



(19)中華民國智慧財產局

(12)發明說明書公告本

(11)證書號數：TW I888632 B

(45)公告日：中華民國 114 (2025) 年 07 月 01 日

(21)申請案號：110132120

(22)申請日：中華民國 110 (2021) 年 08 月 30 日

(51)Int. Cl. : H04L9/06 (2006.01)

H04L9/14 (2006.01)

H04L9/32 (2006.01)

(30)優先權：2020/09/01 美國

63/073,366

2021/08/27 世界智慧財產權組織

PCT/US21/48039

(71)申請人：美商英特爾股份有限公司 (美國) INTEL CORPORATION (US)

美國

(72)發明人：布蘭特 傑森 BRANDT, JASON (US)；葛洛伯曼 史蒂芬 GROBMAN, STEVEN

(US)；沙巴吉 凡德伊斯 SHANBHOGUE, VEDVYAS (US)

(74)代理人：林志剛

(56)參考文獻：

US 8914648B1

US 2015/027114A1

US 2020/0125742A1

審查人員：黎苙婷

申請專利範圍項數：25 項 圖式數：24 共 241 頁

(54)名稱

用於創建、使用、及管理受保護的密碼金鑰的設備、方法及系統

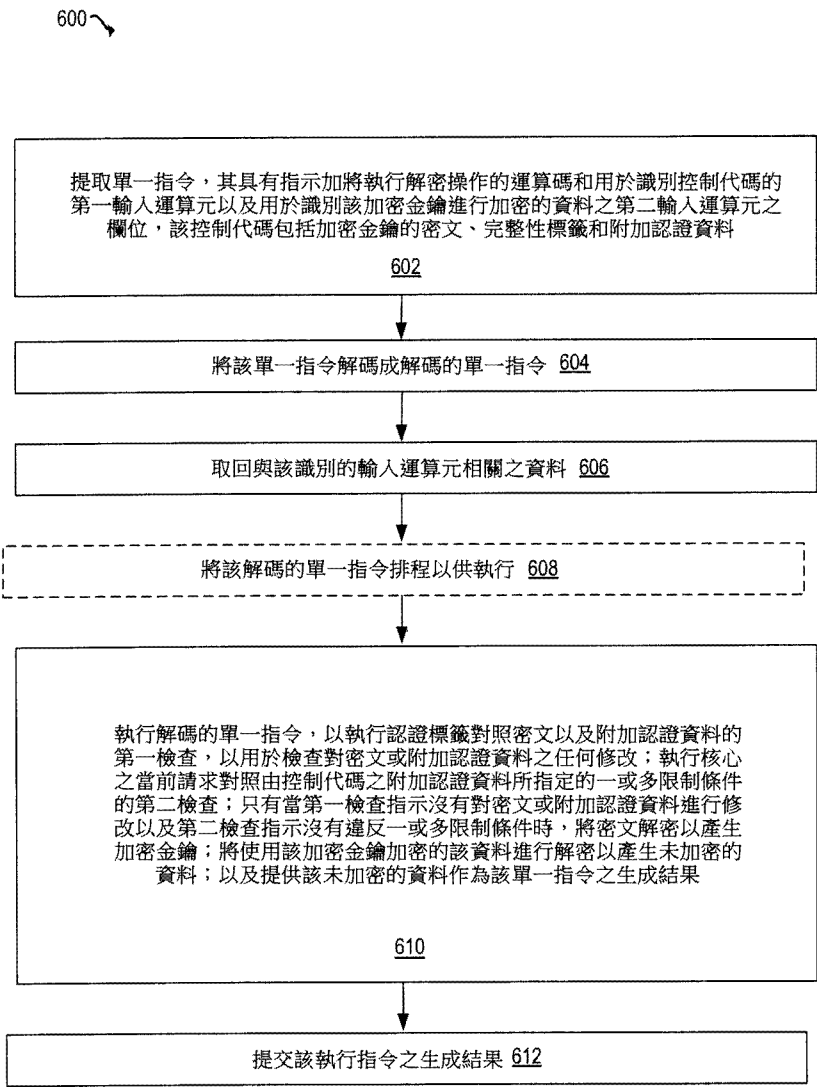
(57)摘要

本發明說明用於創建、使用、及管理受保護的密碼金鑰之系統、方法以及設備實施例。在一個實施例中，一種設備包括解碼器、執行單元和快取。解碼器將單一指令解碼為解碼的單一指令，該單一指令具有用於指定加密資料的第一來源運算元和用於指定控制代碼的第二來源運算元，該控制代碼包括加密金鑰的密文、完整性標籤和附加認證資料。執行單元是用以執行解碼的單一指令，以執行完整性標籤對照密文以及附加認證資料的第一檢查，以用於檢查對密文或附加認證資料之任何修改；執行當前請求對照由控制代碼之附加認證資料所指定的一或多限制條件的第二檢查；只有當第一檢查指示沒有對密文或附加認證資料進行修改以及第二檢查指示沒有違反一或多限制條件時，將密文解密以產生加密金鑰；使用該加密金鑰將該加密資料進行解密以產生未加密的資料；以及提供該未加密的資料作為單一指令之結果。快取是用於儲存控制代碼，其中在控制代碼的查找中僅使用完整性標籤的一部分。

System, method, and apparatus embodiments for creating, using, and managing protected cryptography keys are described. In an embodiment, an apparatus includes a decoder, an execution unit, and a cache. The decoder is to decode a single instruction into a decoded single instruction, the single instruction having a first source operand to specify encrypted data and a second source operand to specify a handle including a first including ciphertext of an encryption key, an integrity tag, and additional authentication data. The execution unit is to execute the decoded single instruction to perform a first check of the integrity tag against the ciphertext and the additional authentication data for any modification to the ciphertext or the additional authentication data, perform a second check of a current request against one or more restrictions specified by the additional authentication data of the handle, decrypt the ciphertext to generate an encryption key only

when the first check indicates no modification to the ciphertext or the additional authentication data and the second check indicates the one or more restrictions are not violated, decrypt the encrypted data with the encryption key to generate unencrypted data, and provide the unencrypted data as a result of the single instruction. The cache is to store the handle, wherein only a portion of the integrity tag is to be used in a lookup of the handle.

