥有原始(超集)缓冲器指针(其可以由指令指针确定)的代码块，则可以产生落入较大缓冲器大小内的较小/相等大小的指针(更具约束性)。例如，存储器管理器模块144可以分配调用栈，并且向调用栈(例如，为栈指针)提供大的存储器分配大小指针。被授权作用于调用栈的代码段然后可以使用此处理器指令来编码指向在栈上隐式创建的缓冲器的子范围指针。编译器可以在这样做时自动扩充代码(因为执行了栈操作(创建局部变量等))，因此，甚至保护在栈上的各个数据结构或各个变量。也就是说，所公开的技术能够将缓冲器大小编码降到各个变量大小(例如，32位整数可以被编码为指向4字节缓冲器的指针)。

[0073] 类似地，通过基于原始代码块为目标/接收代码块生成新编码的指针，例如通过选择较小的缓冲器大小以用于分配到另一代码块，拥有指针的代码块可以使用类似的指令来将控制/所有权转移给另一/不同的代码块。这样的指令将把所得的缓冲器大小、原始数据指针和针对目标代码范围的编码的指针(针对目标代码范围分配了指针)作为输入参数。这样的指令可以使用调用代码块的指令指针作为调整来解码输入的编码的指针，如果输入的范围小于输入的编码的指针则降低范围，并且在产生输出的编码的指针时使用指向目标代码块/范围的输入的编码的指针作为调整的一部分(现在对于指定范围的程度，对新分配的代码块是可访问的)。其它输入参数可以是例如附加的元数据，诸如针对目标代码的读/写/执行许可(可能是原始的子集)。

[0074] 为了提供访问控制，指令指针或包括用类似指数标识的范围、调节和加密的间接地址位的编码的指令指针可以用作调整的一部分。指令指针可以类似地被编码为程序被存储在其中的存储器的可执行范围/缓冲器。当用作对于数据指针(例如，间接地址114)的调整时，指令指针可以控制程序代码的不同片段对数据的访问。此外，编码的指令指针值可由程序查询以用于RIP相对寻址。(例如，指令指针寄存器可以由程序读，并然后用于调用/跳转到程序的有效范围内的相对偏移，或者通过使用编码的指令指针值来读/写程序有效范围内的数据)。

[0075] 另外，数据指针可以由新的处理器指令(或操作系统例程)来创建和转换，允许数据指针(例如，间接地址114)的所有权被扩展到其它代码/程序范围。也就是说，数据指针的所有者程序/代码(其指令指针范围曾被用作针对数据指针的调整的一部分)可以调用例如操作系统例程(或处理器指令)，所述操作系统例程将产生可由另一程序/代码范围使用的新数据指针。在这种情况下，新的指令/操作系统例程将解码如本文所述那样编码的原始数据指针，并使用新的程序/代码范围元数据作为调整来对该范围重新编码，从而产生当从在新的地址范围中操作的指令指针访问时将适当解码的数据指针。新的指令/例程也可采用较小范围的编码作为参数，从而允许拥有原始数据指针的程序将数据缓冲器大小子集化到由新程序/代码范围可访问的存储器的较小区域。

[0076] 此外，64位栈指针可如本文中所述来编码，并且同样地，应由处理器102在栈推入(push)和弹出(pop)、调用和返回时相应地更新，从而符合栈的分配范围。在将MOV指令解码到栈指针之后，处理器102可以选取高速缓存栈指针的解密版本以用于直接存储器访问效率，然而，处理器102可以继续跟踪范围条件以确保不会发生栈溢出。

[0077] 通过指令指针相对寻址，程序计数器寄存器可以被读取并用于计算对于位置无关代码(PIC)和数据的偏移。指令指针也可以被编码，使得传统的指令指针相对位置独立代码仍然将正确运行。在这种情况下，编码的指令指针寄存器可以具有符合于在存储器中重新定位的程序代码和数据(包括文本段)的程度的范围。除了存储器访问之外，PIC程序还可以利用基于RIP相对寻址的调用和间接跳转(JMP)。这样，JMP和CALL指令可以被修改以处置编码的指针地址，将它们转换成类似于MOV指令的实际线性存储器地址。指针的边界之外的调用和指令指针相对跳转可能导致针对跳转/调用指令的损坏的目标地址，这很可能是由故障引起的。加载器还可以修复可重定位的符号表，以便为它们相应的代码段和存储器位置适当地编码函数指针的程度。此指令指针范围指针也可用作灵活的代码块/标识符调整，以访问具有其相关联代码的控制数据指针。此外，调用栈上的编码范围指针可以被加密，以提供调用与返回之间的控制流完整性，同时在返回解密时保留范围编码。并非6位指数元数据的所有值都被实际使用(例如，利用64位寻址)。例如，在64位寻址中，超过48的值将与非规范位冲突，并因此永远不会被利用。因此，高于48/57的指数值可以被重新定义，以指示可以定义调节区域的其它解释。应注意，数字57是基于五级分页的。高阶指数值的这种解释允许未使用/非规范地址位的备选使用以与所公开的地址编码机制共存。其它实施例可以使用这些未定义的值来选择性地确定调节数据是否存在。例如，超过48的指数值可以表示对于缓冲器不存在/不需要调节，并且只有2的幂是有效的，将2的幂设置回开始而不进行调节。通过选择性地确定对于编码的地址需要什么元数据，并且选择性地将可用地址位扩展到先前为调节值保留的空间中，这种方法能够更好地利用地址空间。

[0078] 转到图6，图6是根据本公开的至少一个实施例的以密码方式编码的指针的图。特别地，图6示出了采用加密的基地址(EBA)格式(本文中也称为“EBA指针”)的以密码方式编码的64位指针610(地址)。在某些情况下，EBA指针610可以是图4的过程400的输出。在所示的示例中，EBA指针包括监管者位(S位)601A、编码指示符位(E位)601B、动态上下文查找位(D位)601C、4位标签/版本部分602、32位加密的基地址(EBA)部分604和25位偏移部分606。对于典型的规范编码，与最高有效位相邻并且不是编码的地址的一部分的位具有与最高有效位相同的值，以便被认为是规范的(例如，常规的或未以密码方式编码的指针)。在用于57

位分页的64位寄存器中，最高有效的7位不会用作57个地址位的一部分。在EBA指针610的实施例中，七个最高有效位可以包括三个较高位601A-601C和标签/版本部分602中的四个位。

[0079] 指针的两个较高位(例如，601A、601B)可以被编码以指示该指针是以密码方式编码的指针而不是常规的指针，从而这两种类型的指针可以潜在地被使用在单个地址空间中。例如，监管者位601A被用于指示编码的线性地址是在监管者地址空间(例如，“1”)中还是在用户模式地址空间(例如，“0”)内。编码指示符位601B可以指示指针是以密码方式编码的指针还是传统指针(即，不是以密码方式编码的)。例如，编码指示符位601B可以被设置为监管者位601A的相反值，以指示指针被以密码方式编码，或者可以被设置为与监管者位601A相同的值，以指示指针未被以密码方式编码。在其它实施例中，可以不使用较高位601A和601B。相反，通过在标签/版本部分602中编码特殊值(例如，全为1，全为0)来指示指针未被以密码方式编码，可以在没有专用位的情况下实现传统编码。标签/版本部分中的任何其它值可以指示指针被编码为具有标签/版本部分的以密码方式编码的指针。因此，两种类型的指针(例如，常规的和具有标签/版本部分的以密码方式编码的)可以潜在地被用在相同的地址空间中。在仍有的其它实施例中，如果例如概念没有被实现为与传统程序可兼容，则传统编码可以被完全消除。

[0080] 动态上下文查找位601C可被用于指示如何获得上下文信息，从而用于地址调整以解密指针中的编码的线性基地址片和/或用于数据调整中以对加密的数据进行解密。例如，动态上下文查找位601C可以指示上下文信息是要从指针上下文表(其中每个表条目包含与某个指针相关联的上下文信息)中动态获得，还是要从嵌入在代码流中的指令操作数(其在使用以密码方式编码的指针访问存储器时被供应)中静态获得。此外，当动态获得上下文信息时，也可以从与指针相关联的指针上下文表条目中获得未在EBA指针610中编码的较高地址位。当静态获得上下文信息时，可以从单独的寄存器中获得较高地址位。

[0081] 标签/版本部分602可以用在地址调整和数据调整中，并且对于由特定有效指针所引用的存储器分配或存储器区域可以是唯一的数字或值。在一个示例中，标签/版本部分是位(例如，4位或任何其它合适的大小)的随机化串。在另一示例中，标签/版本部分602是确定性值，诸如在每次为特定存储器分配生成指针时递增预定量的序列号。版本化可以以任何合适的粒度(例如，通过程序、通过特定的线性地址等)来实现。标签/版本部分为地址调整和/或数据调整的加密增加了多样性。然而，使用调整中的上下文信息或者使用调整中的上下文信息和较高地址位而不具有标签/版本部分，地址加密和数据加密可能是足够保密的。此外，尽管具有随机生成的位的标签值对于攻击者来说可能更难以学习，但是在具有其所引用的线性地址的加密片的指针(诸如EBA指针610)中，版本号可能就足够了。这是因为加密保护是由线性地址的加密片提供的，并且版本号可以使用较少的资源来生成。

[0082] 在EBA指针610中，数据的线性基地址被编码。64位指针可能不具有足够的空间来编码数据的整个基地址。因此，基地址的片被加密并存储在指针中。在这个示例中，基地址的3到34位被加密，并构成EBA指针610的加密的基地址部分604。没有嵌入在指针610中的较高地址位(UAB)与指针分开存储。当解码指针时，可以从与数据(或代码)相关联的表条目(例如，在指针上下文表中)、从单独的寄存器、从数据访问指令中的操作数、或者从覆盖多个对象(或代码段)的寄存器中提出较高地址位。指针中剩余的25位包括数据内的偏移。偏移指定了EBA指针610所指向的基地址之外的字节数量。典型地，偏移部分606被设置为零。

然而，应明白，指令可以被定义为基于特定的需要和实现将偏移部分初始化为任何期望的数量。偏移可以由软件操纵例如用于指针算术或其它操作。EBA指针610的加密的基地址部分604(例如，在所示示例中为32位)可以用小的可调整块密码(例如，以32位块大小的SIMON、SPECK或可调整K密码，或其它可变位大小的可调整块密码)加密。

[0083] 此外，尽管EBA指针610是基于对EBA部分604使用32位来示出和描述的，但是指针格式并不意在被如此限制。可以基于容易获得的32位块加密密码来选择要被加密的地址片。然而，也可以改为使用利用任何其它块大小(例如，27、16、可变的等)的加密密码。如果加密的基地址片位的数量被调节(向上或向下)，则剩余的地址位(例如，较高地址位)可以被相应地调节。例如，如果加密的基地址部分被调节为16位，则较高地址位可以被调节为38位，并且指针中可用的16位可以用附加的偏移位和/或附加的元数据(例如，许可、大小等)来编码。

[0084] 当处理器以密码操作模式运行并使用编码的指针(地址)(例如，以与图6的指针610相同或类似方式格式化的指针)来访问存储器以取得实际的线性/虚拟地址存储器位置时，处理器采用编码的地址格式，并使用在指针中编码的元数据(诸如标签/版本部分602)和秘密密钥来解密EBA部分(例如，图6的604)。在一些实例中，在指针中编码的元数据和/或其它元数据或上下文信息可以被包括为用于解密EBA部分604的调整的一部分(本文中也称为“地址调整”)。如果地址解密不正确，则处理器可能会由于试图使用损坏的线性/虚拟地址进行存储器访问而导致一般的保护故障(#GP)或页故障。

[0085] 如本文中所使用的，“上下文信息”意在包括与以下各项相关的任何元数据或其它信息：存储器分配、其相关联的存储器地址、其相关联的指针、为其分配存储器的软件、和/或所分配的存储器的内容。例如，上下文信息可以包括但不限于指示所分配的存储器的字节的大小、包含与存储器地址相关联的随机化位的标签、指示对存储在所分配的存储器中的数据的访问许可的许可信息、可被用于重新分配/撤销先前分配给程序的指针的指针版本号、存储在所分配的存储器中的数据的类型或类别、指示为其分配存储器的软件的用户或监管者模式的特权级别、以及包括对于存储器地址的随机化或确定性唯一值的密码术(以密码方式的)上下文标识符。上下文信息还可以包括编码的基地址的较高地址位，在一个或多个实施例中，其可以用作地址调整的至少一部分。一个或多个指针编码实施例可以使用上下文信息的任何单个项作为调整(地址调整或数据调整)的一部分，或者可以使用上下文信息项的任何合适组合。

[0086] 上下文信息可以存储在任何类型的存储设备中，这可以基于特定的需求和实现。如本文中先前所述，与指针相关联的上下文信息可以存储在指针上下文表(例如，121)的表条目中。在其它示例中，上下文信息的一个或多个项可以嵌入标准大小(例如，64位)的指针(诸如EBA指针610)中。在这种情景中，上下文信息可以代替标签/版本部分602或者与标签/版本部分602一起存储在未使用/非规范位中。对于上下文信息的存储的其它示例类型包括但不一定限于将上下文信息嵌入到已经被放大以适应更多或更大调整的指针(例如，128位指针、256位指针等)中，将上下文信息嵌入在提供给指针加密指令和数据访问指令的另一操作数中，和/或将上下文信息存储在控制寄存器中。控制寄存器可以由指令自动选择以用作密码术输入(例如，如果只有一个寄存器存储这种类型的调整的话)。否则，可以使用一些其它指令操作数来选择控制寄存器，所述指令操作数诸如指针自身中的字段或者被供应有

为特定操作数编码实施例配置的数据访问指令(例如,特殊加载和存储指令)的上下文操作数中的字段。例如,访问指令的索引字段可被用于选择包含用于数据(或代码)的密钥或调整的寄存器。通常,对于仅在切换上下文时更新的调整,用于调整的(一个或多个)项目可能特别适合于存储在寄存器中。与特定指针相关联更紧密的其它调整可能更适合于嵌入在指针中或传递到指令操作数中。然而,如先前所述,上下文信息的任何项都可以嵌入或存储在任何类型的存储设备中。

[0087] 应注意的是,大小和版本元数据在保密性攻击检测中可能特别有利。当某个存储器已经被释放后再被分配给第二指针时,就可能发生释放后使用攻击。如果原始指针再次被使用,而且指向重新分配的存储器内的某个位置并改变数据,这可被用于读或损坏存储器。通过使数据加密/完整性依赖于指针版本,将元数据版本化可以允许在使用错误的版本(旧指针)时防止和/或检测这种攻击(例如,使用以先前释放的版本来编码的指针将不会适当地解密使用以新版本编码的指针分配并写入的数据)。其它实施例可以基于版本值来选择不同的数据加密/解密密钥,例如,通过使用版本作为对密钥表的索引。

[0088] 图7是示例性上下文信息700的图,其可被用于对具有加密的基地址的指针(例如EBA指针610)进行编码和解码。在该示例中,上下文信息700包括较高地址位(UAB)字段702、存储器分配大小字段704、类型字段706和许可字段708。在至少一个实施例中,取决于特定的存储器分配(例如,不可静态寻址),上下文信息700可以是与指针分开存储的一个表条目,并且可以从存储器动态访问该表条目,以检索加密或解密线性地址的片以及可能由该线性地址所引用的数据所需的上下文信息。在一些情景中,较高地址位可以存储在用于静态访问的控制寄存器中,而其它上下文信息可以存储在用于静态访问的存储器操作数中或用于动态访问的存储器中。

[0089] 在一个或多个实施例中,以密码方式编码的指针的至少一部分可以被用作对包含动态加载的上下文信息的指针上下文表的索引。例如,指针610的加密的基地址部分604可被用于索引指针上下文表,以获得包含对于指针610的上下文信息的表条目。该技术通过消除对索引指针上下文表的专用指针位的需要来保存指针610中的指针位。应当注意,诸如上下文信息700的上下文信息的格式可以取决于上下文信息是存储在要动态查找的表条目中还是嵌入在要静态检索的程序代码中而变化,所述这两者将在本文中进一步描述。

[0090] 存储器分配大小字段704可以包含29个位,并且可以用由指针610所引用的存储器位置处的数据的精确大小来编码。因此,存储器大小可以指定为字节粒度或更粗的粒度,以节省空间。存储器分配大小字段704可以被限制为25个位,以匹配指针610中的偏移部分606的宽度。当执行数据访问时,由处理器检查指针610中的偏移部分606,以确保其不超过存储器分配大小字段704中的值。

[0091] 类型字段706可以包含十四个位,并且被用于指定不同类别的编程语言。在某些编程语言中,错误可能会因使用指向特定类的对象的指针来获得不同类的数据(或对象)并然后尝试基于不同类来处理数据而产生。因此,当为数据分配存储器并生成指针时,数据的类型(例如,对象类型)可以被确定并被存储为用于指针的上下文信息700的一部分。当指针在程序中被访问时,作为上下文信息700的一部分指定的类型信息可以与程序中指定的预期类型ID相匹配,以减轻类型混淆弱点。

[0092] 许可字段708可以包含三个许可位(XWR)。这是在诸如读、写和执行的分页许可之

上的一层许可。许可可以基于对分页许可和上下文许可执行的交集操作来确定。例如，如果许可位指示指针仅可读，那么即使页表表明所映射的存储器可写，则写访问也将被处理器拒绝，因为选取了最有约束性的许可。在另一示例中，如果许可位指示存储器可读且可写，但页表表明它仅可读，则写访问可被拒绝。

[0093] UAB字段702包括要与来自指针610的解密的基地址片串接的较高地址位。在该示例中，加密的基地址部分604可以仅包含线性地址的32位(加密的)(例如，位3至34)，上下文信息700中的UAB字段702可以包含22个位，并且最低的三个位(位0-2)可以被设置为“0”，以将基地址与8字节边界对齐。最低的三个位不能在指针中被编码。一旦加密的基地址部分604被解密，可以从存储在存储器中(例如，在表条目中)的上下文信息700中获得或者从寄存器中获得UAB字段702中的位，并且将UAB字段702中的位与解密的基地址片串接。设置为“0”的较低的三个位也可以串接在地址的末端。

[0094] 如果要静态检索上下文信息，则上下文信息可以被嵌入在代码中(例如，嵌入在指令操作数中)，或者如果要动态检索上下文信息，则上下文信息可以存储在存储器中的表中。应当注意，如果较大的寄存器(例如，128位寄存器、256位寄存器等)被用于以密码方式编码的指针，则整个基地址可以编码在指针中。然而，对于64位寄存器，加密的基地址部分(例如604)可能太窄而不能指定完整的基地址。因此，在存储器访问请求(例如，加载、存储)期间，可以从另一存储器位置提出较高基地址位。在某些情景中，可以从寄存器中静态获得较高地址位，而可以从存储器中动态检索上下文信息的部分或全部。如果线性地址是静态可寻址的但是软件的编译器确定对于软件中的指针的每次调用针对指针的上下文信息由编译器是不可探查得出的话，这种情况可能会发生。在这种情景中，表条目中的上下文信息700中的UAB字段702可以被固定为零，因为要与解密的基地址片串接的较高地址位可以由控制寄存器隐式供应。

[0095] 某些定义的存储器区域可以是使用静态上下文信息可寻址的。如本文中使用的“静态上下文信息”意在指的是经由指令操作数或单独的寄存器(例如，控制寄存器)可访问的上下文信息。这种存储器区域可以被指派为快速访问存储器(例如，快速访问存储器122)，其中可以存储数据(或代码)，并且可以快速执行对该数据(或代码)的存储器访问，而不需要对上下文信息执行更耗时的表查找。在示例说明中，指定使用静态上下文信息可寻址的存储器区域的基地址的寄存器可以被命名为STATIC_REGION_BASE。存储器区域的基地址可以由在诸如例如可信任运行时、操作系统、虚拟机管理器等的监管者模式下运行的软件来设置。从该寄存器，静态可寻址区域的排它限制可被定义为：

STATIC_REGION_END=STATIC_REGION_BASE+STATIC_REGION_SIZE=2^(基对齐移位值+加密的基地址片的位宽)，其中“位宽”是以位为单位的长度值。

[0096] 在本文示出的一个示例中，基对齐移位值可以是三个位，以对齐8字节的边界。STATIC_REGION_BASE的值应与STATIC_REGION_SIZE的值对齐，以许可仅通过串接来自STATIC_REGION_BASE的位的片来计算最终的基地址，所述位比由加密的基地址片加上其基对齐移位值所定义的位更有效。在一些实施例中，对于某些存储器分配请求，可以向存储器分配器提供提示，以指示所请求的存储器将被频繁访问。在这种情况下，存储器分配器可以用静态区域内的地址对指针进行编码。

[0097] 如图6-7中所示，使用上下文信息的加密的基地址格式提供了几个优点。首先，EBA

格式允许在不扩展指针的情况下进行以密码方式的计算,因为指针扩展可能带来不期望的存储器开销。第二,使用EBA格式的实施例避免了在调用具有加密的指针的函数时传递附加的元数据,因为要求传递附加的元数据会引入软件启用(enabling)挑战。第三,本文中描述的EBA格式使用8字节基对齐,但是字节粒度限制也是可能的。较大的对齐要求会导致浪费的空间。第四,用于访问存储器的区别许可对于每条指令是可能的,这提供了存储器安全性。最后,在本文中将进一步讨论的多租户环境中,EBA格式导致非常有限的上下文切换开销。

[0098] 转向图8,图8是示出根据至少一个实施例的用于解码采用EBA格式的以密码方式编码的指针的硬件组件的图。在该示例中,示出了基于相关联的上下文信息710对EBA指针610的解码。地址密码操作单元(address cryptography unit)820被用于解码指针610以获得解密的线性地址814。诸如地址生成单元(AGU)822、转译后备缓冲器(TLB)824和页丢失处置器(PMH)826的附加硬件单元例如将解密的线性地址814转换成物理地址,以用于访问由指针610所引用的分配的存储器中的数据。

[0099] 地址密码操作单元820包括用于解密指针610的加密的基地址部分604的以密码方式的算法。EBA部分604是指向(或引用)分配的存储器的存储器地址的加密片。在至少一个实施例中,地址密码操作单元820包括块密码,其基于地址密钥818和第二输入(本文中也称为“调整”或“地址调整”)812来执行对加密的地址片的解密。一般来说,块密码是一种加密算法,它使用对称密钥对数据的块进行加密,以此方式提供财产机密性,诸如位扩散和混淆,这对防止对手可预测地操纵解密的地址片是重要的。取决于特定的块密码要求,至少一些块密码实施例包括初始化向量(IV),其是随机的、伪随机的或不重复的固定大小的输入。对于使用要求初始化向量的块密码的实施例,地址调整812可以组成初始化向量的全部或部分。在一些实施例中,地址调整可以包括来自指针的上下文信息(例如,标签/版本部分602)、来自诸如存储器中的表的存储结构的上下文信息、来自另一寄存器的上下文信息、和/或来自指令操作数的上下文信息的一个或多个项。

[0100] 本文中公开的实施例允许EBA指针编码的变化,并因此允许各种调整。调整(地址调整或数据调整)可以包括随机值、用于不同存储器分配的确定性不同的值、不能被随机化或生成为任意值的语义含义、或其任何合适组合。随机性和/或确定性不同的值可以被用作调整(或作为调整的一部分)来使密码操作多样化。这种调整在本文中被称为“密码术上下文标识符”或“密码术上下文ID”,并且可以采取随机调整(或初始化向量)、由可信软件生成和控制的确定性调整(或初始化向量)或随机密码操作密钥的形式。然而,某些调整可具有语义含义,其不能被随机化或作为任意值生成。例如,上下文信息中的存储器分配大小字段被CPU用来选择为其生成指针的存储器大小。因此,存储器分配大小对每个存储器分配大小值都有定义明确的解释。

[0101] 在图8中所示的实施例中,从指针610获得地址调整812的一部分,并且从其它存储设备位置(例如,寄存器、其它存储器、指令操作数)获得其它部分。地址调整812包括来自EBA指针610的标签/版本部分602和上下文信息,诸如字段704中的存储器分配大小元数据、字段706中的类型元数据和字段708中的许可元数据。在一些情景中,较高地址位702、密码术上下文ID 715和/或其它可变长度元数据713也可以用作地址调整812的一部分。如果不能在程序代码中静态地提供上下文信息,则可以从存储器中的表(指针上下文表121)获得

上下文信息710。然而，如果可以在程序代码中静态地提供上下文信息，则可以在指令的操作数中提供存储器分配大小元数据704、类型元数据706和许可元数据708，并且可以在寄存器中提供较高地址位702。其它可能的调整包括密码术上下文ID 715(其可以是存储在寄存器中的随机生成的值)以及其它可变长度元数据(其可以存储在诸如指针上下文表121的存储器中)。然而，应注意的是，基于特定实现，任何的调整都可以存储在任何合适的存储设备选项中，所述存储设备包括指令操作数、寄存器、和/或表或存储器中的其它存储结构。可以以如本文中先前所述的任何合适方式(例如，关于图1的特权系统组件142和密钥创建模块148)生成或获得地址密钥818。

[0102] 任何合适的块密码的以密码方式的算法都可以被实现为地址密码操作单元820。例如，小的可调整块密码(例如，可以使用32位块大小的SIMON、SPECK或可调整K密码，或者其它可变位大小的可调整块密码)。高级加密标准(AES)提供了各种块密码，这些块密码可以以多种方式实现，以实现诸如密文804的数据块的加密/解密。例如，基于AES异或加密异或(XEX)的具有密文窃取的可调整码本调整模式(AES-XTS)可能是合适的。在其它实施例中，可以实现AES计数器(CTR)操作模式。

[0103] 一旦EBA部分604的解密成功，则地址密码操作单元820可以基于线性基地址的解密片来生成解密的线性地址814。通过将线性基地址的解密片与较高地址位702以及潜在的一定数量的对齐位(例如，对于8字节边界对齐的3个位)串接起来计算的解密的线性基地址。通过添加偏移608来计算由指针610所引用的特定字节的线性地址。要串接的位的顺序包括最高有效位到最低有效位：较高地址位702、解密的线性基地址位、对齐位。此外，解码的线性地址可以通过复制每个未使用/非规范位中的最高有效位来形成。在该示例中，未使用/非规范位包括与最高有效位相邻的下六个位。

[0104] 图9是示出根据至少一个实施例的用于将上下文信息嵌入到编译的代码中的编译器流程900的简化框图。如流程900中所示，EBA格式许可编译器920静态地计算上下文信息，并将其精确地嵌入到程序中所需的点，以提高效率。在编译器流程900中，软件编程代码910可以被提供给编译器920。产生编程代码的编程语言可以是基于特定需求和实现的任何合适的编程语言，包括例如，C++、Rust、Swift等。在许多情景中，编译器920能够提取关于程序的特定部分期望访问哪些对象的信息。所述所提取的信息可以包括上下文信息922，诸如为对象分配的存储器的大小、针对指向对象的指针的许可、以及对象的类型。在该示例中，编译器920提取用于对象X的上下文信息922，并且静态地将上下文信息922嵌入到可执行代码中访问对象X所需的精确位置中。在图9的示例中，上下文信息922作为静态嵌入的上下文信息908被静态地嵌入到函数B 906中，其可被用于从函数B 906访问对象X。

[0105] 编程代码910可以包括函数A 902和函数B 906，其中函数A 902调用函数B 906(或者以其它方式将控制传递给函数B 906)。在该示例中，函数A 902请求为对象X分配存储器，为该对象X生成以密码方式编码的指针904。指针904被传递给函数B 906，其中对于对象X的上下文信息已经被编译器920静态嵌入。因此，函数B 906可以使用指针904来进行对象X的存储器访问，而不需要执行动态上下文信息查找。

[0106] 然而，如果需要基于非默认的扩展，则可能仍然需要动态上下文信息查找。基于非默认的扩展指的是用于基地址的较高地址位。即使编译器能够推导出静态上下文信息，指针仍然可以指示要求动态上下文信息查找来加载非默认的基扩展。如果对象已被分配在使

用存储在对应寄存器中的默认较高基地址位可访问的存储器的范围之外(例如,在快速访问存储器122之外),则这是必需的。指针610中的D位601C被编码以指示相关联的对象已经被分配在始终需要动态上下文查找的区域中。

[0107] 现在参考图10,示出了使用采用EBA格式的以密码方式编码的指针来编译软件程序的示例过程1000。过程1000的部分可以通过运行有编译器程序的计算设备的硬件、固件和/或软件来执行。在一些情景中,计算设备100可以运行编译器程序(例如,920),以使用EBA格式的以密码方式编码的指针来编译软件程序。在其它情景中,这样的软件程序可以由运行在单独的计算设备上的编译器编译,并且编译的代码(例如,可执行代码)可以被分发以用于在诸如计算设备100的计算设备上执行。

[0108] 在1002,启动软件程序的编译。在1004,编译器标识用于对象的存储器分配指令。在1006,做出关于是否将要在快速访问存储器区域(例如,122)内为对象分配存储器的确定,在所述快速访问存储器区域中,使用存储在对应寄存器中的默认基扩展(即,较高基地址位)可访问存储器。

[0109] 如果确定的是要在快速访问存储器区域内为对象分配存储器,则在1008,做出确定,所述确定关于是否可以由编译器为程序代码中的后续存储器访问指令处的对象静态地确定上下文信息。如果确定编译器可以静态地确定其需要执行的访问的上下文信息,则在1010,操作数被嵌入在指针生成指令(例如EncryptBaseAddr)中,其指示不需要针对上下文信息的表条目。

[0110] 在1012,编译器标识用于对象的存储器访问指令。在1014,编译器提取要由存储器访问指令访问的对象的上下文信息(例如,大小、类型、许可)。在1016,编译器在可执行代码中的存储器访问指令处静态嵌入所提取的上下文信息。

[0111] 再次参考1006,如果确定的是要在快速访问存储器区域之外为对象分配存储器,或者如果确定的是要在快速访问存储器区域内为对象分配存储器但在1008确定的是用于对象的后续存储器访问指令的上下文信息不能由编译器推导出,则在1020,操作数被嵌入在指示需要用于上下文信息的表条目的指针生成指令(例如EncryptBaseAddr)中。在1022,编译器标识用于对象的存储器访问指令。在1024,编译了存储器访问指令以允许动态查找与对象相关联的上下文信息。应注意的是,如果可以为存储器访问中的一些(但不是所有的存储器访问)推导出上下文信息,那么仍然需要用于上下文信息的表条目。因此,编译器可以针对用于对象的每个存储器访问指令适当地执行操作1012-1016或1020-1024,并且可以针对软件程序代码中的每个存储器分配请求指令执行操作1004-1024。

[0112] 在一个或多个实施例中,编译器可以使用若干指令以操作化软件程序代码中的采用EBA格式的以密码方式编码的指针的生成和使用。在一个或多个实施例中,这些新指令可以包括加密指针(EncryptBaseAddr)指令、编码上下文信息(EncodeCtx)指令、专用指针(SpecializePtr)指令、从加密的指针加载(LdEP)指令、以及存储到加密的指针(StEP)指令,现在将分别对其进行描述。应注意,为了便于描述,对在分配的存储器中存储和访问的数据做出了参考。这种数据可以包括对象、数组和任何其它数据存储结构。此外,本文中描述的指令也适用于可以在分配的存储器中存储和访问的代码。

[0113] EncryptBaseAddr指令是指针生成指令(例如1010)的一个示例,其用于在分配存储器时生成采用EBA格式的以密码方式编码的指针。指针生成指令(例如EncryptBaseAddr)

的目的是将未处理指针转换成EBA编码格式，并在需要时插入用于动态上下文信息查找的对应表条目。EncryptBaseAddr指令的一种示例格式如下：

EncryptBaseAddr rd、rs2、imm8

rd:包含要以密码方式编码的未处理指针的寄存器操作数(例如64位)

rs1:包含上下文信息的寄存器操作数(例如64位)

imm8:设置为用于指示是否要将包含上下文信息的表条目插入到指针上下文表中的值的立即操作数(例如,8位)。

[0114] 如果对象被存储在静态可寻址存储器区域中(例如,以用于快速访问),并且编译器知道指针将仅在静态已知上下文的地方被使用,则立即操作数imm8可以被设置为预定义值(例如,“0”)以指示表条目不是必需的。在这种情景中,寄存器操作数Rs1包含与对象相关联的上下文信息,诸如大小、许可和可能的类型。在另一种情景中,立即操作数imm8可以被设为不同的预定义值(例如,“1”),以指示具有上下文信息的表条目要被插入在指针上下文表中。

[0115] EncodeCtx指令使函数能够从上下文信息的多个项中编码上下文值。指令从包含上下文信息的所提供操作数输入中生成编码的上下文值。在EncodeCtx指令的一个实施例中,通过使用包含在操作数输入的每个中的上下文信息生成单个64位值来编码上下文值。编码的上下文值可以用作EncryptBaseAddr指令的输入(例如EncryptBaseAddr指令的rs1)。EncodeCtx指令的一种示例格式如下：

EncodeCtx:rd、rs1、rs2

rd:包含要编码的上下文信息的第一项(例如大小元数据)的寄存器操作数(例如64位)输入

rs1:包含要编码的上下文信息的第二项(例如类型元数据)的寄存器操作数(例如64位)输入

rs2:包含要编码的上下文信息的第三项(例如许可元数据)的寄存器操作数(例如64位)输入。

[0116] SpecializePtr指令可以被用于约束或“专门化”指针。SpecializePtr指令可被用于三种特殊实例。首先,SpecializePtr可被用于类型造型,以使用预期的源上下文信息(即,具有原始类型元数据)来解密采用EBA格式的以密码方式编码的指针,并使用目的地上下文信息(即,新类型元数据)来重新加密线性地址。第二,SpecializePtr可被用于边界收窄(例如,生成指向结构体中的字段的采用EBA格式的以密码方式编码的指针)。在这个示例中,SpecializePtr指令可被用于使用父存储器分配(例如,整个结构体)的上下文信息来解密采用EBA格式的以密码方式编码的指针,检查预期的缩窄边界完全处于父边界内,并且生成表示缩窄边界的新的采用EBA格式的以密码方式编码的指针。最后,SpecializePtr可被用来降低许可(例如,将非常量*转换为常量*)。在操作上,SpecializePtr指令使用包括原始许可元数据的原始上下文来解密采用EBA格式的以密码方式编码的指针,并生成具有新许可和/或新大小元数据的新的采用EBA格式的以密码方式编码的指针。SpecializePtr指令的一种示例格式如下：

SpecializePtr:rd、rs1、rs2、imm8

rd:包含原始的采用EBA格式的以密码方式编码的指针的寄存器操作数(例如64位)

rs1:包含原始的编码的上下文值的寄存器操作数(例如64位)输入

rs2:包含新的编码的上下文值的寄存器操作数(例如64位)输入

imm8:设置为指示是否要将包含上下文信息的表条目插入在新的以密码方式编码的指针的指针上下文表中的值的立即操作数(例如8位)。

[0117] 一般来说,SpecializePtr指令根据原始上下文信息解密原始的(或父)以密码方式编码的指针,检查新的上下文信息与原始上下文信息相比没有授予新的许可,并使用新的上下文信息对指针进行重新加密和编码,以获得新的采用EBA格式的以密码方式编码的指针。在至少一个实施例中,原始指针是在没有与其相关联类型的情况下返回的指针(例如,malloc指针)。原始上下文信息是曾用于生成原始指针的上下文信息。用于新指针的基地址是基地址加上从旧指针的偏移(新基地址=原始指针基地址+原始指针偏移)。在一些实施例中,可以以指定新的编码的上下文值并且动态加载旧的编码的上下文值的方式来调用SpecializePtr。

[0118] 如果原始指针具有其动态位设置,那么SpecializePtr指令可以用类似于Ld/StEP指令的方式处置指针。如果在具有其动态位设置的采用EBA格式的以密码方式编码的指针上使用了SpecializePtr指令,那么这就指示了可能的情景,在这种情景中,编译器仅推导出该指令所需的上下文信息中的一些。在这种情景中,指令逻辑检查静态嵌入的类型是否匹配动态获得的类型,以及静态嵌入的许可是否是动态获得的许可的子集。如果检查成功,则指令逻辑可以在指令操作中使用静态嵌入的许可和类型以及动态获得的大小和基(例如,较高地址位)。

[0119] 在一个示例中,SpecializePtr指令中的立即操作数imm8可以被设置为“1”(imm8==1),以指示具有与新指针相关联的新上下文信息的表条目要被插入到指针上下文表中。如果对象存储在分配到快速访问存储器区域的存储器中,并且编译器知道指针将仅在静态已知上下文的地方被使用,则立即操作数imm8可以被设置为“0”(imm8==0),以指示表条目不是必需的。

[0120] LdEP指令对采用EBA格式的以密码方式编码的指针进行解密和解码,以获得存储器位置的最终线性地址,并从该存储器位置来访问(例如,读)数据。LdEP指令的一种示例格式如下:

LdEP:rd、m1、rs2

rd:目的地操作数(例如64位)

m1:供应以密码方式编码的指针的存储器操作数(例如64位)

rs2:包含编码的上下文值的寄存器操作数(例如64位)输入。

[0121] LdEP指令使用来自rs2的上下文值,并执行访问控制检查,以确保尝试的访问完全处于分配的存储器的边界内,并确保由上下文值指示的许可允许在使用以密码方式编码的指针时执行读操作。所分配的存储器的边界由[基,基+大小)来定义,其被计算以确定由LdEP指令计算的线性地址是否处于这些边界内。如果访问检查没有失败,则最终线性地址的位(从以密码方式编码的指针中解码和解密)被加载到目的地操作数rd中。然后可以通过存储器接口(例如数据高速缓存单元360)发出存储器访问请求。如果存储器访问请求成功,则最终线性地址处的位(例如64位)被置于LdEP指令中指定的目的地操作数rd中。

[0122] StEP指令解密并解码采用EBA格式的以密码方式编码的指针,以获得存储器位置

的最终线性地址，并在该存储器位置存储(例如，写)数据。StEP指令的一种示例格式如下：

StEP:m1,rs1,rs2

m1:供应以密码方式编码的指针的存储器操作数(例如64位)

rs1:包含要存储的数据的寄存器操作数(例如64位)

rs2:包含编码的上下文值的寄存器操作数(例如64位)输入。

[0123] StEP指令使用来自rs2的上下文值并执行访问控制检查，以确保尝试的访问完全处于分配的存储器的边界内，并确保由上下文值指示的许可允许在使用以密码方式编码的指针时要执行写操作。分配的存储器的边界由[基，基+大小)来定义，其被计算以确定由StEP指令计算的线性地址是否处于这些边界内。如果访问检查没有失败，则最终线性地址的位(从以密码方式编码的指针解码和解密)可以用于通过存储器接口(例如，数据高速缓存单元360)发出存储器写请求。如果存储器写请求成功，则StEP指令中指定的寄存器操作数rs1中的位(例如64位)被存储在最终线性地址。

[0124] 如果LdEP指令或StEP指令被用在具有其动态位设置的采用EBA格式的以密码方式编码的指针上，那么可能的是上下文信息和较高地址位被存储在存储器中并且可以被动态地检索。动态位还可以指示另一种可能的情景，在其中编译器仅推导出指令所需的上下文信息中的一些，因此，一些上下文信息可以是静态可访问的(例如，存储在操作数和/或寄存器中)，而一些上下文信息可以是动态可访问的(例如，存储在存储器中)。在这种情景中，指令逻辑检查静态嵌入的类型是否匹配动态获得的类型，以及静态嵌入的许可是否不大于动态获得的许可。如果检查成功，则指令逻辑可以在指令操作中使用静态嵌入的许可和类型以及动态获得的大小和较高地址位。

[0125] 如果编译器在程序代码的编译过程中做出不正确的推断，则可能会发生一种可能的访问违例。例如，如果通过LdEP或StEP指令的存储器访问导致了完整性违例，这可能指示编译器不正确地推断出函数预期指向单个对象而不是数组的指针。在这种情景中，数组指针应具有存储的其上下文信息，并可用于动态查找，因此可以在执行动态上下文加载后重试加载或存储。

[0126] 对于新的LdEP和StEP指令以及传统指令，各种方法都是可能的。尽管传统(现有)指令包括存储器操作数，但是指令从存储器中的表中动态加载上下文信息，因为其可用存储器操作数被指派用于数据、指针、立即值等但不用于上下文信息。传统指令提供了许多功能，诸如加法、减法、乘法等。此外，传统指令可以包括在寄存器操作数与存储器操作数之间移动数据(或代码)。在一个或多个实施例中，可以用LdEP和StEP指令代替从存储器中读和写的传统操作，以允许使用静态嵌入的上下文信息。在这些实施例中，可以继续使用具有存储器操作数的其它传统指令(例如，加法、乘法、减法等)，并且可以经由指针上下文表动态地获得上下文信息。

[0127] 在另一实施例中，使用存储器操作数来执行操作(例如，加法、乘法、减法等)的一些或所有传统指令不被包括在由编译器生成的汇编语言输出代码中。相反，通过仅对寄存器数据进行操作而不访问存储器，LdEP和StEP指令与执行相同操作(例如，加法、乘法、减法等)的其它传统指令结合使用。例如，当软件程序代码中包括存储器中的两个值的加法操作时，编译器可以使汇编语言输出代码包括将第一值从存储器加载到第一寄存器中的第一LdEP指令，以及将第二值从存储器加载到第二寄存器中的第二LdEP指令。编译器可以使另

一条传统指令添加到汇编语言输出代码中，以对由LdEP指令加载的两个不同寄存器中的值执行加法操作。然后，可以使用StEP指令将结果值写回存储器。

[0128] 现在参考图11，示出了使用EncryptBaseAddr指令生成采用EBA格式的以密码方式编码的指针的示例过程1100。过程1100的部分可以由计算设备100的硬件、固件和/或软件来执行(例如，由执行地址编码逻辑152的处理器102来执行)。过程1100响应于(例如，通过存储器管理器模块的)存储器分配在1102开始。在1102，由堆分配器或栈分配器利用未处理指针(ptr)、上下文信息(包括存储器分配大小)和是否要将表条目插入到指针上下文表中的指示符来调用EncryptBaseAddr指令。在1104，生成采用EBA格式的以密码方式编码的指针。指针中的加密的基地址片被以密码方式绑定到上下文信息，诸如大小元数据和许可元数据。

[0129] 在1106，做出确定，所述确定关于由以密码方式编码的指针表示的对象是否完全处于使用静态上下文信息可寻址的存储器区域的边界内。静态可寻址区域可以由STATIC_REGION_BASE寄存器来界定，所述寄存器包含所分配的存储器区域的基地址和所分配的存储器区域的限制或结尾。所分配的存储器区域的限制或结尾可以被定义为STATIC_REGION_END(存储器区域的结尾)=STATIC_REGION_BASE(存储器区域的基地址)+STATIC_REGION_SIZE，其中STATIC_REGION_SIZE=2^(基对齐移位值+加密的基地址片的位宽)。在一个示例中，下面的确定可以指示由以密码方式编码的指针表示的对象是否完全处于静态可寻址存储器区域的边界内：

STATIC_REGION_BASE<=指针，并且指针+大小< STATIC_REGION_END?