指定代表圖：



【圖 6】



I888632

【發明摘要】

【中文發明名稱】

用於創建、使用、及管理受保護的密碼金鑰的設備、
方法及系統

【英文發明名稱】

APPARATUS, METHOD, AND SYSTEM FOR
CREATING, USING, AND MANAGING PROTECTED
CRYPTOGRAPHY KEYS

【中文】

本發明說明用於創建、使用、及管理受保護的密碼金鑰之系統、方法以及設備實施例。在一個實施例中，一種設備包括解碼器、執行單元和快取。解碼器將單一指令解碼為解碼的單一指令，該單一指令具有用於指定加密資料的第一來源運算元和用於指定控制代碼的第二來源運算元，該控制代碼包括加密金鑰的密文、完整性標籤和附加認證資料。執行單元是用以執行解碼的單一指令，以執行完整性標籤對照密文以及附加認證資料的第一檢查，以用於檢查對密文或附加認證資料之任何修改；執行當前請求對照由控制代碼之附加認證資料所指定的一或多限制條件的第二檢查；只有當第一檢查指示沒有對密文或附加認證資料進行修改以及第二檢查指示沒有違反一或多限制條件時，將密文解密以產生加密金鑰；使用該加密金鑰將該加

密資料進行解密以產生未加密的資料；以及提供該未加密的資料作為單一指令之結果。快取是用於儲存控制代碼，其中在控制代碼的查找中僅使用完整性標籤的一部分。

【 英文 】

System, method, and apparatus embodiments for creating, using, and managing protected cryptography keys are described. In an embodiment, an apparatus includes a decoder, an execution unit, and a cache. The decoder is to decode a single instruction into a decoded single instruction, the single instruction having a first source operand to specify encrypted data and a second source operand to specify a handle including a first including ciphertext of an encryption key, an integrity tag, and additional authentication data. The execution unit is to execute the decoded single instruction to perform a first check of the integrity tag against the ciphertext and the additional authentication data for any modification to the ciphertext or the additional authentication data, perform a second check of a current request against one or more restrictions specified by the additional authentication data of the handle, decrypt the ciphertext to generate an encryption key only when the first check indicates no modification to the ciphertext or the additional authentication data and the second check indicates the one or more restrictions are not violated, decrypt the encrypted data with the encryption key to generate unencrypted data, and provide the unencrypted data as a result of the single instruction. The cache is to store the handle, wherein only a portion of the integrity tag is to be used in a lookup of the handle.

【指定代表圖】圖 6

【代表圖之符號簡單說明】無

【特徵化學式】無

【發明說明書】

【中文發明名稱】

用於創建、使用、及管理受保護的密碼金鑰的設備、
方法及系統

【英文發明名稱】

APPARATUS, METHOD, AND SYSTEM FOR
CREATING, USING, AND MANAGING PROTECTED
CRYPTOGRAPHY KEYS

【技術領域】

【0001】本揭露一般相關於電子設備，並且更具體地，本揭露的實施例相關於用於實現創建及/或使用資料的指令的電路，該資料的使用方式受到限制。

【先前技術】

【0002】一處理器、或一組的處理器執行來自指令集(例如，指令集架構(ISA))之指令。指令集為相關於程式化之電腦架構之部分，且一般包括本機(native)資料類型、指令、暫存器架構、定址模式、記憶體架構、中斷及異常處理、以及外部輸入及輸出(I/O)。應注意本文術語「指令」可指稱巨集指令，例如被提供到處理器用於執行之指令，或可指稱微指令，例如由處理器解碼器將巨集指令解碼所產生的指令。