在这个确定中，“指针(ptr)”是为存储器分配生成的未处理指针，并且“大小(size)”是存储器中分配的字节的数字。

[0130] 如果确定由以密码方式编码的指针表示的对象不完全处于静态可寻址存储器区域的边界内，则在1108，可以在以密码方式编码的指针中设置D位(例如601C)。D位指示例如要经由指针上下文表动态获得的上下文信息。

[0131] 一旦设置了D位，或者如果确定由以密码方式编码的指针表示的对象完全处于静态可寻址存储器区域的边界内，则在1110，做出确定，所述确定关于指令参数(例如，EncryptBaseAddr指令中的imm8)是否指定要创建上下文表条目。如果上下文信息不一定要被添加到指针上下文表，则在1112，在目的地操作数中返回以密码方式编码的指针(例如，EncryptBaseAddr指令中的rd)。

[0132] 如果确定必须要将上下文信息添加到指针上下文表中，则在1114，做出确定，所述确定关于在以密码方式编码的指针中由加密的基地址片指定的索引处的表条目是否已经被其它上下文信息所占用。如果其它上下文信息在由加密的基地址片指定的索引处占用指针上下文表，则在1116，生成指示表条目冲突的故障。本文中将参考图10进一步描述用于处置表条目冲突的特权软件流程。

[0133] 如果确定由以密码方式编码的指针中的加密的基地址片指定的索引处的指针上下文表中的位置没有被另一条目占用，则在1118，在该索引处插入新的上下文表条目。上下文表条目可以包括例如存储器分配大小元数据、类型元数据和许可元数据。在1120，在目的地操作数中返回以密码方式编码的指针(例如，EncryptBaseAddr指令中的rd)。

[0134] 图12是对采用EBA格式的以密码方式编码的指针执行指针专门化

(SpecializePtr)指令的示例过程1200。特别地,可以执行所述操作来改变与指针相关联的类型,缩窄与指针相关联的存储器区域的边界(例如,降低与指针相关联的大小,并且作为大小降低的一部分潜在地增加基地址),和/或降低由指针可以访问对象的许可。过程1200的部分可以由计算设备100的硬件、固件和/或软件来执行(例如,由处理器102的地址编码逻辑152来执行)。过程1200开始于1202,在那时使用操作数来调用专门化指针指令(SpecializePtr),所述操作数包括采用EBA格式的原始的(旧的)以密码方式编码的指针、与原始的以密码方式编码的指针相关联的原始的(旧的)上下文信息、与要生成的新的以密码方式编码的指针相关联的新的上下文信息、以及指示是否要为新指针创建上下文表条目的标志。

[0135] 在1204,做出确定,所述确定关于新的上下文信息是否将给新的指针对存储器中更宽边界的访问(与旧的指针能够访问的相比)。在至少一个实施例中,该确定可以通过计算新的基地址、新的限制(边界)、原始的限制,并比较新的限制和原始的限制来进行,其中o_ptr是原始的指针,而n_ptr将是新的指针:

new_base_address(新的基地址) = original_base_address [在o_ptr中] + current offset [在o_ptr中]

new_limit(新的限制) = new_base_address + new_size [在新的上下文信息中]

original_limit(原始的限制) = original_base_address [在o_ptr中] + original_size [在 o_ptr中]

如果new_limit ≥ original_limit,则新的上下文信息(即新大小元数据)将给新指针存储器中更宽边界的访问(与原始指针可以访问的相比)。因此,在1206生成故障。否则,如果new_limit < original_limit,则新的大小元数据将给新指针对原始指针的边界内的存储器的访问。因此,不会生成任何故障,并且可以执行另一检查。

[0136] 在1208,做出确定,所述确定关于新的上下文信息(即,许可元数据)是否授予未曾由原始上下文信息授予的任何许可。如果新上下文信息授予未曾由原始上下文信息授予的任何许可(即,新上下文信息扩展了许可),则在1206生成错误。如果新的上下文信息没有授予未曾由旧的上下文授予的任何许可(即,新的上下文信息降低或没有改变原始上下文信息中的许可),则可以基于新的上下文信息创建新的以密码方式编码的指针。

[0137] 在1210,可以计算由使用原始上下文信息的原始指针所表示的明文线性地址,好像正在尝试存储器访问一样。计算明文线性地址可以包括使用以密码方式的算法(例如,块密码)、秘密密钥和地址调整来解密原始指针中的基地址片。地址调整可以包括原始指针的标签/版本部分和原始上下文信息(例如,与原始指针相关联的大小元数据和许可元数据)。在至少一些实施例中,还可以在调整中使用其它上下文信息,包括密码术上下文ID和/或其它可变长度元数据。为了获得原始(解密的)基地址,一旦基地址片已经被解密,它可以与从存储器中的表条目或寄存器中提出的较高地址位串接,并且多个对齐位可以串接在解密的基地址片的末端。在一个示例中,如果地址是8字节对齐的,则使用三个对齐位。为了计算新的明文线性地址,可以将原始指针中的偏移添加到原始(解密的)基地址中。计算出的明文线性地址可以被用作用于新(专用)指针的新的基地址。

[0138] 在1212,可以使用所计算的新的明文基地址作为指针输入并转发新的上下文信息和指示是否需要创建上下文表条目的标志来执行EncryptBaseAddr指令的流程。一旦生成

了专用指针(即,采用EBA格式的新的以密码方式编码的指针),就可以将目的地寄存器设置为EncryptBaseAddr流程的结果(即,基于新的上下文信息的采用EBA格式的新的以密码方式编码的指针),如参考图11所示和所述的那样。

[0139] 在图13中,示出了用于解码和解密采用EBA格式的以密码方式编码的指针的示例过程1300。过程1300可被用于从加密的指针加载(LdEP)指令并用于存储到加密的指针(StEP)指令。过程1300的部分可以由计算设备100的硬件、固件和/或软件来执行(例如,由处理器102中的地址解码逻辑162来执行)。过程1300基于提示指针解码和解密的存储器访问请求开始于1302。例如,可以在1302启动从加密的指针加载(LdEP)指令或到存储到加密的指针(StEP)指令。

[0140] 在1304,做出确定,所述确定关于指针是否指示将静态地(从指令操作数)或动态地(从存储器)检索上下文信息。在一个示例中,该确定可以通过评估指针(例如,601C)的D位来做出。如果未设置D位,则在1305,从指令的操作数中静态检索上下文值。然而,如果设置了D位,则在1304,可以从存储器(例如,指针上下文表121)中动态检索上下文信息。更特别地,上下文值可以从指针上下文表中动态获得。在一些情景中,可以从指令的操作数中检索一些信息,并且可以从控制寄存器中检索一些信息(例如,较高地址位可以存储在控制寄存器中,并且大小、类型和许可元数据可以存储在操作数中)。生成了上下文值以表示与指针相关联的上下文信息(例如,大小元数据、类型元数据、许可元数据)。当指针被以密码方式编码时,上下文值可是使用EncodeCtx指令或现有的传统指令所生成的,并然后被存储在指针上下文表的表条目中。

[0141] 在1306,执行访问控制检查,以确定指针中编码的线性地址是否在所分配的存储器区域的基和边界内,以及与指针相关联的许可是否允许所请求的特定访问。在至少一个实施例中,可以通过确定针对每个指针值的所有偏移值是否都小于上下文信息中指定的大小元数据来执行基和边界检查,其中所述每个指针值涉及要在当前操作中访问的存储器的字节。如果偏移小于或等于上下文信息中指定的大小元数据,则基和边界检查成功。如果任何检查的偏移大于上下文信息中指定的大小元数据,则基和边界检查失败。在一些实施例中,检查少量偏移(例如,仅检查对于当前操作的最大偏移)是足够的。可以通过确定在上下文信息中指定的许可元数据是否允许所请求的特定类型的访问来执行许可检查。如果许可元数据允许所请求的访问的类型,则许可检查也成功。如果许可元数据不允许所请求的访问的类型(例如,写请求被请求并且许可元数据只允许读访问),则许可检查失败。在该实施例中,过程1300允许计算设备100在将指针转换成真实存储器地址之前验证以密码方式编码的指针并实施基和边界检查。

[0142] 如果访问控制检查中的任一个失败,则在1308生成故障。如果两个访问控制检查都成功,则在1310和1312解码以密码方式编码的指针。在1310,使用以密码方式的算法(例如,块密码)、秘密密钥和地址调整来解密指针中的基地址片。地址调整可以包括指针的标签部分和上下文信息(例如,与指针相关联的大小元数据、类型元数据和许可元数据)。在至少一些实施例中,还可以在调整中使用其它上下文信息,包括指针的较高地址位、密码术上下文ID和/或其它可变长度元数据。一旦加密的基地址片已经被解密,则在1312,使用较高地址位、解密的基地址片、对齐位和偏移来计算有效明文线性地址。解密的基地址片可以与从寄存器或指针上下文表中的表条目中提出的较高地址位串接。此外,可以在解密的基地

址片的末端串接多个对齐位，以产生解密的基地址。可以通过将原始指针中的偏移加到解密的基地址上来计算最终的明文线性地址。表示明文线性地址的计算的一个示例等式如下：

(({较高地址位，解密的基地址片})<<3)+偏移。

[0143] 在该示例计算中，三个对齐位被用于强制8字节对齐。然而，在其它示例中，可以使用不同数量的对齐位，或者可以消除对齐位。此外，可以使用任何形式的计算来适当地组合较高地址位、解密的基地址片和对齐位。例如，可以通过将串接的较高地址位和解密的基地址片向左移位三个位来添加对齐位。一旦计算出明文线性地址，就可以在1314执行存储器访问。

[0144] 图14是当试图将新的表条目插入到已经被条目占用的空间中的指针上下文表中时，操作系统捕捉冲突故障的示例过程1400。当与现有条目相关联的以密码方式编码的指针与相关联于新条目的以密码方式编码的指针具有相同的加密片并且该加密片被用于索引指针上下文表时，可能会发生这种情况。即使指针引用不同的对象并且具有不同的大小、类型和/或许可，这也是可能的。如果以密码方式编码的指针的另一部分被用于索引指针上下文表(例如，整个指针、加密的片加上标签部分等)，也会发生这种情况。过程1400的部分可以由计算设备100的硬件、固件和/或软件(例如，由特权系统组件142)来执行。过程1400响应于在执行EncryptBaseAddr指令或SpecializePtr指令期间由于表冲突而生成的故障，开始于1402。例如，在EncryptBaseAddr指令的流程期间，由于表条目冲突，在816处可能生成故障。

[0145] 在1404，表中冲突位置处的现有表条目被拷贝到包含针对该表位置的冲突条目的存储结构。在1406，对应于索引的上下文表中的位置(例如，当前以密码方式编码的指针的加密的基地址片)被标记以指示不存在表条目。在至少一个示例中，标记位置可以通过清除该位置处的表条目中的许可位(例如，许可字段708中的XWR位)来实现。

[0146] 如果故障是通过执行EncryptBaseAddr指令生成的(例如，在816)，则在1408，可以重试EncryptBaseAddr指令。如果故障是通过执行SpecializePtr指令生成的(例如，在1214)，则在1408，可以重试SpecializePtr指令。然而，可能的是如果故障是通过执行SpecializePtr指令生成的，则在1408，可仅执行EncryptBaseAddr指令的流程。在任一种情景中都不应该出现故障，因为表中的空间现在是可用的。在1410，程序执行可以继续。

[0147] 图15是操作系统捕获存储器访问违例并检查指针上下文表中的冲突表条目的示例过程1500。通常，如果先前(例如，在EncryptBaseAddr指令期间)发生指针上下文表中的冲突，需要逐出仍然有效的一个或多个先前条目，则操作系统可以尝试每个有效的、被逐出的条目，以确定是否有任何许可当前访问请求。如果是这样的话，则操作系统可以交换匹配的被逐出的条目和当前加载在指针上下文表中的条目，并恢复程序，然后所述程序将成功地继续运行。过程1500的部分可以由计算设备100的硬件、固件和/或软件(例如，由特权系统组件142)来执行。过程1500响应于由于访问违例而在尝试存储器访问期间生成的故障，开始于1502。存储器访问指令可以是例如LdEP指令或StEP指令。例如，在图13中所示的指针解码和解密流程期间，由于检测到存储器访问违例，在1308可能生成故障。

[0148] 在1504，做出确定，所述确定关于存储结构中的任何表条目(也称为“逐出的表”中的“逐出的条目”)是否包含被用于故障存储器访问的指针(例如，指针的加密的基地址片)

索引的表条目冲突信息。如果由用于故障访问的指针在逐出表中没有索引被逐出的条目，则在1506，故障被处置为正确且不可避免的。因此，可以基于特定的操作系统和实现做出适当的故障检测响应。

[0149] 如果存在于逐出表中的被逐出的条目被用于故障存储器访问的指针所索引，则在1508，做出确定，所述确定关于是否至少一个被逐出的条目将许可故障存储器访问(如果在由用于故障访问的指针索引的指针上下文表的表条目中安装有被逐出的条目的话)。如果由用于故障存储器访问的指针所索引的被逐出的条目都不会许可故障存储器访问，则在1506，故障被处置为正确且不可避免。

[0150] 如果由用于故障存储器访问的指针所索引的至少一个被逐出的条目将许可故障存储器访问(如果该被逐出的条目被安装在指针上下文表中的话)，那么在1510，指针上下文表中的活动表条目和将许可故障存储器访问的来自存储结构的标识的逐出的条目被交换。在1512，故障存储器访问被重试，并且应成功。在1514，程序执行可以继续。

[0151] 在一些实施例中，底层高速缓存访问可以与块大小对齐，使得完整性检查可以操作。指令可以在读整个块之后或在写整个块之前提取或修改块的一部分。逻辑边界检查防止对每个块的未授权部分的访问。在子块访问的示例场景中，考虑具有两个字段的结构:结构a {int x;int y;}。结构的一个实例被分配为与8字节块对齐。生成仅覆盖y整数的指针。针对指针的四字节LdEP加载整个结构，但仅返回从结构中提取的y整数的值。

[0152] 转到图16A-16B，图16A是使用采用EBA格式的以密码方式编码的指针和用于EBA指针的相关连指令的C++编程语言软件代码1600A的示例，并且图16B中的是其对应编译汇编语言输出1600B。在软件代码1600A中，结构定义1602之后是执行各种分配的主函数1604。例如，在1606生成指针，在1608生成指向数组的指针，并且在1610生成指向字段y的专用指针。为了在该示例中简化，假设具有指针参数的任何函数都采用加密指针。因此，如果调用函数，则生成指针。然而，在其它情景中，可以生成并使用指针，而无需将其传递给另一函数。

[0153] 图16B示出了从软件代码1600A中生成的汇编语言输出(或程序代码)1600B。程序代码1600B由两条EncryptBaseAddr指令1612和1614以及一条SpecializePtr指令1622所生成。EncryptBaseAddr指令1612不插入具有上下文信息的表条目，因此利用该指针的后续存储器访问可以静态地获得上下文信息。EncryptBaseAddr指令1614确实插入了具有上下文信息的表条目，因此随后利用该指针的存储器访问从存储器(例如，指针上下文表121)中动态地加载上下文信息。程序代码1600B还包括使用EBA指针从存储器加载数据的LdEP指令1618和1624，以及使用EBA指针将数据存储到存储器的StEP指令1620、1626和1628。

[0154] 也为EBA指针和指令的实现执行运行时软件的某个启用。首先，静态初始化的全局指针要在加载时间被加密。这要求可执行和可链接格式(ELF)元数据来指示这些指针的位置和它们指向的位置，以便加载器可以初始化它们。这可能类似于使用重定位数据。此外，动态链接器被配置为在填充全局偏移表(GOT)和过程链接表(PLT)时使用以密码方式编码的指针格式。

[0155] 图17是根据至少一个实施例的尝试使用以密码方式编码的指针1710来访问存储器的示例对手1720(例如，恶意软件、恶意用户、机器人、有缺陷的软件等)的图1700。如参考EBA指针610所述，以密码方式编码的指针1710可以采用EBA格式来配置。图17中示出的示例示出了使用上下文信息1703(例如，诸如对象的大小、对象的类型、对象的位置、所有权信

息、访问控制、许可等上下文信息)和一些地址位1702(例如,较高地址位)作为地址调整1704来加密存储器地址片以获得指针1710中的加密的基地址片的以密码方式的算法。也可以由以密码方式的算法使用秘密地址密钥1706。然后,具有地址的加密片(例如,密文)的以密码方式编码的指针1710是软件可访问的,并且如果被正确操纵(例如,仅操纵偏移并且偏移保持在所分配的存储器的基和边界内)并且当指针被返回到CPU时提供相同的隐式信息,则正确的原始地址将被解码。如果对手没有正确的隐式信息(诸如访问函数位于错误的存储器位置),或者指针的标签或加密部分被改变,那么处理器将会不正确地解码地址,从而导致坏/故障的存储器访问。例如,在所示示例中,元数据调整(例如,地址位、对象(大小、类型和/或位置)、所有权、访问控制、许可等)和秘密地址密钥被用于加密虚拟地址。如果对手修改了解密过程的输入(例如,改变了编码的指针值或提供了错误的大小信息),如由修改的输入1709所指示的那样,则指针可以被解密到随机(或坏的)地址1722,这可能导致如所示的故障1724。另一方面,如果使用正确的地址调整1704和秘密地址密钥1706,如该图的顶部分支中所示,则可以从解密过程中获得正确的线性地址1712,如果偏移已经改变,则所述线性地址可以不同于线性基地址。

[0156] 图18是根据至少一个实施例的尝试使用以密码方式编码的指针1810来访问存储器的另一示例对手1820(例如,恶意软件、恶意用户、机器人、有缺陷的软件等)的图1800。特别地,在所示的示例中,基于秘密地址密钥1806和包括上下文信息中的三项的地址调整来加密指针1810。在该示例中,上下文信息包括存储器分配大小元数据1803A、类型元数据1803B和许可元数据1803C。然而,如本文中先前所述,上下文信息的任何合适项、存储器地址的部分或其合适组合可用作调整,以对嵌入在指针中的基地址片(例如,EBA部分604)进行加密。可以利用所包括的上下文元数据的多级将嵌入在指针中的加密的基地址片加密多次。在其它实施例中,可以用包含上下文信息的一个或多个项、存储器地址的一部分(例如,较高地址位)或其任何合适组合的地址调整来对嵌入在指针中的基地址片加密一次。

[0157] 数据加密则可能依赖于指针编码。例如,当加密数据在存储器上存储(写)或解密数据在存储器上读(加载)操作时,编码在指针1810中的解码的线性地址(例如,明文基地址加上偏移)可以被生成并用作数据调整以加密/解密由指针1810所引用的数据。来自指针1810的元数据(例如,标签/版本部分602)也可以与解码的线性地址一起使用,以形成数据调整。此外,任何合适的上下文信息可以与解码的线性地址一起用作用于数据的加密/解密的调整。在另一实施例中,当加密(或解密)数据时,指针1810中的编码的地址的加密的基地址部分(例如,604)可以被用作数据调整,以加密/解密由编码指针1810所引用的数据。在此,处理器可以使用可调整的块密码和秘密数据加密密钥来使用加密的基地址部分作为数据调整的一部分以加密数据。取决于密码的调整大小,指针的明文部分(例如,较高地址位)也可以被用作数据调整的部分。例如,64位寄存器的存储可以使用具有64位块大小的PRINCE密码,并且加密的基地址部分(604)和到该64位块的偏移可以用作数据调整,或者从指针加上偏移生成的解码的线性地址(例如,明文基地址加上偏移)可以被用作数据调整。类似地,当以32位模式运行时,可以使用32位块大小的密码(例如SPECK、SIMON、可调整的K密码),或者使用加密的基地址部分(604)和到该32位存储器块的偏移作为数据调整来加密和存储32位大小的寄存器,或者从指针生成的线性基地址的某个部分加上偏移可以用作数据调整。类似地,使用加密的基地址部分(604)和到128位存储器块的偏移作为数据调整,或

者可以使用从指针生成的线性基地址加上偏移作为数据调整，可以使用较大大小的块密码(诸如AES)来将128位AVX寄存器存储到存储器。此外，当调整输入具有可用位时，一些上下文信息也可以被包括在调整中。备选地，较小块大小的密码(例如64位)可被用于以部分方式加密和存储较大的寄存器128位，利用用于上半部分的数据调整加密上半部分，并利用不同的数据调整分开地加密下半部分。应当注意，在至少一些实现中，取决于块密码的块大小和指针大小，整个指针或其任何选择的内容可以用作针对块密码的数据调整或数据调整的一部分。例如，图6的指针610、或从指针610生成的线性基地址或其某部分可以用作数据调整，以用于利用具有128位块大小的AES计数器模式密码来加密数据。

[0158] 如图17中所示，在图18的图中，用无效的编码地址(例如，修改的加密的基地址部分1809)来对加密的数据进行解密1830将会由于以下原因而导致故障：对坏地址1822的指针值解码，或者，即使解密的地址是有效的(无故障)，在该位置的数据也可以用不同的调整来加密(例如，导致无效的密钥流)。因此，即使对手试图进行未授权的访问并猜测使得解密和解码的指针指向目标存储器的地址调整，该存储器也会以绑定到授权指针的方式进行加密。因此，如果完整性被强制执行，则对手的访问很可能仅返回被歪曲的数据或导致可检测到的完整性违例。

[0159] 以这种方式，数据密码操作可能取决于以密码方式的地址编码。存储器管理器可以利用这一点来从堆中分配不同大小的对象，以确保指针中的密文对于每个malloc/新指令始终是不同的。例如，当释放对8个字节的分配时，存储器中的相同位置的下一次重新分配可以被给到4个字节的分配，以便为新分配的缓冲器产生不同的密文(例如，604)。如果释放的8字节分配指针后来被不正确地重新用于访问存储器位置，则当前4字节分配的内容将使用不同的密钥流/调整进行加密，使得释放的8字节分配指针不会适当地解密新数据。

[0160] 在一些实施例中，数据完整性可以由处理器基于熵或模型测试来隐式地验证，因为利用错误的调整/密钥流解密的数据可能相对于将呈现模型的适当解码的数据而言显得随机。隐式完整性是优化，其中处理器每次都查找存储在与(加密的)数据相对应的存储器中的完整性值(例如，消息认证码/MAC)。当处理器从存储器加载寄存器时，它可以验证用于加载寄存器/端口的存储器的对应部分的隐式完整性，或者，如果熵是不确定的，则处理器可以查找用于存储器的该部分的完整性值(例如，基于开始加载寄存器处的存储器地址)以验证存储器的内容属于该密钥流/调整。如果存储器中的完整性值不匹配于被加载到给定当前编码的地址(调整、密钥流)的寄存器中的存储器(数据)内容，则可能会由处理器生成完整性违例(异常/故障)。

[0161] 针对所有权的写操作可被用于将存储器的内容从使用一个密钥流改变为使用另一密钥流，并且这可能是特权操作(例如，由存储器管理器执行，或由利用EncryptBaseAddr的许可确定)，以分配存储器而不会导致完整性违例。针对所有权的写可以改变数据和对应的完整性值(例如，ICV/MAC)，以使所写的数据内容与新的密钥流/调整相匹配。同时，正常的写将首先使用编码的地址从存储器中读旧数据/完整性值，所述编码的地址被用于将寄存器内容存储(写)到存储器中，以首先验证是否使用了正确的密钥流/调整(这是针对所有权检查的读)。通过这种方式，覆写属于其它部分的存储器(不同的密钥/调整)的尝试被检测到并被阻止。

[0162] 图19和20是示出根据本文中描述的实施例的用于解密数据的示例块密码的框图。

图19和20中所示的块密码并不意在是限制性的，因为根据本文中公开的实施例的特定需要和实现，采用各种大小的许多不同的可调整块密码可被用于加密和解密数据(和存储器地址片)。

[0163] 图19示出了具有密文窃取的基于高级加密标准(AES)XEX的调整码本模式(XTS-AES)的解密流程1900，其用于解密由诸如EBA指针610的以密码方式编码的指针所引用的加密数据。AES是对数据的块(例如，固定长度位群)进行操作的块密码。XTS-AES算法是AES算法的操作模式。XTS-AES是可调整的块密码，其使用相同的秘密数据密钥和相同的数据调整来执行XTS-AES加密过程和XTS-AES解密过程。例如，Prince-XTS块密码1906可以基于数据调整1904和秘密数据密钥1905来执行数据加密，并且还可以使用相同的数据调整1904和相同的数据密钥1905来执行对加密数据的相对解密。

[0164] 参考解密流程1900，在数据被Prince-XTS块密码1906加密之后，当加密的数据随后被软件(例如，加载或存储指令等)访问时，加密的数据在1902被加载到缓冲器中，并被馈送到Prince-XTS块密码1906中。数据调整1904和数据密钥1905也被输入到块密码1906。在一个实施例中，数据调整1904可以包括加密的基地址部分(例如，604)和从引用加密数据的存储器位置的指针的偏移。在另一实施例中，数据调整1904可以包括从EBA指针(例如，610)生成的线性基地址加上偏移的至少一部分。块密码1906使用数据调整1904和数据密钥1905对加密的数据进行解密。在1910，解密的数据可以被加载到端口中，以供软件使用。

[0165] 图20示出了高级加密标准(AES)计数器模式(AES-CTR)解密流程2000，其用于解密由以密码方式编码的指针(诸如指针610)所引用的加密数据。AES-CTR算法是AES算法的操作模式。AES-CTR是可以在使用了相同的秘密数据密钥和相同的数据调整的AES-CTR模式加密过程和AES-CTR模式解密过程中使用的块密码。例如，在AES-CTR模式加密过程中，AES-CTR块密码基于秘密数据密钥对数据调整进行加密，以创建密钥流，所述密钥流然后使用异或(XOR)操作对数据的块进行加密。在相对的AES-CTR解密过程中，AES块密码基于相同的秘密数据密钥对相同的数据调整进行加密，以创建密钥流，所述密钥流然后使用异或操作对相应的加密的数据块进行解密。

[0166] 在一个示例中，在调用解密流程2000之前，数据在AES-CTR模式加密过程中被加密。在AES-CTR模式加密过程中，AES-CTR模式块密码2006基于秘密数据密钥2005对数据调整/初始化向量(IV)2004进行加密，以创建密钥流2007。密钥流2007在异或操作2008中被用于加密数据块。这可以用AES-CTR块密码2006对要加密的数据的每个块加密新的数据调整来对多个数据块执行。可以通过递增数据调整中的计数器来计算新的数据调整。计数器(或调整/IV)2004可以包括加密的基地址部分(例如604)和来自引用加密数据的存储器位置的指针的偏移。在至少一个实施例中，数据调整2004可以包括从EBA指针(例如，610)生成的线性地址。可以通过对EBA指针进行解码以生成解密的基地址并将EBA指针中的偏移添加到解密的基地址来计算线性地址。在AES-CTR模式加密过程完成并生成加密的数据之后，当加密的数据被软件(例如，加载或存储指令等)访问时，可以调用解密流程2000。在解密流程2000中，在2002，由AES-CTR模式加密过程生成的加密的数据被加载到缓冲器中。AES-CTR模式块密码2006基于相同的秘密数据密钥2005来加密相同的数据调整2004，以创建密钥流2007。密钥流2007在异或操作2008中被用于对加密的数据的对应块进行解密，所述加密的数据先前是由AES-CTR模式加密过程生成的。这可以使用对每个块的(例如，通过在数据调整中递

增计数器来计算的)新数据调整来对加密的数据的每个块执行。在2010,解密的数据可以被加载到端口中,以供软件使用。

[0167] 还应注意,在一些实施例中,在2003指示的操作可以与为获得要被解密的加密数据的操作并行执行。用于获得加密数据的操作包括解码以密码方式编码的指针以形成线性基地址,并使用线性基地址来定位加密数据。在2002,加密的数据然后可以存储在缓冲器中。此外,应当进一步注意,参考图19-20示出和描述的块密码还可以适用于对线性基地址(例如,线性基地址的片)上执行以密码方式的操作,所述线性基地址被形成为采用EBA格式的以密码方式编码的指针。

[0168] 通常,当加密/解密数据时,有利的是避免对给定分配内的数据的每个块使用相同的调整/IV。因此,调整/IV中的值取决于正在处理的特定块的位置而变化。在某些块密码(诸如AES-CTR模式(例如,2006))中,初始化向量(IV)实施了此概念,因为当生成密钥流的后续块时,所述初始化向量被转换(例如,递增)。然而,本文中的实施例允许软件从任何存储器访问指令中访问分配的任何块。因此,访问的相对偏移必须被并入在调整/IV中。在使用偏移(例如,606)的各种实施例中并入在指针中编码的线性地址的偏移是为实现这一点的一种可能方式,这如本文中所述。

[0169] 图21A-21B示出了获得由采用EBA格式的以密码方式编码的指针2110的实施例所引用的数据的示例过程2100A-2100B的详细流程图,其中数据的加密被绑定到指针的内容。过程2100A-2100B的至少一些部分可以由计算设备100的硬件、固件和/或软件来执行。在所示的示例中,EBA指针2110被实施为64位编码的线性地址,其包括保留部分,所述保留部分包括监管者位(S位)2101A、编码指示符位(E位)2101B和动态上下文查找位(D位)2101C。指针2110还包括标签/版本部分2102、加密的基地址部分2104和偏移部分2106。通常,指针2110具有与指针610类似的配置。

[0170] 过程2100A-2100B的操作被标识为三个阶段:地址解密(阶段I)、地址形成(阶段II)和数据解密(阶段III)。在阶段I中,嵌入指针2110中的线性基地址被解密。特别地,由解密算法解密指针的EBA部分2104,所述算法诸如使用地址密钥2118和地址调整2116的可调整块密码2120。除了外部上下文信息2109的一项或多项之外,地址调整2116可以包括标签/版本部分2102的位。例如,外部上下文信息2109的存储器分配大小元数据2112、类型元数据2113和许可元数据2114的位可以被包括在地址调整2116中。在进一步的实施例中,明文较高地址位2111也可以用作地址调整2116的一部分。在仍有另一实施例中,密码术上下文标识符寄存器2115可以用作地址调整2116的一部分。密码术上下文标识符寄存器2115可以包含与特定功能群(例如,过程、过程的子集、虚拟机(VM)、VM的子集等)相关联的(例如,随机或确定性生成的)唯一值。在至少一些实施例中,可变长度元数据2117的附加一项或多项也可以用作用于可调整块密码2120的地址调整2116的一部分。例如,可变长度元数据可以包括其它上下文信息或元数据(例如,特权级别、位置、所有权等),如本文先前所述。块密码2120可以是任何合适的解密算法(例如,XTS-AES块密码、LRW、AES-CTR模式等),如本文所述。

[0171] 当已经通过可调整的块密码2120将编码的线性地址的EBA部分2104解密成解密的基地址片2124时,则在阶段II中可以形成解码的线性基地址2126。首先,可以从寄存器中的上下文信息中获得(或者从如先前所述的其它存储设备中获得)较高地址位2111,并且将较

高地址位2111串接到解密的基地址片2124。在至少一些实施例中，基对齐位2125与解密的基地址片2124串接，以形成最低有效位。这也可以通过执行适当数量的位的移位的左移位操作来实现。在这个示例中，基对齐位2125包括三个位。基对齐位2125被设置为零，以便将地址对齐到8字节边界。通过将指针2110的偏移部分2106中的偏移添加到解密的基地址2126来形成解码的线性地址2130，以获得线性地址位2134。此外，监管者位2101A被复制到未使用/非规范位(例如，相邻的六个位2132)中，并且所有未使用/非规范位与线性地址位2134串接。在一些实施例中，线性地址位2134的最高有效位被复制以填充解码的线性地址2130中的其它未使用/非规范位。

[0172] 在使用解码的线性地址来执行存储器访问之前，检查由解密的基地址2126和存储器分配大小2112的组合指定的边界。解密的基地址2126指定了下边界，而通过将存储器分配大小2112添加到解密的基地址2126来计算上边界。较窄的边界可以用在引用整体分配的一部分的指针中。如果检查通过，那么存储器访问可以继续。否则，可能会引发故障。

[0173] 解码的线性地址2130被用来找到在阶段III中要进行解密的加密数据的存储器位置。此外，解码的线性地址2130和编码的指针2110可被用于创建用于对加密的数据进行解密的数据调整2144。如图21A的阶段II中所示，数据调整2144可以包括编码指针2110的标签/版本部分2102和解码的线性地址2130的线性地址位2134。此外，可以根据需要包括和复制S位2101A，例如，以填充任何剩余的未使用/非规范位2133。此外，在数据调整2144中可以包括其它上下文信息。例如，密码术上下文ID寄存器2115可以被包括在调整中，和/或任何其它可变长度元数据2136也可以被包括在调整中。可变长度元数据2136可以包括其它上下文信息或元数据(例如，特权级别、位置、所有权、类型、许可、存储器分配大小等)，如本文先前所述。对于数据调整2144来说，可能是特别有利的是包括与以下一项或多项相组合的存储器分配大小和线性地址位2134：(1)类型；(2)标签/版本；(3)类型和标签/版本；(4)密码术上下文ID；或者(5)密码术上下文ID和许可级别。使用IV/数据调整2144中的明文线性地址位2134而不是加密指针位(例如EBA 2104)使得具有不同边界、许可或其它元数据的多个指针能够同时引用相同的分配，并且均能够正确地加密和解密数据，使得可以使用该群中的任何指针来访问所述数据。在IV/数据调整2144中可包括在同时用来访问相同存储器的指针的群内不允许变化的元数据，因为包括允许在该组内变化的元数据可能导致当使用该组中的某个指针访问数据时，如果该数据先前是使用该组中具有包括在IV/数据调整2144中的不同元数据的另一指针加密的，则该数据将被错误地解密。

[0174] 在一些实施例中，可能有利的是添加解密的基地址2126(或解密的基地址片2124)的拷贝作为数据调整2144的一部分。为了说明这种方法的优点，考虑顺序重叠存储器分配，在其中分配和释放对象A，然后在重叠位置分配对象B。即使两个分配在分开的时间重叠，但是不同的基地址具有相同的标签/版本，从而使用指向重叠存储器位置的悬挂指针将不能正确解密甚至对象B的重叠部分，因为它们具有不同的基地址。

[0175] 由解码的线性地址2130获得的加密数据在图21B中所示的阶段III被解密。一旦数据密钥2142变得可用并且已经计算出IV/数据调整2144，就可以着手数据解密，可能要求所述数据解密等待EBA 2104被解密。由诸如密钥流生成器2150的解密算法来对加密的数据进行解密。在至少一个实施例中，密钥流生成器2150可以以特定的大小粒度(任何合适的大小)实现为AES-CTR模式块密码。在该实施例中，从解码的线性地址2130(例如，线性地址位

2134、监管者位2101A、剩余的非规范位2133)和从编码的指针(例如,标签/版本部分2102的标签位)中导出用于密钥流生成器2150的数据调整2144(或初始化向量)。数据调整2144还可以包括外部上下文信息(例如,密码术上下文ID寄存器2115和/或其它可变长度元数据2136)。密钥流生成器2150基于数据密钥2142来加密数据调整2144,以生成密钥流2151。在被用作密钥流生成器的输入之前,数据调整2144的值可以被调节为符合0(以密钥流生成器2150的块大小取模)。数据调整2144的值可以具有设置为0的一些合适数量的最低有效位,以满足该要求,并且可以丢弃密钥流2151的前缀,以解决该调节。可以通过从数据调整2144的未调节的值中减去数据调整2144的调节的值来计算要丢弃的密钥流2151的字节数量。如果要加密的存储器跨一个或多个块对齐的边界,则密钥流生成器2150可以针对随后的块被重新调用,其中每次所述密钥流生成器2150被重新调用时,数据调整2144增加等于块大小的量。生成的密钥流2151的后缀可能是不需要的,并因此被丢弃。然后对密钥流2151和加密的输入数据块(或高速缓存行)2146执行异或操作2152,所述加密的输入数据块(或高速缓存行)2146是从由解码的线性地址所引用的存储器位置中选择的。加密的输入数据块2146的粒度与从密钥流生成器2150输出的密钥流2151相匹配,并且异或操作2152产生解密的输出数据块2154。

[0176] 图22是用于创建和使用具有加密基的以密码方式编码的指针的示例高级过程2200的流程图。过程2200的至少一些部分可以由计算设备100的硬件、固件和/或软件来执行。在图22中所示的示例中,在2202,处理器执行由软件程序的函数所请求的存储器分配。例如,可以在C编程语言软件中使用malloc(存储器分配)指令来为数据(例如,变量、缓冲器、数组等)动态地请求存储器。malloc指令在堆上分配存储器,也在栈上分配指针。在一个或多个实施例中,存储器分配指令(诸如malloc)配置成生成指向所分配的存储器的以密码方式编码的指针。以密码方式编码的指针可以具有任何合适的配置,其包括加密的基地址(或其一部分),诸如指针610和2110或其任何其它合适变体。特别地,以密码方式编码的指针包括指向分配的存储器的线性基地址的加密片。在一种可能的变化中,用于指针的较大的寄存器可以存储整个加密的基地址。以密码方式编码的指针还包括附加的元数据(例如,标签、版本等),以及潜在的保留位。在至少一些实施例中,指针包括针对线性基地址的偏移或可变明文部分。

[0177] 在2204,以密码方式编码的指针被返回给函数。在2206,以密码方式编码的指针可以被加载到通用寄存器(GPR)中。软件程序可以在为编码的指针定义的数据结构的边界内操纵指针。例如,如果指针以与指针610或2110相同或类似的配置被编码,则偏移部分(例如,606、2106)可以在由存储器分配大小元数据(例如,704)定义的边界内被操纵,所述存储器分配大小元数据可以被存储在诸如上下文信息寄存器的存储器操作数中(例如,作为上下文信息700的一部分),或者存储在诸如指针上下文表121的存储器中的表中。

[0178] 在2210,诸如加载指令的数据访问指令由软件程序执行,并且包括用于指针的存储器操作数。存储器操作数可以是包含指针的寄存器。指令还可以包括采用另一寄存器的形式的存储器操作数或用于上下文信息的立即操作数,以及采用寄存器的形式的存储器操作数,以包含用于存储器访问的目的地或解码的线性地址。

[0179] 在2212,指针被解码以获得用于访问加密数据的线性地址。对指针进行解码可以通过执行解密算法(例如,诸如XTS-AES、LRW、AES-CTR模式等的块密码)来实现,以基于秘密

地址密钥和地址调整来对加密的地址片(例如,EBA部分604、2104)进行解密。地址调整包括与指针相关联的至少一个元数据值。例如,元数据值可以包括嵌入在指针中的一个或多个元数据项(例如,标签/版本部分602、2102、其它上下文信息等)、一个或多个外部上下文信息项(例如,存储在上下文信息寄存器中、存储在要动态访问的表中或作为指令中的立即操作数传递的元数据)、和/或密码术上下文标识符寄存器。为形成解码的线性地址,首先,通过将经解密的地址片与线性基地址的另一片(它们是较高地址位(例如,702、2111))串接来生成解密的基地址。指针中的偏移被添加到解密的基地址以计算实际的线性地址,并且附加的高位可以被填充或设置为特定值,这取决于指针编码和地址寄存器大小(例如,最高有效位2101A的复制等)。

[0180] 将随机位的指针的标签部分并入到数据加密/解密中是版本化存储器的方法,并且可以防止某些恶意软件攻击。例如,由于标签位被并入到数据加密中,可以减轻释放后使用攻击(即,在底层对象已经被释放并且甚至潜在地被重新分配给不同的对象之后访问悬摆指针)和越界访问。如果启用了完整性检查,这两种情景都很有可能性导致完整性检查违例。如果未启用完整性检查,释放后使用攻击或越界访问将导致数据损坏,这可能仍足以保护数据机密性。

[0181] 具有用于数据加密/解密的唯一值的标签/版本部分也可以减轻未初始化的使用弱点。当从已分配但未填充有初始值的缓冲器中读数据时,会发生未初始化的使用。通过延迟对象的对所有权的写(write-for-ownership)直到对象被初始化,可以检测到未初始化的使用。对所有权的写指的是在没有从存储器的区域进行首次读的情况下向存储器发出写,其作为对存储器写的惯例。当初始化对象时,软件可以适用于使用对所有权的写,以避免完整性检查违例,否则,如果使用嵌入在指针中的标签/版本值读存储器的区域,而所述标签/版本值不同于先前用于在存储器中写当前值的任何标签/版本值,所述完整性检查违例就会发生。

[0182] 软件还可以适用于检测悬摆指针(例如,释放后使用)或未初始化对象的尝试访问。如果完整性检查被启用,则存储器分配器(例如,malloc指令、释放指令)可以适用于将零密文注入到新分配的(例如,malloc指令)存储器区域或释放的(例如,释放指令)存储器区域中。这可能会导致在访问悬摆指针或未初始化的对象时生成完整性故障。

[0183] 在2214,在通过使用解码的线性地址访问加密的数据之后,可以通过执行解密算法(例如,诸如XTS-AES、LRW、AES-CTR模式等块密码)来获得解密的数据,以基于秘密数据密钥和数据调整来解密数据。数据调整值至少部分地从以密码方式编码的指针中导出。例如,实际的线性地址(其包括指针偏移)可以用于数据调整。数据调整还可以包括以密码方式编码的指针的其它部分,诸如标签/版本部分612或2112,或者甚至以密码方式编码的指针的全部内容。在一些情况下,可能可期望的是添加EBA部分604或2104的加密片。在进一步的实施例中,数据调整可以包括上下文信息的一个或多个项(例如,存储在上下文信息寄存器中、存储在要动态访问的表中、或作为指令中的立即操作数传递的元数据)、和/或密码术上下文标识符寄存器。同样在2214,解密的数据可以被加载到通用寄存器(GPR)中。

[0184] 如果程序对指针正确地解码,那么数据就可以被正确地解密。但是,如果指针被不正确地解码为不正确的线性地址,则可能发生页故障。即使没有发生页故障,不正确的线性地址也可能访问随机存储器,并会试图使用无效的数据调整来解密来自随机存储器中的数

据。因此，在一个或多个实施例中提供了两级的保护。

[0185] 在2216，可以对GPR中的数据执行操作。通常，可以根据软件程序来操纵解密的数据。在2218，确定了存储指令的一个或多个存储器操作数。例如，编码的指针可以作为存储指令的操作数来提供。

[0186] 在2220，以密码方式编码的指针被解码，以形成如本文中先前所述的解码的线性地址。通过执行加密算法(例如，诸如XTS-AES、LRW、AES-CTR模式等块密码)来加密要存储的数据，以基于秘密数据密钥和数据调整来加密GPR中的数据，如先前所述。至少部分地从以密码方式编码的指针(例如，标签/版本部分602或2102)中导出数据调整。对GPR中的当前数据执行数据加密，以逆转先前的解密过程。如果GPR中的当前数据已经被修改，则新加密的数据将不同于先前在2212从存储器中检索的加密数据。一旦数据被加密，它可以基于解码的线性地址存储在存储器中。

[0187] 在2222，存储器可以被解除分配，并因此，以密码方式编码的指针被释放。

[0188] 其它编码的指针格式

相似于采用EBA格式的以密码方式编码的指针，编码指针(以密码方式编码和未以密码方式编码两者)的其它实施例可以用于引用数据和/或代码，并将引用的数据和/或代码绑定到指针。现在将描述编码的指针的两个可能的备选实施例。

[0189] 图23是示出根据至少一个实施例的获得由指针2310的另一实施例所引用的数据的示例过程2300的详细流程图，所述指针2310使用用于数据的线性地址来以密码方式编码，其中数据的加密绑定到指针的内容。过程2300的至少一些部分可以由计算设备100的硬件、固件和/或软件来执行。在所示的示例中，指针2310被实施为使用指数(幂)元数据而不是偏移的采用其基格式的以密码方式编码的64位指针。在所示示例中，编码指针包括上下文信息，诸如标签/版本部分(例如，在所示示例中为2个位)和大小(幂)元数据部分2302(例如，在所示示例中为5位)。编码指针还包括可变位不可变明文部分2306和可变位可变明文部分2308。在所示的示例中，不可变明文部分2306和可变明文部分2308中的每个可以包含0-25位，其中包含在组合部分2306和2308中的位的总数等于25位。

[0190] 在至少一个实施例中，大小(幂)部分2302指示可变明文部分2308的大小，所述大小是可以由用于指针算术的软件自由操纵的低阶地址位的数量。在一些实施例中，大小/幂部分2302可以包括指示基于2的幂的大小的幂(指数)元数据位。大小/幂元数据部分2302也可以用于确定组成不可变明文部分2306的位的数量。在某些实施例中，组成不可变明文部分2306和可变明文部分2308的位的总数可以是恒定的，其中相应部分的大小由大小/幂元数据部分2302来规定。例如，如果大小/幂元数据值为0，则不存在可变明文位，并且所有25个剩余地址位构成不可变明文部分2306。为了进一步说明所示示例中的大小/幂元数据的编码，如果大小/幂元数据值为1，则存在可变明文的一个位和不可变明文的2个位，如果幂元数据值为2，则存在可变明文的两个位和不可变明文的23个位。可以由大小/幂元数据2302指示多达可变明文的25个位，这导致没有不可变明文位(2306)。可变明文部分2308可以由软件操纵，例如用于指针算术或其它操作。

[0191] 不可变明文部分2306可以用作从地址片(即，线性地址的线性地址位的子集)中生成密文部分2304的调整，其中密文部分2304与不可变明文部分2306相邻并且比不可变明文部分2306更重要。指针2310的密文部分2304(例如，在所示示例中的32个位)可以用小的可

调整块密码(例如,以32位块大小的SIMON、SPECK或可调整K密码,或其它可变位大小的可调整块密码)加密。剩余的地址位构成不可变明文部分2306,并被用作针对用于加密密文部分2304的可调整块密码的调整的一部分。虽然这些位也是地址的明文(非加密)部分,但是它们不能像可变明文部分2308的位那样被软件(例如指针算法)修改,而不会导致密文部分2304不正确地解密。指针2310的基指针格式允许以密码方式描述对象大小及其在存储器中的位置。在一些情况下,除了指针之外,指数/幂/大小元数据部分2302可以作为单独的参数来提供;然而,在一些情况下(例如,如所示的),大小/幂元数据部分2302的位可以与指针2310集成,以在某些情况下提供传统兼容性。

[0192] 指针2310的标签/版本部分2301可以是随机的或确定性不同的值。在其它实施例中,标签/版本部分2301可以被省略,并且附加位可以被添加到明文部分2306和2308(例如,总共27个位)或者指针2310的任何其它部分。在仍有其它实施例中,两个较高位可以是保留的位,其允许以密码方式编码的指针与传统指针同时使用。例如,最高有效位可被用于指示地址是位于监督者地址空间(例如“1”)内还是位于用户模式地址空间(例如“0”)内。下一个最高有效位可以被设置为监督者位的相反值,以指示指针被以密码方式编码,或者可以被设置为与监督者位相同的值,以指示指针未被以密码方式编码。在其它实施例中,可以在没有专用保留位的情况下实现传统编码。相反,传统编码可以通过在大小/幂元数据部分2302中编码特定值(例如,全为1、全为0)来实现。如果指针2310包括标签/版本部分2301,则这些位也可以用特定值(例如,全为1、全为0)编码,以允许同时使用传统和常规编码。在仍有其它实施例中,如果例如概念没有被实现为与传统程序兼容,则传统编码可以被完全消除。

[0193] 应当注意,大小/幂元数据部分2302可以不被加密,因为它被用于确定指针的可变和不可变明文部分中的位的数量,并因此确定地址调整(例如,不可变明文部分2306)中使用的位的数量。然而,标签/版本部分2301不被用于确定地址调整的大小。因此,标签/版本部分2301可备选地被包括作为地址的加密部分(即,密文2304)的一部分,只要标签/版本部分2301没有在地址调整中被使用。在该备选实施例中,块密码将具有对应的更大的块大小以适合标签/版本部分,或者被包括在密文中的地址位将被降低,并且地址位的对应数量将被包括在明文部分(即,2306和2308)中。此外,应注意,尽管过程2300利用指针2310(其包括标签/版本(或保留的位)部分2301)中示出的编码来示出,但是过程2300可以利用具有不包括标签/版本(或保留的位)部分的大小/幂元数据部分的其它指针编码来执行。在这种情景中,标签/版本(或保留的位)部分可以简单地从地址调整中消除。

[0194] 过程2300的操作被标识为三个阶段:地址解密(阶段I)、地址形成(阶段II)和数据解密(阶段III)。在阶段I中,嵌入指针2310中的线性地址被解密。特别地,编码线性地址的密文部分2304由解密算法来解密,所述解密算法诸如是使用地址密钥2318和地址调整2316的可调整块密码2320。地址调整2316可以包括在指针中编码的上下文信息,诸如标签/版本部分2301和大小/幂元数据部分2302。除了零填充2311之外,地址调整2316还可以包括不可变明文部分2306。大小/幂元数据部分2302被用于确定不可变明文部分2306中的位的数量和可变明文部分2308中的位的数量(其等于在地址调整2316中用于填充2311的位的数量)。在至少一些实施例中,可变长度元数据2313的附加一项或多项也可以用作用于可调整块密码2320的地址调整2316的一部分。例如,可变长度元数据2313可以包括其它上下文信息或元数据(例如,许可、特权级别、类型、位置、所有权等),如本文先前所述。在仍有另外的实施

例中，密码术上下文标识符寄存器2315可以用作地址调整2316的一部分。密码术上下文标识符寄存器2315可以包含与特定功能群(例如，过程、过程的子集、虚拟机(VM)、VM的子集等)相关联的唯一值(例如，随机或确定性生成的)。块密码2320可以是任何合适的解密算法(例如，32位块大小密码的可调整版本，诸如SIMON、SPECK、K-密码或其它可变块大小密码，或者对于更大的地址，PRINCE、XTS-AES块密码、LRW、AES-CTR模式等可以被使用)，如本文中所述的。

[0195] 当编码的线性地址的密文2304部分已经被可调整的块密码2320解密成解密的地址片2324时，那么在阶段II中可以形成解码的线性地址2330。如果作为地址的加密部分(即密文2304)的一部分包括标签/版本部分2301，则在形成解码的线性地址2330时，解密的基地址片2324中的标签/版本部分应被符号扩展位2322所覆写。在至少一个实施例中，最高位(例如，标签/版本部分2301和大小/幂部分2302)可以被设置为相同的位值(例如，0或1)。此外，不可变明文部分2306和可变明文部分2308的位构成解码的线性地址2330的较低位。在一些实施例中，处理器可以检查解密的地址片2324中的较高位的片是否具有预期值，所述期望值作为解密的地址片2324是否被不正确解密的指示。例如，在一些分页模式中，要求了一定数量的较高地址位都具有相同的值(即，全为0或全为1)。如果解密的地址片2324中的对应位具有不同的值，则这表明解密的地址片2324被不正确地解密。在这种情况下，一些实施例可能会生成故障。在使用解码的线性地址2330时，一些其它实施例可以依赖于现有的规范性检查来在那种情况下生成故障。即使较高位都具有相同的值，这也不能最终指示解密的地址片2324被正确地解密。一些实施例可以对在当前操作中要访问的最小和最大地址两者的预期位值执行前述检查，使得如果访问的任何部分越界，将可能生成故障。其它实施例可能仅要求访问的特定部分(例如，第一字节)处于指针的边界内，并且因此仅针对访问的该部分执行对指针的预期位值的前述检查。其它实施例可以检查写操作的最小和最大地址，但是仅检查读的单个指针值，其依赖于数据密码操作来可能地防止部分越界读返回正确的明文。

[0196] 解码的线性地址2330被用于在阶段III中找到要解密的加密数据的存储器位置。加密数据由诸如密钥流生成器2350的解密算法解密。在至少一个实施例中，密钥流生成器2350可以以特定的大小粒度(任何合适的大小)实现为AES-CTR模式块密码。在该实施例中，以密码方式编码的指针的内容被用作初始化向量(IV)或数据调整2344，其中明文移位(例如，2308)被用作计数器值(CTR)。密钥流2351的生成可以在不等待加密地址片2304完成的情况下着手。密钥流生成器2350基于数据密钥2342加密数据调整2344，以生成密钥流2351。在被用作密钥流生成器的输入之前，数据调整2344的值可以被调节为符合0(以密钥流生成器2350的块大小取模)。数据调整2344的值可以具有设置为0的一些合适数量的最低有效位，以满足该要求，并且可以丢弃密钥流2351的前缀，以解决该调节。可以通过从数据调整2344的未调节值中减去数据调整2344的调节值来计算要丢弃的密钥流2351的字节的数量。这种调节可以修改指向小于块大小的对象的指针中的不可变明文2306的值。然而，数据加密可以间接绑定到修改的不可变位的值，因为这些位可以被并入到用于生成密文2304的调整中。如果要加密的存储器跨一个或多个块对齐的边界，则密钥流生成器2350可以针对随后的块被重新调用，其中每次所述密钥流生成器2350被重新调用时，数据调整2344增加等于块大小的量。生成的密钥流2351的后缀可能是不需要的，并因此被丢弃。然后对密钥流

2351和从由解码的线性地址2330所引用的存储器位置中选择的加密输入数据块(或高速缓存行)2346执行异或操作2352。加密的输入数据块2346的粒度与从密钥流生成器2350输出的密钥流2351的粒度相匹配,并且异或操作2352产生解密的输出数据块2354。相似地,当将数据存储回高速缓存/存储器时,可以执行相同的操作,其中来自处理器寄存器的明文数据可以与用于编码的地址(这是数据调整2344)的密钥流输出2351进行异或操作,并且将所得加密数据写回到存储器。

[0197] 图24是示出获得由指针2410的另一实施例所引用的数据的示例过程2400的详细流程图,所述指针2410用数据的线性地址来编码,其中数据的加密被绑定到指针的内容。过程2400的至少一些部分可以由计算设备100的硬件、固件和/或软件来执行。在所示的示例中,指针2410被实施为64位编码的线性地址,其包括7位标签/版本部分2402和57位可变明文地址部分2408。标签/版本部分2402可以包括临时安全性位,每次为所请求的存储器分配编码指针时,所述临时安全性位就随机生成。备选地,标签/版本部分2402可以包括版本号或用于所请求的存储器分配的任何确定性唯一值。例如,标签/版本部分2402可以是每次为所请求的存储器分配来编码指针时生成的顺序递增的数。尽管可以使用任一种方法,但是在没有地址位被加密的该实施例中,具有随机生成的位的标签可以提供抵抗攻击的更好保护,因为随机生成的位比确定性不同的值(诸如顺序递增的数)更难确定。对于本文中公开的其它实施例,其中地址的一部分被加密,版本可能是更期望的,因为加密提供了对抗攻击的保护,并且版本号使用较少的资源来生成。

[0198] 应当注意,标签/版本部分2402和地址部分2408可以配置有不同数量的位(例如,5位标签/版本部分和59位地址部分2408等)。在其它实施例中,可以对附加位进行编码,以允许编码指针2410与传统指针同时使用。例如,编码指针可以分配有2位保留部分、5位标签/版本部分和57位明文地址部分。在该实施例中,2位保留部分可以被指定为监管者位和编码位指示符。监管者位可以指示地址是位于监管者地址空间(例如“1”)内还是位于用户模式地址空间(例如“0”)内。编码位指示符可以被设置为监管者位的相反值以指示指针被编码(例如,用标签/版本位),或者可以被设置为与监管者位相同的值以指示指针未被编码。在其它实施例中,可以在没有专用位的情况下实现传统编码。相反,传统编码可以通过在标签/版本部分2402中编码特定值(例如,全为1、全为0)来实现。在仍有其它实施例中,如果例如概念没有被实现为与传统程序兼容,则传统编码可以被完全消除。

[0199] 过程2400的操作被标识为两个阶段:地址形成(阶段I)和数据解密(阶段II)。过程2400不包括地址解密阶段,因为编码的线性地址没有被加密。相反,线性地址被编码在可变明文地址部分2408中。

[0200] 在阶段I中,可以从编码指针2410中形成解码的线性地址2430。在至少一个实施例中,不作为线性地址的一部分的最高位2422(标签/版本部分2402)可以被设置为相同位值(例如,0或1)。可变明文地址部分2408的位构成解码的线性地址2430的较低位。

[0201] 解码的线性地址2430被用于在阶段II中找到要解密的加密数据的存储器位置。加密数据由诸如密钥流生成器2450的解密算法来解密。在至少一个实施例中,密钥流生成器2450可以以特定的大小粒度(任何合适的大小)实现为AES-CTR模式块密码。在该实施例中,编码指针的内容被用作初始化向量(IV)或数据调整2444。特别地,与明文地址部分2408的位串接的标签/版本部分2402的随机生成的位形成了用于对加密数据进行解密的数据调整

(或IV)。密钥流生成器2450基于数据密钥2442来加密数据调整2444,以生成密钥流2451。然后对密钥流2451和从由解码的线性地址2430所引用的存储器位置中选择的加密输入数据块(或高速缓存行)2446执行异或操作2452。加密的输入数据块2446的粒度与从密钥流生成器2450输出的密钥流2451的粒度相匹配,并且异或操作2452产生解密的输出数据块2454。此外,在一些实施例中,其它上下文信息也可以用在数据调整2444中,其包括例如存储器分配大小、类型、许可、特权级别、位置、所有权、密码术上下文ID寄存器或其任意组合。

[0202] 将指针的标签/版本部分并入到数据加密/解密中可以被用作版本化存储器的一种方式,并且可以防止某些恶意攻击,即使线性地址完全作为明文嵌入在指针中(即,没有加密片)。例如,由于标签/版本位被并入到数据加密中,可以减轻释放后使用攻击(即,在底层对象已经被释放并且甚至潜在地被重新分配给不同的对象之后访问悬摆指针)和越界访问。如果启用了完整性检查,则这两种情景都很有可能导致完整性检查违例。如果未启用完整性检查,则释放后使用攻击或越界访问将导致数据损坏,这可能仍足以保护数据机密性。

[0203] 数据加密/解密中使用的标签/版本部分也可以减轻未初始化的使用弱点。当从已分配但未填充有初始值的缓冲器中读数据时,会发生未初始化的使用。通过延迟对象的对所有权的写直到对象被初始化,可以检测到未初始化的使用。对所有权的写指的是在没有从存储器的区域进行首次读的情况下向存储器发出写,就像对存储器写的惯例。当初始化对象时,软件可以适用于使用对所有权的写,以避免完整性检查违例,否则,如果使用嵌入在指针中的标签/版本值读存储器的区域,而所述标签/版本值不同于先前用于在存储器中写当前值的任何标签/版本值,则所述完整性检查违例就会发生。

[0204] 软件还可以适用于检测悬摆指针(例如,释放后使用)或未初始化对象的尝试访问。如果完整性检查被启用,则存储器分配器(例如,malloc指令、释放指令)可以适用于将零密文注入到新分配的(例如,malloc指令)存储器区域或释放的(例如,释放指令)存储器区域中。这可能会导致在访问悬摆指针或未初始化的对象时生成完整性故障。

[0205] 约束代码访问

如本文先前所述,诸如如本文中描述的用于访问数据的采用EBA格式的以密码方式编码的指针等编码指针也可以适用于控制代码获取。这包括具有标签/版本部分的以密码方式编码的指针(例如610、2110)以及具有标签/版本部分的未以密码方式编码的指针。此外,如果添加了标签/版本部分,例如,通过增加地址寄存器大小,通过降低嵌入在指针中的地址位的数量,和/或降低嵌入在编码指针中的其它元数据位的数量,也可以使用其它编码指针格式来控制代码获取。

[0206] 通常,代码获取是由处理器基于指令指针寄存器(诸如64位指令指针寄存器的RIP)的内容自动执行的。在用于控制代码获取的实施例中,标签/版本位被包括在诸如RIP(例如,64位寄存器)的指令指针寄存器中,以将代码区域的加密绑定到该RIP。本文中公开的各种指针中的任何一个,包括例如具有标签/版本部分的指针(例如,610、2110、2310、2410)和没有标签/版本部分的指针(例如,没有标签/版本部分的EBA指针610和2110,没有标签/版本部分的指针2310)可以被存储在RIP中以获取代码区域并将代码区域的数据加密绑定到引用该代码区域的指针。对于包括其它元数据(例如,指示指针中可变和不可变明文位的数量的大小位)的指针格式,则该元数据也可以被包括在RIP中。在一个或多个实施例中,在代码获取期间供使用的上下文信息(例如,较高地址位、存储器分配大小元数据、许可

元数据和针对EBA指针的类型元数据)可以存储在与RIP寄存器并排的专用寄存器中。间接分支指令可以例如在立即操作数中指定更新的上下文信息。当在函数之间发生代码跳转时(例如,当一个函数调用另一函数时),则特定的分支指令可以指定要放入专用寄存器的新的上下文值,以便新函数中的获取将会成功。因此,一些可能的扩展分支指令将在本文中参考多租户架构进一步描述。当执行函数调用时,与RIP相关联的上下文寄存器的内容也可以被推到与RIP并排的栈上,以便在对应的返回过程中将其恢复到与RIP相关联的上下文寄存器中。

[0207] 对于在加密和解密操作中不包括上下文信息的指针格式,可以使用RIP寄存器的内容来加密/解密代码区域。对于EBA格式指针和其它使用上下文信息的指针,可以使用RIP寄存器、专用寄存器和立即操作数(在某些情景中)来加密/解密代码区域。代码区域的这种加密/解密通过防止分支目标注入来帮助限制由程序可以执行的代码。分支目标注入可被用于ROP和JOP攻击,在所述攻击中,对手在代码的片断之间跳转,以执行不意在由代码执行的一系列操作。根据本文中公开的实施例格式化的代码指针迫使潜在的对手猜测或获得标签位值。每次加载程序以加强代码加密保护时,可以将标签位随机化成对每个代码区域的不可预测的值。通过迫使潜在的对手猜测或获得对目标代码区域的有效加密基地址片以及该区域的有效标签,使用具有在EBA实施例之下分层的带标签的数据加密的加密基地址指针(例如,610,2110)可以进一步加强代码加密保护。

[0208] 具有存储器分配大小上下文信息的EBA格式指针可以指示程序(例如,函数)的特定部分可以执行的边界。当执行代码访问时由处理器隐式地使用持有上下文信息的专用寄存器。当执行分支或调用指令时,新的扩展分支/调用指令被用于为专用寄存器指定更新的上下文信息(例如,在指令的立即操作数中),使得在新函数中的访问能够成功。

[0209] 通过加载时生成和修正采用EBA格式和采用包含标签部分的其它指针格式的指针,或运行时生成相同的指针,可以概率性地检查实现使用合法目标的所有间接分支的正向代码指针保护。这可以应用于函数和其它间接分支目标(例如,C编程语言中的切换语句)。

[0210] 编码指针也可以用来保密导入的函数。导入的函数是库文件,其根据需要动态地被提出到程序的地址空间中。因此,函数可以被提出到不同程序中的不同位置中。此外,用于加密/解密指针的密钥也会改变。因此,当指针值被加载到不同程序中时,它可能需要改变。通常,大多数二进制结构都具有包含关于导入的函数的地址信息的节。在本文中这个节被称为全局函数表。全局函数表的一个示例是用于可执行和链接格式(ELF)文件的过程链接表(PLT)。二进制内对导入的函数本身的任何引用都表示为从全局函数表的开始的偏移(例如,在全局偏移表(GOT)中)。对于使用具有标签部分但没有上下文信息的指针格式的实施例,在加载时,通过生成和包括新生成的标签位,全局函数表中的实际地址可以被指针所替换。对于使用具有标签部分和上下文信息的指针格式(例如,EBA格式指针610、2110)的实施例,在加载时,通过生成和包括新生成的标签位,全局函数表中的实际地址可以被EBA指针所替换,其中EBA指针被绑定到指示目的地代码区域(例如,函数、摆脱控制流转移的基块或编译单元)的大小的上下文信息。

[0211] 对于普通指针(例如,不引用导入的函数),与可执行代码(例如,ELF文件)相关联的元数据被存储在可执行文件中。元数据指示在可执行文件中代码指针的位置以及它们引

用的对象。一旦可执行文件被加载，则可执行文件中的代码指针要被更新。当可执行文件被加载时，软件组件(例如，加载器)可以浏览代码指针的列表，并根据它们的特定编码格式(例如，诸如EBA的加密片、非加密地址位等)对它们进行编码。

[0212] 对应的代码区域(例如，函数、摆脱控制流转移的基块或编译单元)则可以被相应地加密。函数可以包含一个或多个基块，其可以具有单个入口点和出口点。通常，与直接分支连接的基块(例如，要跳转的字节的数量被嵌入在代码中而不是寄存器中)可以具有相同的上下文信息。然而，在基块之间转移的间接分支(例如，C编程语言中的切换语句)可能使用不同的上下文信息。编译单元是可能包括文件的文件。因此，基于特定的实现和需求，可以在不同的粒度上定义代码的区域。

[0213] 更细粒度的策略可能会减轻更多的弱点。然而，如果使用基于块密码的代码加密方案(其要求利用区分的标签加密的代码与块边界对齐)，则这种策略可能会增加代码大小。备选地，可以通过使用基于流密码的加密方案(例如，AES-CTR模式)来实现不增加代码大小的细粒度保护。

[0214] 可以执行类似的指针标记操作来更新被嵌入在全局变量中的代码指针、指向未经由全局函数表调用的函数或其它代码的间接分支目标、或者非位置独立程序中的代码指针。

[0215] 如上所述，可以通过控制每个程序可以访问哪些编码指针来实现对程序可以访问什么数据进行控制。例如，可以通过将指针作为参数传递或将指针作为全局变量提供来提供访问。然而，诸如EBA指针的以密码方式编码的指针也可以被用来约束哪些代码段可以访问特定数据。在至少一个实施例中，这可以通过将代码段的散列绑定到用于加密指针的上下文信息中来实现。

[0216] 首先，当利用间接调用或返回进入代码段时，可以计算每个代码段的散列(因为EBA指针610或2110或者利用大小/幂元数据2310编码的指针指定了代码段的边界)并且所述代码段的散列被用来初始化具有标识当前函数的散列的CURRFUNCID(当前函数ID)寄存器。对于函数调用的当前栈的CURRFUNCID值的序列也可以被累加到表示当前控制流的CURRCFID(当前控制流ID)寄存器中。可以使用可逆累加算法(例如异或)，使得对应的返回(例如RET)可以自动回到CURRCFID中的先前散列值，而不需要存储它。例如，对于间接调用，这可以在CURRFUNCID被更新为表示被调用者之后通过计算CURRCFID=CURRCFID异或CURRFUNCID来实现，并且对于对应的返回，可以在CURRFUNCID仍然表示正在返回的被调用者时通过计算CURRCFID=CURRCFID异或CURRFUNCID来实现。CURRCFID可以在开始软件的定义单元时被初始化为常数值，例如，当程序内的线程开始时被初始化为零。来自使用直接调用来调用的函数激活的返回可能不需要回到CURRCFID中的散列值。为检测返回是否对应于先前的间接调用，可以将指示符编码到返回地址中，例如作为位设置。备选地，为了保存指针位，可以由调用指令自动地或由软件手动地将指示推到与返回地址并排的栈上(例如作为与RIP相关联的上下文值的一部分)，并由返回指令自动地或由软件手动地检查。在后一种返回情况下，软件可以取决于CURRCFID是否需要更新来在返回指令的多个变体之间进行选择。在间接调用时，CURRFUNCID寄存器值也可以拷贝到PREVFUNCID(先前函数ID)寄存器中。对应地，CURRCFID寄存器值也可以在间接调用时拷贝到PREVCFID(先前控制流ID)寄存器中。对代码段的散列进行计算可能是慢的，因此备选实施例可以用传递到间接调用或返

回的指针的加密片或指向包含目的地函数的代码段的基的整个指针替换代码段的散列。可以从间接调用的目的地地址以及返回地址两者中计算代码段的基地址,因此它可以支持在调用和返回两者之后更新CURRFUNCID和CURRCFID寄存器。

[0217] 此外,指令要在ENDBRANCH指令(或其它类似的指派指令)之后被使用,以检查某个控制流是否已用于到达该点。如果检查成功(例如,跳转控制流完整性指令(JCFI)),则一条指令可以执行直接分支,如果检查失败(例如,控制流完整性检查(CFIC)),则另一条指令可以生成故障。到紧接着CFIC而在其之后的点的一系列JCFI分支可能接着的是单个CFIC,以支持对授权控制流的任意大的集进行编码。每个指令的操作数可以指定与PREVFUNCID寄存器或PREVCFID寄存器中的值进行比较的表示授权控制流的散列值。可以使用操作数来选择在比较中使用那些寄存器中的哪一个,或者可以定义不同的指令变量或前缀来在那些寄存器之间进行选择。

[0218] 控制流完整性(CFI)鉴别散列也可以被包括在用于一些代码或数据加密的数据调整中,以约束控制流路径可以根据其来访问代码或数据的函数。例如,地址调整和代码调整或数据调整的某种组合可以绑定到当前代码段的标识(例如,使用CURRFUNCID值)或函数调用的相关序列(例如,使用CURRCFID值)。代码段或数据对象被绑定到存储在特定寄存器中的CFI鉴别散列的指示符可以被编码到引用该代码段或数据对象的指针中、相关联的上下文中、或者导致存储器访问的指令中。EncryptBaseAddr指令可以被扩展以将授权散列及其含义(例如,对应于CURRFUNCID或CURRCFID)指定为一个或多个附加操作数。如果SpecializePtr指令被嵌入在指针中,则所述SpecializePtr指令可以将约束从旧指针传播到新指针。为初始化仅从特定函数(例如,如在CURRFUNCID中标识的)或在控制流中的特定点(例如,如在CURRCFID中标识的)可访问的存储器,可以定义WRCFMEM(写控制流存储器)指令,其接受为相关寄存器指定覆盖值的操作数(例如,CURRFUNCID或CURRCFID)、指向目的地存储器位置的指针以及要写的代码或数据值。然后,WRCFMEM可以加密数据,就像在将数据写到存储器之前相关寄存器已经使用指定值被覆盖一样。

[0219] 在其它实施例中,可以扩展用于采用EBA格式的以密码方式编码的指针的上下文信息,以通过将控制流表示或特定代码段的散列(例如,对应于CURRFUNCID值)或函数调用的序列(例如,对应于CURRCFID值)并入上下文信息中(作为使用该散列作为数据调整、代码调整或地址调整的补充或者替代)来指定哪些函数或控制流被授权访问特定指针。EncryptBaseAddr指令可以被扩展以将授权散列及其含义(例如,对应于CURRFUNCID或CURRCFID)指定为一个或多个附加操作数。如果SpecializePtr指令被嵌入在指针中,则SpecializePtr指令可以将约束从旧指针传播到新指针。在该实施例中,可以将WRCFMEM定义为接受指针和上下文编码,这可以避免需要单独的操作数来指定散列值。

[0220] 多租户架构

如本文中所描述的具有EBA格式的以密码方式编码的指针,或者任何其它以密码方式编码的指针,也可以用于支持单个地址空间中的多租户软件架构。租户可以被定义为由处理器102执行的代码的集合(例如,函数、函数的群、过程、虚拟机或程序代码的其它集合)。在使用以密码方式编码的指针(例如610、810、2310)或未以密码方式编码的指针(例如2410)的多租户环境中,密码操作强制租户之间的隔离而非页表或扩展页表之间的隔离。

[0221] 图25是示出示例多租户环境2500的简化框图,所述环境2500包括支持租户A

2520A和租户B 2520B的单个地址空间2510。使用以密码方式编码的指针在租户之间实行隔离。地址空间2510包括可信运行时2530，其促进租户2520A和2520B之间的仿真的过程间通信2532。为了简单和易于描述，图25中仅示出了两个租户。然而，应注意，本文中描述的概念可以应用于具有许多租户的规模实现。

[0222] 可以实现若干指令规则，以使用本文中先前示出和描述的涉及以密码方式编码的指针(例如，610、2110、2310)和未以密码方式编码的指针(例如，2410)的概念来实例化多租户软件架构(例如，2500)中的以密码方式的隔离。首先，租户不被许可生成以未处理指针作为输入的指针，因为这可能使租户能够访问未授权的存储器。通常，在多租户环境中，租户不受其它租户或可信运行时的信任。如果每个租户都可以访问地址空间的任何部分，则恶意租户就可以访问另一租户的数据或代码，损坏另一租户的数据或代码。因此，生成指向单个地址空间的任何部分的加密指针的能力受到约束。例如，在一个或多个实施例中，租户被约束执行EncryptBaseAddr指令。该约束由以密码方式编码的指针和未以密码方式编码的指针来实行，这将在本文中进一步描述。第二个指令规则包括允许租户执行SpecializePtr指令，因为这产生的指针最多提供已由允许租户访问的现有指针供应的许可。

[0223] 前面的指令规则将每个租户仅仅约束在由指针所覆盖的存储器区内，所述指针由租户的监管者(例如操作系统或可信运行时)作为输入提供到该租户。在至少一个实施例中，指令规则可以通过在运行程序代码之前扫描程序代码(例如，汇编语言输出)以作为被禁止的指令的不存在、通过定义那些指令在其中会故障的新模式、通过将某些代码页标记为被禁止使用那些指令、或者通过它们的任何合适的组合来实行。指令约束也可以被编码在与指令指针寄存器(RIP)相关联的上下文信息中，并且被以密码方式绑定到RIP(例如，作为新指派为指示不允许EncryptBaseAddr的上下文信息中的一个位)。在另一实施例中，指令约束可以在源代码级别实行，其中要由租户执行的源代码由可信编译器(例如，在云中)编译，这防止了在可执行代码中生成EncryptBaseAddr指令。

[0224] 在一个或多个实施例中，可信运行时2520包括受租户信任的软件，但不一定是特权软件。除了可信软件，可信运行时2530还可以包括任何合适组合中的固件和/或硬件。当被执行时，负责将租户代码加载到存储器中，以用于调度租户，并用于配置处理器存储器访问控制功能，以授予每个租户对其被授权访问的存储器的精确访问。它还可以提供服务，诸如为租户分配或解除分配私有或共享存储器的区域，在租户之间发送消息，终止租户，以及向被授权访问第一租户的一个或多个其它租户提供第一租户的密码密钥。

[0225] 在多租户环境中，可信运行时2530将每个租户约束在其自己的授权存储器区域内。可信运行时2530可以生成一个或多个指针，所述指针引用一个或多个“私有”存储器区域，特定租户被授权访问所述“私有”存储器区域，但是其它租户没有被授权访问所述“私有”存储器区域。可以由可信运行时2530用未处理指针(例如，使用EncryptBaseAddr指令)生成指向授权存储器区域的指针。此外，可信运行时在加载租户代码时对其进行加密，并生成指向租户代码的指针。由可信运行时2530生成的指针可以在租户代码被启动时或者在执行期间(例如，当从另一租户接收到消息时)被提供给租户代码。在一个示例说明中，当可信运行时2530启动租户A 2520A时，可信运行时2530生成覆盖租户A 2520A被授权访问的特定存储器区域的EBA指针，然后向租户A 2520A提供EBA指针以在其执行期间使用。应注意，授权的存储器区域可以包含存储器区域内的子区域，所述子区域可以包含它们自己的指针。

此外，可信运行时2530还可以生成一个或多个指针，所述指针引用一个或多个“共享”存储器区域，允许单个地址空间内的所有租户访问所述“共享”存储器区域，或者允许单个地址空间内的租户的子集访问所述“共享”存储器区域。

[0226] 在一个或多个实施例中，租户被提供有指向授权存储器区域的至少一个指针，并且可以生成指向授权存储器区域内的子区域的专用指针。在一个示例中，可以允许租户使用由可信运行时2530提供的指针作为用于指令的输入来执行本文中先前描述的SpecializePtr指令。SpecializePtr指令可以采用输入指针，其覆盖针对租户的授权存储器区域，并在较大的授权区域内生成覆盖较小区域(或子区域)的专用指针。因此，租户可以将指针传递到其整个存储器区域，并请求指向该存储器区域内特定对象的指针。SpecializePtr指令可以验证租户已请求访问的对象被输入指针所引用(即，位于输入指针所指向的存储器区域内)。

[0227] 如本文中先前所述，秘密密码密钥被用于加密EBA指针(例如610、2110)的基地址片和使用其它以密码方式编码格式的指针(例如2310)中的地址片，并加密由所述指针所引用的数据或代码。在多租户环境中，密钥可以在租户之间交换，使得每个租户使用不同于其它租户的密钥来为其指针生成加密基地址片(以及使用其它以密码方式编码格式的指针中的地址片)，并加密由所述指针所引用的数据或代码。切换密钥可以加强隔离，而不要求转译后备缓冲器(TLB)逐出，或者像常规的页表切换那样施加附加开销的其它动作。实施例还允许在租户2522之间切换存储器访问，其中一个租户转移控制，使得另一租户可以访问第二租户的代码和数据。第一租户(例如，租户A 2520A)可以使正被访问的租户(例如，租户B 2520B)的密钥被激活以供使用，以便成功地访问正被访问的租户的数据和代码。

[0228] 图26是示出可与多租户环境中的租户相关联的可能保密性句柄(handle)2600的框图。在一个或多个实施例中，保密性句柄2600对于每个租户可以是唯一的，并且可以存储在存储器中。保密性句柄2600可以包括密钥三件套2610，其包括地址密钥2612、数据密钥2614和代码密钥2616。在以密码方式的算法中可以使用地址密钥来加密/解密要嵌入到引用该基地址的指针中的基地址片。数据密钥可以在以密码方式的算法中用于加密/解密以密码方式编码的指针所指向的数据。在以密码方式的算法中可以使用代码密钥来加密/解密以密码方式编码的指针所指向的代码。在至少一个实施例中，一个或多个以密码方式的算法可以包括一种或多种类型的块密码，如本文中先前描述的。尽管在单个保密性句柄2600中针对指针地址、数据和代码示出了单独的密钥，并且可以提供更高级别的保密性，但是应当理解，基于特定的需要和实现，可以使用单个密钥或者可以以任何合适的组合使用两个密钥(例如，用于地址的一个密钥以及用于代码和数据的不同的密钥，用于数据的一个密钥以及用于地址和代码的不同的密钥，用于代码的一个密钥以及用于地址和数据的不同的密钥)。此外，还应注意，取决于特定的实现，可以仅包括某些密钥。例如，可以包括地址和数据密钥而不具有代码密钥，可以包括地址和代码密钥而不具有数据密钥，或者可以包括数据和代码密钥而不具有地址密钥。

[0229] 保密性句柄2600还可以包括与密钥三件套2610相关联的授权入口点指针2620。可选地，如果授权入口点指针未被加密，则授权入口点指针可以包括其它元数据。授权入口点指针可以包括指向被授权使用密钥三件套2610来解密以密码方式编码的指针以及来解密由指针所引用的数据或代码的存储器区域中的地址的指针。基于嵌入在目的地租户的保密

性句柄2600中的入口点指针2620，加载目的地租户的保密性句柄的密钥可以导致到目的地租户的授权入口点的分支。这确保了其它租户只能在其授权代码位置调用目的地租户。这可有助于防止恶意租户调用受害者租户中可能危及受害者的保密性的功能(例如，读私钥或其它数据并将其返回给调用者的功能)。此外，由于保密性句柄是存储器中的结构，本文中先前描述的EBA指令可以用于通过将授权的存储器区域设置为仅包括某些句柄来控制每个租户可以调用哪些其它句柄。

[0230] 在至少一个实施例中，保密性句柄(例如，2600)可以由第一租户(可以是不可信软件)使用，以向处理器供应第二租户的入口点指针以及第二租户的密钥或密钥的句柄，以实现到第二租户的切换，而不调用诸如可信运行时的可信软件。在示例中，如果租户A想调用租户B，则租户A可被给予对用于租户B的保密性句柄的访问。例如，如果租户A被授权访问租户B，则可信运行时可以将租户B的保密性句柄存储在租户A的存储器区域中。租户A可以执行新的指令，其接受保密性句柄作为存储器操作数，并且基于保密性句柄的内容，为不同的密钥类型(例如，地址、数据和/或代码)更新指令指针和密钥寄存器。例如，新指令的执行可以包括将保密性句柄中指定的地址密钥(例如，2612)加载到寄存器中，以用于解密采用EBA指针(例如，610，2110)或其它以密码方式编码的指针(例如，2310)中的加密片，并且更新指向租户B的保密性句柄(例如，2600)中指定的授权入口点指针(例如，2620)的指令指针。这防止了租户A分支到租户B中的任意代码，租户B可能希望保护所述代码免受外部(其它租户)访问，例如，如果所述代码包含敏感数据和/或代码的话。此外，如果保密性句柄包括数据密钥和/或代码密钥，那么这些密钥也可以被加载到适当的寄存器中，使得它们被激活以用于实现对租户B的数据和代码的成功访问。

[0231] 鉴别和加密可被用于维护密钥和授权入口点指针的保密性和完整性。可以通过基于密钥三件套2610和授权入口点指针2620生成消息鉴别码2630来评估保密性句柄2600的完整性。可以使用MAC算法和密钥来生成MAC。试图使用保密性句柄2600的租户可以通过基于密钥三件套2610和授权入口点指针2620来使用相同的MAC算法和密钥生成第二MAC来验证其完整性(即，其内容尚未被修改)。第二MAC与保密性句柄2600中的MAC 2630进行比较。如果它们匹配，则验证保密性句柄2600的完整性，并且租户可以使用保密性句柄2600来访问相关联的租户。但是，如果MAC不匹配，则会生成故障，并且阻止试图使用保密性句柄的租户这样做。MAC不匹配可指示密钥中的一个或多个、授权入口点指针或MAC已被修改。

[0232] 密钥和可选的授权入口点指针2620可以通过具有包装密钥(wrapping key)的鉴别和加密来保密。在第一示例中，包装密钥2602可被用于加密密钥三件套2610和授权入口点指针2620(单独地或组合地)。在第二示例中，包装密钥2602可以用于加密密钥三件套2610(单独地或组合地)，但是授权入口点指针可以不被加密。当租户加载保密性句柄时，包装密钥2602随后可被用于解密密钥和授权入口点指针(如果加密的话)。这些加密和解密步骤可以与MAC生成步骤分开发生，或者它们可被组合。例如，一些具有相关联数据的鉴别加密(AEAD)(或者如果没有未加密的鉴别数据，则仅鉴别的加密)模式组合了这些步骤，而其它模式则分开地执行了这些步骤。在一个实施例中，包装密钥2602可以存储在处理器核中，以提供附加的保密性。在其它实施例中，包装密钥可以单独存储在存储器中。

[0233] 在一些实施例中，可以为每个租户定义用于每种密钥类型(例如，地址密钥、数据密钥、代码密钥)的多于一个密钥，以支持多个租户之间的存储器区域的选择性共享。例如，

可以为单个租户定义单个代码密钥、多个地址密钥和多个数据密钥。当为租户的群定义多个密钥时，可以使用共享存储器区域来实现群租户之间的直接通信，而不是严格隔离并通过可信运行时发送所有消息。

[0234] 特定租户可以被授权访问任意数量的存储器区域，并且每个单独的存储器区域可以由指针或其上下文信息来标识。指针或其上下文信息可以指示该存储器区域对于当前租户是私有的、在所有租户之间全局共享的还是由租户的子集所共享的。因此，特定租户可与一个或多个指针或上下文信息实例相关联，诸如仅针对该租户授权的存储器区域的私有指针或上下文信息、针对所有租户授权的存储器区域的广播指针或上下文信息、针对可由租户读但不可写的存储器区域的单播指针或上下文信息(例如，另一租户可以填充该存储器区域，并且租户可以消耗来自该区域的数据)、和/或可以由多个租户读和写的存储器区域的多播指针或上下文信息。在一个或多个实施例中，上下文信息可以包括以密码方式的上下文ID，其指示对存储器区域的授权(例如，私有、单播、多播、广播)。在一些实施例中，特定租户可以与针对这些类型的指针和上下文信息实例中的任何一个的多个指针或上下文信息实例相关联。

[0235] 对于共享的存储器区域中的每个，选择了特定的密钥来加密地址或数据。因此，单独的加密密钥可以被用于不同类型的共享的存储器区域。在由租户的子集所共享的存储器区域中，子集中的每个租户可以为其私有存储器区域使用不同的数据密钥，但是所有租户可以为共享的存储器区域使用相同的数据密钥。例如，考虑了对租户A和租户B的共享的存储器区域。租户A可以具有对用于加密/解密分配给租户A的第一私有存储器区域中的数据的第一数据密钥的访问。租户B可以具有对用于加密/解密分配给租户B的第二私有存储器区域中的数据的第二数据密钥的访问。然而，租户A和租户B两者都可以具有对用于加密/解密共享的存储器区域中的数据的共享数据密钥的访问。可以定义单独的寄存器来为密码术上下文中的每个持有不同的密钥。可能是有益的是，如果支持了大量的多播群以降低硅面积要求，则将用于多播群的密钥存储在由多播群ID索引的存储器中的表中。

[0236] 当在多租户环境中使用以密码方式编码的指针时，切换租户时要更新一个或多个密钥。租户在本文中也可以称为“隔室”。在至少一个实施例中，取决于哪个软件可信来访问和更新密钥，可以使用可从用户模式和监管者模式中的一个或两个模式中可执行的指令来完成在切换租户时更新密钥。例如，诸如图25的可信运行时2530等管理租户的可信运行时可以使用可以在用户模式下执行的密钥更新指令(例如，环级别3代码)。当在租户之间转移控制时，可信运行时可以执行密钥更新指令来更新(一个或多个)相关的密钥寄存器。如本文先前关于某些其它指针指令(例如EncryptBaseAddr)所述，某些密钥更新指令可以被约束到仅可信运行时。

[0237] 当密钥被用于加密和解密代码、数据和/或地址时，可以使用用于读和写密钥寄存器值的指令，其可以由可信运行时执行或者由租户直接执行。然而，如果代码是使用单独的密钥为单独的租户加密的并且代码没有被安排成使得目的地租户的代码在切换代码密钥的WRKEY指令之后立即开始，则这种指令可能不可被用于切换代码密钥。在一个示例中，这种读(RDKEY)和写(WRKEY)指令可以被定义如下：

RDKEY:rs1/xmm、rs2

rs1/xmm:密钥值将被写到其的寄存器操作数输入。操作数的大小可以取决于密钥的大

小而变化(例如,针对64位通用寄存器的r64或针对128位寄存器的xmm)。

rs2:寄存器操作数(例如,64位),其指定要读的密钥寄存器的索引(例如,当存在多个密钥时,诸如租户的不同对之间的共享对象)。

WRKEY:rs1/xmm、rs2

imm64或rs1/xmm:包含要写的密钥值的立即操作数或寄存器操作数输入。如果密钥值太大,不能放入立即操作数中,则可以使用寄存器操作数。操作数的大小可以取决于密钥的大小而变化(例如,对于64位通用寄存器的r64或128位寄存器的xmm)。

rs2:寄存器操作数(例如,64位),其指定要更新的密钥寄存器的索引(例如,当存在多个密钥时,诸如租户的不同对之间的共享对象)。

[0238] 如果密钥没有被用于加密和解密代码,则仅从用户模式更新密钥的指令是可用的。然而,如果使用密钥中的一个来解密代码,则使用用于分支和更新密钥两者的单个指令,使得在可以使用新密钥正确解密的位置处执行继续。例如,如果分支和更新不是在单个指令中被执行的,那么如果利用密钥加密代码并且执行写密钥指令,则密钥将被更新,但是它可能处于使用先前密钥(即,在执行写密钥指令之前)加密的代码区域中。除非在写密钥指令之后立即利用新密钥对代码进行重新加密,否则用于解密代码的新更新密钥可能会失败。因此,执行分支以及密钥更新的密钥指令防止了这种情景。

[0239] 用于分支和还更新密钥寄存器值的示例指令可以被定义为JKEY。JKEY指令可以指定租户跳转的目的地地址以及针对密钥寄存器的新值,以更新要在目的地执行的代码的代码密钥。一旦控制被转移给目的地租户,就可以由目的地租户更新其它的密钥值(例如,地址和数据密钥)。JKEY指令可被用于直接在租户之间、从租户到可信运行时、或者从可信运行时到租户转移控制。对于用于分支和更新密钥寄存器值的JKEY指令的示例格式可以被定义如下:

JKEY:rs1、rs2/xmm、rs3

rs1:寄存器操作数(例如,64位)输入,其包含用于分支到目的地(例如,租户、可信运行时)的目标地址

rs2/xmm:寄存器操作数输入,其包含用于解密代码的密钥寄存器的新值。备选地,代码密钥的新值可以作为立即操作数被嵌入。操作数的大小可能取决于密钥的大小而变化(例如,对于64位通用寄存器的r64或对于128位寄存器的xmm)。

rs3:寄存器操作数,如果目的地地址是以密码方式编码的指针,如果外部上下文信息用于生成以密码方式编码的指针,并且如果外部上下文信息是静态可用的,则其包含上下文信息。

[0240] 用于分支和还更新密钥寄存器值的另一示例指令可以被定义为JHND。JHND指令可以指定指向存储器中的保密性句柄的存储器操作数。保密性句柄可以包含新的密钥值(例如,地址、数据、和代码密钥)来更新适当的密钥寄存器和指向租户正在跳转的授权入口点指针。JHND指令可被用于直接在租户之间、从租户到可信运行时、或者从可信运行时到租户转移控制。分支和更新用于地址、数据和代码密钥的密钥寄存器值的JHND指令的示例格式可以被定义如下:

JHND:m1,rs2/xmm

m1:存储器操作数(例如,64位),其指向目的地(例如租户、可信运行时)的保密性句柄。

[0241] 如果保密性句柄中的授权入口点指针是以密码方式编码的指针，并且如果上下文信息被用于生成以密码方式编码的指针，则保密性句柄可以被配置为包括上下文信息。

[0242] 图27是示出当使用以密码方式编码的指针时用于从一个租户跳转到另一租户的指令(诸如JKEY)的示例操作的简化框图。图27包括租户A代码页#0 2710和具有代码页#0 2730和代码页#1 2740的租户B 2720。在该示例中，常规跳转指令2724被用于从租户B 2720中的代码页#0 2730跳转到也在租户B 2720中的代码页#1 2740。从相同区域内的一个代码页跳转到另一代码页不要求JKEY指令，因为代码页的代码密钥在相同租户内是相同的。图27还示出了从租户A代码页#0 2710进行分支的租户到租户(tenant-to-tenant)尝试，租户A代码页#0 2710包括三个JKEY指令2712A、2712B和2712C。JKEY指令中的每个都尝试访问租户B的代码页2730和2740中的一个。

[0243] 从至少保密性角度来看，限制租户在调用另一租户或可信运行时可以使用的入口点可能是可期望的。在至少一个实施例中，可以通过在每个入口点处使用指派的所需指令来限制入口点。例如，可以使用英特尔®控制流实行技术(CET)ENDBRANCH指令。在这个示例中，JKEY指令可以被设计成要求它的目的地解密成ENDBRANCH指令。然而，ENDBRANCH也可以在每个租户内来约束控制流，这可以许可任何有效的函数入口点同样可被用作租户入口点。因为这通常是不期望的，所以可以进一步约束入口点。特别地，JKEY指令可以被设计成检查其目的地地址是否与页边界(例如4KB)对齐，否则生成故障。这可以防止JKEY落在不处于页的开始处的ENDBRANCH指令上。此外，对编译器进行了修改，以避免将函数入口点或利用ENDBRANCH指令开始的其它代码结构放在作为无效入口点的页的开始处。这可以通过将ENDBRANCH指令向后移位一个字节并在页的开始处插入NOP指令来纠正。

[0244] 在图27中，JKEY指令2712A是成功的，因为它的目的地解密到了租户A 2720中的代码页#0 2730的所需指令(ENDBRANCH指令2722)。JKEY指令2712B失败，因为它的目的地不处于页的开始(即，不是页对齐的)。尽管JKEY指令2712C跳转到代码页#1 2740的开始，但是JKEY指令2712C也失败了，因为它的目的地没有解密成ENDBRANCH指令。

[0245] 可以相互合作地使用多种技术来保护密钥不被公开。首先，密钥可以被嵌入到标记为仅执行的代码页中，以防止软件直接读密钥。然而，对手可能仍然能够跳转到密钥本身内的非预期指令偏移，并试图通过观察处理器将密钥值解释为指令的效果来确定密钥值的全部或部分。这可以通过启用CET来防止到非预期移位的分支来减轻。可能的是，用于ENDBRANCH指令的字节序列可能碰巧出现在密钥值中，因此在使用之前应扫描密钥值，以验证它们不包含嵌入的ENDBRANCH序列。通过移除嵌入在密钥内的ENDBRANCH序列，这可以防止对手跳转到密钥内的非预期偏移，从而允许处理器执行可能碰巧在密钥值中表示的“代码”，并试图从观察处理器响应中了解密钥的内容。

[0246] 要求JKEY指令落在ENDBRANCH指令上的备选实施例是定义专门用于此目的而不是重用ENDBRANCH的新指令，例如LANDKEY。

[0247] 另一备选实施例是定义在存储器中使用描述符表的调用门(call gate)。调用门是代码入口点和资源(例如，密钥或密码术上下文ID)的规范，其可由该租户代码使用，所述租户代码由监管软件授权由一个或多个其它租户来调用。监管软件可以是传统的操作系统、本身由操作系统监管的可信运行时等。调用门既更新指向授权入口点的指令指针(例如，在RIP中)也更新一个或多个密钥寄存器值。例如，授权的调用门可以被存储在描述符表

中,所述描述符表包含多个描述符,所述描述符包含关于当前租户可以调用的其它租户的信息。描述符表可以由监管者模式管理,并且可以生成指向描述符表的指针。可以为租户定义用户模式指令(例如CALLCOMPARTMENT),以用于调用其它租户。CALLCOMPARTMENT指令被设计成接受到描述符表中的索引,在表中查找描述符。这个描述符为CALLCOMPARTMENT指令给予了新租户入口点和密钥值。CALLCOMPARTMENT指令然后可以跳转到入口点并更新密钥值。

[0248] CALLCOMPARTMENT指令提供了若干优点。其将避免检查入口点的对齐以及检查在入口点处是否存在ENDBRANCH/LANDKEY指令的需要,因为入口点值已被信任。它还将消除在租户的入口点之后用来更新附加的密钥寄存器的WRKEY指令的需要,因为它们都可以作为通过调用门传递的一部分而被更新。

[0249] 在仍有的另一实施例中,代替在租户之间切换时切换密钥,使用上下文调整值的增强格式,对象可以被绑定到不同的以密码方式的上下文。以密码方式的上下文是特定的存储器区域,其可以被授权给处理器(例如,102)的单个租户、两个或更多个租户的集、或所有的租户。因此,以密码方式的上下文可以是单个租户可访问的私有存储器区域、或者是两个或更多个租户可访问的共享存储器区域。特定租户可以被授权访问任意数量的存储器区域,并且每个单独的存储器区域可以由以密码方式的上下文标识符(或密码术上下文ID)来标识。用于加密/解密指针中的地址片、数据或代码的调整可以扩展到包括此密码术上下文ID。因此,代替在租户之间切换控制时切换密钥,可以为每个以密码方式的上下文切换密码术上下文ID值。因此,可以消除或降低切换密钥的开销,尤其是当生成具有子密钥的密钥调度时。

[0250] 特定租户可以与一个或多个密码术上下文ID相关联,所述密码术上下文ID诸如用于仅针对该租户进行授权的存储器区域的私有密码术上下文ID、用于针对所有租户进行授权的存储器区域的广播密码术上下文ID、用于可由租户读但不能写的存储器区域的单播密码术上下文ID(例如,另一租户可以填充存储器空间,并且租户可以消耗来自该空间的数据)、和/或用于可由多个租户但不是所有租户读和写的存储器区域的多播密码术上下文ID。在一些实施例中,特定租户可以与多个密码术上下文ID相关联,以用于这些类型的密码术上下文ID中的任一种。

[0251] 在特定实施例中,密码术上下文ID可以由例如可信运行时生成,所述可信运行时可以为多租户环境的多个(或所有)租户生成密码术上下文ID。在一些实施例中,密码术上下文ID是随机数(例如,随机64位值)。

[0252] 对于正被访问的存储器位置,以密码方式的上下文ID可以标识当前租户对该存储器位置的访问的范围。通常,只要使用相同的密钥,密码术上下文ID对于每个以密码方式的上下文可能是唯一的,因为密码术上下文ID在多租户环境中使用的每个密钥内是唯一的。例如,如果两个租户共享特定的存储器区域,如果单个密钥被用于该区域,则它们每个都可能与用于该区域的相同密码术上下文ID相关联。虽然数据共享也可以通过让租户共享密码操作密钥来实现,但是使用密码术上下文ID可能是共享数据的更有效的方式(例如,交换密钥的开销可能是繁重的,尤其是当必须生成子密钥时)。

[0253] 可期望的是,防止不可信的软件伪造密码术上下文ID,以防止恶意软件执行未授权的数据访问。例如,密码术上下文ID可以被存储在嵌入在程序代码中的上下文信息中(例

如,作为操作数),所述程序代码已经被可信运行时扫描以确保它仅使用授权的密码术上下文ID,或者在免受恶意访问的表中。然而,这要求附加的扫描和存储器访问约束,由于它们引入系统的复杂性,这可能是不期望的。

[0254] 图28示出了使用秘密密码术上下文ID值来概率性地防止对手伪造针对给定对象的正确ID值的备选实施例。特别地,对于使用EBA格式作为其以密码方式编码的指针的多租户环境,增强的上下文信息2800可以被用于指定秘密密码术上下文ID。增强的上下文信息2800可以包括具有二十个位的较高地址位字段2802、具有二十九个位的存储器分配大小字段2804、具有十二个位的类型字段2806、具有三个位的许可字段2808以及具有六十四个位的密码术上下文ID字段2810。在该示例中,增强的上下文信息2800从64位字段增加到128位字段。

[0255] 生成以密码方式编码的指针(例如EncryptBaseAddr、SpecializePtr)的软件可以直接将秘密密码术上下文ID插入到上下文调整中。在至少一个实施例中,密码术上下文ID可以被调整大小以防止对手成功猜测它们。因此,如果密码术上下文ID是大的,可优选的是将相关ID存储在寄存器和/或表中,所述寄存器和/或表在切换以密码方式的上下文时更新。然后,软件可以指定较小的密码术上下文索引来随后用于查找密码术上下文ID,代替软件直接将完整的密码术上下文ID指定为操作数。则这将缩小包含范围,在所述范围内,相对于针对完整密码术上下文ID的要求,密码术上下文索引必须是唯一的。例如,如果在切换过程时更新了密码术上下文ID寄存器,即使单个密钥被用于多个过程,则密码术上下文索引也只需要在过程之间是唯一的。如同任何上下文信息的情况一样,密码术上下文索引备选地可直接编码到编码指针中,而不是单独的上下文值中。一些实施例可以对于私有、单播、多播或广播密码术上下文ID类别中的每个中的一个或多个支持多个密码术上下文ID或密钥,以表示这些类别内的数据的单独版本。每个密码术上下文索引值都将映射到特定的密码术上下文ID或密钥。版本化密码术上下文索引可以直接编码到编码指针中。因此,每条指令都可以访问数据的任何版本。在该实施例中,编译器可以不静态地在上下文值中指定版本信息。

[0256] 另一实施例包括为每次存储器访问并入来自单个密码术上下文ID寄存器的密码术上下文ID值,这将不要求指定密码术上下文索引。但是,对存储器进行共享可能会导致频繁更新密码术上下文ID寄存器(即使在租户执行的单个会话中)。这可能由于租户在访问其自己的私有对象和与其它租户共享的对象之间进行切换而发生,因为密码术上下文ID寄存器将需要包含针对被访问的当前对象的对应密码术上下文ID。为避免需要如此多的密码术上下文ID寄存器更新,可以支持某些更多数量的寄存器。例如,除了针对私有对象的密码术上下文ID寄存器之外,还可以为在所有函数之间全局共享的对象定义附加的密码术上下文ID寄存器,其类似于网络广播。可以定义另外的密码术上下文ID寄存器或甚至表来持有针对要与一个或多个其它函数共享的对象的密码术上下文ID,其分别类似于网络单播或多播分组。

[0257] 图29示出了具有密码术上下文索引的备选增强的上下文信息2900,以及使用来自多个寄存器和/或存储器中的表的密码术上下文索引选择密码术上下文ID 2930。备选增强的上下文信息2900与扩展的上下文信息2800的不同之处在于,密码术上下文索引字段2910包含比扩展的上下文信息2800的密码术上下文ID字段2810中的密码术上下文ID(例如64