【圖式簡單說明】

【0003】 在所附圖式中的該等圖中，藉由例示之方式而非限定之方式說明本揭露，其中相似參考編號指向類似元件，並且該等圖式其中：

【0004】 [圖1]描繪根據本揭露實施例的運用控制代碼(handle)之多核心硬體處理器之方塊圖。

【0005】 [圖2]描繪根據本揭露實施例的耦接至儲存器之硬體處理器，該儲存器包括一或多運用控制代碼(handle)之加密或解密指令。

【0006】 [圖3]描繪根據本揭露實施例的耦接至儲存器之硬體處理器，該儲存器包括一或多載入內部金鑰指令。

【0007】 [圖4]描繪根據本揭露實施例的耦接至儲存器之硬體處理器，該儲存器包括一或多控制代碼(handle)產生指令。

【0008】 [圖5]描繪了根據本揭露的實施例的用於載入第一訪客金鑰的第一「載入內部金鑰」指令的執行，和用於載入第二訪客金鑰的第二「載入內部金鑰」指令的執行。

【0009】 [圖6]描繪根據本揭露實施例的運用控制代碼(handle)之解密指令的處理方法。

【0010】 [圖7A]是描繪根據本揭露實施例之通用向量友善指令格式與其A類別(class A)指令模板的方塊圖。

【0011】[圖 7B]是描繪根據本揭露實施例之通用向量友善指令格式與其B類別(class B)指令模板的方塊圖。

【0012】[圖 8A]是描繪根據本揭露實施例之圖 7A與 7B中通用向量友善指令格式之欄位的方塊圖。

【0013】[圖 8B]是描繪根據本揭露一實施例且構成完整運算碼欄位之圖 8A中特定向量友善指令格式的欄位之方塊圖。

【0014】[圖 8C]是描繪根據本揭露一實施例且構成暫存器索引欄位之圖 8A中特定向量友善指令格式的欄位之方塊圖。

【0015】[圖 8D]是描繪根據本揭露一實施例且構成擴充操作欄位 750之圖 8A中特定向量友善指令格式的欄位之方塊圖。

【0016】[圖 9]是根據本揭露一實施例的暫存器架構之方塊圖。

【0017】[圖 10A]係說明根據本揭露實施例的例示性依序管線及例示性暫存器更名、失序發出/執行管線兩者的方塊圖。

【0018】[圖 10B]係說明根據本揭露實施例將被包括於處理器中的例示性依序架構核心之實施例及例示性暫存器更名、失序發出/執行架構核心兩者的方塊圖。

【0019】[圖 11A]是根據本揭露實施例子之單一處理器核心，以及其到晶粒上互連網路以及到其本地子集的第2階(L2)快取之連接的方塊圖。

【0020】[圖11B]係根據本揭露實施例的圖11A中處理器核心的部分的展開圖。

【0021】[圖12]係根據本揭露實施例可能具有多於一個核心、可能具有整合記憶體控制器、及可能具有整合圖形的處理器的方塊圖。

【0022】[圖13]係根據本揭露一實施例的系統的方塊圖。

【0023】[圖14]係根據本揭露實施例的更特定例示性系統之方塊圖。

【0024】[圖15]示出根據本揭露實施例的第二更特定例示性系統之方塊圖。

【0025】[圖16]示出根據本揭露實施例的系統單晶片(SoC)之方塊圖。

【0026】[圖17]係對比根據本揭露實施例之使用軟體指令轉換器以將源指令集中的二進制指令轉換成目標指令集中的二進制指令的方塊圖。

【0027】[圖18]是根據一個實施例在執行基於主機的加密時保護對稱加密金鑰的電腦實現的方法的流程圖。

【0028】[圖19]係示出根據本一實施例創建轉換後金鑰之機制的方塊圖。

【0029】[圖20]是根據一實施例使用對稱加密金鑰進行加密的電腦實作方法的流程圖。

【0030】[圖21]描繪根據本發明實施例用於載入與虛擬化密碼金鑰之資訊處理系統。

【0031】[圖22]描繪可以在根據如圖21中的實施例的系統中的處理器或多核處理器的執行核心。

【0032】[圖23]描繪根據實施例的多核心處理器。

【0033】[圖24]描繪根據實施例用於載入與虛擬化密碼金鑰之方法。

【發明內容】與【實施方式】

【0034】在以下說明中陳述數種特定細節。然而，應瞭解可實作本揭露之實施例而不具有此些特定細節。在其他情況中，已知電路、結構及技術並未被詳細顯示以避免模糊對本說明之理解。

【0035】在本說明書中對「一實施例」、「實施例」、「例示實施例」等之參照指示該所述實施例可包括特定特徵、結構、或特性，但每一個實施例可不需要包括該特定特徵、結構、或特性。此外，此類詞語不必然係在參照同一個實施例。此外，當特定特徵、結構、或特性係結合實施例而被描述時，無論有無明確描述，對結合其他實施例之此類特徵、結構、或特性做出改變係被理解為在該領域中具有通常知識者之知識內的。

【0036】硬體處理器可以使用加密金鑰(encryption key，例如，密碼金鑰(cryptographic key))來加密及/或解密資料，例如，該資料為期望受保護免受攻擊者的資料。本文中實施例減輕了竊取底層金鑰(例如，根據進階加密標準(AES)的金鑰)的硬體和軟體攻擊情境。本文中某些實

施例將操作(operation)限制為它們旨在的使用的方式，使得竊取加密資料的攻擊者無法存取它。在一個實施例中，用於資料(例如，磁碟)加密的加密金鑰可以被限制於僅在OS中可用而不在應用程式中可用。在特定實施例中，處理器(例如，應軟體的請求)將金鑰轉換為不透露金鑰值的相應「控制代碼(handles)」(例如，然後將原始金鑰從記憶體中刪除)，然後處理器以使用本文揭露一或多指令之控制代碼執行加密及/或解密。在特定實施例中，特定控制代碼不能在其他系統上或在撤銷之後(例如，在重啟之後)使用。控制代碼可以透過限制條件來指定何者為允許的，例如，該控制代碼僅由作業系統(OS)使用，該控制代碼僅用於加密，或該控制代碼僅用於解密。此處的指令和方法的非限制性用途是用於密碼庫、磁碟加密、網路連結、使用加密標準(例如AES)的任何軟體。在一個實施例中，密碼庫的使用是從應用程式中抽象化(例如，隱藏)控制代碼的使用。

【0037】在某些實施例中，硬體處理器的指令集架構包括在本文討論的一或多個指令，以創建及/或使用資料(例如，原始資料)，該資料的使用方式受到限制。儘管以下使用密碼金鑰作為受保護的原始資料的實例，但這僅僅是例示性，其他用途也是可能的。在一個實施例中，加密(例如，密碼)金鑰被限制為僅在特定模式下使用，例如，僅由OS(作業系統)、僅由特定應用程式、僅由特定可信執行環境(例如，指定位址空間(enclave))、僅由特定的虛擬

機、僅在特定的指令指標處等。

【0038】因此，竊取(例如，通過旁通道或資料洩漏)加密資料(例如，包括金鑰)的攻擊者不能在受限方式之外之方式使用它。對於某些攻擊情境，這將完全阻止了攻擊者使用該資料(例如，金鑰)。

【0039】這裡的實施例是對電腦功能的改進，因為它們保護金鑰免於外洩(例如，其中被竊取的「控制代碼」在另一個系統/處理器上是沒有用處)。這裡的實施例是對電腦功能的改進，因為它們很容易被撤銷(例如，在偵測到安全問題之後)，例如，控制代碼的使用確保竊取控制代碼的攻擊者不會獲得真正的金鑰(例如，撤銷(例如，重新啟動)後該控制代碼將沒有用，因此攻擊者必須保持其存取)。如下文進一步討論的，控制代碼的使用可以限於特定請求(例如，基於模式/使用)。例如，將控制代碼的使用限制為僅在OS(例如Ring 0(內層環))、僅加密、僅解密、特定於虛擬機(VM)(例如，假設虛擬機監控器(VMM)未設置訪客以共享控制代碼空間)、特定於特定處理(process)、特定於(例如，安全的)受信執行環境或其任何組合。在一個實施例中，可信執行環境是(例如，安全的)指定位址空間(enclave)。在一個實施例中，可信執行環境是可信域(Trusted Domain，TD)，例如，正在工作的TD，其中整個VM(例如，訪客)以保護其免受來自管理其之惡意VMM的攻擊的方式運行。