位)具有更少位数(例如,3位)的密码术上下文索引。为了维持用于备选增强的上下文信息2900的64位值,可以向下调节较高地址位字段2902的位数(例如,17个位)。在该示例中,其它字段(例如,存储器分配大小字段2904、类型字段2906和许可字段2908)中的位数没有改变。因此,对于采用EBA格式的指针,需要在指针的一个或多个部分中调节。例如,较高地址位字段2902被调节的位数可被用于增加EBA指针中的对齐位(例如,2125)的数量。备选地,可以使用任何其它合适的调整。

[0258] 在该实施例中,存储器中的多个寄存器和/或表存储了以密码方式的上下文ID。在至少一个实现中,寄存器可以是64位寄存器。例如,私有密码术上下文ID寄存器2912可以存储可由单个租户访问的以密码方式的上下文的私有以密码方式的上下文ID,广播密码术上下文ID寄存器2914可以存储可由所有租户访问的共享存储器区域的广播密码术上下文ID,以及单播和多播以密码方式的上下文ID表2916可以存储可在多租户环境中的所有租户的子集(例如,两个租户、三个租户等)之间共享的共享存储器区域的各种单播和多播密码术上下文ID(或其它定制以密码方式的上下文ID)。密码术上下文ID基于字段2910中的密码术上下文索引可以是可选择的。

[0259] 字段2910中的密码术上下文索引可以以任何合适的方式被供应给处理器102(例如,使用本文中描述的用于提供上下文信息700的任何方式)。在一些实施例中,字段2910中的密码术上下文索引可以在输入寄存器中供应,或者可以从存储器中的表中加载。在该示例中,密码术上下文索引被供应给多路复用器2925,以与私有密码术上下文ID寄存器2912的索引0 2920(A)、广播密码术上下文ID寄存器的索引1 2920(B)以及单播和多播密码术上下文ID表2916的索引2-7 2920(C)进行比较。选择了索引0-7的对应索引,并提供了其相关联的密码术上下文ID 2930,以用作用于相关以密码方式的指令(例如EncryptBaseAddr、SpecializePtr、JKEY、JHND、CALLCOMPARTMENT、WRKEY等)的全部或部分调整。

[0260] 如上所述,密码术上下文ID(例如,字段2810中的715、2115)可以由可信运行时或其它可信实体来生成。相同实体可以存储带有其适当句柄的以密码方式的上下文ID。例如,保密性句柄(例如,2600)可以包括用于保密性句柄被分配到的以密码方式的上下文(例如,存储器区域)(例如,租户A的私有存储器区域、共享存储器区域、租户B的私有存储器区域、广播存储器区域等)的密码术上下文ID。当在处理器102处的执行切换到特定租户时,该租户可以调用指令来利用用于该租户的密码术上下文ID加载寄存器2912和2914。当在处理器102处的执行切换到不同的租户时,新租户可以调用指令以利用用于该新租户的密码术上下文ID来覆写寄存器2912和2914的值。

[0261] 本文中公开多租户实施例的许多其它实施例或变型。例如,密码术上下文索引可以被映射到单独的密钥,而不是密码术上下文ID。在另一示例中,来自配置寄存器的其它状态可以被并入到上下文调整(例如,用于当前特权级别的标识符)中以防止其它特权级别访问相关联的数据。此外,可以为密码术上下文ID寄存器定义类似于本文中先前公开的指令(例如,RDKEY、WRKEY、JKEY、JHND、LANDKEY、CALLCOMPARTMENT)的指令。因此,指令将在密码术上下文ID值中操作,而不是在密钥值上操作。

[0262] 现在参考图30-34,示出了针对多租户环境中的操作的示例过程,其中使用以密码方式编码的指针来保密存储器地址、数据和/或代码。图30示出了与在多租户环境(例如,2500)中加载当前租户相关联的示例过程3000。过程3000的部分可以由计算设备的硬件、固

件和/或软件(例如,通过执行可信运行时)来执行。在一些情景中,具有计算设备(例如,100)的一个或多个核的处理器(例如,102)可以运行管理多个租户(例如,租户A 2520A、租户B 2520B)的可信运行时(例如,2530)。

[0263] 在3002,可信运行时可以启动加载租户以用于在单个地址空间中执行。在3004,可信运行时可以为租户生成(或请求生成)唯一的密钥。唯一的密钥可以包括地址密钥、数据密钥和代码密钥的任意组合,以用于分别加密/解密地址、数据和代码。此外,可信运行时可以标识或生成共享密钥(如果有的话),当前租户被授权使用这些密钥来访问共享存储器。此外,可信运行时还可以标识当前租户被授权访问(例如,分支到)的其它租户的任何保密性句柄。

[0264] 在3006,生成的密钥可以被扫描以用于嵌入的字节序列(例如,ENDBRANCH指令字节序列),其可以被诸如恶意租户的外部代码访问。如果密钥被存储在仅执行代码中,那么当处理器试图执行嵌入在密钥中的字节序列时,恶意代码可能会分支到字节序列并监测处理器动作。如果指令中的任何字节序列碰巧代表指令,那么恶意代码可以使用处理器动作来确定密钥的内容(或部分内容)。因此,在3008,如果在密钥中发现任何字节序列(诸如ENDBRANCH),则在3010,可信运行时生成新的唯一密钥,而没有ENDBRANCH指令(或可由外部代码访问的其它指令)。

[0265] 如本文中先前所述,在3012,可信运行时可以利用生成的密钥、租户的授权入口点和消息鉴别码生成保密性句柄。

[0266] 在3014,可信运行时可以生成指向用于存储当前租户的数据的私有存储器区域的以密码方式编码的数据指针(本文中也称为“私有数据指针”)。基于地址调整和在3004生成的地址密钥,可以使用以密码方式的算法(例如,块密码)来生成私有数据指针。私有数据指针可以根据任何合适的以密码方式的编码格式(诸如例如EBA格式)生成。备选地,私有数据指针可以根据包括私有存储器区域的地址的至少一部分的加密的任何其它格式来生成。

[0267] 在一个或多个实施例中,私有数据指针也可被用于加密私有存储器区域中的数据。基于数据调整和在3004生成的数据密钥,可以使用以密码方式的算法(例如,块密码)对数据加密。然而,在本文中进一步描述的其它实施例中,私有数据区域的不同部分被单独加密,以在私有数据区域内实行存储器安全性,这可以提供更细粒度的保密性。

[0268] 在3016,可信运行时可以生成指向用于存储当前租户的可执行代码的私有代码区域的以密码方式编码的代码指针(本文中也称为“私有代码指针”)。基于地址调整和在3004生成的地址密钥,可以使用以密码方式的算法(例如,块密码)来生成私有代码指针。私有代码指针可以根据任何合适的以密码方式的编码格式(诸如例如EBA格式)生成。备选地,私有代码指针可以根据包括私有代码区域的地址的至少一部分的加密的任何其它格式来生成。

[0269] 私有代码指针也可被用于加密私有代码区域中的可执行代码。可执行代码可以基于代码调整和在3004生成的代码密钥使用以密码方式的算法(例如,块密码)来加密。

[0270] 在3018,由可信运行时向当前租户提供了为当前租户创建的保密性句柄。在一个示例中,可信运行时可以将当前租户的保密性句柄拷贝到当前租户的私有存储器区域中的一个中。

[0271] 在3020,对于当前租户被授权访问的地址空间中的每个其它租户,由可信运行时向当前租户提供了用于(一个或多个)其它租户的(一个或多个)保密性句柄。在一个示例

中,可信运行时可以将(一个或多个)其它租户的(一个或多个)保密性句柄拷贝到当前租户的私有存储器区域中的一个中。此外,对于当前租户被授权访问的任何共享存储器区域,在3004标识(或生成)的(一个或多个)共享密钥被提供给当前租户。在一个示例中,可信运行时可以将共享密钥拷贝到当前租户的私有存储器区域。

[0272] 在3022,可信运行时完成了加载当前租户以用于执行。应当注意,为了简单起见,过程3000是参考密钥、私有存储器区域和私有代码区域的单个实例来描述的。然而,租户可以具有多个私有存储器区域和/或私有代码区域、以及用于执行线性地址、数据和/或代码的加密/解密的多个密钥。

[0273] 图31示出了与在多租户环境(例如,2500)中从第一租户分支到第二租户相关联的示例过程3100。过程3100的部分可以由计算设备的硬件、固件和/或软件(例如,通过执行可信运行时)来执行。在一些情景中,具有计算设备(例如,100)的一个或多个核的处理器(例如,102)可以运行管理多个租户(例如,租户A 2520A、租户B 2520B)的可信运行时(例如,2530)。

[0274] 在3104,第一租户请求对第二租户的访问。可以使用任何合适的机制来请求对另一租户的访问。在一个示例中,第一租户可以调用软件应用编程接口(API)来指示它希望调用第二租户。可以由可信运行时将该API导出到第一租户。

[0275] 在3106,可信运行时可以确定第一租户是否被授权访问第二租户。例如,如果第一租户向可信运行时提供了第二租户的有效保密性句柄,那么这可以指示第一租户被授权访问第二租户。在另一示例中,可信运行时检查指示用于租户之间访问的授权的表。例如,表可以指示被授权访问第二租户的所有租户。表还可以指示第一租户被授权访问的所有租户。

[0276] 如果做出了第一租户未被授权访问第二租户的确定,则在3108,访问可被可信运行时拒绝,并且可向第一租户发送响应。还可以发送消息来警告第二租户(或第二租户的用户)关于未授权的访问曾试图访问。在一些实现中,可能会生成故障,并且第一租户可能会被终止。

[0277] 如果做出了第一租户被授权访问第二租户的确定,则在3110,可信运行时可以调用指令(例如,WRKEY)来用第二租户的数据密钥、代码密钥和/或地址密钥更新密钥寄存器。地址密钥、数据密钥和代码密钥对于可信运行时是已知的,因为可信运行时在加载第二租户时创建了密钥(例如,3004)。此外,可信运行时还为第二租户生成指向私有存储器区域的以密码方式编码的指针(例如,3014)。此外,当加载第二租户时,可信运行时可能已经为第二租户生成了入口点指针。在一种情景中,当使用以密码方式编码的指针(例如,610、2110、2310)时,可以用第二租户的地址密钥更新地址密钥寄存器。代码密钥寄存器和数据密钥寄存器可以分别用第二租户的代码密钥和数据密钥来更新。在另一种情景中,如果使用未以密码方式编码的指针(例如,2410)或未加密的线性地址,则可能不需要地址密钥,并且只有数据密钥寄存器和/或代码密钥寄存器可以被更新。此外,应当注意,在一些实现中,数据和/或代码可以未以密码方式绑定到它们的指针或者以其它方式加密。在这些实现中,如果数据未以密码方式绑定到其指针,则在切换租户时可能不需要更新数据密钥寄存器。类似地,如果代码未以密码方式绑定到它的指针,那么在切换租户时代码密钥寄存器可能不需要更新。在一个实施例中,为每个被更新的密钥调用用于更新密钥寄存器的指令(例如,

WRKEY)。

[0278] 在3112,可信运行时可以用第二租户的授权入口点指针更新指令指针寄存器(例如,RIP),这有效地将流程转移给第二租户的授权入口点。在3114,基于在RIP中的授权入口点指针中编码的线性地址,执行被允许继续进行。应当注意,如果代码被加密,并且因此代码密钥寄存器用目的地代码密钥来更新,则必须使用目的地代码密钥来加密获取的下一指令。

[0279] 图32A示出了在多租户环境(例如,2500)中可以与将控制从一个租户直接转移给另一租户、从租户转移给可信运行时、或者从可信运行时转移给租户的示例指令(例如,JKEY)相关联的过程3200A。过程3200A在不使用保密性句柄的情况下执行。相反,在一个实施例中,用于目的地租户的代码密钥可以作为指令(例如,JKEY)的立即操作数被嵌入在代码中。代码本身可以被标记为“仅执行”,以保护嵌入的密钥。使用单独的代码和数据密钥的另一益处是,数据加载器不能正确地解密代码,并且数据存储也不能正确地加密代码,从而使得代码在页表中有效地仅执行,即使代码没有被标记为仅执行。更新页表标记可能会引入附加的开销。密钥太大而不能放在单个立即操作数中的备选实施例可以将密钥的不同部分嵌入到加载到寄存器中的仅执行代码中的多个立即操作数中,使得它们可以被组合到JKEY的一个或多个寄存器操作数中。过程3200A的部分可以由计算设备的硬件、固件和/或软件(例如,通过执行分支租户、被调用的租户和/或可信运行时)来执行。在一些情景中,具有计算设备(例如,100)的一个或多个核的处理器(例如,102)可以运行管理多个租户(例如,租户A 2520A、租户B 2520B)的可信运行时(例如,2530)。尽管与过程3200A相关联的指令可以用于在租户和可信运行时之间转移控制,但是为了简单起见,过程3200A是参照仅在第一(原始)租户和第二(目的地)租户之间转移控制来描述的。应理解,这些概念也适用于在租户和可信运行时之间转移控制。

[0280] 在至少一个实施例中,如果第一租户被授权访问第二租户,则用于第二租户的代码密钥和表示第二租户的授权入口点的目的地地址可以嵌入在第一租户的代码中,以使得第一租户能够在第二租户的授权入口点访问第二租户。例如,在分支/跳转指令(诸如JKEY)中,第二租户的代码密钥可以作为立即操作数被嵌入。寄存器操作数可以包含第二租户的目的地地址。目的地地址可以是授权入口点的未加密线性地址,或者是指向授权入口点的编码指针。编码指针可以是以密码方式编码的(例如610、2110、2310)或未以密码方式编码的(例如2410)。此外,如果目的地地址被实施为使用外部上下文信息(例如,610、2110、2310的一些变体)生成的以密码方式编码的指针,则外部上下文信息可以被嵌入为JKEY指令的立即操作数,或者经由存储器中的上下文表可以是可访问的。代码页可以标记为“仅执行”以保护代码密钥。

[0281] 在3202,第一租户调用指令(例如,JKEY)以在第二租户的授权的入口点将控制转移给第二租户。指令可以包括用于目的地地址的操作数和第二租户的代码密钥。在一些实施例中,指令还可以包括用于上下文信息的操作数。如果目的地地址是以密码方式编码的指针,那么它可以被解码成线性地址,如例如图21中所示的。在这种情景中,JKEY指令可以接受地址密钥来启用解码,其包括对加密地址片的解密。在进一步的实施例中,JKEY指令还可以接受数据密钥,使得地址密钥和数据密钥都不需要使用WRKEY被加载到目的地租户中。

[0282] 在3204,执行检查以确定目的地地址是否包含ENDBRANCH指令。在至少一个实施例

中，要求JKEY指令落在ENDBRANCH指令(或ENDBRANCH指令的变体，诸如ENDBR64等)上，以便限制一个租户(例如，第一租户)在调用另一租户(例如，第二租户)时可以使用的入口点。如果目的地地址不包含ENDBRANCH指令，则在3206，可生成故障。

[0283] 如果目的地地址确实包含ENDBRANCH指令，则在3208，执行另一检查以确定目的地地址是否与页边界(例如，4KB)对齐。这防止了JKEY指令落在不处于页的开始的ENDBRANCH指令上。如果ENDBRANCH指令没有与页边界对齐，则在3206可生成故障。

[0284] 以其它方式，如果ENDBRANCH指令与页边界对齐，则在3210，用第二租户的代码密钥来更新用于代码密钥的密钥寄存器。在至少一个实施例中，用作为JKEY指令的立即操作数而被嵌入的代码密钥来更新密钥寄存器。应注意的是，如果第二租户的代码没有被以密码方式绑定到其指针或以其它方式加密，则代码密钥寄存器可能不需要被更新。

[0285] 在3212，与指令指针寄存器(例如，RIP)相关联的用于上下文信息的专用寄存器可以取决于目的地地址的编码(或缺少编码)来更新。如果目的地地址是至少部分基于外部上下文信息(即，没有在指针中编码的上下文信息)被以密码方式编码的指针，则在3212，可以用表示针对指针的外部上下文信息的上下文值来更新专用寄存器。表示外部上下文信息的上下文值可以作为立即操作数被嵌入在指令中。备选地，上下文值可以包含在存储器中的指针上下文表中。在一个或多个实施例中，可以创建指针上下文表，使得它由多租户环境中共享指针的所有租户可访问。然而，在其它实施例中，可以为不同的租户创建单独的指针上下文表。如果用在指针中而不在指针外部编码的上下文信息来以密码方式编码授权的入口点指针(例如，2310的一些实施例)，或者如果授权的入口点指针是未以密码方式编码的(例如，2410)，则在3212，可以不用上下文信息来更新专用寄存器。类似地，如果授权的入口点指针是未加密的(明文)线性地址，则在3212，可能没有用外部上下文信息来更新专用寄存器。

[0286] 在3214，用目的地地址来更新指令指针寄存器(例如，RIP)。在至少一个实施例中，用JKEY指令的寄存器操作数中的目的地地址来更新指令指针寄存器。如果目的地地址是以密码方式编码的指针，那么它可以被解码成线性地址，如例如图21或23中所示的。控制然后被有效地从第一租户转移给第二租户。

[0287] 在3216，一旦控制已经转移给第二租户，则第二租户调用指令(例如，WRKEY)以利用第二租户的数据密钥和/或地址密钥更新数据和/或地址密钥寄存器。在一种情景中，当使用了以密码方式编码的指针(例如，610、2110、2310)时，地址密钥寄存器和数据密钥寄存器两者可以分别用第二租户的地址密钥和数据密钥来更新。在另一种情景中，如果使用了未以密码方式编码的指针(例如，2410)或未加密的线性地址，则可能不需要地址密钥，并且仅可更新数据密钥寄存器。此外，应注意的是，在一些实现中，数据可能没有被以密码方式绑定到其指针或者以其它方式被加密。在这些实现中，当切换租户时，可能不需要更新数据密钥寄存器。在至少一个实施例中，为每个被更新的密钥执行用于更新密钥寄存器的指令(例如，WRKEY)。在过程3200A完成时，为授权的租户执行第二租户的可访问代码所需的所有密钥(例如，代码、数据和/或地址密钥)以及可能的外部上下文信息被更新，以实现可访问代码的成功执行。

[0288] 应当注意，在另一实施例中，可以定义新的指令并使用其来代替ENDBRANCH，以作为包含在JKEY指令中的目的地地址的存储器位置中的所需指令。例如，这个新指令可以被

命名为LANDKEY,如本文中先前所述。

[0289] 图32B示出了在多租户环境(例如,2500)中可以与将控制从一个租户直接转移给另一租户、从租户转移给可信运行时、或者从可信运行时转移给租户的示例指令(例如,JHND)相关联的过程3200B。过程3200B通过使用目的地租户的保密性句柄来执行。存储在保密性句柄中的密钥被包装(即加密)以用于保护,并由指令加载以将控制转移给目的地租户。过程3200B的部分可以由计算设备的硬件、固件和/或软件(例如,通过执行分支租户和/或可信运行时)来执行。在一些情景中,具有计算设备(例如,100)的一个或多个核的处理器(例如,102)可以运行管理多个租户(例如,租户A 2520A、租户B 2520B)的可信运行时(例如,2530)。尽管与过程3200B相关联的指令可以用于在租户和可信运行时之间转移控制,但是为了简单起见,过程3200B是参照仅在第一(原始)租户和第二(目的地)租户之间转移控制来描述的。应理解,这些概念也适用于在租户和可信运行时之间转移控制。

[0290] 在至少一个实施例中,如果第一租户被授权访问第二租户,则包含用于第二(目的地)租户的密钥和指针的保密性句柄(例如,2600)可以被存储在第一(原始)租户的私有存储器区域中。保密性句柄可以包括用于解密和加密第二租户的代码、第二租户的数据和/或指向第二租户的存储器区域的以密码方式编码的指针中的地址的密钥。第二租户的存储器区域可以是私有的或者是与被授权访问第二租户的租户共享的。保密性句柄还可以包括授权的入口点指针,其可以被实施为以密码方式编码的指针(例如610、2110、2310)、未以密码方式编码的指针(例如2410)或未加密的线性地址。在一个实现中,第二租户的保密性句柄可以在加载第一租户以用于执行期间(或者稍后,如果第二租户指示稍后第一租户被授权访问第二租户的话)由可信运行时(例如,2530)提供给第一租户。在一个示例实现中,在确定第一租户被授权访问第二租户时,可信运行时可以将第二租户的保密性句柄存储在第一租户的私有存储器区域中。

[0291] 在3222,第一租户调用指令(例如,JHND)以在第二租户的授权的入口点将控制转移给第二租户。该指令可以包括用于保密性句柄的存储器操作数。在一些实施例中,指令还可以包括用于上下文信息的操作数。

[0292] 在3224,分别用第二租户的地址密钥、数据密钥和/或代码密钥来更新用于地址密钥、数据密钥和/或代码密钥的密钥寄存器。在一种情景中,当使用了以密码方式编码的指针(例如,610、2110、2310)时,用第二租户的地址密钥来更新地址密钥寄存器。在另一种情景中,如果使用了未以密码方式编码的指针(例如,2410)或未加密的线性地址,则在切换租户时可能不需要地址密钥。如果第二租户的数据被以密码方式绑定到其指针或以其它方式加密,则数据密钥寄存器可以用第二租户的数据密钥来更新。然而,如果第二租户的数据没有以密码方式绑定到其指针或以其它方式加密,则数据密钥寄存器可能不需要被更新。类似地,如果第二租户的代码被以密码方式绑定到其指针或以其它方式加密,则代码密钥寄存器用第二租户的代码密钥来更新。但是,如果第二租户的代码没有被以密码方式绑定到其指针或以其它方式加密,那么代码密钥寄存器可能不需要被更新。可以从存储在存储器中的保密性句柄中检索切换到第二租户所需的密钥。可以使用指令中的存储器操作数来访问保密性句柄。地址、数据和代码密钥可以被包装在保密性句柄中,因此,在密钥被加载到适当的密钥寄存器中之前,可以执行适当的解包装(或解密)。

[0293] 在3226,可以取决于存储在保密性句柄中的授权入口点指针的编码(或缺少编码)

来更新与指令指针寄存器(例如,RIP)相关联的用于上下文信息的专用寄存器。如果至少部分基于外部上下文信息(即,没有在指针中编码的上下文信息)以密码方式编码授权入口点指针,则在3226,专用寄存器可以用表示针对指针的外部上下文信息的上下文值来更新。表示外部上下文信息的上下文值可以作为立即操作数被嵌入在指令中。备选地,上下文值可以包含在存储器中的指针上下文表中。在一个或多个实施例中,可以创建指针上下文表,使得它被多租户环境中共享指针的所有租户可访问。然而,在其它实施例中,可以为不同的租户创建单独的指针上下文表。如果授权的入口点指针是用在指针中而不在指针外部编码的上下文信息来以密码方式编码的(例如,2310的一些实施例),或者如果授权的入口点指针是未以密码方式编码的(例如,2410),则在3226,专用寄存器可以不用上下文信息来更新。类似地,如果授权的入口点指针是未加密的(明文)线性地址,则在3226,专用寄存器可能没有用外部上下文信息来更新。

[0294] 在3228,用授权的入口点指针更新指令指针寄存器(例如,RIP)。在至少一个实施例中,用从第二租户的保密性句柄中检索的授权入口点指针(其存储在第一租户的私有存储器区域中)更新指令指针寄存器。授权的入口点指针可能没有被包装。然而,如果授权的入口点指针被包装,则可以在更新指令指针寄存器之前执行适当的解包装(例如,解密)。如果目的地地址是以密码方式编码的指针,那么它可以被解码成线性地址,例如如图21或23中所示。控制然后被有效地从第一租户转移给第二租户。在过程3200B的完成时,为授权的租户执行第二租户的可访问代码所需的所有密钥(例如,代码、数据和/或地址密钥)以及可能的外部上下文信息被更新,以实现可访问代码的成功执行。

[0295] 这些实施例中的许多提供了相对于过去强制租户之间隔离并许可租户之间受控制的切换的方法的益处。对于本文中公开的实施例,不一定要定义新的模式、特权级别或具有由可信运行时占用的特殊特权的代码段来执行生成保密性句柄或更新密钥的指令。在过去的方法中,这种规定的必要性增加了系统复杂性和开销。在许多实施例中,通过恶意执行指令以生成未授权的保密性句柄或执行未授权的密钥更新,租户难以猜测将许可对另一租户进行未授权访问的密钥值。因此,本文中公开的可信运行时由于其对每个租户的正确密钥的了解,能够唯一地实行隔离并对租户之间的控制转移施加控制。

[0296] 图33示出了在多租户环境(例如,2500)中可以与将控制从一个租户直接转移给另一租户、从租户转移给可信运行时、或者从可信运行时转移给租户的示例指令(例如,CALLCOMPARTMENT)相关联的过程3300。过程3300的部分可以由计算设备的硬件、固件和/或软件(例如,通过执行分支租户、被调用的租户和/或可信运行时)来执行。在一些情景中,具有计算设备(例如,100)的一个或多个核的处理器(例如,102)可以运行管理多个租户(例如,租户A 2520A、租户B 2520B)的可信运行时(例如,2530)。尽管与过程3300相关联的指令可以用于在租户和可信运行时之间转移控制,但是为了简单起见,过程3300是参照仅在第一租户和第二租户之间转移控制来描述的。应理解,这些概念也适用于在租户和可信运行时之间转移控制。

[0297] 过程3300使用新的调用门来实现将控制从第一租户转移给第二租户,并且用由第二租户所需的数据、代码和地址密钥来更新密钥寄存器。在3304,第一租户调用指令(例如,CALLCOMPARTMENT)以在授权的入口点将控制转移给第二租户。指令指定了描述符表中的索引。描述符表是包含被索引的描述符的存储器中的表。对应于特定租户的描述符可以包括

指向特定租户的授权的入口点指针和特定租户的密钥值(例如,地址密钥、数据密钥、代码密钥)。

[0298] 在3006,对于CALLCOMPARTMENT指令中指定的索引来搜索表。如果没有找到索引,则在3308可生成故障。如果找到索引,则在3310,对应于指定索引的描述符被标识。

[0299] 在3312,用第二租户的密钥值(例如,地址密钥、数据密钥、代码密钥)加载密钥寄存器,所述密钥值是从所标识的描述符中获得的。在3314,用来自所标识的描述符的授权入口点指针(例如,以密码方式编码的指针)来更新寄存器指令指针(RIP)。

[0300] 在3316,RIP可被用于在第二租户的授权入口点获取并执行指令。存储在RIP中的以密码方式编码的指针可以基于第二租户的更新地址密钥进行解码。要在调整中使用的上下文信息可以被存储在用于获取操作的专用寄存器中。一旦获得了用于授权入口点的线性地址,就可以基于第二租户的更新的代码密钥来获取和解密在该线性地址处的代码。要在调整中使用的上下文信息可以被存储在用于获取操作的专用寄存器中。一旦代码被解密,在其授权入口点处的第二租户的执行就可以继续进行。

[0301] 图34示出了与在多租户环境(例如,2500)中选择密码术上下文ID相关联的示例过程3400,其中针对不同的以密码方式的上下文来切换密码术上下文ID。过程3400的部分可以由计算设备的硬件、固件和/或软件(例如,通过执行可信运行时或租户)来执行。在一些情景中,具有计算设备(例如,100)的一个或多个核的处理器(例如,102)可以运行管理多个租户(例如,租户A 2520A、租户B 2520B)的可信运行时(例如,2520)。过程3400或其变体可在指令的执行期间被执行,其中用于特定存储器区域的密码术上下文ID被用于形成调整的全部或部分,所述调整用于解密/加密数据、代码或指向数据或代码的指针中的线性地址的至少一部分。

[0302] 在3402,从上下文信息中获得密码术上下文索引。对于利用以密码方式编码的指针来访问数据的一些软件指令,密码术上下文索引可以借助于包含其它上下文信息的操作数来静态获得,所述操作数也可以用作调整的至少一部分。对于利用以密码方式编码的指针来访问数据的其它软件指令,密码术上下文索引可以例如从存储器中的表(例如,指针上下文表121)中动态地获得。对于利用以密码方式编码的指针来访问代码的处理器指令(例如,获取),密码术上下文索引可以从与寄存器指令指针相关联的专用寄存器中获得。

[0303] 在3404,将密码术上下文索引与多租户环境中用于以密码方式的上下文(或存储器区域)的密码术上下文ID的索引进行比较。

[0304] 在3406,做出确定,所述确定关于密码术上下文索引是否匹配(或以其它方式对应于)可以被存储在寄存器中的用于私有密码术上下文ID(例如,用于单个租户的私有存储器区域)的索引。如果密码术上下文索引与用于私有密码术上下文ID的索引匹配,则在3408,选择用于私有存储器区域的私有密码术上下文ID。在3418,取决于特定指令或处理器流水线,用于私有存储器区域的所选择密码术上下文ID被用于解密/加密以密码方式编码的指针中的代码、数据或地址的加密片。

[0305] 如果在3406确定了密码术上下文索引不匹配(或以其它方式不对应于)用于私有密码术上下文ID的索引,则在3410做出确定,所述确定关于密码术上下文索引是否匹配(或以其它方式对应于)可以被存储在寄存器中的用于广播密码术上下文ID的索引(例如,用于所有租户的广播存储器区域)。如果密码术上下文索引与用于广播密码术上下文ID的索引

匹配，则在3412，选择用于广播存储器区域的广播密码术上下文ID。在3418，取决于特定指令或处理器流水线，用于广播存储器区域的所选择密码术上下文ID被用于解密/加密以密码方式编码的指针中的代码、数据和/或地址的加密片。

[0306] 如果在3410确定了密码术上下文索引不匹配(或以其它方式不对应于)用于广播密码术上下文ID的索引，则在3414做出确定，所述确定关于密码术上下文索引是否匹配(或以其它方式对应于)用于单播或多播密码术上下文ID的索引(例如，用于租户的子集的共享存储器区域)，在一个示例中，所述用于单播或多播密码术上下文ID的索引可以被存储在存储器中的表中。如果密码术上下文索引与用于单播或多播密码术上下文ID的索引匹配，则在3416，选择用于共享存储器区域的单播或多播密码术上下文ID。在3418，取决于特定指令或处理器流水线，用于共享存储器区域的所选择密码术上下文ID被用于解密/加密以密码方式编码的指针中的代码、数据和/或地址的加密片。

[0307] 如果在3414确定了密码术上下文索引与用于单播或多播密码术上下文ID的索引不匹配(或以其它方式不对应)，则没有索引与密码术上下文索引匹配，并且在3420生成故障。然而，如果密码术上下文索引在3406、3410或3414匹配，则在3418将所选择密码术上下文ID用于适当的加密/解密。

[0308] 隐藏的内嵌元数据

隐藏的内嵌元数据(HIM)可以与采用EBA格式的以密码方式编码的指针相组合，以消除仅为检索大小和类型信息而对集中式上下文表执行动态查找的需要。然而，如果对象被存储在由存储在STATIC_REGION_BASE寄存器中的默认较高地址位所表示的区域之外，那么可能仍然需要查找集中式上下文表。

[0309] 通过要求对象与缓存线边界对齐并在缓存线内分配足够的存储来持有大小和类型信息，可以为每次访问加载大小和类型上下文，而不需要任何附加的高速缓存线加载。在这种方法中，大小和类型将被直接检查，而不是作为用于加密基地址片的调整被并入。仅许可位仍将作为地址调整被并入。因此，具有不同许可的指针可以引用相同的底层存储，因为它们不是作为数据调整被并入的。即使静态已知，大小和类型信息也可不再作为静态上下文的一部分由软件供应，因为大小和类型信息仍然将需要与隐藏的内嵌元数据进行比较，以确保它们匹配并且没有被软件伪造。

[0310] 许可位也可以存储为隐藏的内嵌元数据，而不是并入到用于加密基地址片的调整中。然而，非常常见的是，程序的某些部分使用指向数据的读写指针，而程序的其它部分使用指向该相同数据的只读指针，因此将该相同数据作为与对象存储相关联的隐藏内嵌元数据来进行嵌入可能导致程序源代码中表达的保密性策略与由硬件实行的策略之间的不匹配。

[0311] 作为隐藏内嵌元数据存储的大小应当是相对于当前高速缓存行的，以避免需要查找对象的开始(这可能引入附加的开销)。一种可能的压缩编码是以高速缓存行的增量来指示对象超出当前高速缓存行的量。备选地，可以用更细粒度的单位来指定长度，以支持更精确的边界检查，例如对象超出当前高速缓存行的开始处的字节的数量。

[0312] 加密基地址片可以绑定到对象存储，以使用隐藏的内嵌元数据来减轻悬摆和伪造的指针弱点。特别地，将数据加密绑定到指针的一部分(诸如标签或加密基地址片)的备选方法是将指针的边界部分的拷贝存储为隐藏的内嵌元数据。然而，这可能无法像加密一样

有效地减轻物理攻击。

[0313] 为了利用空间高效的HIM存储提供细粒度的边界检查,EBA片的旋转子片可以被存储为隐藏的内嵌元数据。例如,为了支持每16字节数据块具有8位的隐藏内嵌元数据的16字节边界粒度,可以基于指针的位4和5来选择8位的EBA片作为隐藏的内嵌元数据来存储。这可以表示为公式的集,其用于为数据的给定组块选择要存储的隐藏内嵌元数据:

EBA=指针[56:25]

slice_idx=指针[5:4]

HIM元数据= EBA[((slice_idx+1)*8-1):(slice_idx*8)]。

[0314] 指针中的标签片可以作为上下文被并入到用于加密和解密基地址片的地址调整中。

[0315] 示例架构

图35是示出根据至少一个实施例的示例以密码方式的计算环境3500的框图。在所示的示例中,以密码方式的寻址层3510延伸跨过示例计算向量中央处理单元(CPU)3502、图形处理单元(GPU)3504、人工智能(AI)3506和现场可编程门阵列(FPGA)3508。例如,对于存储在存储器3512中的数据,CPU 3502和GPU 3504可以共享相同的虚拟地址转译并且密码操作地址可以建立在这个共享的虚拟存储器上。对于给定的执行流程,它们可以共享相同的过程密钥,并遵循相同的以密码方式的算法计算相同的调整来解密以密码方式编码的地址和解密由这些编码的地址所引用的数据。

[0316] 结合起来,本文中描述的能力可以实现以密码方式的计算。存储器3512可以在存储器层级的每一级(从高速缓存的第一级到高速缓存的最后一级并进入到系统存储器中)被加密。将密码操作地址编码绑定到数据加密可能会允许极细粒度的对象边界和访问控制,实现细粒度的保密容器一直向下到甚至到单个函数及其对象,以实现函数即服务。对调用栈上的返回地址(取决于它们的位置)进行以密码方式编码也可以实现控制流程完整性,而不需要影子栈元数据。因此,数据访问控制策略和控制流程中的任一个都可以简单地取决于以密码方式的寻址和相应的密码操作数据绑定而被以密码方式执行。

[0317] 图36-38是可根据本文公开的实施例使用的示例性计算机架构的框图。通常,可以使用处理器和计算系统领域中已知的任何计算机架构设计。在示例中,现有技术中已知的用于膝上型电脑、台式电脑、手持PC、个人数字助理、平板电脑、工程工作站、服务器、网络设备、服务器、仪器、网络集线器、路由器、交换机、嵌入式处理器、数字信号处理器(DSP)、图形设备、视频游戏设备、机顶盒、微控制器、智能电话、移动设备、可穿戴电子设备、便携式媒体播放器、手持设备和各种其它电子设备的系统设计和配置也适用于本文中所述的计算系统的实施例。一般而言,用于本文公开的实施例的合适的计算机架构可以包括但不限于图36-38中所示的配置。

[0318] 图36是根据实施例的处理器的示例说明。处理器3600是硬件设备的类型的示例,其可以结合本文中示出和描述的实现(例如,处理器102)来使用。处理器3600可以是任何类型的处理器,诸如微处理器、嵌入式处理器、数字信号处理器(DSP)、网络处理器、多核处理器、单核处理器或其它执行代码的设备。尽管图36中仅示出了一个处理器3600,但是处理元件可以备选地包括多于一个图36中示出的处理器3600。处理器3600可以是单线程核,或者对于至少一个实施例,处理器3600可以是多线程的,因为它可以包括每核多于一个的硬件

线程上下文(或“逻辑处理器”)。

[0319] 图36还示出了根据实施例的耦合到处理器3600的存储器3602。存储器3602可以是本领域技术人员已知的或可用的各种各样的存储器(包括存储器层次的各种层)中的任何一种。这种存储器元件可以包括但不限于随机存取存储器(RAM)、只读存储器(ROM)、现场可编程门阵列(FPGA)的逻辑块、可擦除可编程只读存储器(EPROM)和电可擦除可编程只读存储器(EEPROM)。

[0320] 处理器3600可以执行与本文中详细描述的算法、过程或操作相关联的任何类型的指令。通常,处理器3600可以将元素或物品(例如,数据)从一种状态或事物变换到另一种状态或事物。

[0321] 可以是要由处理器3600执行的一个或多个指令的代码3604可以被存储在存储器3602中,或者可以被存储在软件、硬件、固件或其任何合适的组合中,或者在适当的情况下并基于特定需求被存储在任何其它内部或外部组件、设备、元件或对象中。在一个示例中,处理器3600可以遵循由代码3604指示的指令的程序序列。每个指令进入前端逻辑3606,并由一个或多个解码器3608处理。解码器可以生成微操作作为其输出,所述微操作诸如采用预定义格式的固定宽度微操作,或者可以生成反映原始代码指令的其它指令、微指令或控制信号。前端逻辑3606还包括寄存器重命名逻辑3610和调度逻辑3612,它们通常分配资源并将对应于指令的操作排队以用于执行。

[0322] 处理器3600还可以包括具有执行单元3616a、3616b、3616n等的集的执行逻辑3614。一些实施例可以包括专用于特定函数或函数集的多个执行单元。其它实施例可以仅包括一个执行单元或者可以执行特定功能的一个执行单元。执行逻辑3614执行由代码指令指定的操作。

[0323] 在完成由代码指令指定的操作的执行之后,后端逻辑3618可以撤回代码3604的指令。在一个实施例中,处理器3600允许无序执行,但要求有序的指令退出。退出逻辑3620可以采取各种已知的形式(例如,重新排序缓冲器等)。以这种方式,处理器3600在代码3604的执行期间至少在由解码器生成的输出、由寄存器重命名逻辑3610使用的硬件寄存器和表以及由执行逻辑3614修改的任何寄存器(未示出)方面被变换。

[0324] 尽管图36中未示出,但是处理元件可以包括具有处理器3600的芯片上的其它元件。例如,处理元件可以包括连同处理器3600的存储器控制逻辑。处理元件可以包括I/O控制逻辑和/或可以包括与存储器控制逻辑集成的I/O控制逻辑。处理元件还可以包括一个或多个高速缓存。在一些实施例中,非易失性存储器(诸如闪存或熔丝)也可以包括在具有处理器3600的芯片上。

[0325] 图37A是示出根据本公开的一个或多个实施例的示例性有序流水线和示例性寄存器重命名、无序发布/执行流水线两者的框图。图37B是示出根据本公开的一个或多个实施例的要在处理器中包括的有序架构核和示例性寄存器重命名、无序发布/执行架构核两者的示例性实施例的框图。图37A-37B中的实线框示出了有序流水线和有序核,而虚线框的可选添加示出了寄存器重命名、无序发布/执行流水线和核。给定有序方面是无序方面的子集,将描述无序方面。

[0326] 在图37A中,处理器流水线3700包括获取阶段3702、长度解码阶段3704、解码阶段3706、分配阶段3708、重命名阶段3710、调度(也称为分派或发布)阶段3712、寄存器读/存储

器读阶段3714、执行阶段3716、写回/存储器写阶段3718、异常处置阶段3722和提交阶段3724。

[0327] 图37B示出了处理器核3790，其包括耦合到执行引擎单元3750的前端单元3730，并且两者都耦合到存储器单元3770。处理器核3790和存储器单元3770是可结合本文中所示和所述的实现(例如，处理器102、存储器120)一起使用的硬件的类型的示例。核3790可以是精简指令集计算(RISC)核、复杂指令集计算(CISC)核、超长指令字(VLIW)核、或者混合或备选核类型。作为仍有的另一选项，核3790可以是专用核，诸如例如网络或通信核、压缩引擎、协处理器核、通用计算图形处理单元(GPGPU)核、图形核等。此外，处理器核3790及其组件表示可被用于实现逻辑处理器及其相应组件的示例架构。

[0328] 前端单元3730包括耦合到指令高速缓存单元3734的分支预测单元3732，所述指令高速缓存单元3734耦合到指令转译后备缓冲器(TLB)单元3736，所述指令转译后备缓冲器单元3736耦合到指令获取单元3738，所述指令获取单元3738耦合到解码单元3740。解码单元3740(或解码器)可以解码指令，并生成一个或多个微操作、微代码入口点、微指令、其它指令或其它控制信号作为输出，所述微操作、微代码入口点、微指令、其它指令或其它控制信号是从原始指令中解码的、或者以其它方式反映原始指令的、或从原始指令中导出的。解码单元3740可以使用各种不同的机制来实现。合适机制的示例包括但不限于查找表、硬件实现、可编程逻辑阵列(PLA)、微代码只读存储器(ROM)等。在一个实施例中，核3790包括微代码ROM或存储(例如，在解码单元3740中或以其它方式在前端单元3730内)用于特定宏指令的微代码的其它介质。解码单元3740耦合到执行引擎单元3750中的重命名/分配器单元3752。

[0329] 执行引擎单元3750包括耦合到退出单元3754和一个或多个调度器单元3756的集的重命名/分配器单元3752。(一个或多个)调度器单元3756表示任何数量的不同调度器，其包括预约站、中央指令窗口等。(一个或多个)调度器单元3756耦合到(一个或多个)物理寄存器堆单元3758。(一个或多个)物理寄存器堆单元3758中的每个表示一个或多个物理寄存器堆，其中不同的物理寄存器堆存储一个或多个不同的数据类型，诸如标量整数、标量浮点、打包整数、打包浮点、向量整数、向量浮点、状态(例如，作为要执行的下一个指令的地址的指令指针)等。在一个实施例中，(一个或多个)物理寄存器堆单元3758包括向量寄存器单元、写掩码寄存器单元和标量寄存器单元。这些寄存器单元可以提供架构向量寄存器、向量掩码寄存器和通用寄存器(GPR)。在本文中描述的至少一些实施例中，寄存器单元3758是可以结合本文中示出和描述的实现使用的硬件类型(例如，寄存器112)的示例。(一个或多个)物理寄存器堆单元3758与退出单元3754重叠，以示出可以实现寄存器重命名和无序执行的各种方式(例如，使用(一个或多个)重新排序缓冲器和(一个或多个)退出寄存器堆)；使用(一个或多个)未来文件、(一个或多个)历史缓冲器和(一个或多个)退出寄存器堆；使用寄存器映射和寄存器的池；等等。退出单元3754和(一个或多个)物理寄存器堆单元3758耦合到(一个或多个)执行集群3760。(一个或多个)执行集群3760包括一个或多个执行单元3762的集和一个或多个存储器访问单元3764的集。执行单元3762可以对各种类型的数据(例如，标量浮点、打包整数、打包浮点、向量整数、向量浮点)执行各种操作(例如，移位、加法、减法、乘法)。虽然一些实施例可以包括专用于特定函数或函数集的多个执行单元，但是其它实施例可以包括仅一个执行单元或多个执行单元，其都执行所有函数。执行单元3762还可

以包括地址生成单元(例如,822),以计算由核用来访问主存储器(例如,存储器单元3770)和页未命中处理器(PMH)(例如,826)的地址。

[0330] (一个或多个)调度器单元3756、(一个或多个)物理寄存器堆单元3758和(一个或多个)执行集群3760被示为可能是复数个,因为某些实施例为某些类型的数据/操作创建了单独的流水线(例如,标量整数流水线、标量浮点/打包整数/打包浮点/向量整数/向量浮点流水线、和/或存储器访问流水线,其每个都具有它们自己的调度器单元、(一个或多个)物理寄存器堆单元和/或执行集群,并且在单独的存储器访问流水线的情况下实现了某些实施例,其中仅有该流水线的执行集群具有(一个或多个)存储器访问单元3764)。还应当理解,在使用分开的流水线的情况下,这些流水线中的一个或多个可能是无序发布/执行的,而其余的是有序的。

[0331] 存储器访问单元3764的集耦合到存储器单元3770,所述存储器单元3770包括耦合到数据高速缓存单元3774的数据TLB单元3772,所述数据高速缓存单元3774耦合到2级(L2)高速缓存单元3776。在一个示例性实施例中,存储器访问单元3764可以包括加载单元、存储地址单元和存储数据单元,其每个都耦合到存储器单元3770中的数据TLB单元3772。指令高速缓存单元3734进一步耦合到存储器单元3770中的2级(L2)高速缓存单元3776。L2高速缓存单元3776耦合到一个或多个其它级别的高速缓存,并最终耦合到主存储器。此外,如果在数据TLB单元3772中没有找到匹配,则页未命中处置器(例如,页未命中处置器826)也可以被包括在核3790中,以在页表中查找地址映射。

[0332] 以示例的方式,示例性寄存器重命名、无序发布/执行核架构可如下实现流水线3700:1)指令获取3738执行获取和长度解码阶段3702和3704;2)解码单元3740执行解码阶段3706;3)重命名/分配器单元3752执行分配阶段3708和重命名阶段3710;4)(一个或多个)调度器单元3756执行调度阶段3712;5)(一个或多个)物理寄存器堆单元3758和存储器单元3770执行寄存器读/存储器读阶段3714;执行集群3760执行执行阶段3716;6)存储器单元3770和(一个或多个)物理寄存器堆单元3758执行写回/存储器写阶段3718;7)异常处置阶段3722中可能涉及各种单元;以及8)退出单元3754和(一个或多个)物理寄存器堆单元3758执行提交阶段3724。

[0333] 核3790可以支持一个或多个指令集(例如,x86指令集(具有一些已经添加有较新版本的扩展);加州桑尼维尔市的MIPS技术公司的MIPS指令集;加州桑尼维尔市的ARM控股公司的ARM指令集(带有可选的附加扩展,诸如NEON)),所述一个或多个指令集包括本文中所述的(一个或多个)指令。在一个实施例中,核3790包括支持打包数据指令集扩展(例如,AVX1、AVX2)的逻辑,从而允许由许多多媒体应用所使用的操作被使用打包数据来执行。

[0334] 应当理解,核可以支持多线程(执行操作或线程的两个或更多个并行集),并且可以以各种方式来支持多线程,所述方式包括时间分片多线程、同时多线程(其中单个物理核为线程中的每个提供逻辑核,物理核是同时多线程的),或者其组合(例如,时间分片获取和解码以及此后的同时多线程,诸如在英特尔®超线程技术中)。因此,在至少一些实施例中,可以支持多线程包围区。

[0335] 虽然寄存器重命名是在无序执行的上下文中描述的,但是应理解,寄存器重命名可以在有序架构中使用。虽然处理器的示出的实施例还包括单独的指令和数据高速缓存单元3734/3774以及共享的L2高速缓存单元3776,但是备选实施例可以具有用于指令和数据

两者的单个内部高速缓存，例如1级(L1)内部高速缓存或者多级内部高速缓存。在一些实施例中，系统可以包括内部高速缓存和在核和/或处理器外部的外部高速缓存的组合。备选地，所有高速缓存可以在核和/或处理器的外部。

[0336] 图38示出了根据实施例的采用点对点(PtP)配置布置的计算系统3800。特别地，图38示出了其中处理器、存储器和输入/输出设备通过多个点对点接口进行互连的系统。通常，本文中描述的计算系统或计算设备(例如，计算设备100)的一个或多个可以以与计算系统3800相同或类似的方式配置。

[0337] 处理器3870和3880可以被实现为单核处理器3874a和3884a或者多核处理器3874a-3874b和3884a-3884b。处理器3870和3880可以每个包括由它们相应的一个或多个核使用的高速缓存3871和3881。共享的高速缓存(未示出)可以被包括在任意处理器中或两个处理器的外部，但是经由P-P互连与处理器连接，使得如果处理器被置于低功率模式，则任一个或两个处理器的本地高速缓存信息可以被存储在共享的高速缓存中。应当注意，本文中描述的一个或多个实施例可以在诸如计算系统3800的计算系统中实现。此外，处理器3870和3880是可以结合本文中示出和描述的实现使用的硬件类型(例如，处理器102)的示例。

[0338] 处理器3870和3880还可以每个包括集成存储器控制器逻辑(MC)3872和3882，以与存储器元件3832和3834通信，所述存储器元件3832和3834可以是本地附接到相应处理器的主存储器的部分。在备选实施例中，存储器控制器逻辑3872和3882可以是与处理器3870和3880分离的离散逻辑。存储器元件3832和/或3834可以存储要由处理器3870和3880在实现本文中概述的操作和功能时使用的各种数据。

[0339] 处理器3870和3880可以是任何类型的处理器，诸如结合其它附图讨论的那些处理器。处理器3870和3880可以分别使用点对点接口电路3878和3888经由点对点(PtP)接口3850交换数据。处理器3870和3880可以每个使用点对点接口电路3876、3886、3894和3898经由个别点对点接口3852和3854与输入/输出(I/O)子系统3890交换数据。I/O子系统3890也可以使用接口电路3892经由高性能图形接口3839与高性能图形电路3838交换数据，所述接口电路3892可以是PtP接口电路。在一个实施例中，高性能图形电路3838是专用处理器，诸如例如高吞吐量MIC处理器、网络或通信处理器、压缩引擎、图形处理器、GPGPU、嵌入式处理器等。I/O子系统3890还可以与显示器3833通信，以用于显示由人类用户可查看的数据。在备选实施例中，图38中所示的任何或所有PtP链接可以被实现为多点总线(multi-drop bus)，而不是PtP链接。

[0340] I/O子系统3890可以经由接口电路3896与总线3810通信。总线3810可以具有通过它进行通信的一个或多个设备，诸如总线桥3818、I/O设备3814以及一个或多个其它处理器3815。经由总线3820，总线桥3818可以与其它设备通信，所述设备诸如用户接口3822(诸如键盘、鼠标、触摸屏或其它输入设备)、通信设备3826(诸如调制解调器、网络接口设备或可以通过计算机网络3860进行通信的其它类型的通信设备)、音频I/O设备3824和/或数据存储设备3828。数据存储设备3828可以存储代码和数据3830，其可以由处理器3870和/或3880执行。在备选实施例中，总线架构的任何部分可以用一个或多个PtP链接来实现。

[0341] 诸如代码3830的程序代码可以应用于输入指令，以执行本文中描述的功能并生成输出信息。输出信息可以以已知的方式应用于一个或多个输出设备。出于本申请的目的，处

理系统可以是计算系统3800的一部分，并且包括具有处理器的任何系统，所述处理器诸如例如数字信号处理器(DSP)、微控制器、专用集成电路(ASIC)或微处理器。

[0342] 程序代码(例如3830)可以用高级过程编程语言或面向对象的编程语言来实现，以与处理系统通信。如果期望的话，程序代码也可以用汇编语言或机器语言实现。事实上，本文中描述的机制在范围上不限于任何特定的编程语言。在任何情况下，语言可以是编译语言或解释语言。

[0343] 可以通过存储在机器可读介质上的代表性指令来实现至少一个实施例的一个或多个方面，所述代表性指令代表处理器内的各种逻辑，当其被机器读时，使得机器制造逻辑来执行本文中描述的一个或多个技术。这种被称为“IP核”的表示可以被存储在有形的机器可读介质上，并被供应给各种客户或制造设施，以加载到实际制造逻辑或处理器的制造机器中。

[0344] 这种机器可读存储介质可以包括但不限于由机器或设备制造或形成的制品的非暂态有形布置，其包括诸如硬盘的存储介质、任何其它类型的盘(包括软盘、光盘、紧致盘只读存储器(CD-ROM)、紧致盘可重写设备(CD-RW)和磁光盘)、半导体设备(诸如只读存储器(ROM)、诸如动态随机存取存储器(DRAM)的随机存取存储器(RAM)、静态随机存取存储器(SRAM)、可擦除可编程只读存储器(EPROM)、闪存、电可擦除可编程只读存储器(EEPROM))、相变存储器(PCM)、磁卡或光卡，或适于存储电子指令的任何其它类型的介质。

[0345] 因此，本公开的实施例还包括包含指令或包含诸如硬件描述语言(HDL)的设计数据的非暂态有形机器可读介质，所述设计数据定义了本文中描述的结构、电路、装置、处理器和/或系统特征。这些实施例也可以被称为程序产品。

[0346] 图38中描绘的计算系统是可被用于实现本文中所讨论的各种实施例的计算系统的实施例的示意图。将显而易见的是，图38中描述的系统的各种组件可以被组合在片上系统(SoC)架构中或者能够实现本文中提供的示例和实现的功能和特征的任何其它合适的配置中。

[0347] 在某些情况下，指令转换器可被用于将指令从源指令集转换成目标指令集。例如，指令转换器可以将指令转译(例如，使用静态二进制转译、包括动态编译的动态二进制转译)、变形、仿真或以其它方式将指令转换成要由核处理的一个或多个其它指令。指令转换器可以用软件、硬件、固件或其组合来实现。指令转换器可以在处理器上、在处理器外、或部分在处理器上部分在处理器外。

[0348] 图39是根据本公开的实施例的对比使用软件指令转换器将源指令集中的二进制指令转换成目标指令集中的二进制指令的框图。在示出的实施例中，指令转换器是软件指令转换器，尽管备选地，指令转换器可以用软件、固件、硬件或其各种组合来实现。图39示出了采用高级语言3902的程序，其可以使用x86编译器3904编译，以生成x86二进制代码3906，所述代码可以由具有至少一个x86指令集核的处理器3916原生地执行。具有至少一个x86指令集核的处理器3916代表能够通过兼容地执行或以其它方式处理以下项来执行与具有至少一个x86指令集核的英特尔处理器基本相同的功能，以便实现与具有至少一个x86指令集核的英特尔处理器基本相同的结果的任何处理器，所述项为：(1)英特尔x86指令集核的指令集的主要部分，或(2)目标为在具有至少一个x86指令集核的英特尔处理器上运行的应用或其它软件的目标代码版本。x86编译器3904表示可操作来生成x86二进制代码3906(例如，

目标代码)的编译器,其可以在具有或不具有附加链接处理的情况下,在具有至少一个x86指令集核的处理器3916上执行。类似地,图39示出了可以使用备选指令集编译器3908来编译采用高级语言3902的程序,以生成备选指令集二进制代码3910,所述备选指令集二进制代码3910可以由没有至少一个x86指令集核的处理器3914(例如,具有执行加州桑尼维尔市的MIPS技术公司的MIPS指令集和/或执行加州桑尼维尔市的ARM控股公司的ARM指令集的核的处理器)原生地执行。指令转换器3912被用于将x86二进制代码3906转换成可由没有x86指令集核的处理器3914原生地执行的代码。该转换的代码不太可能与备选指令集二进制代码3910相同,因为能够做到这样的指令转换器是难以制造的;然而,转换的代码将完成通用操作,并由来自备选指令集的指令组成。因此,指令转换器3912代表软件、固件、硬件或其组合,其通过仿真、模拟或任何其它过程来允许不具有x86指令集处理器或核的处理器或其它电子设备执行x86二进制代码3906。

[0349] 尽管已经根据某些实现和通常相关联的方法描述了本公开,但是本领域技术人员将明白这些实现和方法的变更和置换。例如,本文中描述的动作可以以不同于所描述的顺序来执行,并且仍然实现期望的结果。作为一个示例,附图中描绘的过程不一定要求所示的特定顺序或序列顺序来实现期望的结果。在某些实现中,多任务和并行处理可能是有利的。其它变化也处于以下权利要求的范围内。

[0350] 本文中呈现的架构仅通过示例的方式提供,并且意在是非排它性和非限制性的。此外,所公开的各个部分仅意在进行逻辑划分,并且不一定代表物理上分开的硬件和/或软件组件。某些计算系统可以在单个物理存储设备中提供存储器元件,并且在其它情况下,存储器元件可以在功能上跨许多物理设备分布。在虚拟机管理器或监管程序的情况下,可以以运行在虚拟化层上的软件或固件的形式来提供全部或部分功能,以提供所公开的逻辑功能。

[0351] 注意,利用本文中提供的示例,交互可以按照单个计算系统来描述。然而,这样做只是为了清楚和示例。在某些情况下,仅通过引用单个计算系统来描述流程的给定集的一个或多个功能可能更容易。此外,用于深度学习和恶意软件检测的系统是易于可扩展的,并且可以跨大量的组件(例如,多个计算系统)以及更复杂/精巧的布置和配置来实现。因此,所提供的示例不应限制计算系统的范围或禁止其广泛教导,因为其潜在地应用于无数的其它架构。

[0352] 如本文中所使用的,除非明确声明相反,否则短语“至少一个”的使用是指命名的项、元素、条件或活动的任何组合。例如,“X、Y和Z中的至少一个”意在表示以下任何一个:1)至少一个X,但不是Y且不是Z;2)至少一个Y,但不是X且不是Z;3)至少一个Z,但不是X且不是Y;4)至少一个X和至少一个Y,但不是Z;5)至少一个X和至少一个Z,但不是Y;6)至少一个Y和至少一个Z,但不是X;或7)至少一个X、至少一个Y和至少一个Z。

[0353] 此外,除非明确声明相反,否则术语“第一”、“第二”、“第三”等意在区分特定名词(例如,元素、条件、模块、活动、操作、权利要求元素等),它们修饰但不意在指示被修饰名词的任何类型的顺序、等级、重要性、时间序列或层级。例如,“第一X”和“第二X”意在表示两个单独的X元素,其不必受到这两个元素的任何顺序、等级、重要性、时间序列或层次的限制。

[0354] 说明书中对“一个实施例”、“实施例”、“一些实施例”等的引用指示所描述的(一个或多个)实施例可以包括特定的特征、结构或特性,但是每个实施例可以或可以不一定包括

该特定的特征、结构或特性。此外，这些短语不一定指相同实施例。

[0355] 尽管本说明书包含许多特定的实现细节，但这些细节不应被解释为对任何实施例的范围或可能要求保护的内容的限制，而是对特定于特定实施例的特征的描述。本说明书中在单独实施例的上下文中描述的某些特征也可以在单个实施例中组合实现。相反，在单个实施例的上下文中描述的各种特征也可以在多个实施例中单独实现或者以任何合适的子组合实现。此外，尽管特征可以在上面被描述为在某些组合中起作用，并且甚至初始也是这样要求保护的，但是在某些情况下，来自所要求保护的组合的一个或多个特征可以从该组合中删除，并且所要求保护的组合可以针对子组合或子组合的变体。

[0356] 类似地，上述实施例中的各种系统组件和模块的分离不应理解为在所有实施例中都要求这样的分离。应当理解，所描述的程序组件、模块和系统通常可以一起集成在单个软件产品中，或者封装到多个软件产品中。

[0357] 因此，已经描述了主题的特定实施例。其它实施例处于本公开的范围内。本领域的技术人员可以确定许多其它的变化、替换、变体、变更和修改，并且本公开意在涵盖落入所附权利要求范围内的所有这些变化、替换、变体、变更和修改。

[0358] 其它注释和示例

以下示例涉及根据本说明书的实施例。系统、装置、方法和机器可读存储介质实施例可以包括以下示例中的一个或组合：

示例A1提供了一种装置、系统、处理器、机器可读介质、方法和/或基于硬件、基于固件和/或基于软件的逻辑，其中A1的示例包括执行第一加密指令以生成指向用于存储对象的存储器中的存储器位置的第一以密码方式编码的指针，包括：至少部分地基于与对象相关联的第一上下文信息和第一密钥来计算存储器位置的基地址的加密片；将基地址的加密片存储在第一以密码方式编码的指针的第一位中；并且基于确定对象要被存储在静态可寻址的存储器区域中，在第一以密码方式编码的指针中指示：要从第一指令操作数中获得要被用于解码第一以密码方式编码的指针的第一上下文信息。

[0359] 在示例A2中，示例A1的主题可以可选地包括，其中第一上下文信息包括：指定对象的大小的大小元数据和指定第一以密码方式编码的指针访问存储器位置的一个或多个访问许可的第一许可元数据。

[0360] 在示例A3中，示例A1-A2中任一个的主题可以可选地包括，其中与基地址的明文片相邻的较高地址位被存储在寄存器中，其中计算包括基于第一密钥和至少包括第一上下文信息的调整来加密基地址的明文片。

[0361] 在示例A4中，示例A1-A3中任一个的主题可以可选地包括，其中执行第一加密指令还包括：计算对象要被完全存储在由存储器区域的第一地址和存储器区域的第二地址界定的存储器区域内。

[0362] 在示例A5中，示例A1-A4中任一个的主题可以可选地包括，其中存储器区域的第二地址等于存储器区域的第一地址加上存储器区域大小，并且其中存储器区域大小至少部分地基于存储器位置的基地址的加密片的位宽。

[0363] 在示例A6中，示例A1-A5中任一个的主题可以可选地包括，其中执行第一加密指令还包括：响应于确定第一加密指令中的第二指令操作数指示第一上下文信息要被存储在存储器中，将第一上下文信息存储在存储器中的表的表条目中，其中表条目被映射到第一以

密码方式编码的指针。

[0364] 在示例A7中，示例A1-A6中任一个的主题可以可选地包括，其中第一元数据被存储在第一以密码方式编码的指针的第二位中，并且移位被存储在第一以密码方式编码的指针的第三位中。

[0365] 在示例A8中，示例A1-A7中任一个的主题可以可选地包括，其中部分地基于被存储在第一以密码方式编码的指针的第二位中的第一元数据来计算基地址的加密片，其中第一元数据表示随机生成的值或确定性不同的值。

[0366] 在示例A9中，示例A1-A8中任一个的主题可以可选地包括执行第二加密指令以生成指向第二对象的存储器中的第二存储器位置的第二以密码方式编码的指针，包括：至少部分地基于与第二对象相关联的第二上下文信息来计算第二存储器位置的第二基地址的第二加密片；将第二存储器位置的第二基地址的加密片存储在第二以密码方式编码的指针中；并且基于确定第二对象要被存储在非静态可寻址的第二存储器区域中，在第二以密码方式编码的指针中指示要从存储器中动态获得要被用于解码第二以密码方式编码的指针的第二上下文信息。

[0367] 在示例A10中，示例A1-A9中任一个的主题可以可选地包括，其中存储器中的表包含多个表条目，其中包含第二上下文信息的新表条目要由第二以密码方式编码的指针的至少一部分来进行索引。

[0368] 在示例A11中，示例A1-A10中任一个的主题可以可选地包括，其中执行第二加密指令进一步包括尝试使用第二基地址的加密片作为索引，在表中存储包含第二上下文信息的新表条目。

[0369] 在示例A12中，示例A1-A11中任一个的主题可以可选地包括，其中执行第二加密指令进一步包括，响应于确定表中的现有表条目由第二基地址的加密片索引并且包含其它上下文信息：将其它上下文信息从表中的现有表条目拷贝到包含冲突表条目的存储结构；以及将第二上下文信息存储在表中的现有表条目中。

[0370] 在示例A13中，示例A1-A12中任一个的主题可以可选地包括执行第三指令，所述第三指令包括：包含第一以密码方式编码的指针的第一操作数；包含第一上下文信息的第二操作数；以及包含新上下文信息的第三操作数，其中执行第三指令包括生成指向存储器位置的第三以密码方式编码的指针，包括：根据从第一操作数中获得的第一以密码方式编码的指针来计算基地址，其中至少部分地基于从第二操作数和第一密钥中获得的第一上下文信息来计算基地址；以及至少部分地基于从第三操作数中获得的新上下文信息来计算存储器位置的基地址的新加密片。

[0371] 在示例A14中，示例A1-A13中任一个的主题可以可选地包括，其中生成第三以密码方式编码的指针包括：确定新上下文信息指示小于或等于与第一以密码方式编码的指针相关联的第一存储器边界的新存储器边界；以及确定新上下文信息包括新许可元数据，所述新许可元数据向存储器位置授予与第一上下文信息中的第一许可元数据向存储器位置授予的访问许可相比相同或更少的访问许可。

[0372] 在示例A15中，示例A1-A14中任一个的主题可以可选地包括，其中新上下文信息包括指定对象的大小的大小元数据、指定第三以密码方式编码的指针访问存储器位置的一个或多个访问许可的新许可元数据、以及指定对象的类别的类型元数据。

[0373] 示例B1提供了一种装置、系统、处理器、机器可读介质、方法和/或基于硬件、基于固件和/或基于软件的逻辑，其中B1的示例包括执行第一*加密*指令以生成指向用于存储对象的存储器中的第一存储器位置的第一以密码方式编码的指针，包括：至少部分地基于与对象相关联的第一上下文信息和第一密钥来计算第一存储器位置的基地址的加密片；将基地址的加密片存储在第一以密码方式编码的指针的第一位中；并且基于确定第一上下文信息将不被存储器访问指令所静态访问，尝试将包含第一上下文信息的第一条目存储在存储器的第一表中。

[0374] 在示例B2中，示例B1的主题可以可选地包括，其中第一以密码方式编码的指针中的基地址的加密片被用作索引，以尝试将包含第一上下文信息的第一条目存储在存储器中的第一表中。

[0375] 在示例B3中，示例B1-B2中任一个的主题可以可选地包括，其中第一上下文信息包括指定要被存储在第一存储器位置的对象的大小的大小元数据和指定第一以密码方式编码的指针访问第一存储器位置的一个或多个访问许可的许可元数据。

[0376] 在示例B4中，示例B1-B3中任一个的主题可以可选地包括，其中第一表中的第一条目进一步包含基地址的较高地址位。

[0377] 在示例B5中，示例B1-B4中任一个的主题可以可选地包括，其中基地址的较高地址位被存储在控制寄存器中。

[0378] 在示例B6中，示例B1-B5中任一个的主题可以可选地包括，其中确定第一上下文信息不能被存储器访问指令静态访问进一步包括：确定对象要被存储在非静态可寻址的存储器区域中；以及在第一以密码方式编码的指针中指示要从存储器中动态获得第一上下文信息。

[0379] 在示例B7中，示例B1-B6中任一个的主题可以可选地包括，其中确定第一上下文信息不能被存储器访问指令静态访问进一步包括：确定第一*加密*指令的指令参数指示第一上下文信息要被添加到存储器中的第一表；以及在第一以密码方式编码的指针中指示要从存储器中动态获得第一上下文信息。

[0380] 在示例B8中，示例B1-B7中任一个的主题可以可选地包括，其中第一以密码方式编码的指针的至少一部分被用作第一索引，以尝试将第一表中的第一条目存储在存储器中。

[0381] 在示例B9中，示例B1-B8中任一个的主题可以可选地包括，其中执行第一*加密*指令进一步包括：响应于确定包含第二上下文信息的现有条目被存储在具有与第一索引等效的现有索引的第一表中：将第二上下文信息从第一表中的现有条目中拷贝到用于存储冲突的表条目的逐出表中的逐出条目；以及将第一上下文信息存储在第一表中的现有条目中。

[0382] 在示例B10中，示例B1-B9中任一个的主题可以可选地包括执行第二指令以确定第一表中的现有条目的现有索引等效于从第二以密码方式编码的指针的至少一部分中导出的第二索引；以及响应于第一确定，第二以密码方式编码的指针正试图访问至少部分地存储在与第一上下文信息相关联的存储器分配的上边界之外的数据：搜索逐出表中具有等效于第二索引的被逐出索引的被逐出条目；以及响应于第二确定，所述确定关于包含在被逐出条目中的第二上下文信息将允许第二以密码方式编码的指针访问数据，将第一表的现有条目中的第一上下文信息与被逐出表中的被逐出条目中的第二上下文信息交换。

[0383] 在示例B11中，示例B1-B10中任一个的主题可以可选地包括，其中执行第二指令进

一步包括:从存储在第一表中的现有条目中的第一上下文信息中获得存储器分配大小元数据,其中第二以密码方式编码的指针正尝试至少部分地访问与第一上下文信息相关联的存储器分配的上边界之外的数据的第一确定是基于将来自存储在第一表中的现有条目中的第一上下文信息的存储器分配大小元数据与第二以密码方式编码的指针中的偏移进行比较。

[0384] 在示例B12中,示例B1-B11中任一个的主题可以可选地包括执行第二指令以从存储器中的第一表中检索第一上下文信息;以及解码第一以密码方式编码的指针,包括:对第一以密码方式编码的指针中的基地址的加密片执行以密码方式的算法,以至少部分地基于包含在存储器中的第一表中的第一密钥和第一上下文信息来计算基地址的解密片;以及至少部分地基于基地址的较高地址位、基地址的解密片以及存储在第一以密码方式编码的指针的第二部分中的偏移来生成线性地址。

[0385] 示例C1提供了一种装置、系统、处理器、机器可读介质、方法和/或基于硬件、基于固件和/或基于软件的逻辑,其中C1的示例包括在第一寄存器中存储指向存储器位置的第一以密码方式编码的指针,其中存储器位置的基地址的加密片被存储在第一以密码方式编码的指针的第一部分中;执行第一指令以解码第一以密码方式编码的指针,包括:对基地址的加密片执行以密码方式的算法,以至少部分地基于第一密钥和第一调整来计算基地址的解密片,所述第一调整包括包含在第一指令的第一操作数中的上下文信息;以及至少部分地基于基地址的较高地址位、基地址的解密片以及存储在第一以密码方式编码的指针的第二部分中的偏移来生成明文线性地址。

[0386] 在示例C2中,示例C1的主题可以可选地包括,其中第一调整包括表示随机生成的值或确定性不同的值的第一元数据。

[0387] 在示例C3中,示例C1-C2中任一个的主题可以可选地包括,其中第一元数据被存储在第一以密码方式编码的指针的第三部分中。

[0388] 在示例C4中,示例C1-C3中任一个的主题可以可选地包括,其中较高地址位被存储在控制寄存器中。

[0389] 在示例C5中,示例C1-C4中任一个的主题可以可选地包括,其中上下文信息包括指定要存储在存储器位置的对象的大小的大小元数据和指定第一以密码方式编码的指针访问存储器位置的一个或多个访问许可的许可元数据。

[0390] 在示例C6中,示例C1-C5中任一个的主题可以可选地进一步包括:将多个最低有效位串接到基地址的解密片,其中最低有效位中的每一个被设置为零,以基于最低有效位的数量将基地址与字节边界对齐。

[0391] 在示例C7中,示例C1-C6中任一个的主题可以可选地包括,其中响应于以下项在基地址的加密片上执行以密码方式的算法:确定存储在第一以密码方式编码的指针的第二部分中的移位小于上下文信息中指定的大小元数据;以及确定根据在上下文信息中指定的许可元数据来允许与第一指令相关联的访问的类型。

[0392] 在示例C8中,示例C1-C7中任一个的主题可以可选地包括,其中执行第一指令进一步包括:使用通过解码第一以密码方式编码的指针生成的明文线性地址来访问在存储器位置处的加密数据。

[0393] 在示例C9中,示例C1-C8中任一个的主题可以可选地包括,其中执行第一指令还包

括:基于第二密钥和第二调整来对加密数据进行解密,所述第二调整包括至少部分地从第一以密码方式编码的指针中导出的一个或多个位。

[0394] 在示例C10中,示例C1-C9中任一个的主题可以可选地包括,其中第二调整包括第一元数据和通过解码第一以密码方式编码的指针而生成的明文线性地址,其中第一元数据被存储在第一以密码方式编码的指针的第三部分中。

[0395] 在示例C11中,示例C1-C10中任一个的主题可以可选地包括,其中第二调整包括基地址的解密片。

[0396] 在示例C12中,示例C1-C11中任一个的主题可以可选地包括,其中第二调整包括以下至少一项:表示与由第一以密码方式编码的指针所引用的存储器分配相关联的随机生成的值的标签元数据;指示对存储器位置授权的访问的级别的许可元数据;指示存储在存储器位置中的加密数据的类别的类型元数据;表示与由第一以密码方式编码的指针所引用的存储器分配相关联的确定性不同的值的版本元数据;指示用户级别或监管者级别的特权级别元数据;以及用于表示函数的特定群的唯一值的密码术上下文标识符。

[0397] 在示例C13中,示例C1-C12中任一个的主题可以可选地包括在第二寄存器中存储指向第二存储器位置的第二以密码方式编码的指针,其中第二存储器位置的第二基地址的第二加密片被存储在第二以密码方式编码的指针中;执行第二指令以解码第二以密码方式编码的指针,包括:对第二基地址的第二加密片执行以密码方式的算法,以至少部分地基于第一密钥和第二调整来计算第二基地址的第二解密片,所述第二调整包括从存储器中检索的第二上下文信息;以及至少部分地基于第二基地址的第二较高地址位、第二基地址的第二解密片以及存储在第二以密码方式编码的指针中的第二移位来生成第二明文线性地址。

[0398] 示例D1提供了一种装置、系统、处理器、机器可读介质、方法和/或基于硬件、基于固件和/或基于软件的逻辑,其中D1的示例包括由多租户环境中的可信运行时执行第一指令以:为存储器中的私有存储器区域生成第一地址密钥;并且生成指向存储器中的私有存储器区域的第一以密码方式编码的指针,包括:将与私有存储器区域相关联的第一上下文信息存储在第一以密码方式编码的指针的第一位中;以及至少部分地基于第一地址密钥和第一调整,对私有存储器区域的第一线性地址的片执行以密码方式的算法,所述第一调整包括第一上下文信息;以及许可多租户环境中的第一租户访问第一地址密钥和指向私有存储器区域的第一以密码方式编码的指针。

[0399] 在示例D2中,示例D1的主题可以可选地包括,其中执行第一指令进一步包括:基于确定第一租户被授权访问第二租户,许可第一租户访问第二地址密钥、第二数据密钥和指向第二租户的授权入口点地址的第二以密码方式编码的指针。

[0400] 在示例D3中,示例D1-D2中任一个的主题可以可选地包括,其中执行第一指令进一步包括:从第一租户中接收访问第二租户的请求;并且响应于确定第一租户被授权访问第二租户:将第二租户的第二数据密钥存储在数据密钥寄存器中;将第二租户的第二地址密钥存储在地址密钥寄存器中;并将控制转移给第二租户。

[0401] 在示例D4中,示例D1-D3中任一个的主题可以可选地包括将控制转移给第二租户,包括:在指令指针寄存器中存储指向为第二租户分配的第二存储器区域中的授权入口点的第二以密码方式编码的指针。

[0402] 在示例D5中,示例D1-D4中任一个的主题可以可选地包括执行第一租户的第二指

令以:将第二租户的第二代码密钥存储在代码密钥寄存器中;并且使用目的地地址来访问为第二租户分配的第二存储器区域中的授权入口点;并且执行第二租户的一个或多个第三指令以:将第二租户的第二地址密钥存储在地址密钥寄存器中;并将第二租户的第二数据密钥存储在数据密钥寄存器中。

[0403] 在示例D6中,示例D5的主题可以可选地包括,其中响应于确定授权入口点包含授权指令并且目的地地址与存储器页边界对齐,使用目的地地址来访问第二存储器区域中的授权入口点。

[0404] 在示例D7中,示例D1-D6中任一个的主题可以可选地包括,其中核执行第一租户的第三跳转指令以从第二租户的保密性句柄中检索第二地址密钥、第二数据密钥和第二代码密钥,将第二地址密钥加载到地址密钥寄存器中,将第二数据密钥加载到数据密钥寄存器中,将第二代码密钥加载到代码密钥寄存器中,并使用第二租户的保密性句柄中的授权入口点指针来访问与第二租户相关联的存储器区域中的授权入口点。

[0405] 在示例D8中,示例D7的主题可以可选地包括,其中第二租户的保密性句柄被存储在与第一租户相关联的私有存储器区域中。

[0406] 在示例D9中,示例D1-D8中任一个的主题可以可选地包括执行第一租户的第二指令以将控制转移给第二租户,包括使用存储在第二指令的操作数中的索引来标识存储器中的表中的描述符,其中描述符包括指向为第二租户分配的第二私有存储器区域中的授权入口点的至少第三以密码方式编码的指针和第二租户的第二代码密钥;将第二租户的第二代码密钥存储在代码密钥寄存器中;将第三以密码方式编码的指针存储在寄存器指令指针中;以及使用寄存器指令指针在授权入口点获取下一条指令。

[0407] 在示例D10中,示例D1-D9中任一个的主题可以可选地包括,在第二寄存器中存储指向存储器中的共享存储器区域的共享的以密码方式编码的指针;执行可信运行时的第一指令以进一步:生成指向共享存储器区域的共享的以密码方式编码的指针,包括:将与共享存储器区域相关联的共享上下文信息存储在共享的以密码方式编码的指针的第一位中;以及至少部分地基于共享地址密钥和共享上下文信息,对共享存储器区域的第二线性地址的片执行以密码方式的算法;以及许可第一租户和第二租户访问共享地址密钥、共享数据密钥和指向共享存储器区域的共享的以密码方式编码的指针。

[0408] 在示例D11中,示例D10的主题可以可选地包括解码共享的以密码方式编码的指针,包括:对第二线性地址的加密片执行以密码方式的算法,以至少部分地基于共享地址密钥和共享上下文信息来计算第二线性地址的解密片;至少部分地基于第二线性地址的解密片来生成第二线性地址;以及访问共享存储器区域中第二线性地址处的数据。

[0409] 在示例D12中,示例D11的主题可以可选地包括对共享存储器区域中的第二线性地址处的加密数据执行第二以密码方式的算法,以至少部分地基于至少部分地从共享的以密码方式编码的指针中导出的调整和共享地址密钥来计算解密数据。

[0410] 示例E1提供了一种装置、系统、处理器、机器可读介质、方法和/或基于硬件、基于固件和/或基于软件的逻辑,其中E1的示例包括在多租户环境中存储指向存储器区域的第一以密码方式编码的指针,其中存储器区域的基地址的加密片将被存储在第一以密码方式编码的指针的第一部分中;以及执行第一指令以解码第一以密码方式编码的指针,包括:检索与第一以密码方式编码的指针相关联的第一上下文信息;从第一上下文信息中获得以密

码方式的上下文索引;基于以密码方式的上下文索引选择以密码方式的上下文标识符;至少部分地基于第一密钥、以密码方式的上下文标识符和第一上下文信息的至少一部分,对基地址的加密片执行以密码方式的算法,以计算基地址的解密片;以及至少部分地基于基地址的解密片来生成明文线性地址。

[0411] 在示例E2中,示例E1的主题可以可选地包括,其中从存储器或第一指令的操作数中检索第一上下文信息。

[0412] 在示例E3中,示例E1-E2中任一个的主题可以可选地包括,其中选择以密码方式的上下文标识符包括:将以密码方式的上下文索引与用于分别索引多个以密码方式的上下文标识符的多个以密码方式的上下文索引进行比较,其中多个以密码方式的上下文标识符分别与多租户环境中的多个存储器区域相关联。

[0413] 在示例E4中,示例E1-E3中任一个的主题可以可选地包括,其中多个以密码方式的上下文标识符被存储在由多个以密码方式的上下文索引所索引的表中。

[0414] 在示例E5中,示例E1-E4中任一个的主题可以可选地包括,其中多个以密码方式的上下文标识符包括私有以密码方式的上下文标识符、多播以密码方式的上下文标识符和广播以密码方式的上下文标识符中的至少一个。

[0415] 在示例E6中,示例E1-E5中任一个的主题可以可选地包括,其中私有以密码方式的上下文标识符被授权由第一以密码方式编码的指针使用以访问私有存储器区域。

[0416] 在示例E7中,示例E1-E6中任一个的主题可以可选地包括,其中多播以密码方式的上下文标识符被授权由多租户环境中的第一以密码方式编码的指针和至少一个其它以密码方式编码的指针使用以访问共享存储器区域。

[0417] 在示例E8中,示例E1-E7中任一个的主题可以可选地包括,其中广播以密码方式的上下文标识符被授权由多租户环境中的所有以密码方式编码的指针使用以访问共享存储器区域。

[0418] 实例Y1提供了一种装置,所述装置包括用于执行上述示例A1至E8中任一个的方法的部件。

[0419] 在示例Y2中,示例Y1的主题可以可选地包括,用于执行所述方法的部件包括至少一个处理器和至少一个存储器元件。

[0420] 在示例Y3中,示例Y2的主题可以可选地包括,其中至少一个存储器元件包括机器可读指令,当所述指令被执行时,其使得装置执行以上示例A1-A15、B1-B12、C1-C13、D1-D12至E1-E8中任一个的方法。

[0421] 在示例Y4中,示例Y1-Y3中任一个的主题可以可选地包括,所述装置是计算系统或片上系统中的一个。

[0422] 示例X1提供了包括指令的至少一种机器可读存储介质,其中在所述指令被执行时实现上述示例A1-A15、B1-B12、C1-C13、D1-D12至E1-E8中任一个中的方法、实现装置、或实现系统。

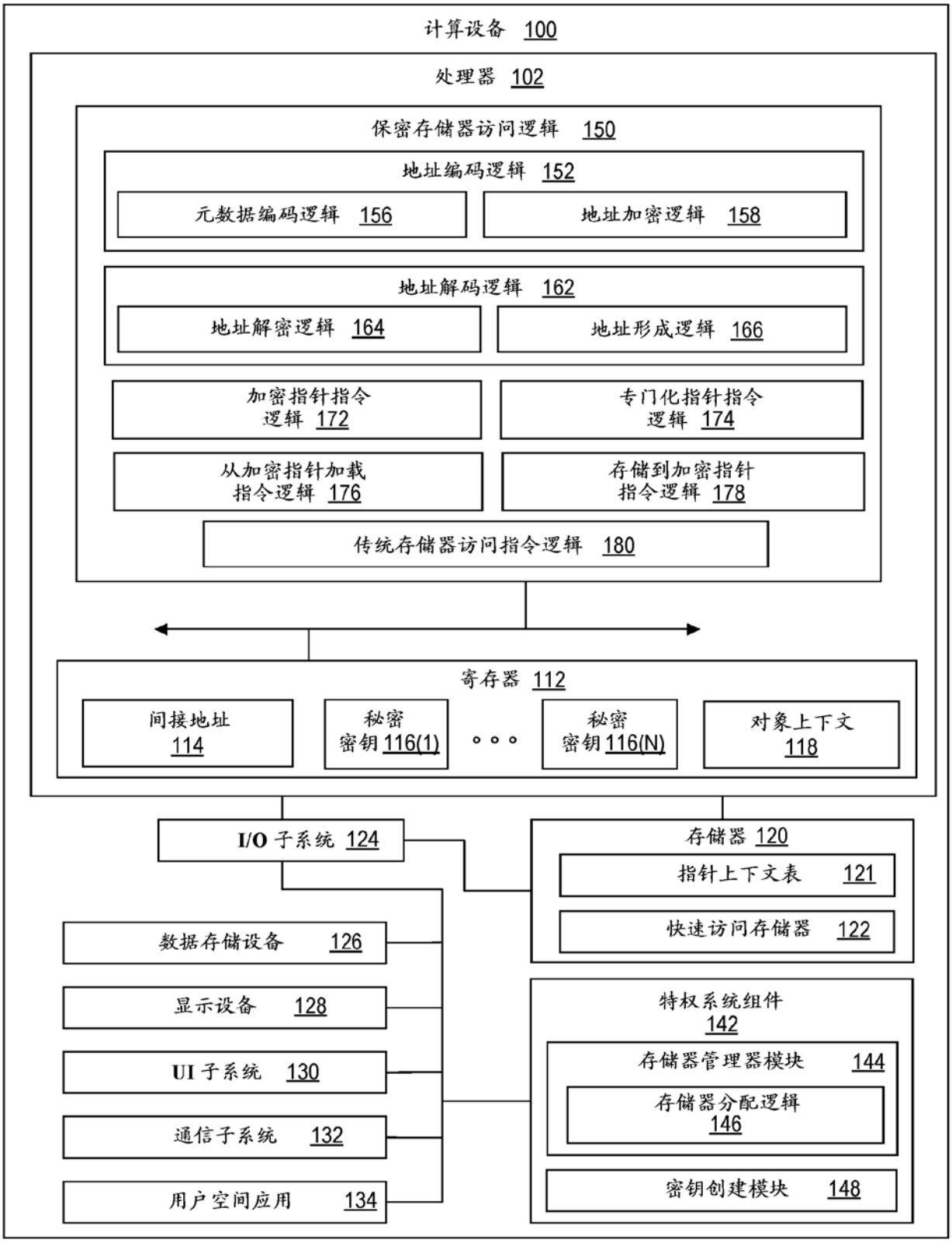

计算设备 100
处理器 102
保密存储器访问逻辑 150
地址编码逻辑 152
元数据编码逻辑 156
地址加密逻辑 158
地址解码逻辑 162
地址解密逻辑 164
地址形成逻辑 166
加密指针指令逻辑 172
专门化指针指令逻辑 174
从加密指针加载指令逻辑 176
存储到加密指针指令逻辑 178
传统存储器访问指令逻辑 180
寄存器 112
间接地址 114
秘密密钥 116(1)
秘密密钥 116(N)
对象上下文 118
I/O 子系统 124
存储器 120
指针上下文表 121
快速访问存储器 122
数据存储设备 126
显示设备 128
UI 子系统 130
通信子系统 132
用户空间应用 134
特权系统组件 142
存储器管理器模块 144
存储器分配逻辑 146
密钥创建模块 148

图 1

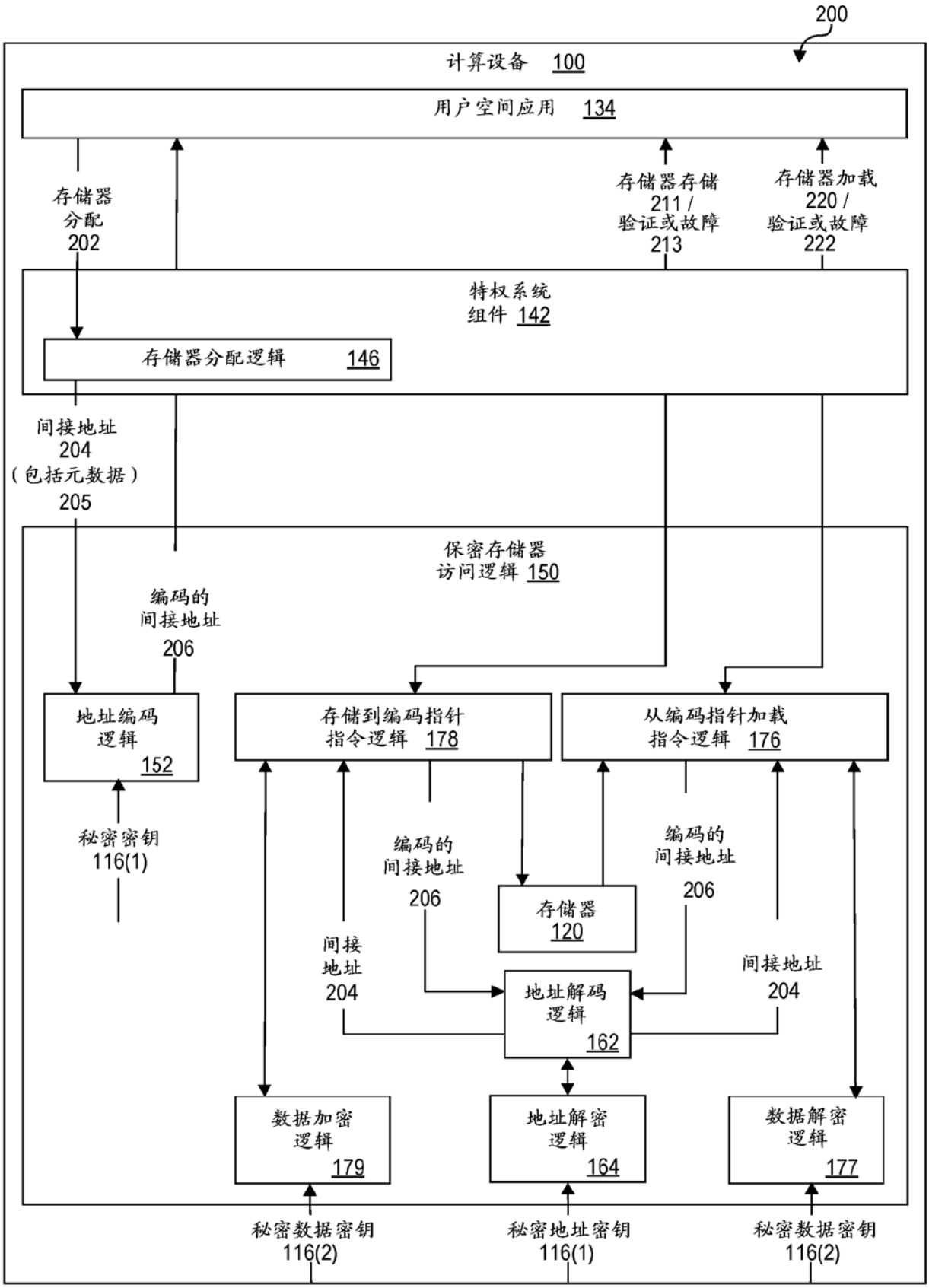

200
计算设备 100
用户空间应用 134
存储器分配 202
存储器存储 211 / 验证或故障 213
存储器加载 220 / 验证或故障 222
特权系统组件 142
存储器分配逻辑 146
间接地址 204 (包括元数据) 205
保密存储器访问逻辑 150
编码的间接地址 206
地址编码逻辑 152
秘密密钥 116(1)
存储到编码指针指令逻辑 178
从编码指针加载指令逻辑 176
编码的间接地址 206
存储器 120
编码的间接地址 206
间接地址 204
地址解码逻辑 162
间接地址 204
数据加密逻辑 179
地址解密逻辑 164
数据解密逻辑 177
秘密数据密钥 116(2)
秘密地址密钥 116(1)
秘密数据密钥 116(2)

图 2

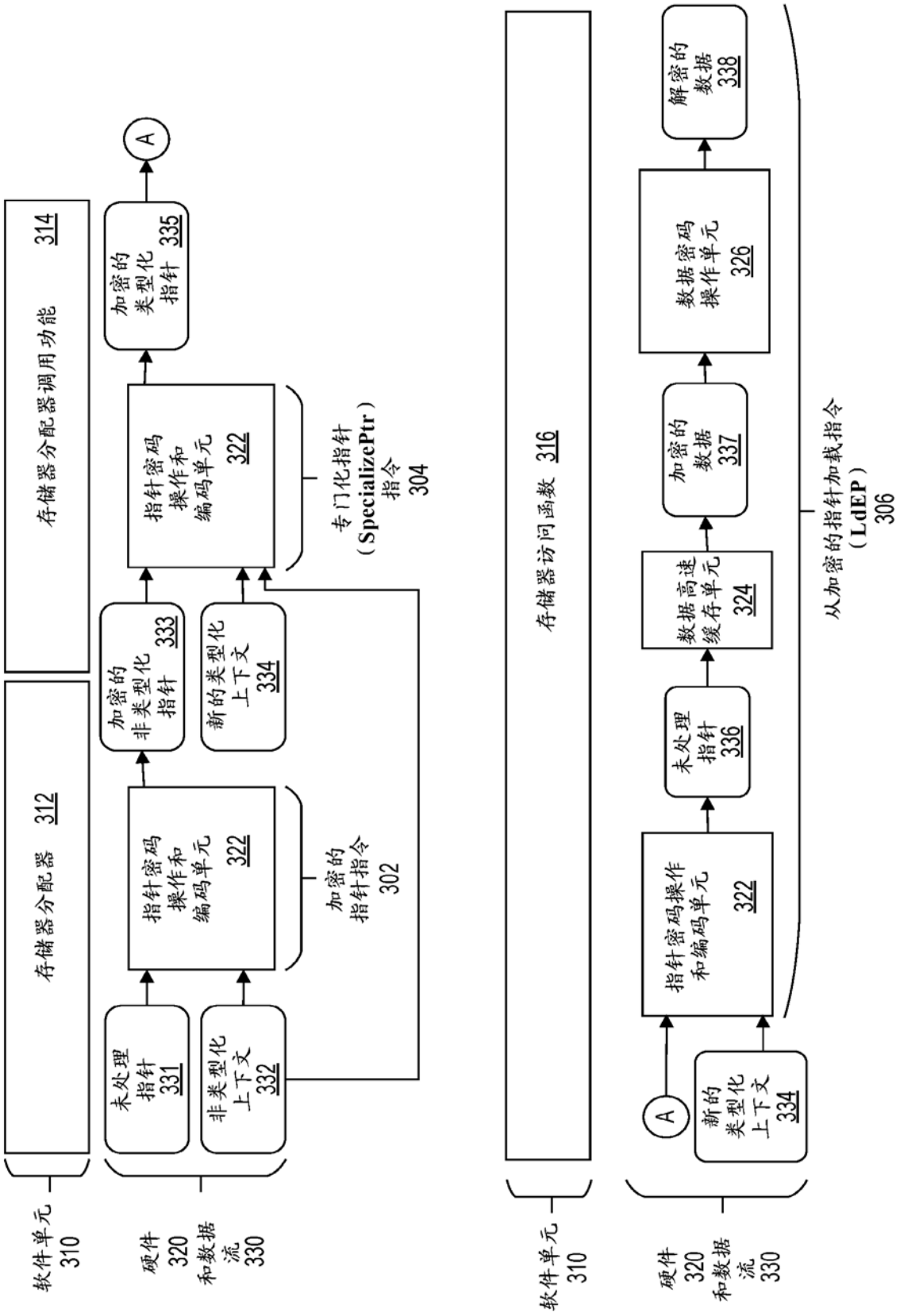

软件单元 310
存储器分配器 312
存储器分配器调用功能 314
硬件 320 和数据流 330
未处理指针 331
非类型化上下文 332
指针密码操作和编码单元 322
加密的指针指令 302
加密的非类型化指针 333
新的类型化上下文 334
专门化指针（SpecializePtr）指令 304
加密的类型化指针 335
A
软件单元 310
存储器访问函数 316
硬件 320 和数据流 330
新的类型化上下文 334
A
指针密码操作和编码单元 322
未处理指针 336
数据高速缓存单元 324
加密的数据 337
数据密码操作单元 326
解密的数据 338
从加密的指针加载指令（LdEP）306