【0040】本文中某些實施例是對電腦功能的改進，因

為本文討論之使用指令允許保護資料與切換到不同環境(例如，安全虛擬機或可信執行環境)以為了使用資料(例如，金鑰)相比，後者會有高負擔且比直接使用原生處理器指令使用資料相比對軟體進行管理方面而言較複雜。

【0041】圖1描繪根據本揭露實施例的運用控制代碼(handle)126之多核心硬體處理器100之方塊圖。硬體處理器100包括複數個核心104(1)至104(N)，例如，其中N是任何整數(例如，單核心類型的處理器)或更大(例如，多核心類型的處理器)。硬體處理器100被描繪為耦接到系統記憶體102，例如，形成計算系統101。在所描繪的實施例中，硬體處理器100的核心(例如，每個核心)包括複數個邏輯核心(例如，邏輯處理元件或邏輯處理器)，例如，其中M是1或更大的任何整數。在特定實施例中，實體核心104(1)至104(N)中各者支援多執行緒(例如，在第一與第二邏輯核心上執行二或更多平行組之操作或執行緒)，且可以各種方法來達成上述操作，該等方法包括時間切割多執行緒、同時多執行緒(例如，其中單一實體核心提供各別邏輯核心給實體核心正同時執行多執行緒的各個執行緒)、或其組合(例如，時間切割提取及解碼及之後的同時多執行緒)。在某些實施例中，每個邏輯核心對軟體(例如，作業系統(OS))表現為不同的處理單元，例如，使得軟體(例如，OS)可以為並發排程兩個處理(例如，兩個執行緒)進行執行。

【0042】所描繪的硬體處理器100包括核心104(1)的

暫存器 106。在某些實施例中，每個核心包括其自己的一組暫存器 106。暫存器 106 可以包括一個或多個通用(例如，資料)暫存器 108，以執行(例如，邏輯或算術)運算，例如附加地或替代地存取(例如，載入或儲存)記憶體 102 中的資料。暫存器 106 可以包括片段暫存器 110，例如，用於儲存指示在邏輯核心上運行的軟體的當前特權級別的資料，例如，單獨地用於每個邏輯核心。在一個實施例中，當前特權級別儲存在片段暫存器 110 的碼片段選擇器暫存器的當前特權級別(CPL)欄位中。在某些實施例中，處理器 100 需要一定級別的特權來執行某些動作，例如由特定邏輯核心請求的動作(例如，由在該特定邏輯核心上運行的軟體所請求的動作)。

【0043】暫存器 106 可包括一或多個模型特定暫存器 112。在一個實施例中，模型特定暫存器 112 是組態及/或控制暫存器。在某些實施例中，每個實體核心包括其自己的一組暫存器 106。在某些實施例中，每個邏輯核心(例如，單一實體核心的多個邏輯核心)包括其自己的一組暫存器 106。在某些實施例中，每個邏輯核心包括其自己的組態及/或控制暫存器。在一個實施例中，一個或多個(例如，模型特定)暫存器(例如，僅)在處理器上運行的 OS 的請求下被寫入，例如，其中 OS 在特權(例如，系統)模式下運行但不在非特權(例如，使用者)模式下運行。在一個實施例中，模型特定暫存器只能由運行在監督者模式中的軟體寫入，而不能由運行在使用者模式中的軟體寫入。暫

存器 106 可以包括能力暫存器 116，例如，以指示處理器 (例如，核心) 是否能夠執行這裡討論的指令或其他功能。

【0044】暫存器 106 (例如，模型特定暫存器 110) 可包括一或多個控制暫存器 114、能力暫存器 116、或內部金鑰暫存器 118 (例如除了其他控制暫存器以外)。在一實施例中，各邏輯核心有其自己的控制暫存器 114、能力暫存器 116、內部金鑰暫存器 118 或其組合。在一個實施例中，複數個邏輯核心共享單一暫存器，例如共享一個或多個通用 (例如，資料) 暫存器 108。

【0045】在特定實施例中，各邏輯核心包括其自己 (例如，不與其他邏輯核心共享之) 控制暫存器 114、能力暫存器 116、及/或內部金鑰暫存器 118 (例如，與資料暫存器 108 分開 (separate))。在一個實施例中，內部金鑰暫存器 118 是唯寫暫存器 (例如，它只能由軟體寫入，而不能由軟體讀取)。在某些實施例中，控制暫存器 114 及/或能力暫存器 116 每個都是讀取和寫入暫存器，例如，當寫入請求者 (例如，軟體) 具有適當的 (例如，允許的) 特權級別 (及/或預測器模式) 時允許寫入及/或對任意特權級別皆允許之讀取。每個暫存器可以是唯讀的 (例如，由以低於臨界值的特權級別操作的邏輯核心)，或讀取及寫入 (例如，由以高於臨界值的特權級別操作的邏輯核心為可寫的)。在某些實施例中，讀取和寫入暫存器僅在監督者特權級別是可讀和可寫的。在某些實施例中，唯寫暫存器僅在監督者特權級別是可寫的，並且在任何特權級別是不可讀的。在某些

實施例中，唯讀暫存器僅在監督者特權級別是可讀的，並且在任何特權級別是不可寫的。

【0046】記憶體可以包括加密金鑰 124、控制代碼 126 及 / 或加密資料 128 (例如，加密資料的多個區塊 128(1) 至 128(X)，其中 X 是大於 1 的任何整數)。在一個實施例中，每個加密資料塊由其自己的加密金鑰 124 進行加密。在一個實施例中，多個加密資料塊由單一加密金鑰 124 進行加密。

【0047】記憶體 102 可以包括 (例如，儲存) 以下軟體中的一個或多個 (例如，其任何組合)：作業系統 (OS) 碼 130、第一應用程式碼 132、第二 (或更多) 應用程式碼 134、虛擬機監控器碼 136 或其任何組合。第一應用程式碼 132 或第二應用程式碼 134 可以是各自的使用者程式。

【0048】請注意，此處的圖式可能未描繪所有資料通訊連接。本技術領域的具有通常知識者將理解，這不是為了掩蓋圖式中的某些細節。請注意，圖中的雙頭箭頭可能不需要雙向通訊，例如，它可以指示單向通訊 (例如，去往或來自該組件或裝置)。在本文的某些實施例中可以利用通訊路徑的任何者或所有組合。在一個實施例中，處理器 100 具有單核心。在某些實施例中，計算系統 101 及 / 或處理器 100 包括在下文中討論 (例如參照本文任何圖式) 之一個或多個特徵及 / 或組件。

【0049】此處的某些實施例使用控制代碼 126 而不是用於加密資料 128 的金鑰來執行加密 / 解密。在一個實施例

中，一種方法(例如，指令)使用被轉換成控制代碼 126 的資料。在某些實施例中，控制代碼包括用金鑰(例如，加密金鑰 124)加密的原始資料、完整性測量(例如，認證標籤)和指定對如何使用資料之限制條件的附加認證資料(例如，元資料)(例如，附加認證資料也受到完整性測量的保護)。例示性限制條件是在 CPL(例如，Ring)大於零時，不能使用控制代碼(例如，在控制代碼內加密的金鑰)進行加密/解密，不能用於加密，及/或不能用於解密。控制代碼 126 的例示性格式是用於第一尺寸的金鑰(例如，128 位元金鑰)的(例如，384 位元)控制代碼，其中第一欄位(例如，控制代碼 126 的位元 [127:0])是附加認證資料、第二欄位(例如，控制代碼 126 的位元 [255:128])是認證標籤、以及第三欄位是原始資料(例如，加密金鑰)的加密版本(例如，密文)(例如，控制代碼 126 的位元 [383:256])。控制代碼 126 的另一個例示性格式是用於第二尺寸的金鑰(例如，256 位元金鑰)的(例如，512 位元)控制代碼，其中第一欄位(例如，控制代碼 126 的位元 [127:0])是附加認證資料、第二欄位(例如，控制代碼 126 的位元 [255:128])是認證標籤、第三欄位是加密金鑰的加密版本的一部分(例如，[127:0] 中密文位元中位元的前半部)(例如，控制代碼 126 的位元 [383:256])、以及第四欄位是加密金鑰的加密版本的另一部分(例如，[255:128] 中密文位元中位元的後半部)(例如，控制代碼 126 的位元 [511:383])。附加認證資料的例示性格式包括以下中的一個或多個：當設置(例如，為一)時的第

一位元位置(例如，索引[0])指示控制代碼在CPL>0中不可用，當設置(例如，為一)時的第二位元位置(例如，索引[1])指示控制代碼不能用於加密，當設置(例如，為一)時的第三位元位置(例如，索引[2])指示控制代碼不能用於解密，另一位元位置(例如，位元[27:24])指示金鑰類型(例如，0為128位元金鑰，而1為256位元金鑰)。

【0050】在一個實施例中，附加(例如，認證)資料受到完整性保護但未加密。在一個實施例中，附加(例如，認證)資料受到完整性保護且也被加密進行機密性保護。例如，元資料可能是「明文」的(意味著它可以在控制代碼內部觀察到)，但限制條件的概念也可以透過加密的元資料來完成(例如，這樣使得獲得控制代碼的攻擊者不會知道對控制代碼施加了哪些限制條件)。

【0051】在某些實施例中，藉由對加密金鑰124的原始資料(例如，用於將加密資料128中的資料塊進行加密的加密金鑰124)進行加密並在附加認證資料中設置期望的位元來創建控制代碼。在某些實施例中，加密的「加密金鑰」124的密文和(例如，加密的或未加密的)附加認證資料被綁定在一起以形成認證標籤(authentication tag)，例如，藉由將相關聯的資料(例如，附加認證資料)綁定至密文和它應該出現/使用的上下文(context)，以便偵測和拒絕嘗試將有效密文「剪切並貼上」到不同的上下文中，例如，藉由使用相關資料執行經過身份驗證的加密(例如，以允許接收者檢查訊息中加密和未加密資訊的完整性)。

在某些實施例中，使用 **nonce**(隨機值)來創建控制代碼。在某些實施例中，不使用 **nonce**(隨機值)創建控制代碼。在一個實施例中，嘗試修改密文或具有特定認證標籤的附加認證資料將導致無法使用該認證標籤(例如，以及加密金鑰 120及/或完整性金鑰 122)解密密文。在某些實施例中，在產生相應控制代碼之後，刪除加密金鑰 124。下面參考圖 4 更詳細地討論控制代碼之產生。

【0052】在某些實施例中，為將使用控制代碼之處理器(例如，核心 104(1))的未來請求(例如，模式/使用)(例如，與附加認證資料中指定的限制條件相應的請求)產生控制代碼 126。例如，未來請求(例如，模式及/或使用)可能僅用於執行 OS(例如 Ring 0)、加密、解密、執行特定虛擬機(VM)、執行特定處理、在(例如，安全的)指定位址空間(enclave)或其任何組合中執行。在一個實施例中，由核心 104(1)接收到要解密或加密的請求(例如，參考圖 2 所討論的指令)，並且作為回應，核心要讀取控制代碼 126 和內部金鑰暫存器 118(例如，具有用於創建加密控制代碼 126 的金鑰)。在某些實施例中，用於創建控制代碼的金鑰對核心 104(1)而言是私有的，例如儲存在內部金鑰暫存器 118 中(例如，不被讀取 MSR 或寫入 MSR 指令存取，但僅藉由參考圖 3 中討論的指令進行操作)。