图 3

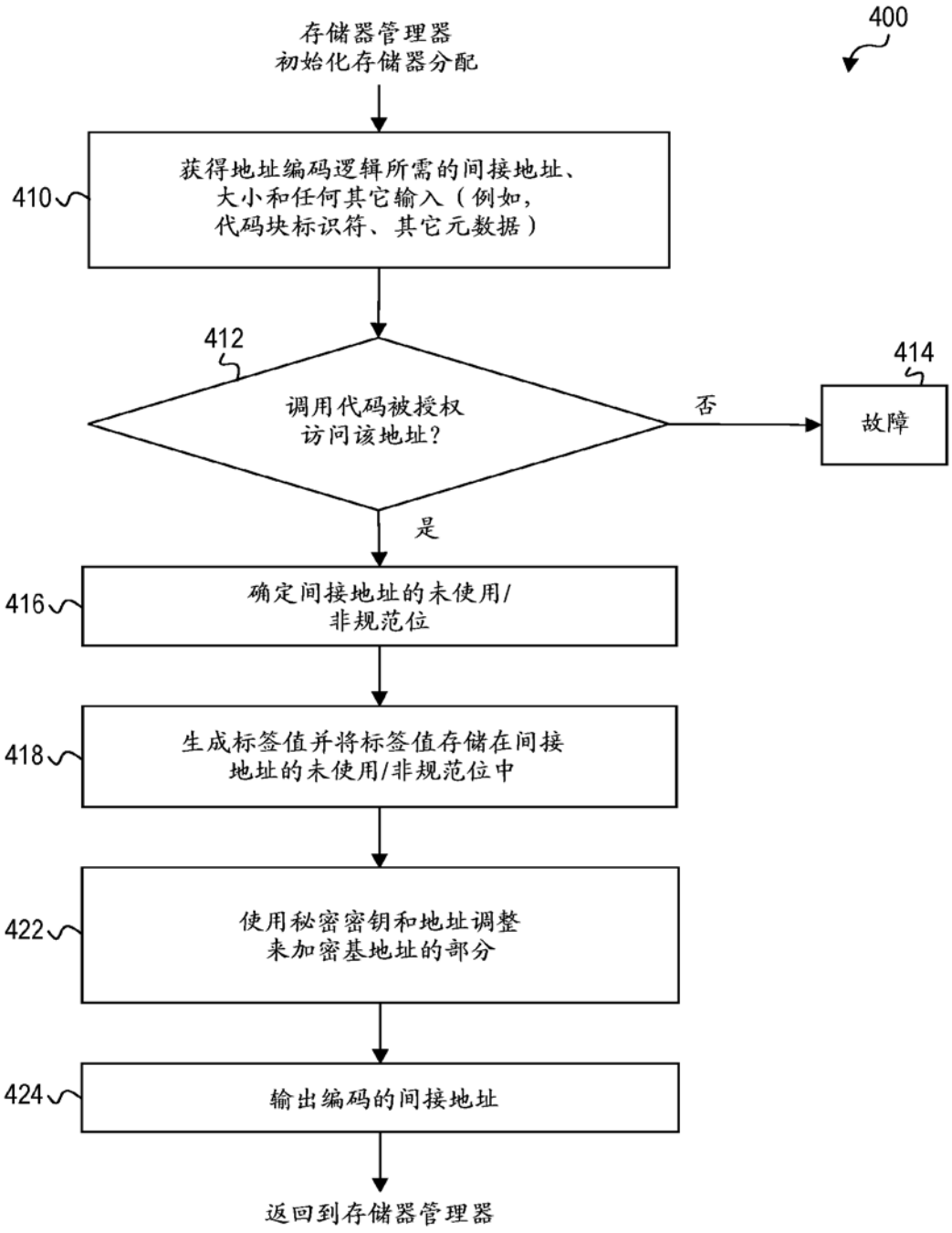

400
存储器管理器
初始化存储器分配
410
获得地址编码逻辑所需的间接地址、
大小和任何其它输入（例如，
代码块标识符、其它元数据）
412
调用代码被授权
访问该地址？
否
414
故障
是
416
确定间接地址的未使用/
非规范位
418
生成标签值并将标签值存储在间接
地址的未使用/非规范位中
422
使用秘密密钥和地址调整
来加密基地址的部分
424
输出编码的间接地址
返回到存储器管理器

图 4

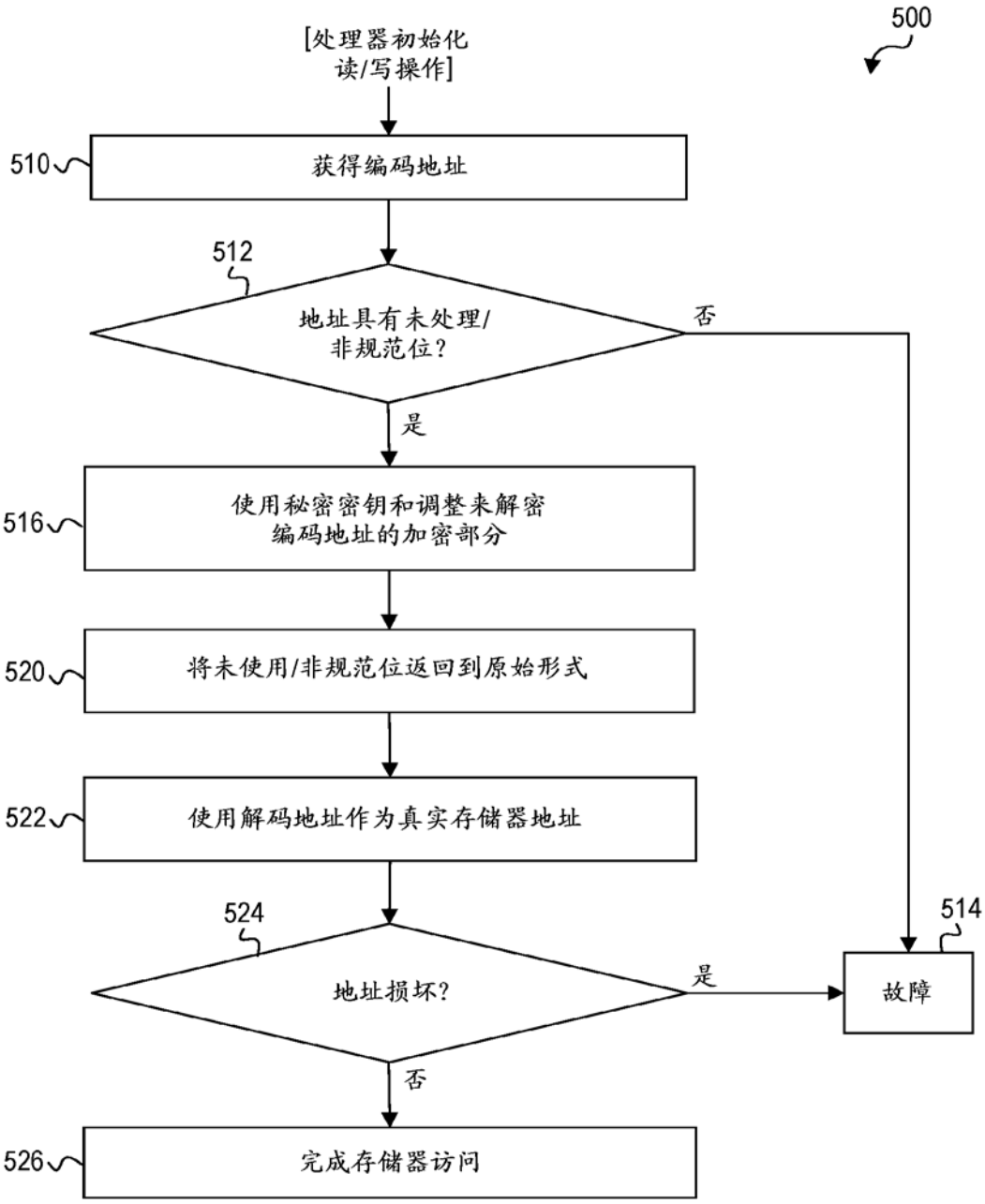

500
[处理器初始化
读/写操作]
510
获得编码地址
512
地址具有未处理/
非规范位?
否
是
516
使用秘密密钥和调整来解密
编码地址的加密部分
520
将未使用/非规范位返回到原始形式
522
使用解码地址作为真实存储器地址
524
地址损坏?
是
514
故障
否
526
完成存储器访问

图 5

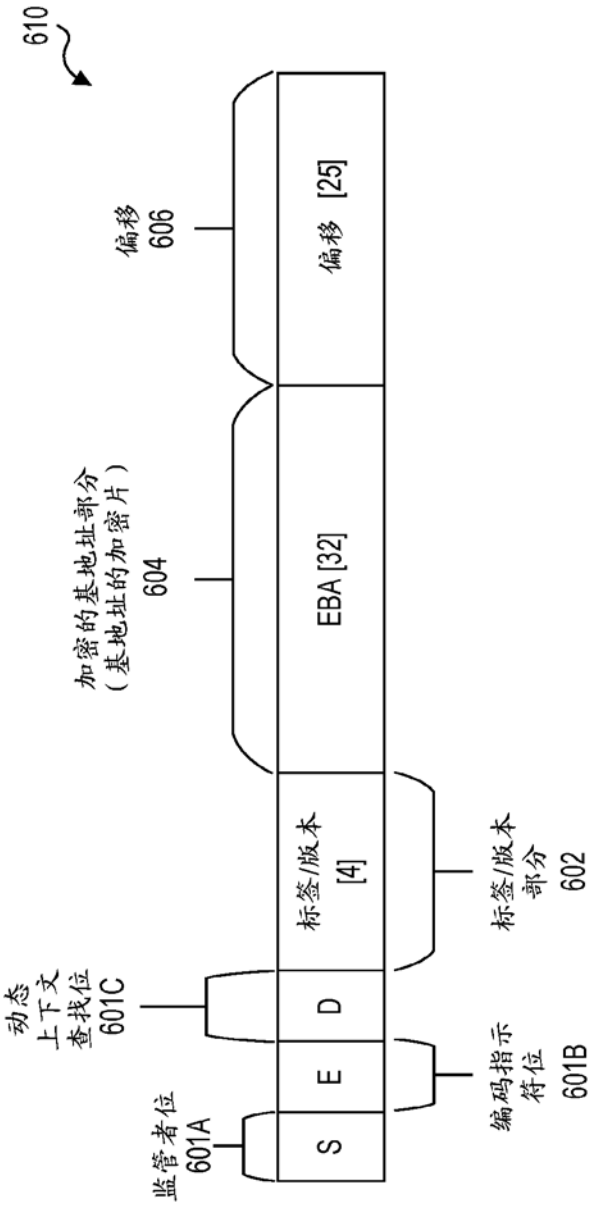

610
偏移 606
偏移 [25]
加密的基地址部分
(基地址的加密片)
604
EBA [32]
标签/版本
[4]
标签/版本
部分
602
动态
上下文
查找位
601C
D
E
S
编码指示
符位
601B
监管者位
601A

图 6

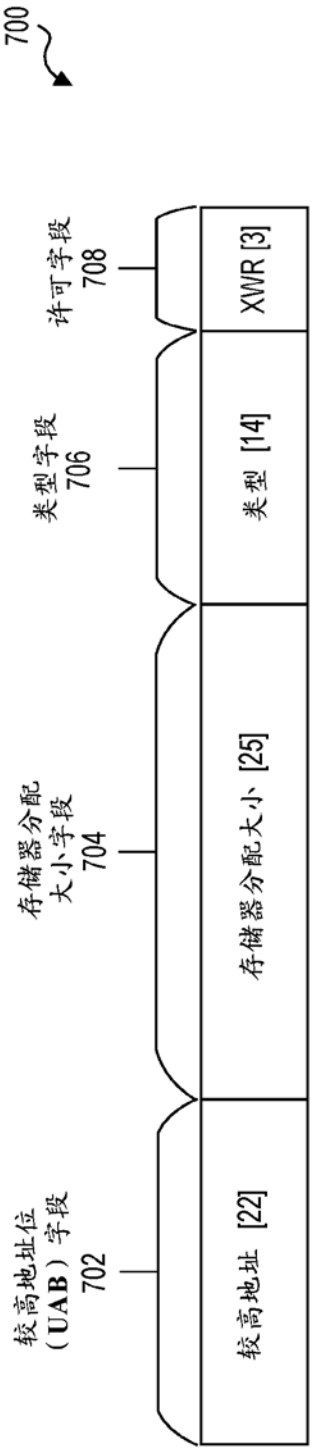

700
较高地址位（UAB）字段 702
存储器分配大小字段 704
类型字段 706
许可字段 708
较高地址 [22]
存储器分配大小 [25]
类型 [14]
XWR [3]

图 7

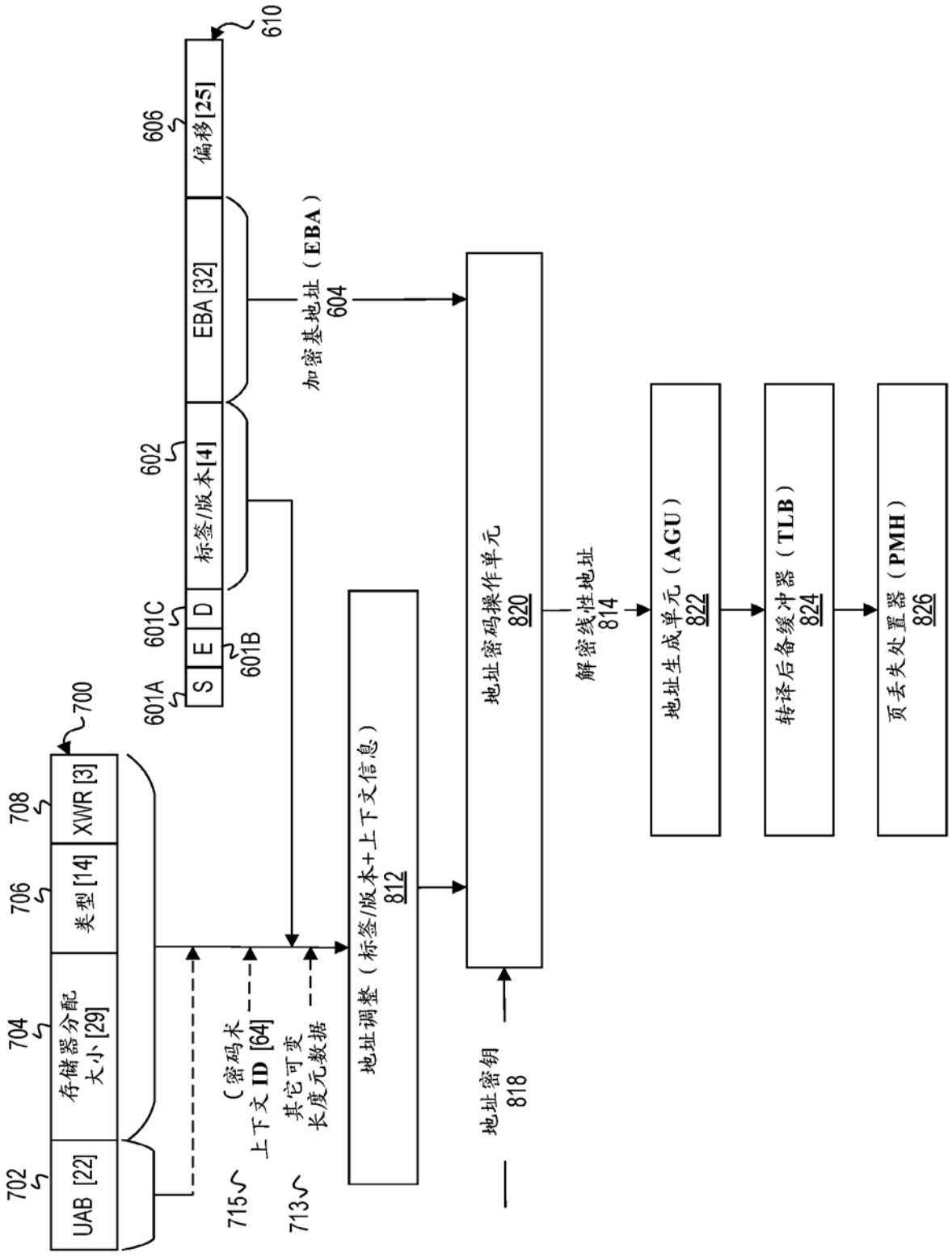

610
606
偏移[25]
EBA[32]
602
标签/版本[4]
601C
E
D
601B
601A
S
加密基地址（EBA）
604
700
708
XWR [3]
706
类型 [14]
704
存储器分配
大小 [29]
702
UAB [22]
715
（密码术
上下文 ID [64]
713
其它可变
长度元数据
地址调整（标签/版本+上下文信息）
812
地址密钥
818
地址密码操作单元
820
解密线性地址
814
地址生成单元（AGU）
822
转译后备缓冲器（TLB）
824
页丢失处置器（PMH）
826

图 8

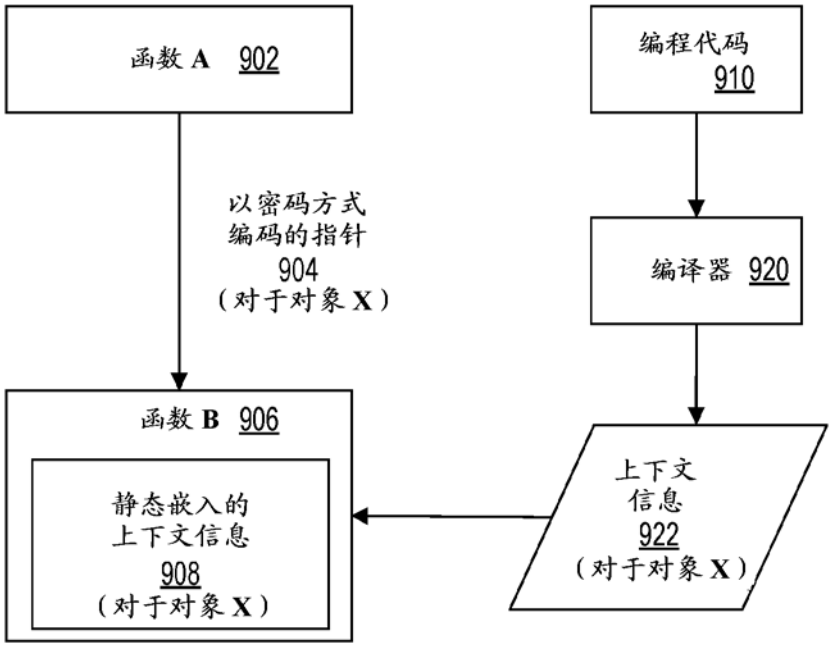

函数 A 902
编程代码
910
以密码方式
编码的指针
904
（对于对象 X）
编译器 920
函数 B 906
静态嵌入的
上下文信息
908
（对于对象 X）
上下文
信息
922
（对于对象 X）

图 9

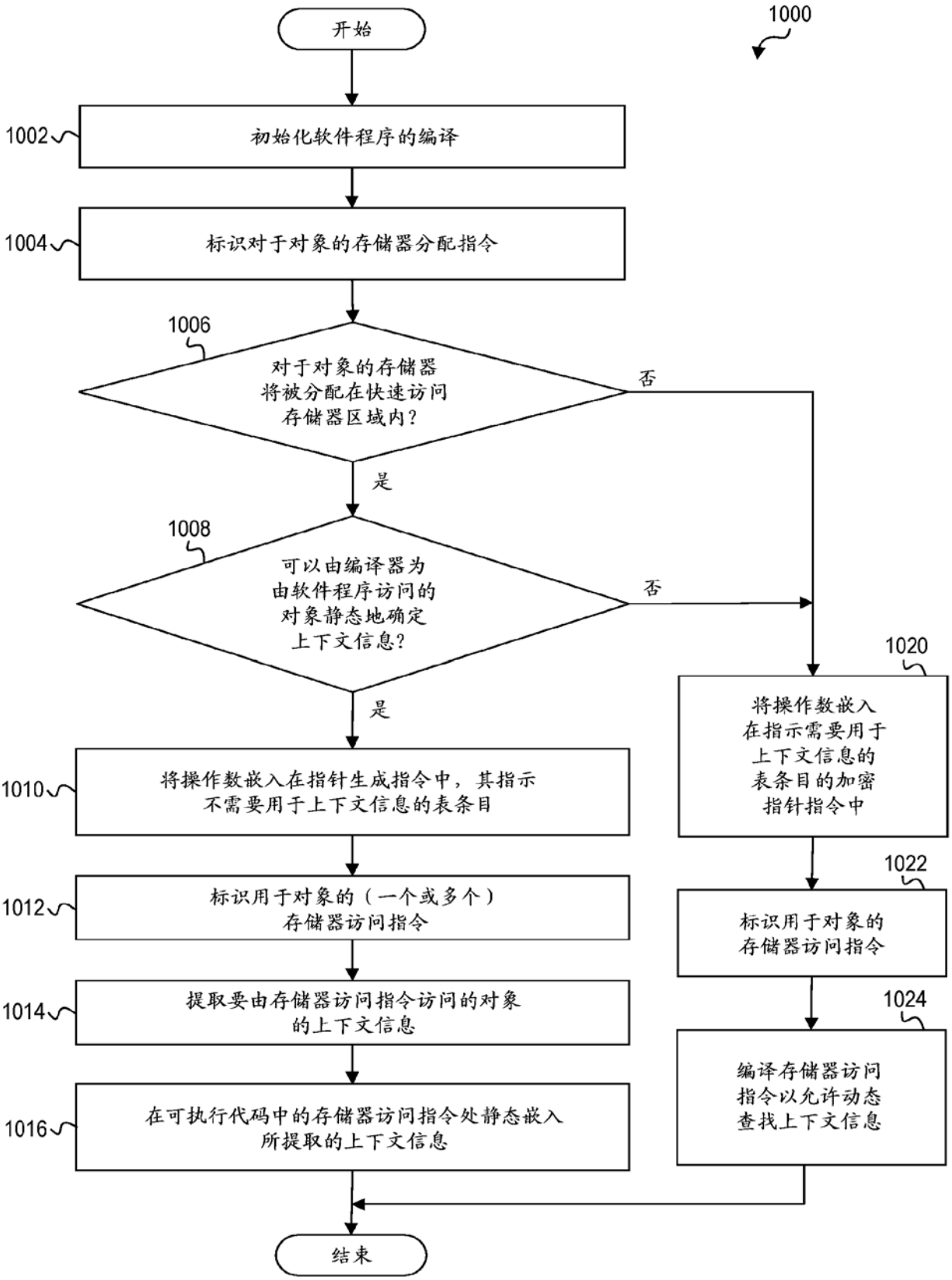

1000
开始
1002
初始化软件程序的编译
1004
标识对于对象的存储器分配指令
1006
对于对象的存储器
将被分配在快速访问
存储器区域内?
否
是
1008
可以由编译器为
由软件程序访问的
对象静态地确定
上下文信息?
否
是
1010
将操作数嵌入在指针生成指令中，其指示
不需要用于上下文信息的表条目
1012
标识用于对象的（一个或多个）
存储器访问指令
1014
提取要由存储器访问指令访问的对象
的上下文信息
1016
在可执行代码中的存储器访问指令处静态嵌入
所提取的上下文信息
1020
将操作数嵌入
在指示需要用于
上下文信息的
表条目的加密
指针指令中
1022
标识用于对象的
存储器访问指令
1024
编译存储器访问
指令以允许动态
查找上下文信息
结束

图 10

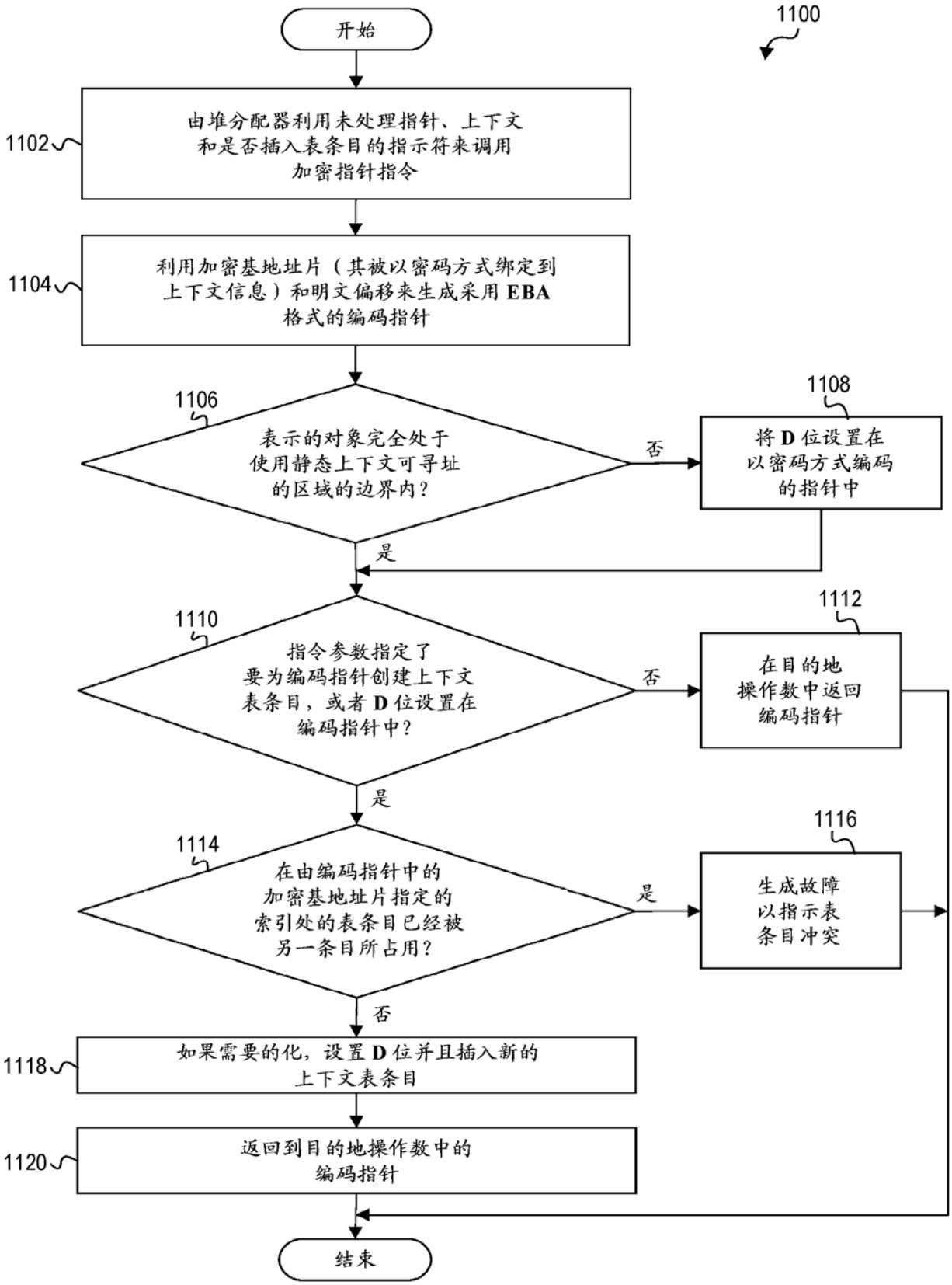

1100
开始
1102
由堆分配器利用未处理指针、上下文和是否插入表条目的指示符来调用加密指针指令
1104
利用加密基地址片（其被以密码方式绑定到上下文信息）和明文偏移来生成采用 EBA 格式的编码指针
1106
表示的对象完全处于使用静态上下文可寻址的区域的边界内？
否
1108
将 D 位设置在以密码方式编码的指针中
是
1110
指令参数指定了要为编码指针创建上下文表条目，或者 D 位设置在编码指针中？
否
1112
在目的地操作数中返回编码指针
是
1114
在由编码指针中的加密基地址片指定的索引处的表条目已经被另一条目所占用？
是
1116
生成故障以指示表条目冲突
否
1118
如果需要的化，设置 D 位并且插入新的上下文表条目
1120
返回到目的地操作数中的编码指针
结束

图 11

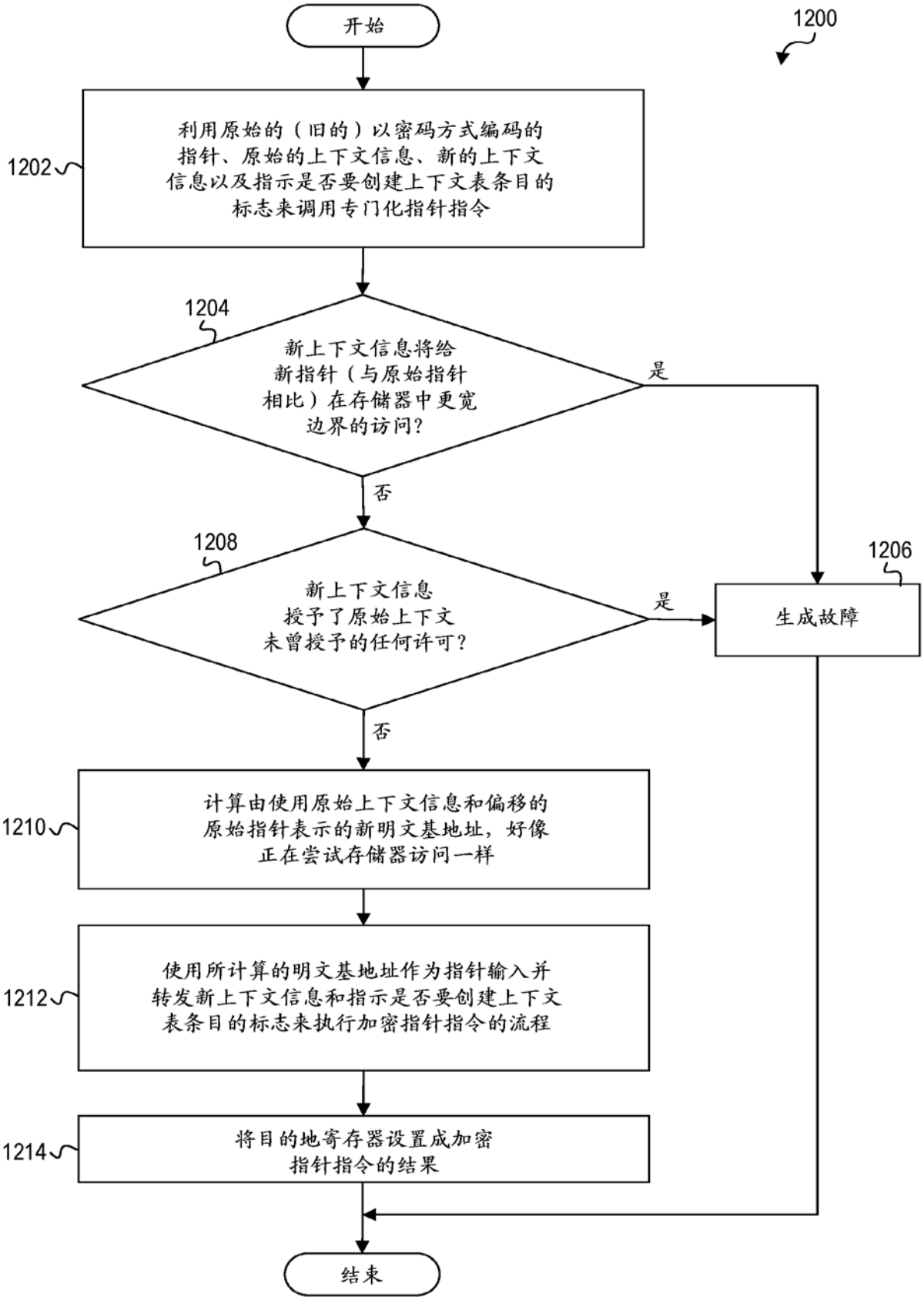

开始
1200
利用原始的（旧的）以密码方式编码的指针、原始的上下文信息、新的上下文信息以及指示是否要创建上下文表条目的标志来调用专门化指针指令
1202
1204
新上下文信息将给新指针（与原始指针相比）在存储器中更宽边界的访问?
是
否
1208
新上下文信息授予了原始上下文未曾授予的任何许可?
是
1206
生成故障
否
计算由使用原始上下文信息和偏移的原始指针表示的新明文基地址，好像正在尝试存储器访问一样
1210
使用所计算的明文基地址作为指针输入并转发新上下文信息和指示是否要创建上下文表条目的标志来执行加密指针指令的流程
1212
将目的地寄存器设置成加密指针指令的结果
1214
结束

图 12

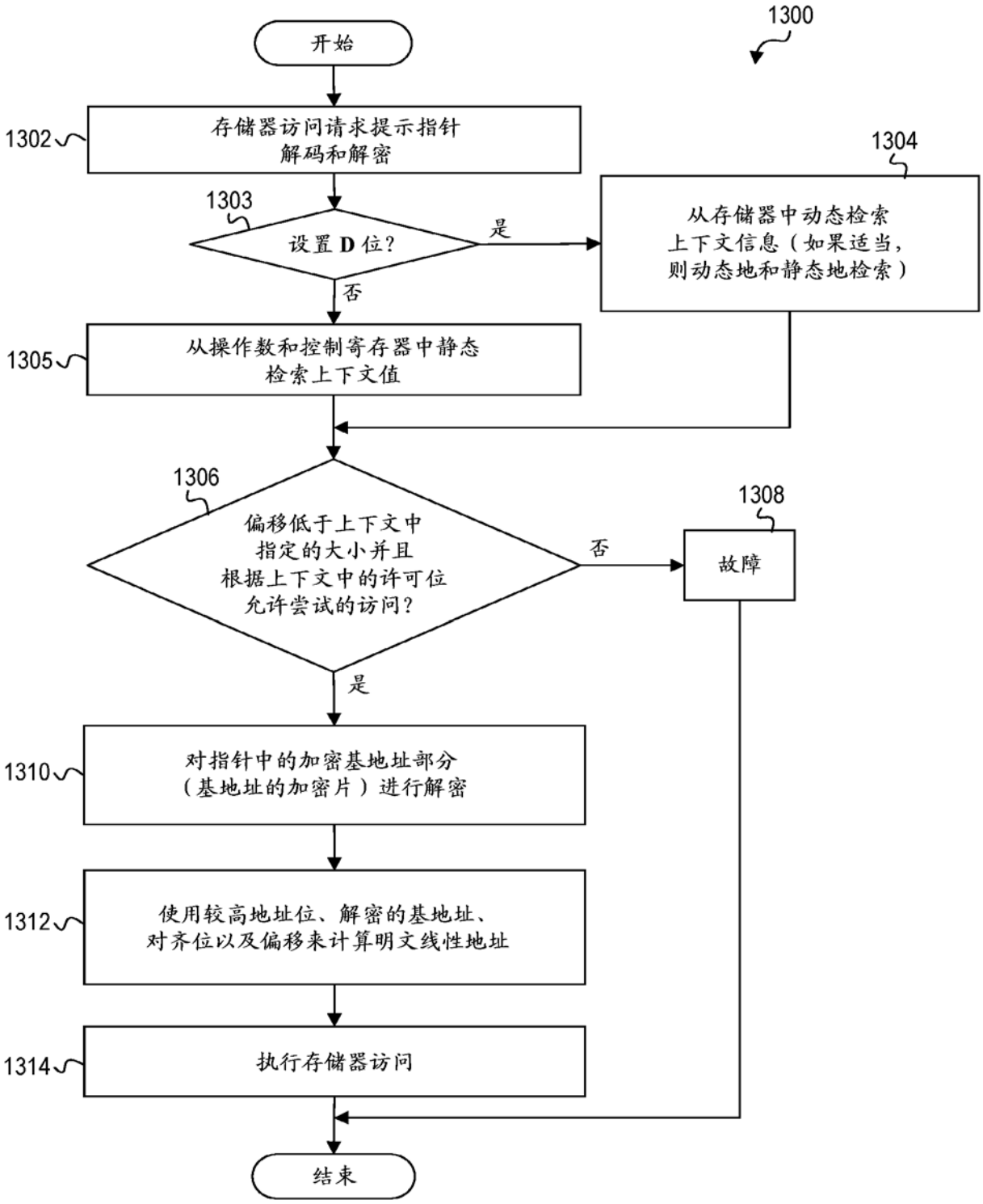

1300
开始
1302
存储器访问请求提示指针
解码和解密
1303
设置 D 位?
是
否
1304
从存储器中动态检索
上下文信息（如果适当，
则动态地和静态地检索）
1305
从操作数和控制寄存器中静态
检索上下文值
1306
偏移低于上下文中
指定的大小并且
根据上下文中的许可位
允许尝试的访问?
否
1308
故障
是
1310
对指针中的加密基地址部分
（基地址的加密片）进行解密
1312
使用较高地址位、解密的基地址、
对齐位以及偏移来计算明文线性地址
1314
执行存储器访问
结束

图 13

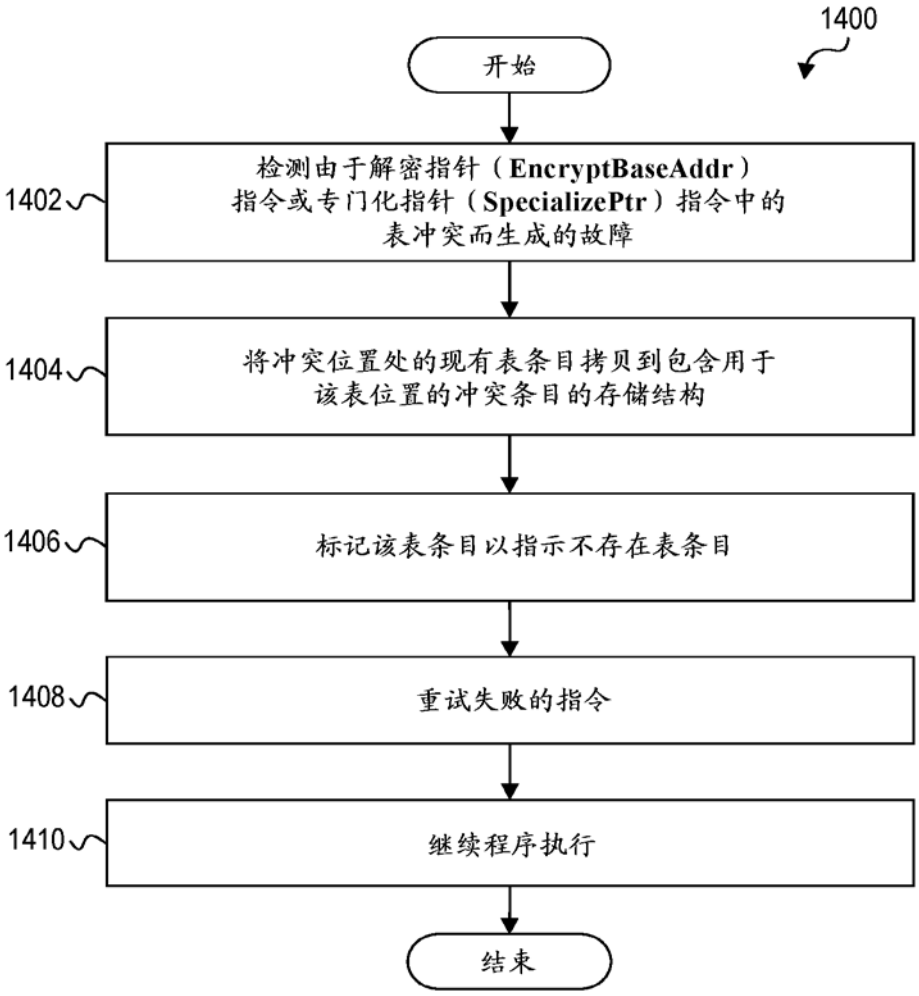

1400
开始
1402
检测由于解密指针（EncryptBaseAddr）指令或专门化指针（SpecializePtr）指令中的表冲突而生成的故障
1404
将冲突位置处的现有表条目拷贝到包含用于该表位置的冲突条目的存储结构
1406
标记该表条目以指示不存在表条目
1408
重试失败的指令
1410
继续程序执行
结束

图 14

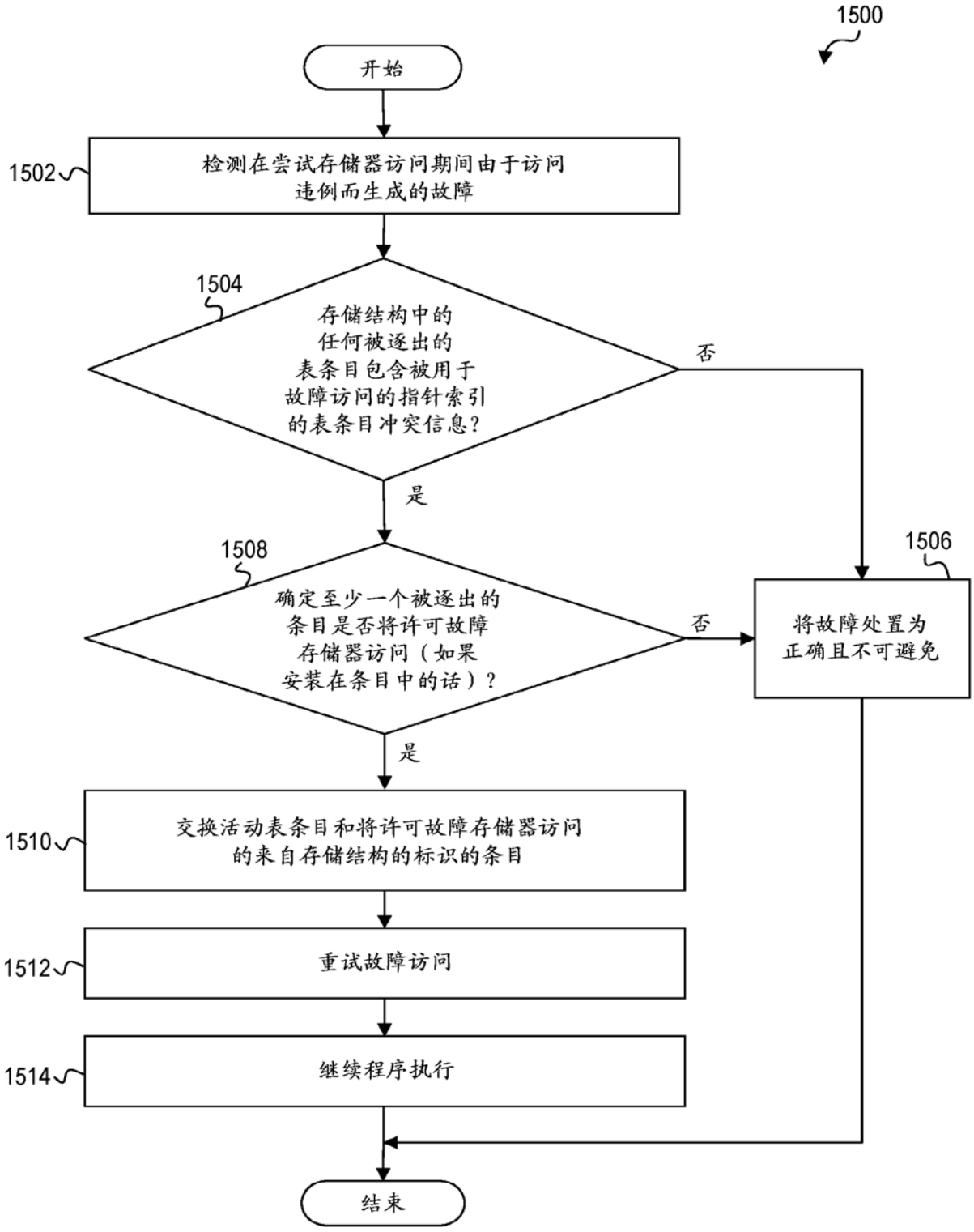

1500
开始
1502
检测在尝试存储器访问期间由于访问违例而生成的故障
1504
存储结构中的任何被逐出的表条目包含被用于故障访问的指针索引的表条目冲突信息?
否
是
1508
确定至少一个被逐出的条目是否将许可故障存储器访问（如果安装在条目中的话）?
否
1506
将故障处置为正确且不可避免
是
1510
交换活动表条目和将许可故障存储器访问的来自存储结构的标识的条目
1512
重试故障访问
1514
继续程序执行
结束

图 15

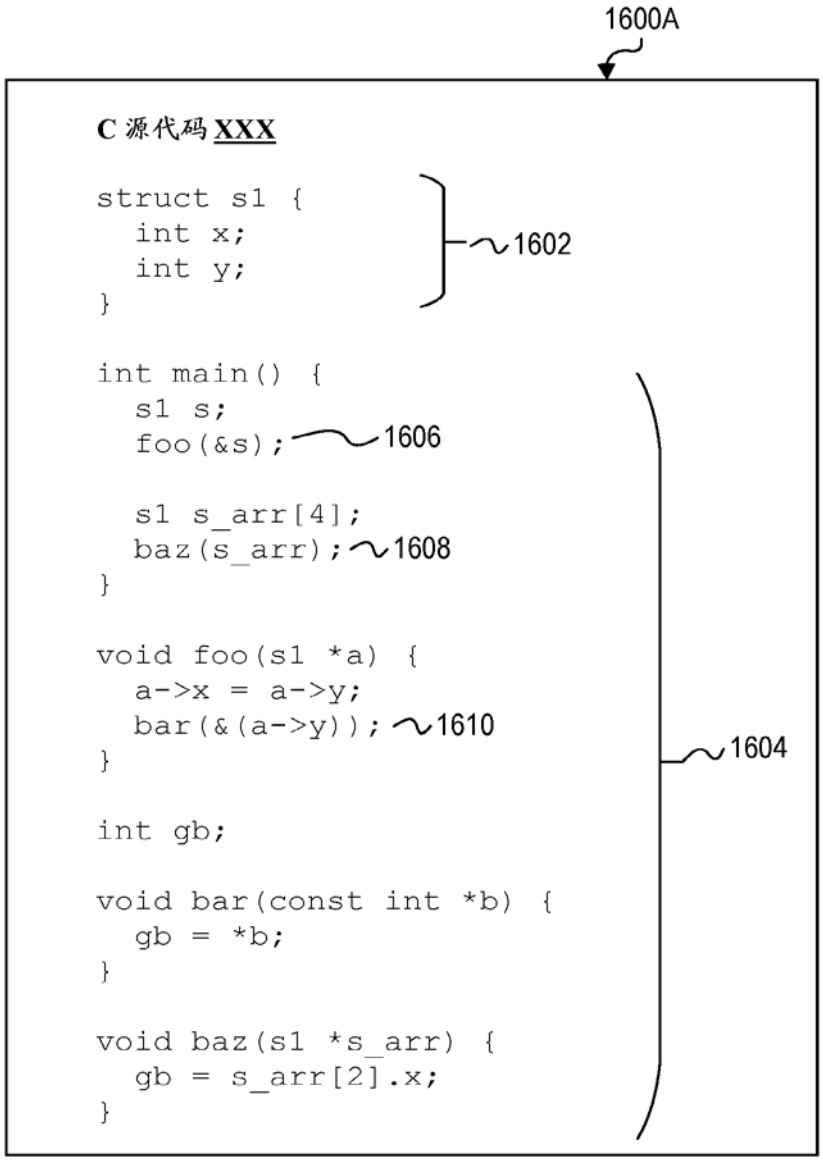

1600A
C 源代码 XXX
struct s1 {
int x;
int y;
}
1602
int main() {
s1 s;
foo(&s);
1606
s1 s_arr[4];
baz(s_arr);
1608
}
void foo(s1 *a) {
a->x = a->y;
bar(&(a->y));
1610
}
1604
int gb;
void bar(const int *b) {
gb = *b;
}
void baz(s1 *s_arr) {
gb = s_arr[2].x;
}

图 16A

1600B

具有 EBA 指令的汇编语言输出代码:

```
  sub $8, %rsp
  movabs (constant encoding context with size 8, type ID for
struct s1, RW permissions), %rax
  mov %rsp, %rdi
1612 EncryptBaseAddr $0, %rax, %rdi
  call foo
  sub $32, %rsp
  mov %rsp, %rdi
  movabs (constant encoding context with size 32, type ID for
array of size 4 with elements of type struct s1, RW
permissions), %rax
1614 EncryptBaseAddr $1, %rax, %rdi // 这将插入上下文表条目

  call baz
  xor %eax, %eax
  ret

foo:
  movabs (constant encoding context with size 8, type ID for
struct s1, RW permissions), %rcx
1618 LdEP %rcx, 4(%rdi), %eax
1620 StEP %rcx, %eax, (%rdi)
  mov (constant encoding context with size 4, type ID for
int, RO permissions), %rdx
  add $4, %rdi
1622 SpecializePtr $0, %rdx, %rcx, %rdi
  call bar
  ret

bar:
  mov (constant encoding context with size 4, type ID for
int, RW permissions), %rcx
  mov (constant encoding context with size 4, type ID for
int, RO permissions), %rdx
1624 LdEP %rdx, (%rdi), %eax
1626 StEP %rcx, %eax, gb
  ret

baz:
  mov 16(%rdi), %eax // 这将动态加载适当的上下文表条目

  mov (constant encoding context with size 4, type ID for
int, RW permissions), %rcx
1628 StEP %rcx, %eax, gb
```

图 16B

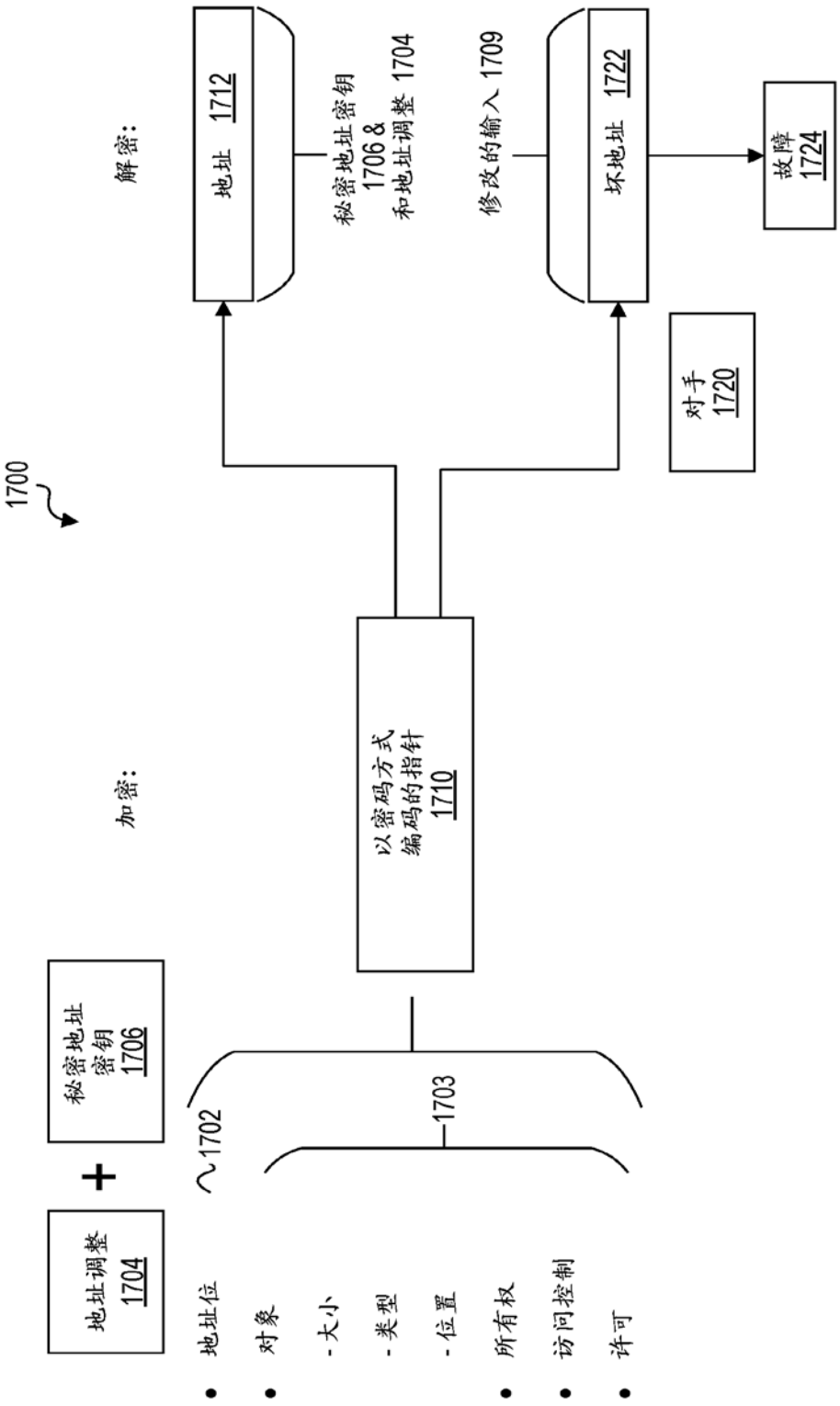

1700
加密:
解密:
地址调整 1704
+
秘密地址密钥 1706
~1702
• 地址位
• 对象
- 大小
- 类型
- 位置
• 所有权
• 访问控制
• 许可
1703
以密码方式编码的指针 1710
地址 1712
秘密地址密钥 1706 & 和地址调整 1704
修改的输入 1709
坏地址 1722
故障 1724
对手 1720

图 17

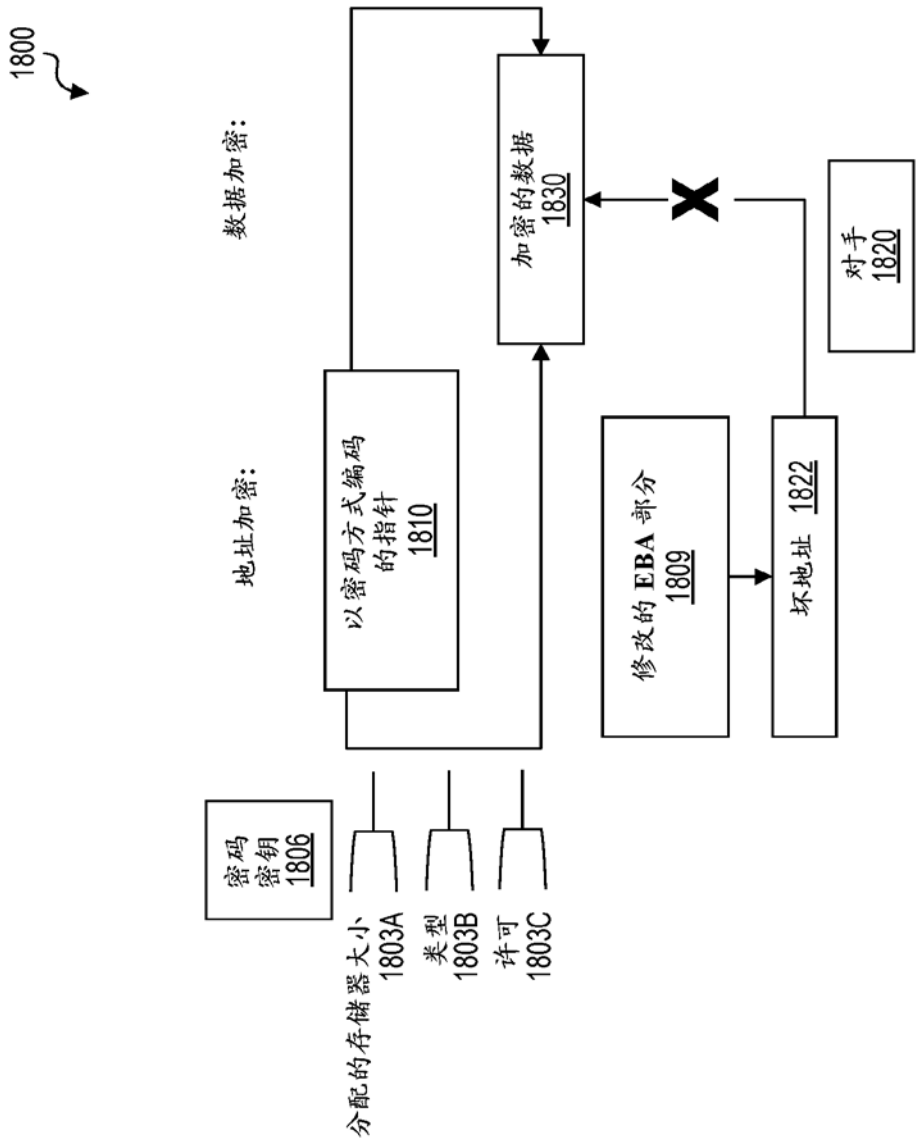

1800
数据加密：
地址加密：
加密的数据
1830
对手
1820
以密码方式编码
的指针
1810
修改的 EBA 部分
1809
坏地址 1822
密码
密钥
1806
分配的存储器大小
1803A
类型
1803B
许可
1803C

图 18

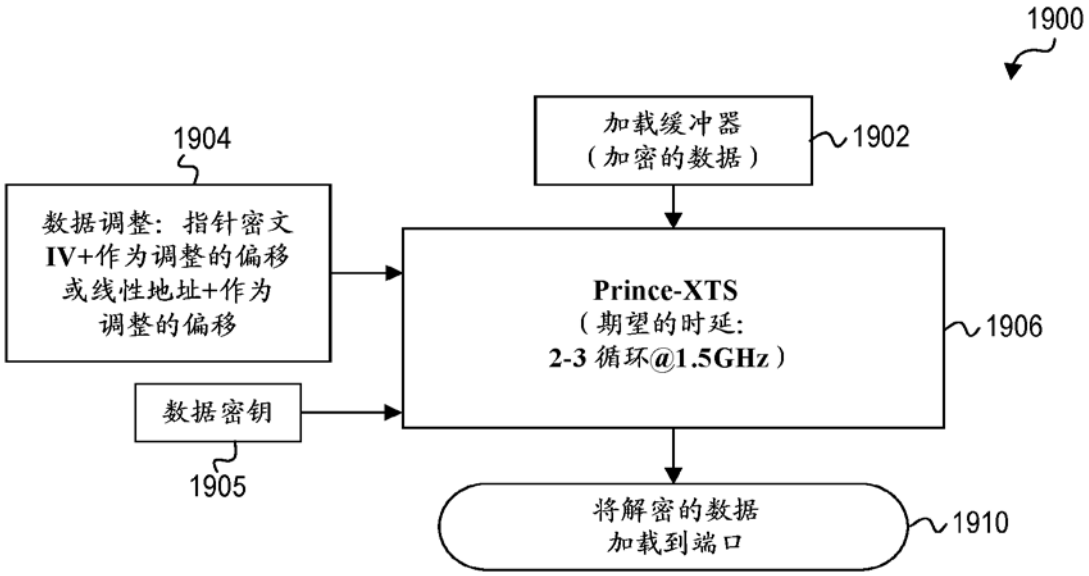

1900
加载缓冲器
（加密的数据）
1902
1904
数据调整：指针密文
IV+作为调整的偏移
或线性地址+作为
调整的偏移
Prince-XTS
（期望的时延：
2-3 循环@1.5GHz）
1906
数据密钥
1905
将解密的数据
加载到端口
1910

图 19

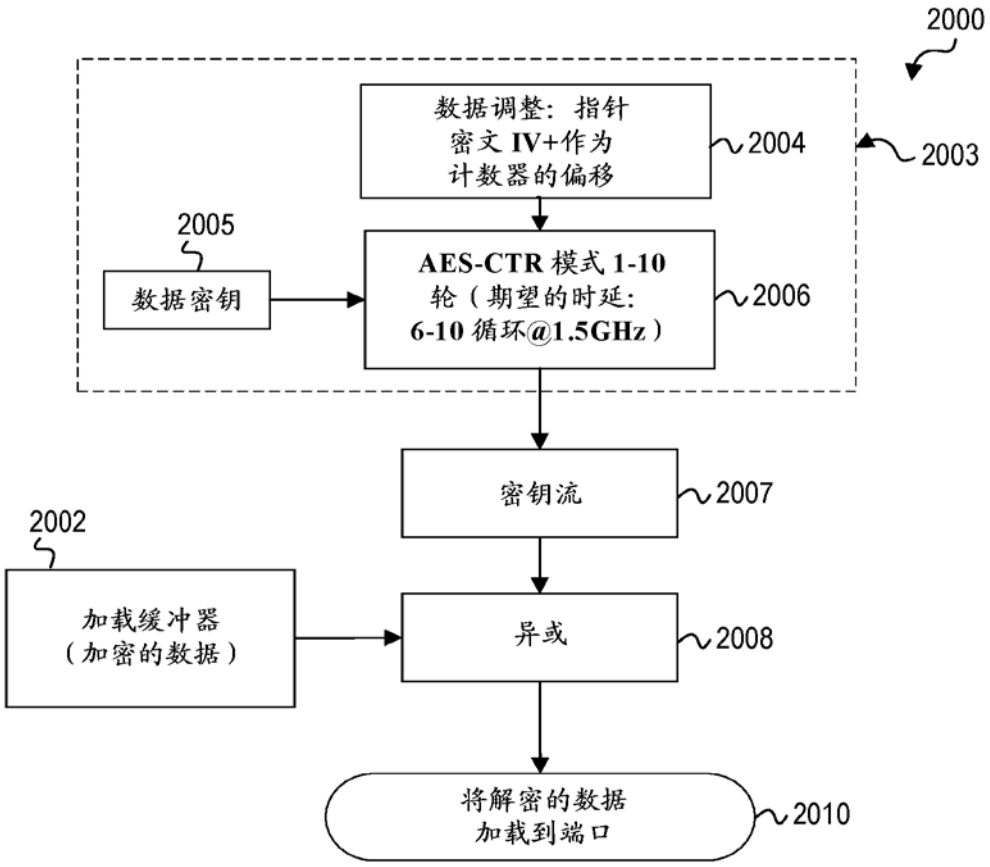

2000
数据调整：指针
密文 IV+作为
计数器的偏移
2004
2003
2005
数据密钥
AES-CTR 模式 1-10
轮（期望的时延：
6-10 循环@1.5GHz）
2006
密钥流
2007
2002
加载缓冲器
（加密的数据）
异或
2008
将解密的数据
加载到端口
2010

图 20

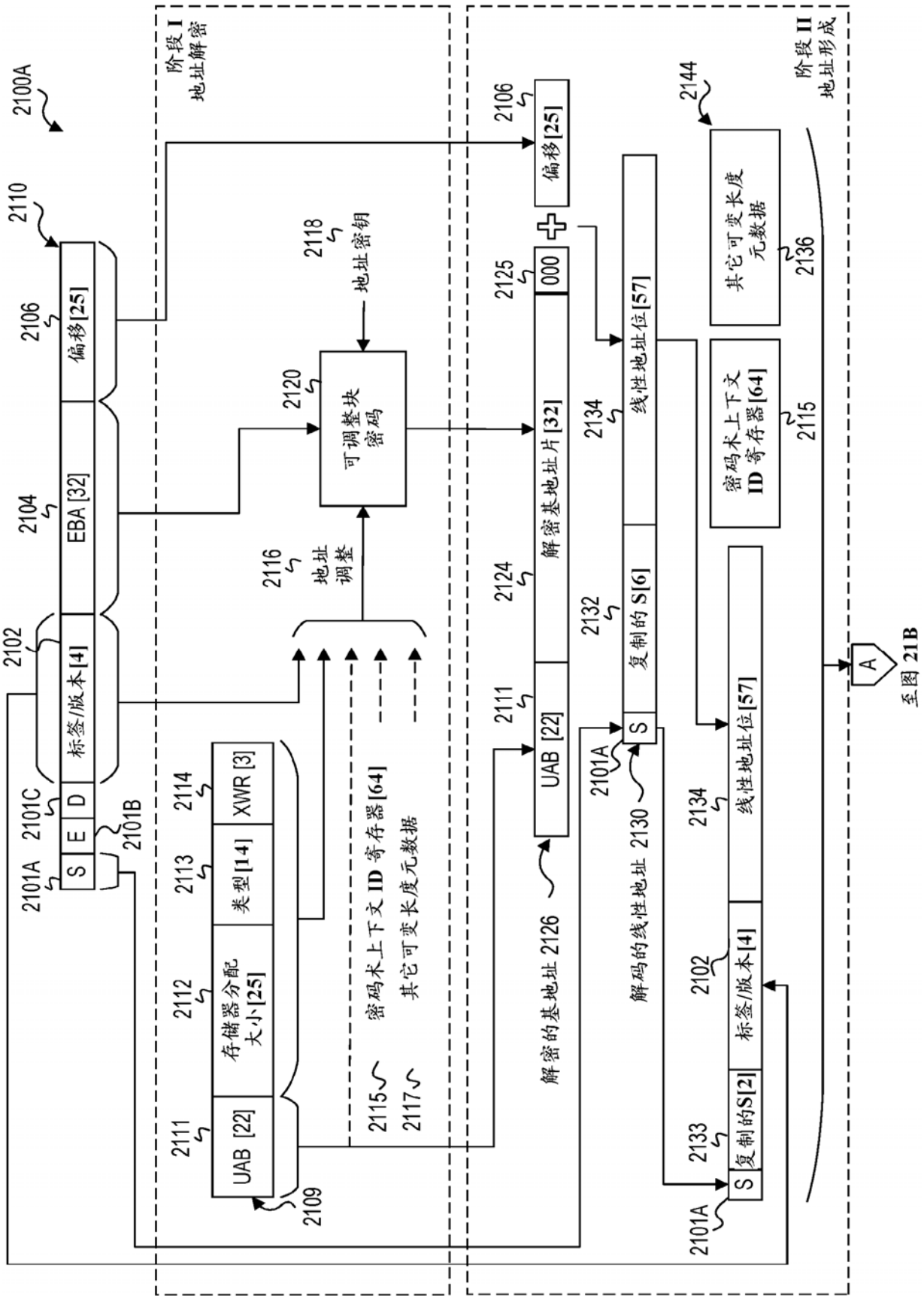

2100A
阶段 I
地址解密
阶段 II
地址形成
2110
2106
偏移[25]
2104
EBA [32]
2102
标签/版本[4]
2101A
2101C
S
E
D
2101B
2118
地址密钥
2120
可调整块密码
2116
地址调整
2111
UAB [22]
2112
存储器分配大小[25]
2113
类型[14]
2114
XWR [3]
2109
2115
密码术上下文 ID 寄存器[64]
2117
其它可变长度元数据
2125
000
2124
解密基地址片[32]
2132
复制的 S[6]
2134
线性地址位[57]
2144
其它可变长度元数据
2136
密码术上下文 ID 寄存器[64]
2115
解密的基地址 2126
2101A
2130
解码的线性地址
2133
复制的S[2]
A
至图 21B

图 21A

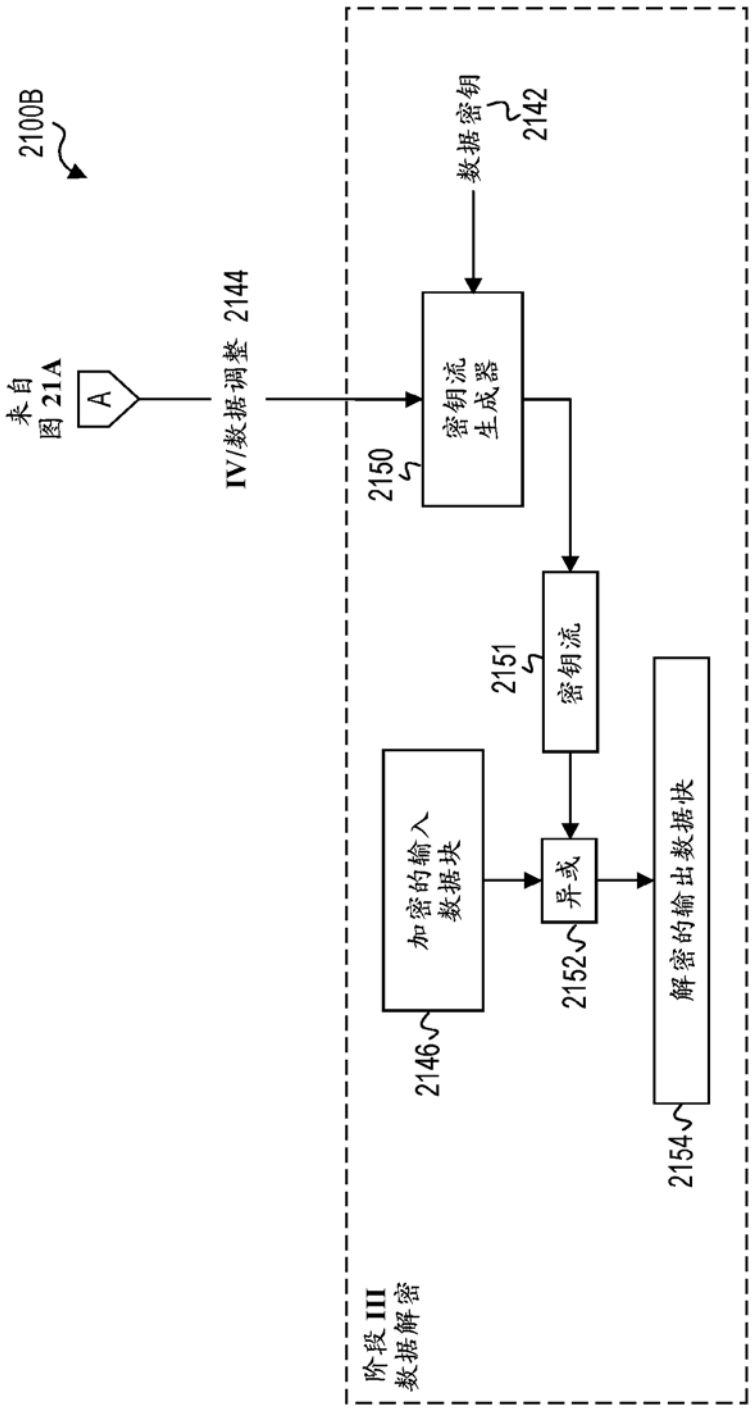

2100B
来自
图 21A
A
IV/数据调整 2144
数据密钥
2142
密钥流
生成器
2150
密钥流
2151
加密的输入
数据块
2146
异或
2152
解密的输出数据块
2154
阶段 III
数据解密

图 21B

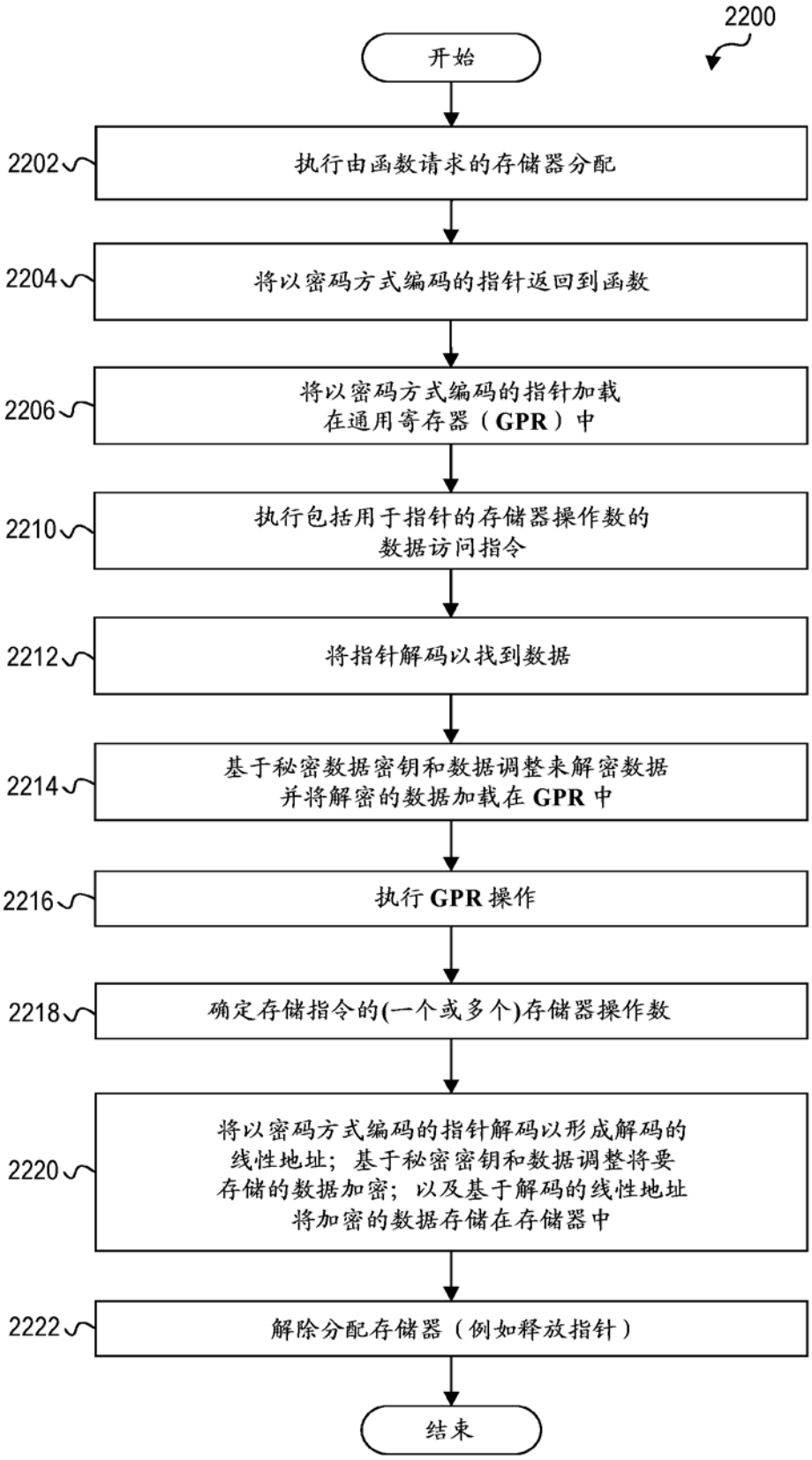

2200
开始
2202
执行由函数请求的存储器分配
2204
将以密码方式编码的指针返回到函数
2206
将以密码方式编码的指针加载
在通用寄存器（GPR）中
2210
执行包括用于指针的存储器操作数的
数据访问指令
2212
将指针解码以找到数据
2214
基于秘密数据密钥和数据调整来解密数据
并将解密的数据加载在 GPR 中
2216
执行 GPR 操作
2218
确定存储指令的(一个或多个)存储器操作数
2220
将以密码方式编码的指针解码以形成解码的
线性地址；基于秘密密钥和数据调整将要
存储的数据加密；以及基于解码的线性地址
将加密的数据存储在存储器中
2222
解除分配存储器（例如释放指针）
结束

图 22

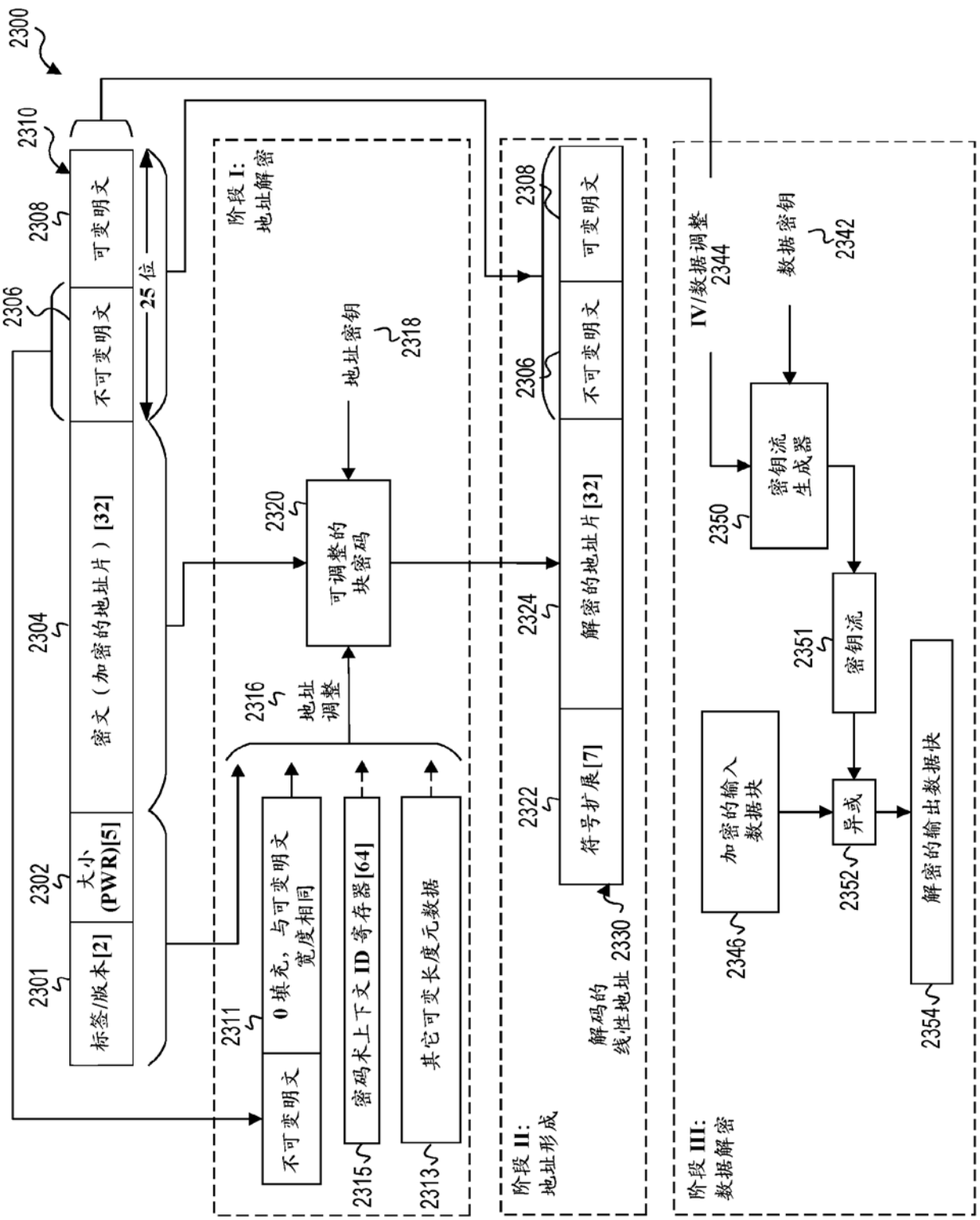

2300
2310
2308
可变明文
2306
不可变明文
25 位
2304
密文（加密的地址片）[32]
2302
大小 (PWR)[5]
2301
标签/版本[2]
阶段 I: 地址解密
地址密钥
2318
2320
可调整的块密码
2316
地址调整
2311
0 填充，与可变明文宽度相同
不可变明文
密码术上下文 ID 寄存器[64]
2315
其它可变长度元数据
2313
阶段 II: 地址形成
2308
可变明文
2306
不可变明文
2324
解密的地址片[32]
2322
符号扩展[7]
解码的线性地址 2330
IV/数据调整 2344
数据密钥 2342
2350
密钥流生成器
2351
密钥流
加密的输入数据块
2346
异或
2352
解密的输出数据块
2354
阶段 III: 数据解密

图 23

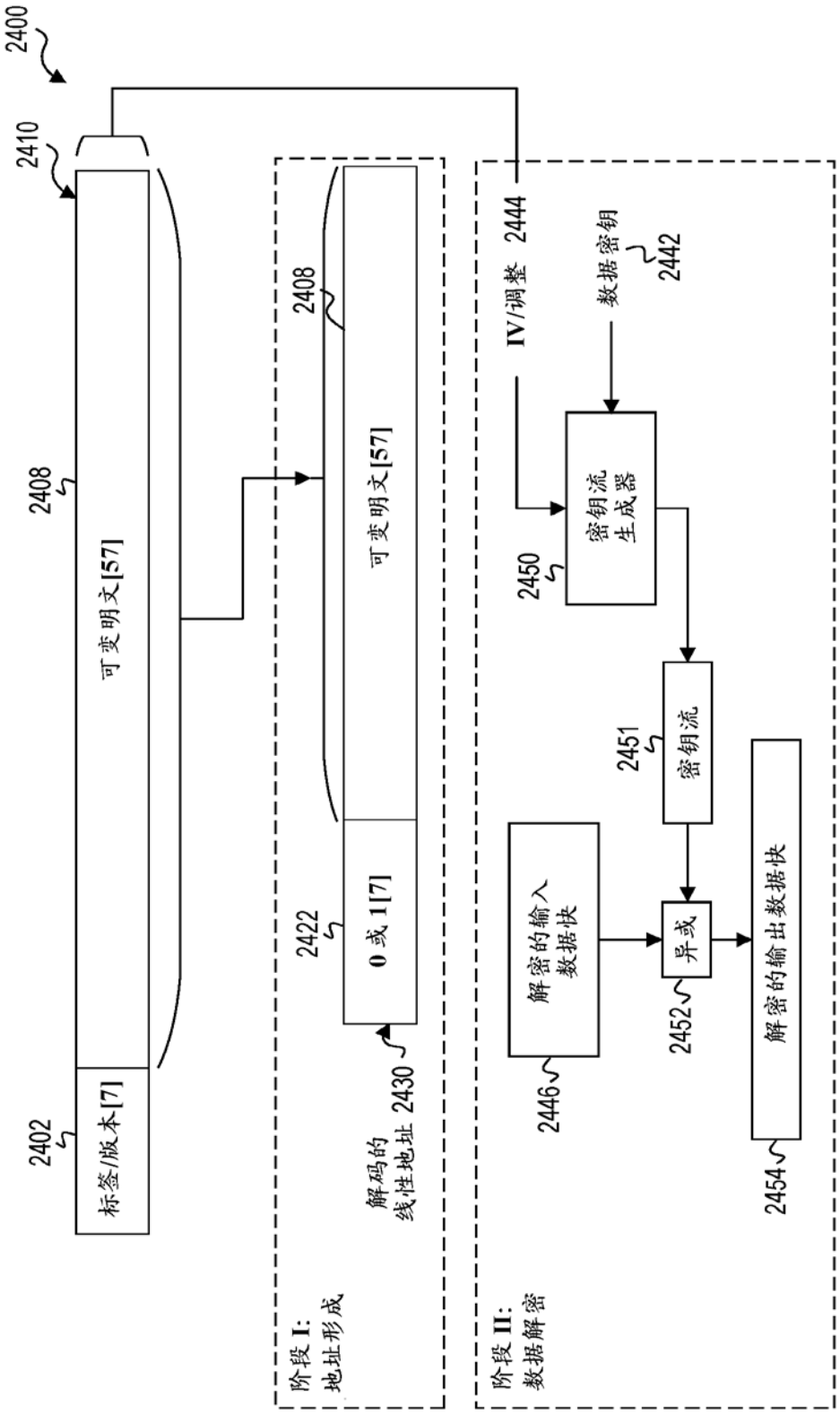

2400
2410
2408
可变明文[57]
2402
标签/版本[7]
2408
可变明文[57]
2422
0 或 1[7]
解码的
线性地址 2430
阶段 I:
地址形成
IV/调整 2444
数据密钥
2442
2450
密钥流
生成器
2451
密钥流
解密的输入
数据块
2446
2452
异或
解密的输出数据块
2454
阶段 II:
数据解密

图 24

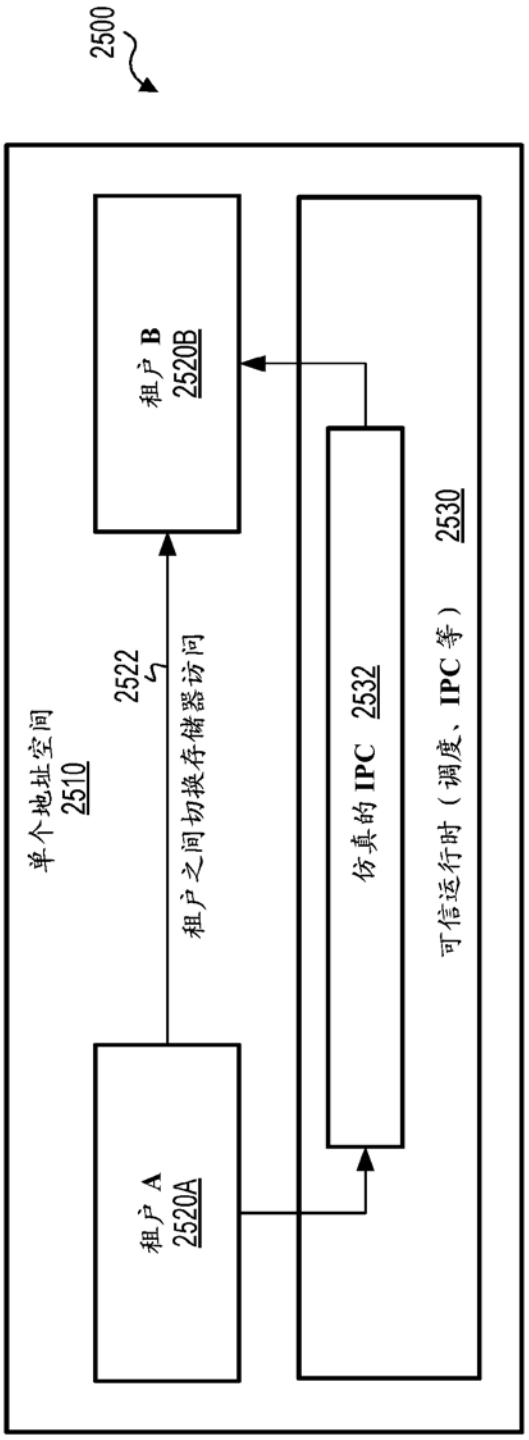

2500
单个地址空间
2510
租户 A
2520A
租户 B
2520B
2522
租户之间切换存储器访问
仿真的 IPC 2532
可信运行时（调度、IPC 等） 2530

图 25

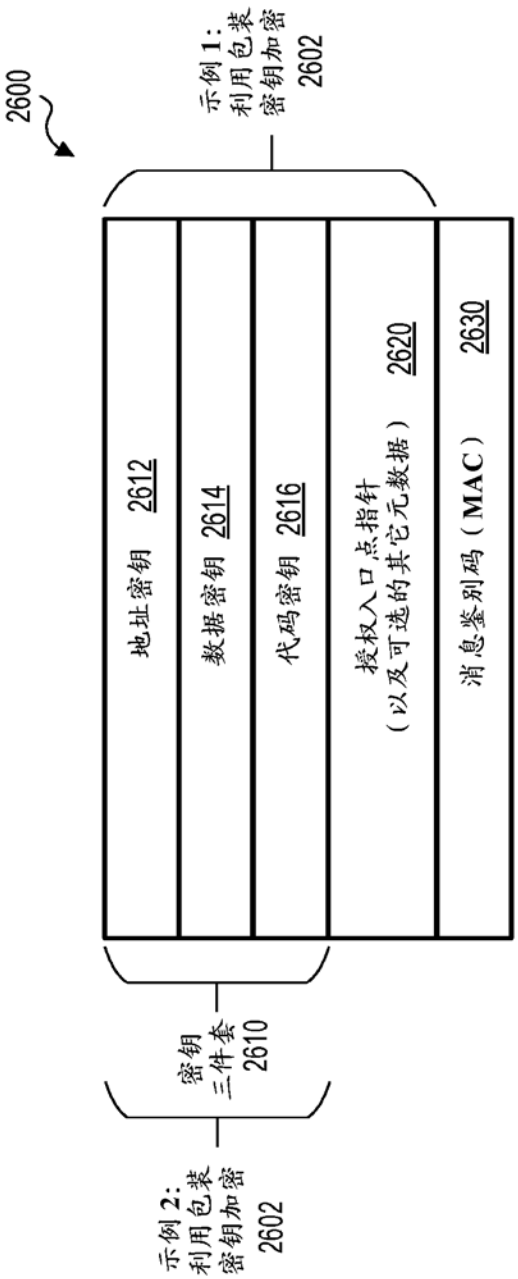

2600
示例1：
利用包裹
密钥加密
2602
地址密钥 2612
数据密钥 2614
代码密钥 2616
授权入口点指针
（以及可选的其它元数据） 2620
消息鉴别码（MAC） 2630
密钥
三件套
2610
示例2：
利用包裹
密钥加密
2602

图 26

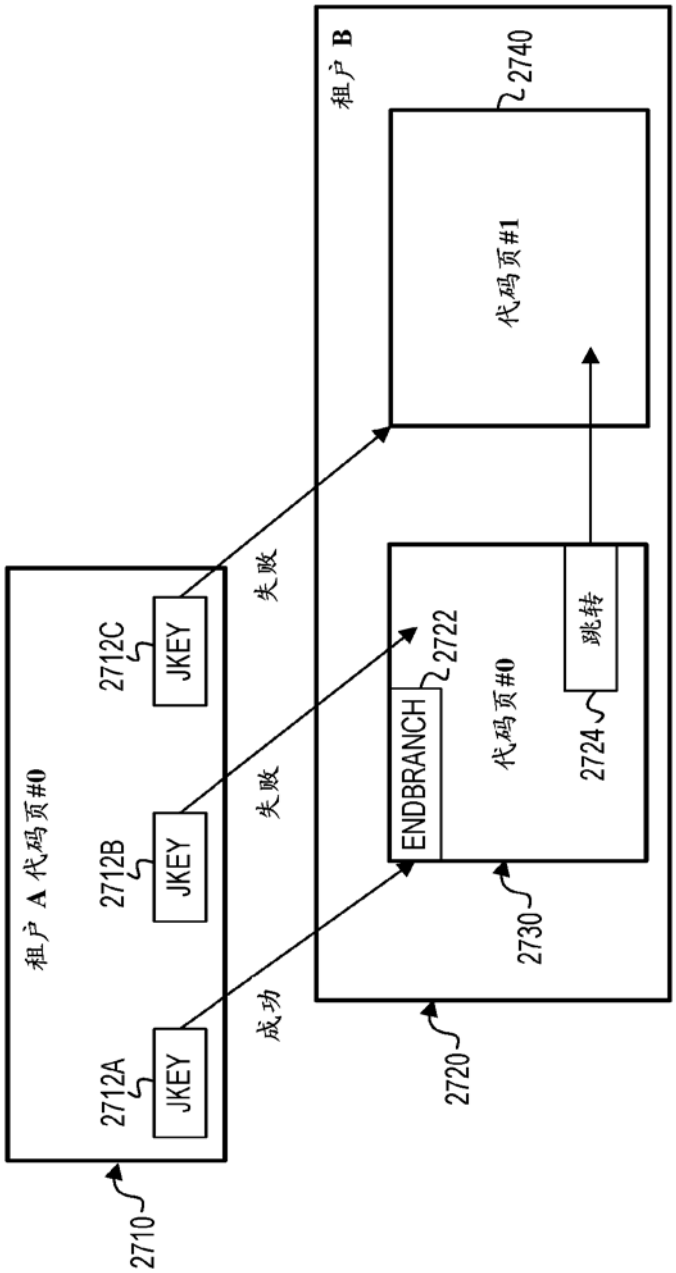

租户 B
2740
代码页#1
2722
ENDBRANCH
代码页#0
跳转
2724
2730
2720
租户 A 代码页#0
2712C
JKEY
2712B
JKEY
2712A
JKEY
失败
失败
成功
2710

图 27

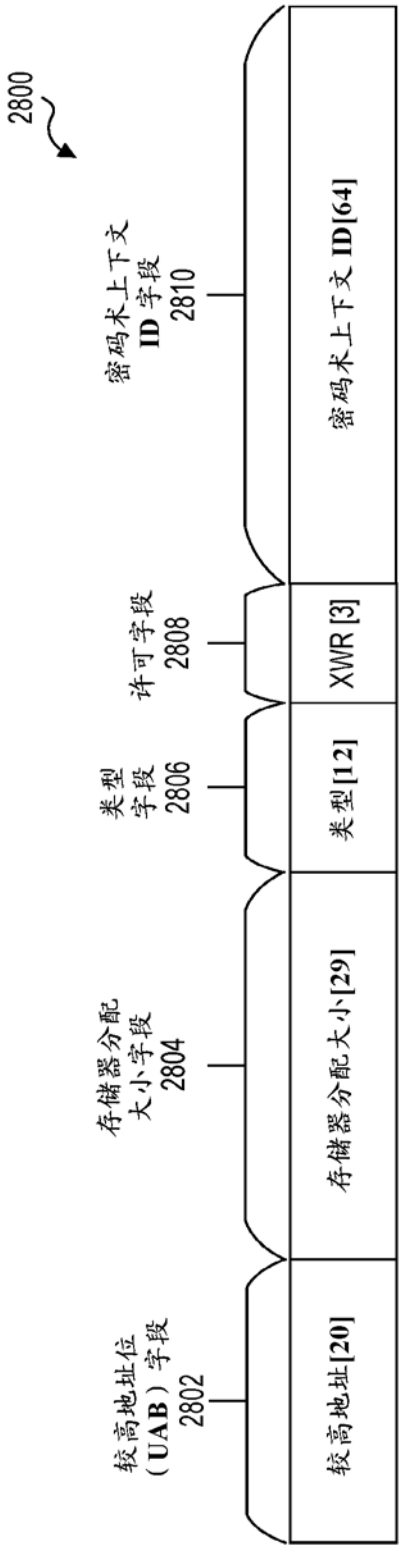

2800
较高地址位（UAB）字段 2802
存储器分配大小字段 2804
类型字段 2806
许可字段 2808
密码术上下文 ID 字段 2810
较高地址[20]
存储器分配大小[29]
类型[12]
XWR [3]
密码术上下文 ID[64]

图 28

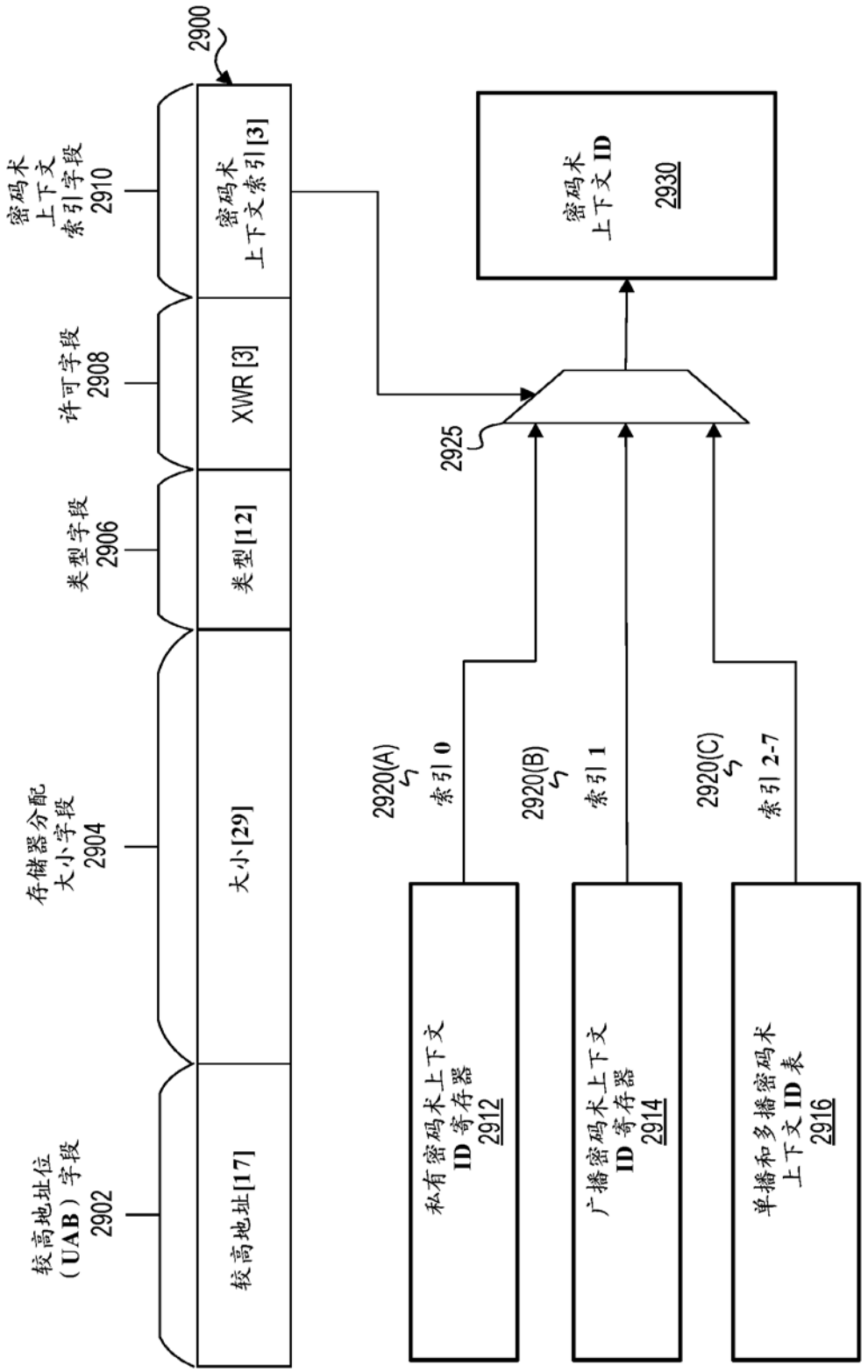

2900
较高地址位（UAB）字段 2902
较高地址[17]
存储器分配大小字段 2904
大小[29]
类型字段 2906
类型[12]
许可字段 2908
XWR [3]
密码术上下文索引字段 2910
密码术上下文索引[3]
2925
密码术上下文 ID 2930
2920(A) 索引 0
私有密码术上下文 ID 寄存器 2912
2920(B) 索引 1
广播密码术上下文 ID 寄存器 2914
2920(C) 索引 2-7
单播和多播密码术上下文 ID 表 2916