【0053】在某些實施例中，在讀取控制代碼 126 和用於創建加密控制代碼 126 的金鑰之後，核心 104(1)對照控制代碼 126 的密文(在一個實例中，是加密資料 128 的加密

金鑰)和控制代碼 126 的附加認證資料，檢查對密文或附加認證資料的任何修改，如果通過，則執行核心 104(1)當前請求(例如，模式/使用)對照由控制代碼 126 的附加認證資料所指定的一或多個限制條件的檢查，並且只有當第一檢查指示沒有對密文或附加認證資料進行修改，且第二檢查指示並沒有違反一或多限制條件時將密文解密以產生密文之原始資料(例如，明文)(例如，在一實例中的加密金鑰)。核心 104(1)然後可以用未加密的加密金鑰解密或加密資料，例如，用現在是明文格式的加密金鑰將加密資料 128 解密。

【0054】在某些實施例中，處理器(例如，CPU)將從控制代碼到原始資料的明文格式的映射快取，以避免在普通情況下解密原始資料的密文格式所需的潛時或功率。在一個實施例中，當快取中不存在「控制代碼」到「原始資料」的映射時，仍然需要解密。某些實施例為此目的使用特殊快取，例如，不是資料/指令快取。

【0055】圖 2 描繪根據本揭露實施例的耦接至儲存器 202 之硬體處理器 200，該儲存器包括一或多運用控制代碼(handle)216 之加密或解密指令 204。在某些實施例中，加密或解密指令是根據本文的任何揭露內容。在一個實施例中，例如，回應於執行操作的請求，從儲存器 202 提取指令(例如，巨集指令)並將其發送到解碼器 206。在所描繪的實施例中，解碼器 206(例如，解碼器電路)將指令解碼成解碼指令(例如，一個或多個微指令或微操作)。解碼的

指令然後被發送用於執行，例如，經由排程器電路208來排程解碼的指令以供執行。

【0056】在某些實施例中，(例如，其中處理器/核心支援失序(OoO)執行)，處理器包括耦接到暫存器檔案/記憶體電路210(例如，單元)的暫存器更名/分配器電路以分配資源和對暫存器(例如，與指令關聯的暫存器)執行暫存器更名。在某些實施例中，(例如，對於失序執行)，處理器包括耦接到解碼器的一個或多個排程器電路208。排程器電路可以將與解碼指令相關聯的一個或多個操作排程，該操作包括從加密或解密指令解碼的一個或多個操作，以便在執行電路212上執行。

【0057】在某些實施例中，包括寫回電路214以將指令的結果寫回目的地(例如，將它們寫到暫存器及/或記憶體)，例如，使得這些結果在處理器內為可見(例如，在產生這些結果的執行電路之外為可見)。

【0058】這些組件中的一個或多個(例如，解碼器206、暫存器更名/暫存器分配器/排程器208、執行電路212、暫存器檔案/記憶體210或寫回電路214)可以是在硬體處理器的單核心內(例如，以及多個核心，且每個核心都有這些組件的一個實例)。

【0059】在加密指令的一個實施例中，執行電路212(例如，單元)將從控制代碼216(例如，如本文所討論的)判定加密金鑰，並且如果沒有異常(例如，失敗)，則使用該加密金鑰將未加密資料218加密為加密資料220。

【0060】在解密指令的一個實施例中，執行電路212(例如，單元)將從控制代碼216(例如，如本文所討論的)判定加密金鑰，並且如果沒有異常(例如，由於不與認證標籤匹配所造成失敗)時，則使用該加密金鑰將加密資料220解密為未加密(例如，解密)資料218。

【0061】在某些實施例中，加密或解密指令具有下述格式：OPCODE{ENC,DEC}{128,256}KL用於加密(ENC)或解密(DEC)單一(例如，128位元)資料塊。例如，使用暫存器108中(例如，在XMM0-7暫存器之一中)的來源和目的地資料。例如，使用記憶體引數指定的記憶體中的控制代碼。

【0062】在某些實施例中，加密或解密指令具有下述格式：OPCODE{ENC,DEC}WIDE{128,256}KL使用相同單一金鑰加密/解密以加密(ENC)或解密(DEC)多個(例如，任意複數個，例如八個)資料塊(例如，128位元)。例如，使用暫存器108(例如，在XMM0-7暫存器)中的來源和目的地資料。例如，使用記憶體引數指定的記憶體中的控制代碼。

在某些實施例中，若控制代碼真實性失敗，則加密或解密指令失敗(例如，拋出一異常)。在一個實施例中，失敗導致一組異常旗標(例如，EFLAGS.ZF)並且不修改目的地資料。在目的地可能保存原始明文/密文的一個實施例中，軟體檢查異常旗標(例如，EFLAGS.ZF以避免問題(例如，將明文放入結果中))。例示性真實性失敗是：如果沒

有使用當前(例如，包裝)金鑰(例如，來自內部金鑰暫存器 118 的內部金鑰)創建(例如，包裝)控制代碼，如果控制代碼指示的金鑰尺寸與指令指定的不同，及/或如果附加身份驗證資料指定違反的規則(例如，使用超出指定限制條件的控制代碼)。

【0063】在某些實施例中，控制代碼是從原始資料創建的，並且金鑰是本討論中「原始資料」的實例，例如，使得存在用於從「原始資料」創建控制代碼的內部金鑰(IKey)，在下面的例子中，原始資料也恰好是一個不同的金鑰。