图 29

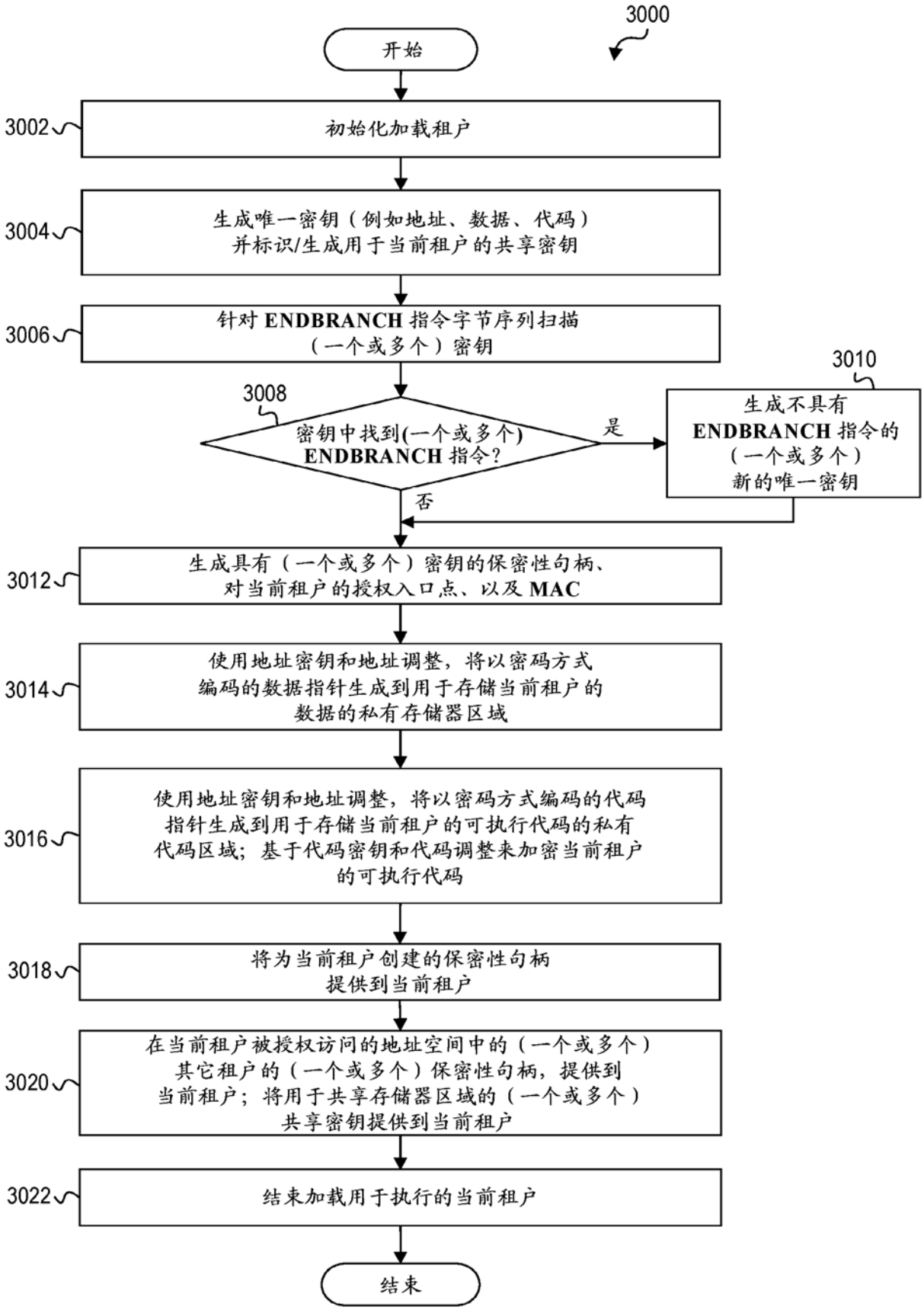

3000
开始
3002
初始化加载租户
3004
生成唯一密钥（例如地址、数据、代码）
并标识/生成用于当前租户的共享密钥
3006
针对 ENDBRANCH 指令字节序列扫描
（一个或多个）密钥
3008
密钥中找到(一个或多个)
ENDBRANCH 指令?
是
否
3010
生成不具有
ENDBRANCH 指令的
（一个或多个）
新的唯一密钥
3012
生成具有（一个或多个）密钥的保密性句柄、
对当前租户的授权入口点、以及 MAC
3014
使用地址密钥和地址调整，将以密码方式
编码的数据指针生成到用于存储当前租户的
数据的私有存储器区域
3016
使用地址密钥和地址调整，将以密码方式编码的代码
指针生成到用于存储当前租户的可执行代码的私有
代码区域；基于代码密钥和代码调整来加密当前租户
的可执行代码
3018
将为当前租户创建的保密性句柄
提供到当前租户
3020
在当前租户被授权访问的地址空间中的（一个或多个）
其它租户的（一个或多个）保密性句柄，提供到
当前租户；将用于共享存储器区域的（一个或多个）
共享密钥提供到当前租户
3022
结束加载用于执行的当前租户
结束

图 30

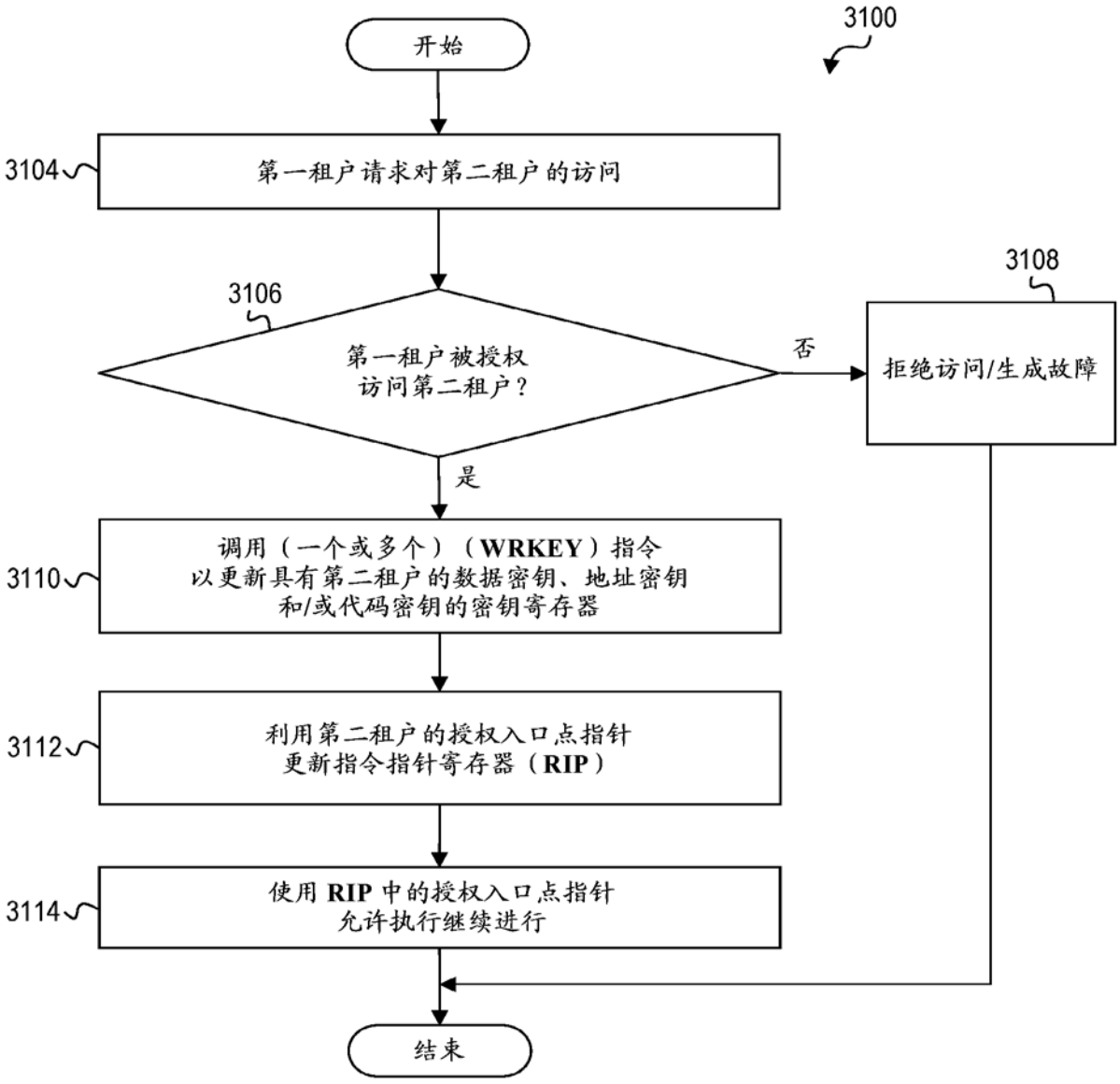

3100
开始
3104
第一租户请求对第二租户的访问
3106
第一租户被授权
访问第二租户？
否
3108
拒绝访问/生成故障
是
3110
调用（一个或多个）（WRKEY）指令
以更新具有第二租户的数据密钥、地址密钥
和/或代码密钥的密钥寄存器
3112
利用第二租户的授权入口点指针
更新指令指针寄存器（RIP）
3114
使用 RIP 中的授权入口点指针
允许执行继续进行
结束

图 31

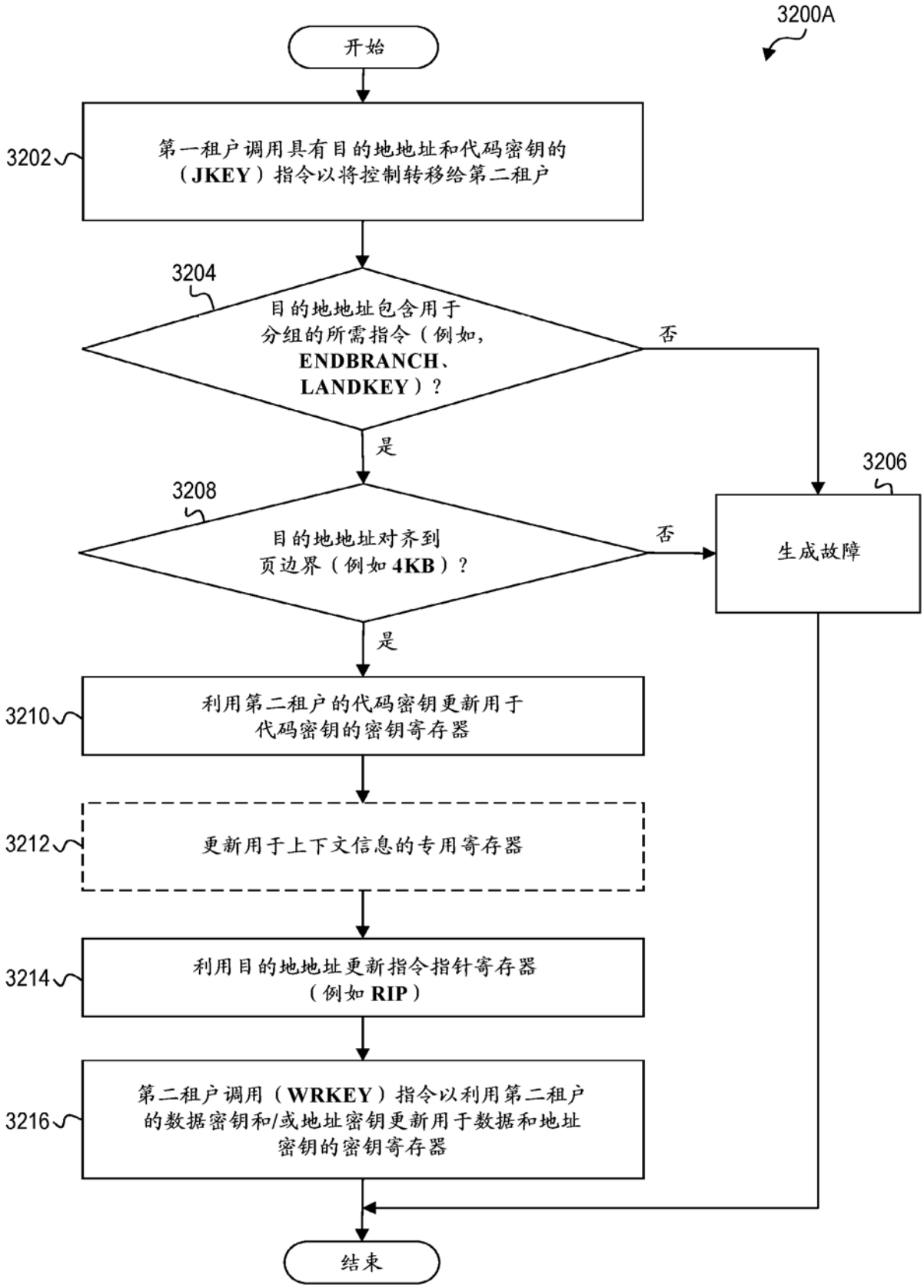

3200A
开始
3202
第一租户调用具有目的地地址和代码密钥的（JKEY）指令以将控制转移给第二租户
3204
目的地地址包含用于分组的所需指令（例如，ENDBRANCH、LANDKEY）？
否
是
3206
生成故障
3208
目的地地址对齐到页边界（例如4KB）？
否
是
3210
利用第二租户的代码密钥更新用于代码密钥的密钥寄存器
3212
更新用于上下文信息的专用寄存器
3214
利用目的地地址更新指令指针寄存器（例如RIP）
3216
第二租户调用（WRKEY）指令以利用第二租户的数据密钥和/或地址密钥更新用于数据和地址密钥的密钥寄存器
结束

图 32A

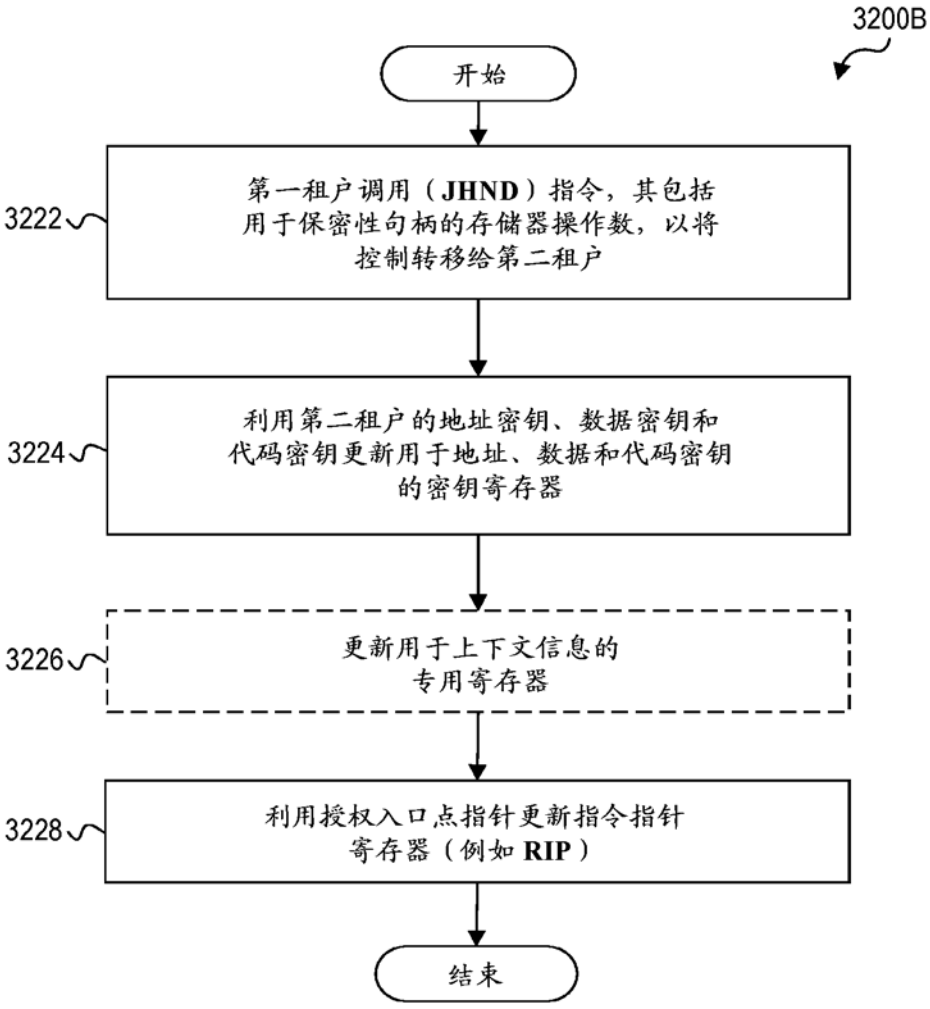

3200B
开始
3222
第一租户调用（JHND）指令，其包括用于保密性句柄的存储器操作数，以将控制转移给第二租户
3224
利用第二租户的地址密钥、数据密钥和代码密钥更新用于地址、数据和代码密钥的密钥寄存器
3226
更新用于上下文信息的专用寄存器
3228
利用授权入口点指针更新指令指针寄存器（例如 RIP）
结束

图 32B

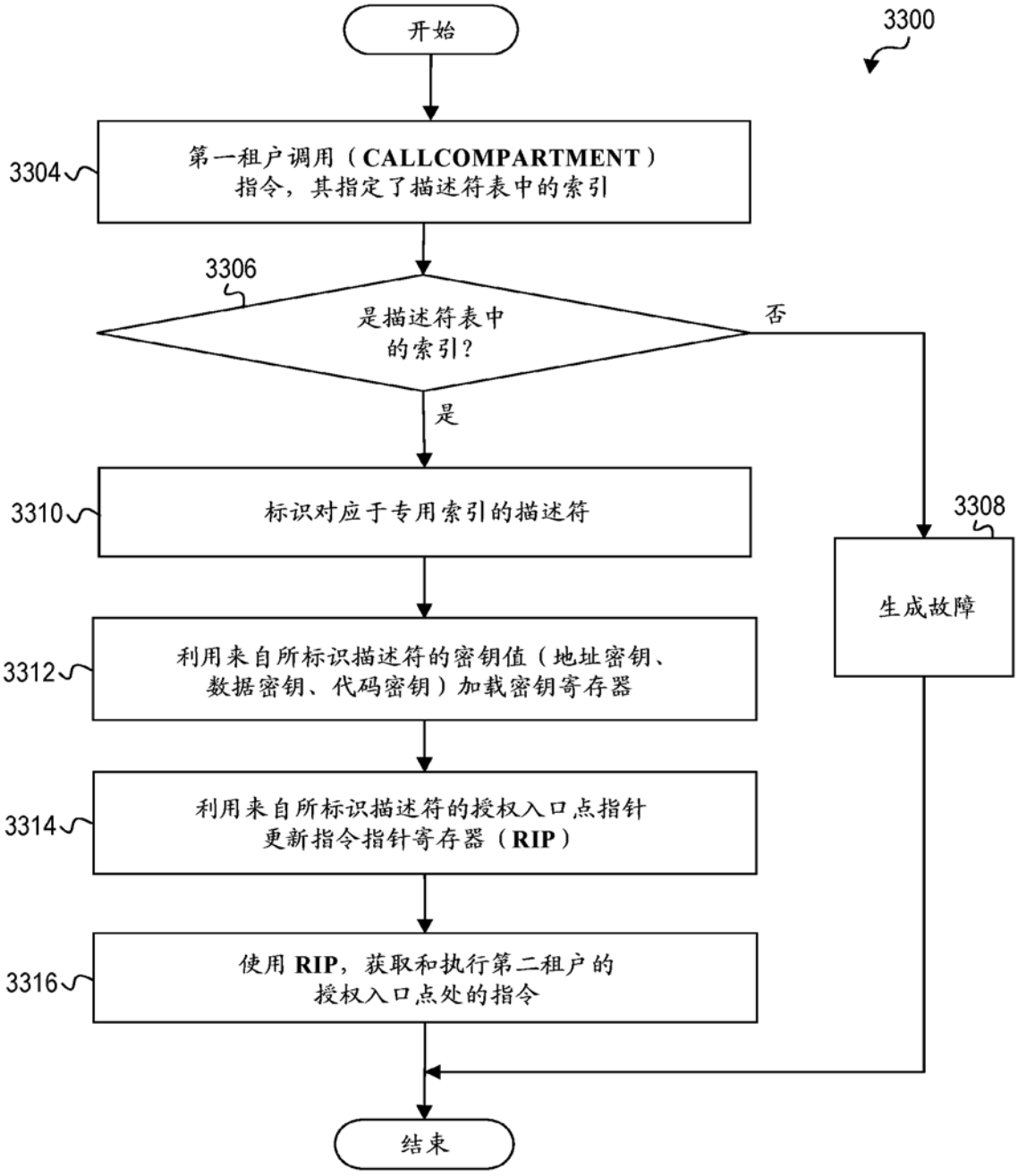

开始
3300
3304
第一租户调用（CALLCOMPARTMENT）指令，其指定了描述符表中的索引
3306
是描述符表中的索引？
否
是
3308
生成故障
3310
标识对应于专用索引的描述符
3312
利用来自所标识描述符的密钥值（地址密钥、数据密钥、代码密钥）加载密钥寄存器
3314
利用来自所标识描述符的授权入口点指针更新指令指针寄存器（RIP）
3316
使用 RIP，获取和执行第二租户的授权入口点处的指令
结束

图 33

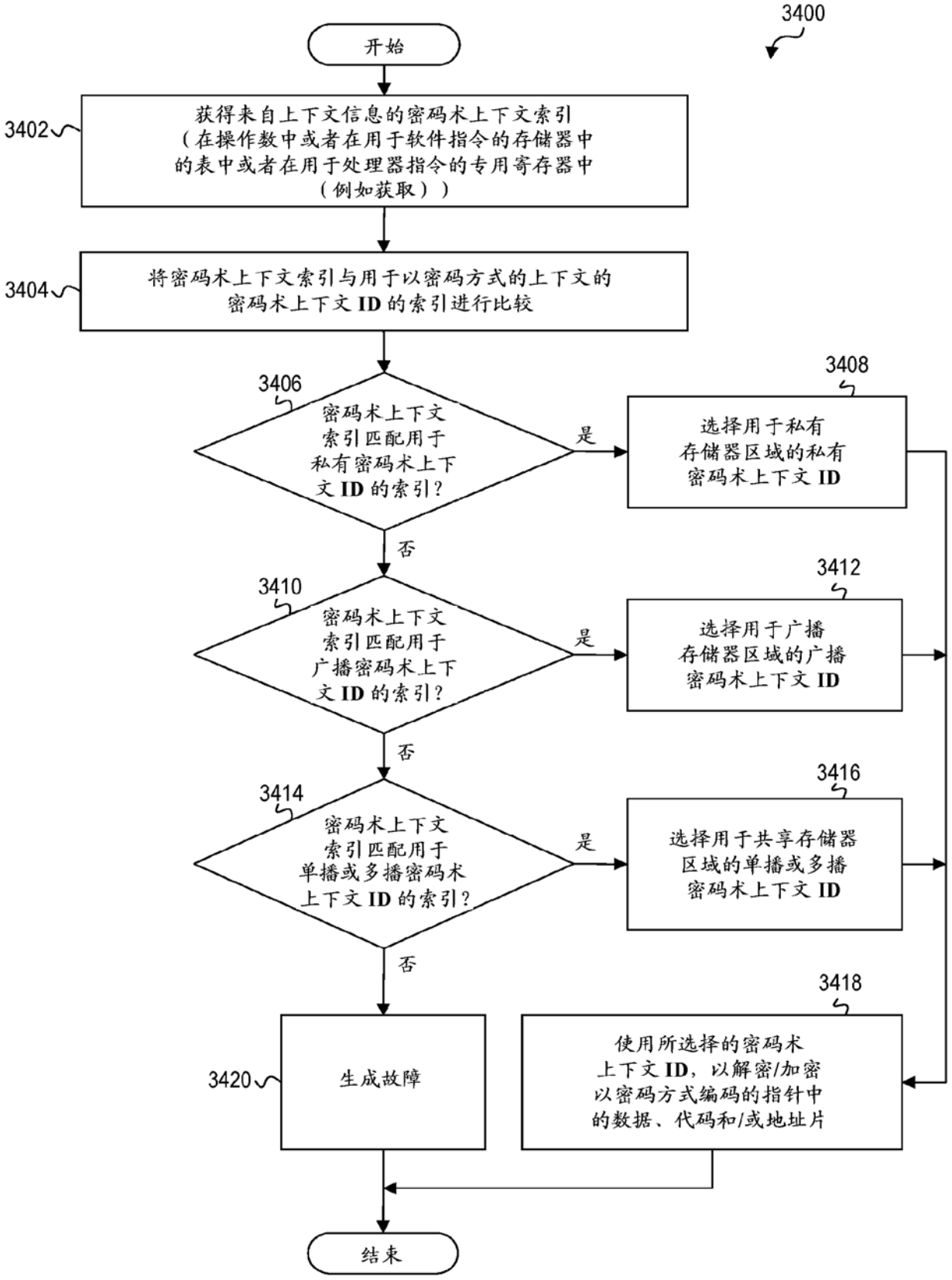

3400
开始
3402
获得来自上下文信息的密码术上下文索引（在操作数中或者在用于软件指令的存储器中的表中或者在用于处理器指令的专用寄存器中（例如获取））
3404
将密码术上下文索引与用于以密码方式的上下文的密码术上下文 ID 的索引进行比较
3406
密码术上下文索引匹配用于私有密码术上下文 ID 的索引？
是
3408
选择用于私有存储器区域的私有密码术上下文 ID
否
3410
密码术上下文索引匹配用于广播密码术上下文 ID 的索引？
是
3412
选择用于广播存储器区域的广播密码术上下文 ID
否
3414
密码术上下文索引匹配用于单播或多播密码术上下文 ID 的索引？
是
3416
选择用于共享存储器区域的单播或多播密码术上下文 ID
否
3418
使用所选择的密码术上下文 ID，以解密/加密以密码方式编码的指针中的数据、代码和/或地址片
3420
生成故障
结束

图 34

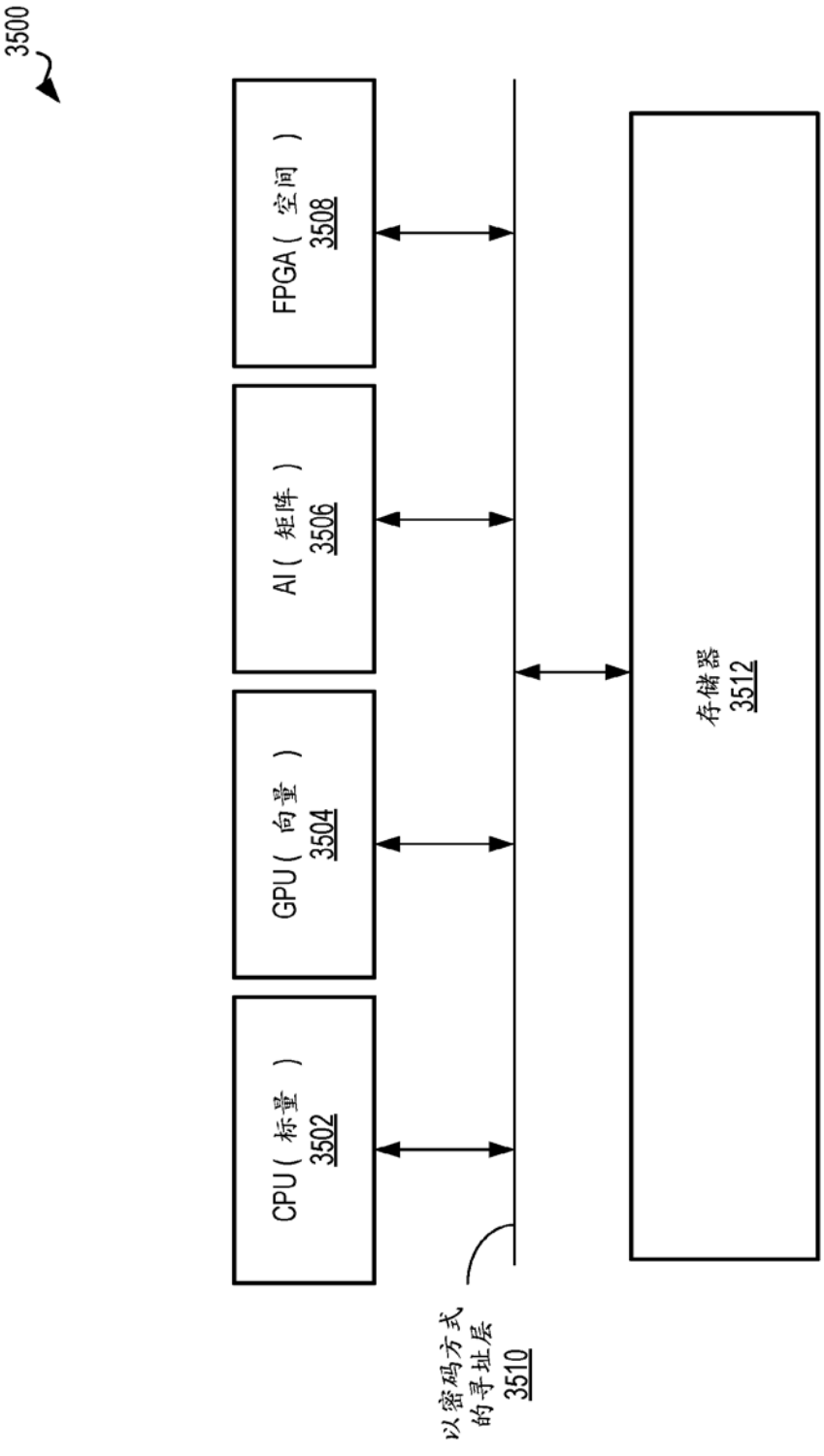

3500
FPGA（空间）
3508
AI（矩阵）
3506
GPU（向量）
3504
CPU（标量）
3502
存储器
3512
以密码方式的寻址层
3510

图 35

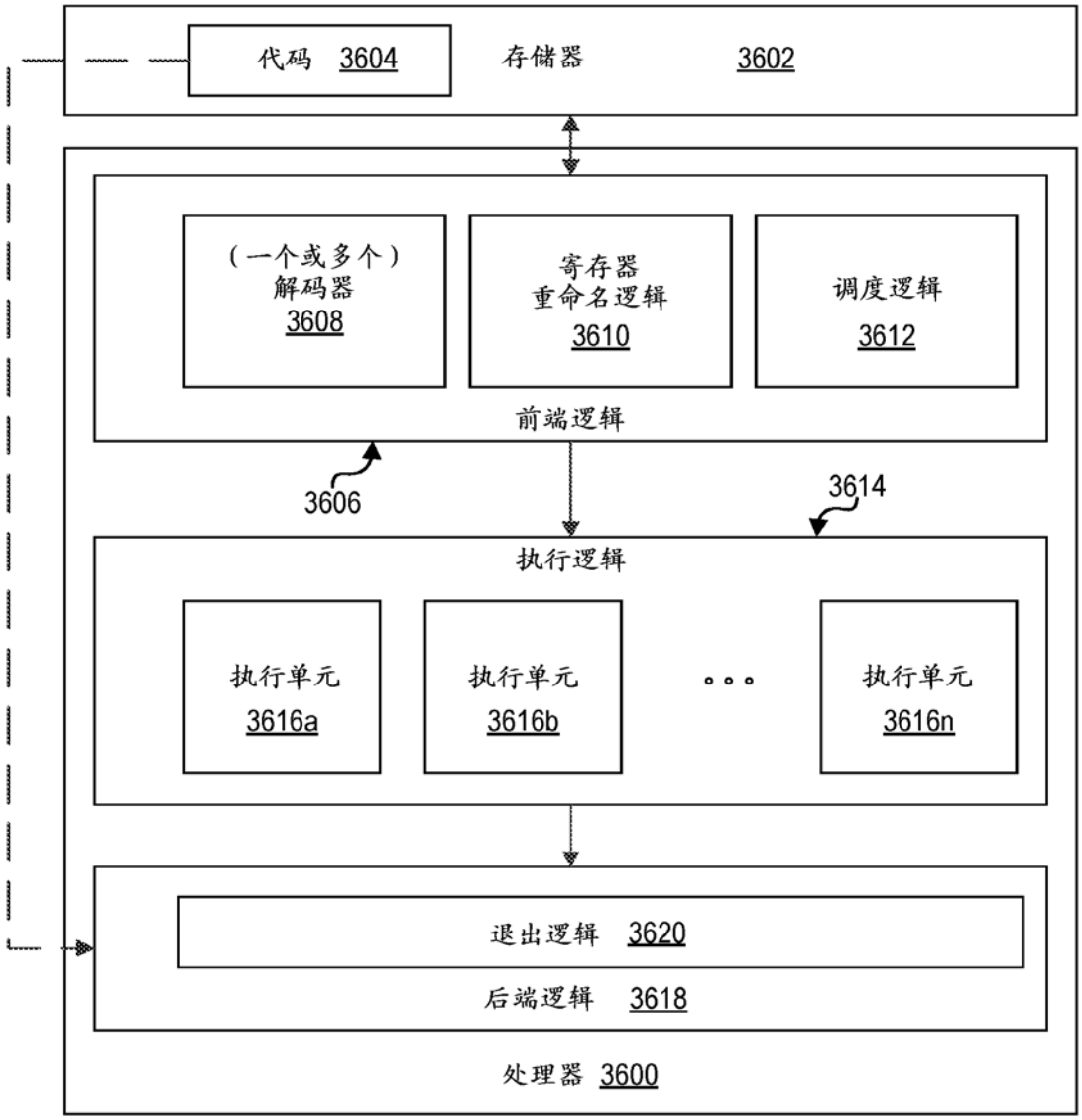

代码 3604
存储器 3602
(一个或多个)
解码器
3608
寄存器
重命名逻辑
3610
调度逻辑
3612
前端逻辑
3606
3614
执行逻辑
执行单元
3616a
执行单元
3616b
执行单元
3616n
退出逻辑 3620
后端逻辑 3618
处理器 3600

图 36

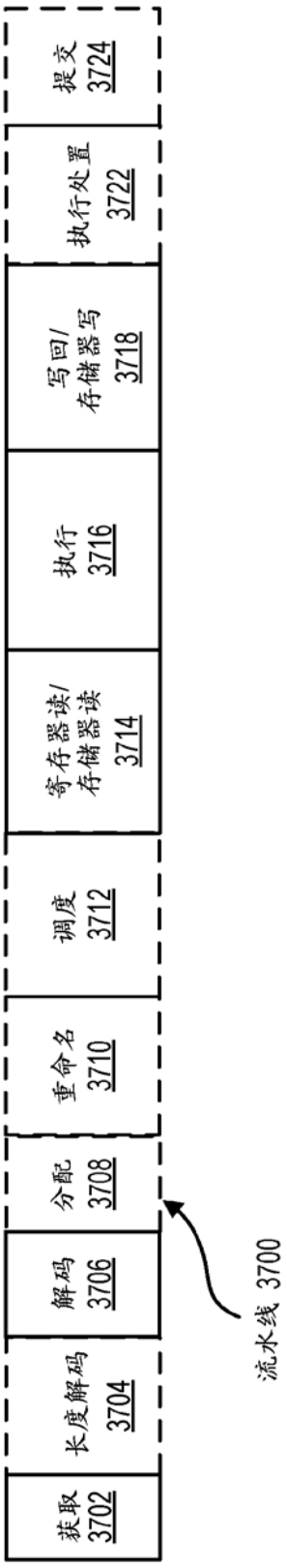

获取 3702
长度解码 3704
解码 3706
分配 3708
重命名 3710
调度 3712
寄存器读/存储器读 3714
执行 3716
写回/存储器写 3718
执行处置 3722
提交 3724
流水线 3700

图 37A

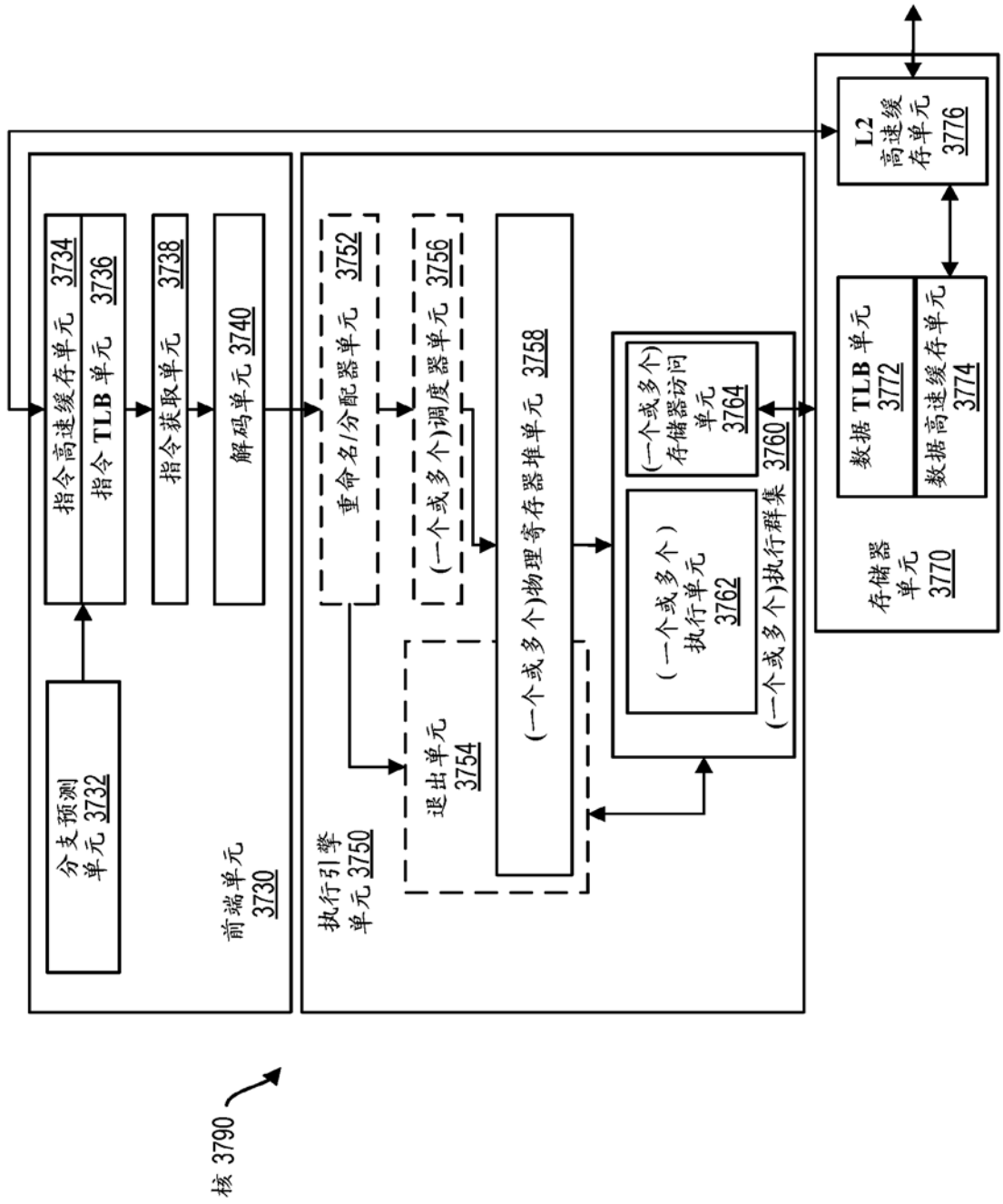

核 3790
前端单元 3730
分支预测单元 3732
指令高速缓存单元 3734
指令TLB单元 3736
指令获取单元 3738
解码单元 3740
执行引擎单元 3750
重命名/分配器单元 3752
退出单元 3754
(一个或多个)调度器单元 3756
(一个或多个)物理寄存器堆单元 3758
(一个或多个)执行群集 3760
(一个或多个)存储器访问单元 3764
(一个或多个)执行单元 3762
存储器单元 3770
数据TLB单元 3772
数据高速缓存单元 3774
L2高速缓存单元 3776

图 37B

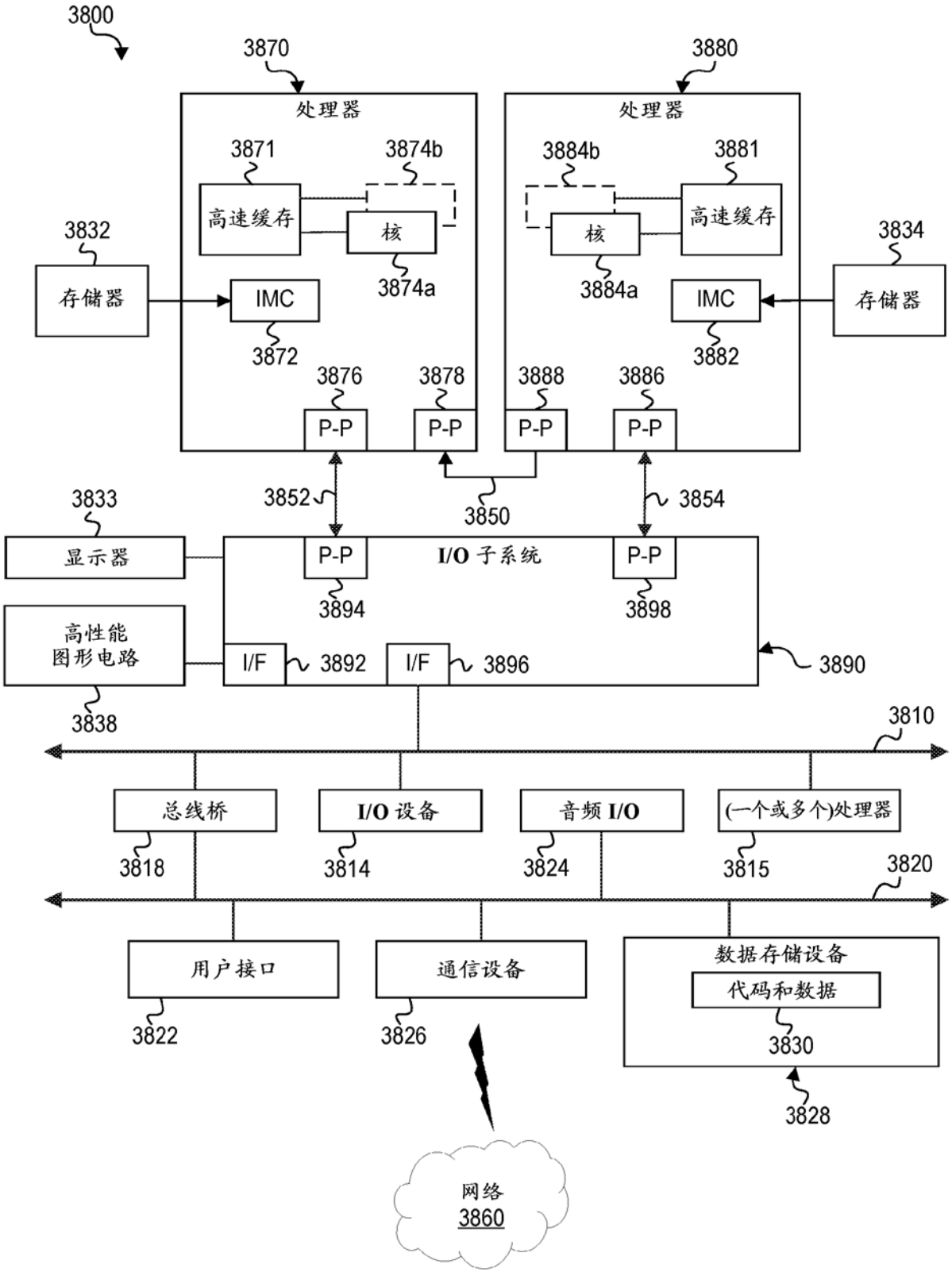

3800
3870
处理器
3871
3874b
高速缓存
核
3874a
3832
存储器
IMC
3872
3876
3878
P-P
P-P
3880
处理器
3884b
3881
核
高速缓存
3884a
3834
存储器
IMC
3882
3888
3886
P-P
P-P
3852
3850
3854
3833
显示器
P-P
I/O 子系统
P-P
3894
3898
高性能
图形电路
I/F
3892
I/F
3896
3890
3838
3810
总线桥
I/O 设备
音频 I/O
(一个或多个)处理器
3818
3814
3824
3815
3820
用户接口
通信设备
数据存储设备
代码和数据
3822
3826
3830
3828
网络
3860

图 38

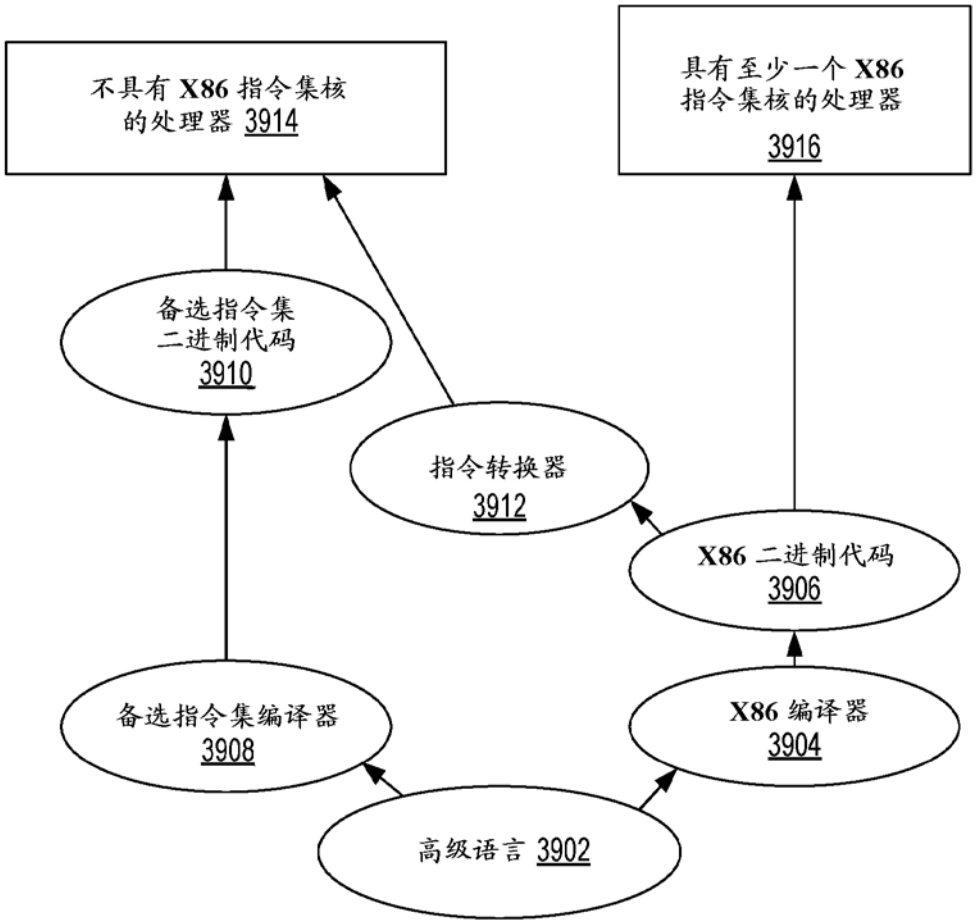

不具有 X86 指令集核
的处理器 3914
具有至少一个 X86
指令集核的处理器
3916
备选指令集
二进制代码
3910
指令转换器
3912
X86 二进制代码
3906
备选指令集编译器
3908
X86 编译器
3904
高级语言 3902

图 39