【0064】在某些實施例中，內部金鑰(IKey)(例如，內部包裝金鑰(IWKey))用於從資料加密(例如，以創建加密資料 220)或用於資料解密(例如，創建未加密(例如，解密)資料 218)的不同金鑰創建控制代碼。在一個實施例中，允許軟體(例如，OS 或應用程式)將內部金鑰寫入處理器，但不允許軟體從處理器讀取內部金鑰，例如，軟體可以寫入圖 1 中的內部金鑰暫存器 118 中，但不能從內部金鑰暫存器 118 讀取。在一個實施例中，藉由執行載入內部金鑰指令(例如，LOADIKEY 指令或暫存器寫入(例如，WRMSR)指令)將內部金鑰寫入處理器(例如，寫入內部金鑰暫存器 118)。內部金鑰可以是軟體指定的或硬體隨機的。某些實施例允許內部金鑰僅由軟體在特殊安全模式(例如，系統管理模式(SMM)或特殊特權指定位址空間(enclave))下被讀出。

【0065】在一個實施例中，軟體可以將內部金鑰保存到平台範圍的狀態(platform-scoped state，例如，IKeyBackup)並使用新的架構MSR從其儲存器中恢復它。在一個實施例中，作業系統可以使用它來跨睡眠狀態(例如，S3/S4睡眠狀態)保存/恢復內部金鑰。虛擬機監控器(VMM)可以使用其來跨虛擬機執行保存/恢復超管理器內部金鑰。處理器可以使用其來將內部金鑰安全地分發給其他邏輯處理器(例如，在同一核心內)。

【0066】在某些實施例中，內部金鑰由軟體或硬體程式化，例如，軟體使用「載入內部金鑰」指令(例如，在啟動初期)指定隨機或特定值。在一個實施例中，軟體為所有邏輯處理器(例如，邏輯核心)(例如，單一核心中的所有邏輯處理器)設置相同的內部(例如，核心內部)金鑰。在一個實施例中，內部金鑰值沒有被鎖定，因此可以被覆寫。控制代碼產生指令的執行(例如，如參考圖4所討論的)可以返回關於哪個實體寫入當前內部金鑰的資訊，例如，並指示內部金鑰或隨機產生的內部金鑰的軟體指定值。返回資訊允許程式碼決定它是否信任內部金鑰(例如，可信執行環境中的程式碼可能只信任隨機內部金鑰)。

【0067】圖3描繪根據本揭露實施例的耦接至儲存器302之硬體處理器300，該儲存器包括一或多載入內部金鑰指令304。在某些實施例中，載入內部金鑰指令是根據本文的任何揭露內容。在一個實施例中，例如，回應於執行

操作的請求，從儲存器302提取指令(例如，巨集指令)並將其發送到解碼器306。在所描繪的實施例中，解碼器306(例如，解碼器電路)將指令解碼成解碼指令(例如，一個或多個微指令或微操作)。解碼的指令然後被發送用於執行，例如，經由排程器電路308來排程解碼的指令以供執行。

【0068】在某些實施例中，(例如，其中處理器/核心支援失序(OoO)執行)，處理器包括耦接到暫存器檔案/記憶體電路310(例如，單元)的暫存器更名/分配器電路以分配資源和對暫存器(例如，與指令關聯的暫存器)執行暫存器更名。在某些實施例中，(例如，對於失序執行)，處理器包括耦接到解碼器的一個或多個排程器電路308。排程器電路可以將與解碼指令相關聯的一個或多個操作排程，該操作包括從載入內部金鑰指令解碼的一個或多個操作，以便在執行電路312上執行。

【0069】在某些實施例中，包括寫回電路314以將指令的結果寫回目的地(例如，將它們寫到暫存器及/或記憶體)，例如，使得這些結果在處理器內為可見(例如，在產生這些結果的執行電路之外為可見)。

【0070】這些組件中的一個或多個(例如，解碼器306、暫存器更名/暫存器分配器/排程器308、執行電路312、暫存器檔案/記憶體310或寫回電路314)可以是在硬體處理器的單核心內(例如，以及多個核心，且每個核心都有這些組件的一個實例)。

【0071】在載入內部金鑰指令的一個實施例中，執行電路312(例如，單元)將輸出內部金鑰(例如，到內部金鑰儲存器316中)。在某些實施例中，內部金鑰(IKey)具有以下格式：一(例如，256位元)加密金鑰(例如，圖1中的加密金鑰120)(例如，加密原始資料以使用控制代碼中的密文)，一(例如，128位元)完整性金鑰(例如，圖1中的完整性金鑰122)(例如，允許接收者檢查加密資訊(例如控制代碼中的密文)和未加密資訊(例如，控制代碼中的附加認證資料))，一(例如，384位元)「內部金鑰持有者」身份(例如，持有者ID)，一單位元「無備份」旗標(當設置時，表示內部金鑰於內部金鑰備份儲存器(例如，在核心的快取中)的備份是不允許的)，及/或來源欄位(其中第一個值(例如，零)意味著軟體請求了特定的內部金鑰(例如，通過執行相應的載入內部金鑰指令)和第二個值(例如，一)意味著軟體請求了一個隨機的內部金鑰(例如，透過執行相應的載入內部金鑰指令))。在某些實施例中，不允許軟體從處理器讀取內部金鑰。

【0072】在某些實施例中，載入「內部金鑰」指令具有下述格式：OPCODE(例如，對於LOADIKEY或LOADIWKEY助記符)來源2(例如，xmm_reg_src2)，來源1(例如，xmm_reg_src1)。在某些實施例中，例如，只有OS(例如，ring 0)可以請求執行載入「內部金鑰」指令，而應用程式不能請求執行載入該指令。在一個實施例中，使用隱含暫存器(例如，XMM0和EAX)。在一個實施例中，

IKey.Encryption_Key=來源 1 和 來源 2 的串聯(例如，xmm_src_reg_2、xmm_src_reg1)和 IKey.Integrity_Key= XMM0。在一個實施例中，如果隱含暫存器(例如，EAX [1])已設置，則 IKey.Encryption_Key 和 IKey.Integrity_Key 與隨機數(例如，來自核心的隨機數產生器)進行 XOR 運算。在某些實施例中，如果隨機性不足，則這將失敗(例如，不修改 IKey)並設置異常(例如，ZF)旗標。在某些實施例中，隱含暫存器(例如，EAX[0])指定 IKey.NoBackup 值。在某些實施例中，IKey.KeySource 被設置以指示為軟體指定(例如，EAX[1]=0)或隨機(例如，EAX[1]=1)。在一個實施例中，虛擬機監控器被組態為藉由設置「LoadIKey 退出」VM 執行控制來導致 LOADIKEY VM 退出。在某些實施例中，加密金鑰(例如，Encryption_Key)用於將原始資料加密成控制代碼密文，而內部金鑰(例如，Integrity_Key)用於產生和驗證認證標籤(例如，其包含原始資料和附加認證資料)。

【0073】在某些實施例中，內部金鑰用於控制代碼的完整性及/或加密/解密。例如，其中控制代碼包括用於加密資料(例如，圖 1 中的資料 128)所用金鑰的加密形式。在某些實施例中，軟體經由控制代碼產生指令從金鑰和其他輸入值產生控制代碼(例如，ENCODEKEY128：從 128 位元金鑰產生控制代碼；以及 ENCODEKEY256：從 256 位元金鑰產生控制代碼)。

【0074】圖 4 描繪根據本揭露實施例的耦接至儲存器

402之硬體處理器400，該儲存器包括一或多控制代碼產生指令404。在某些實施例中，控制代碼產生指令是根據本文的任何揭露內容。在一個實施例中，例如，回應於執行操作的請求，從儲存器402提取指令(例如，巨集指令)並將其發送到解碼器406。在所描繪的實施例中，解碼器406(例如，解碼器電路)將指令解碼成解碼指令(例如，一個或多個微指令或微操作)。解碼的指令然後被發送用於執行，例如，經由排程器電路408來排程解碼的指令以供執行。

【0075】在某些實施例中，(例如，其中處理器/核心支援失序(OoO)執行)，處理器包括耦接到暫存器檔案/記憶體電路410(例如，單元)的暫存器更名/分配器電路以分配資源和對暫存器(例如，與指令關聯的暫存器)執行暫存器更名。在某些實施例中，(例如，對於失序執行)，處理器包括耦接到解碼器的一個或多個排程器電路408。排程器電路可以將與解碼指令相關聯的一個或多個操作排程，該操作包括從控制代碼產生指令解碼的一個或多個操作，以便在執行電路412上執行。

【0076】在某些實施例中，包括寫回電路414以將指令的結果寫回目的地(例如，將它們寫到暫存器及/或記憶體)，例如，使得這些結果在處理器內為可見(例如，在產生這些結果的執行電路之外為可見)。

【0077】這些組件中的一個或多個(例如，解碼器406、暫存器更名/暫存器分配器/排程器408、執行電路

412、暫存器檔案/記憶體 410或寫回電路 414)可以是在硬體處理器的單核心內(例如，以及多個核心，且每個核心都有這些組件的一個實例)。

【0078】在控制代碼產生指令的一個實施例中，執行電路 412(例如，單元)將從(例如，AES)加密金鑰和內部金鑰(例如，內部加密金鑰和內部完整性金鑰)(例如，如本文所述)產生控制代碼 416。在一個實施例中，執行單元利用加密和完整性演算法(例如，AES 演算法)(例如，AES-GCM-SIV)來產生控制代碼 416。在某些實施例中，控制代碼 416被寫入暫存器(例如，圖 1 中的暫存器 108)。在某些實施例中，控制代碼(例如，圖 1 中的控制代碼 126)被寫入記憶體 102。

【0079】在某些實施例中，控制代碼產生指令的執行還導致目的地暫存器被更新具有一值，該值指示內部金鑰(例如，軟體或隨機內部金鑰)的載入器以及指示是否允許備份內部金鑰。在一個實施例中，如果非隨機的，則只需要隨機內部金鑰的軟體可以刪除控制代碼。在某些實施例中，在所有特權級別(例如，OS、應用程式、指定位址空間(enclave)等)中都允許執行控制代碼產生指令。在某些實施例中，當利用載入內部金鑰(例如，加密和完整性金鑰)指令時，加密/完整性演算法不使用金鑰推衍函數。例如，控制代碼產生指令在其執行中不使用 nonce(隨機值)(例如，在加密/完整性演算法中不使用 nonce(隨機值))。

【0080】在某些實施例中，控制代碼產生指令不僅將

金鑰作為輸入，而且還將對控制代碼如何使用的限制條件作為輸入。在某些實施例中，控制代碼產生指令採用來源(例如，來源暫存器)中的限制條件類型欄位。例示性限制條件不允許在大於零的環上使用控制代碼進行加密/解密。在某些實施例中，可以在產生控制代碼的附加認證資料中看到限制條件類型(例如，在位元127:0中)。在某些實施例中，可以在產生控制代碼的附加認證資料中看到限制條件類型與資格(例如，在位元127:0中)。在一實施例中，產生控制代碼之附加認證資料(AAD)是完整性受保護(例如，藉由認證標籤)，但沒有加密。在某些實施例中，單一控制代碼可以指示多個限制條件，例如，僅在Ring0中工作並且僅加密(不解密)的控制代碼。在某些實施例中，限制條件類型在控制代碼中是顯而易見的，但是在不破壞密碼的情況下不能改變限制條件(例如，它會破壞原始資料，在上面的實例中是在(例如，AES)加密金鑰中的原始資料)。

【0081】在某些實施例中，控制代碼產生指令(例如，用於正在轉換為密文的128位元金鑰)具有下表1中的格式。

表 1：控制代碼產生指令(例如，ENCODEKEY128指令)的例示性格式。

輸入	說明
XMM0	Input key(例如，[127:0])
Src(GPR)	Bit 23-0:控制代碼限制條件類型。 如果指定了保留編碼，則異常(例如，#GP) Bit 31-24是保留的，且如果非0則#GP。忽略上32位元(例如，忽略 REX.W、CS.D)
輸出	說明
XMM0- XMM2	控制代碼之輸出： XMM0儲存控制代碼127:0(AAD) XMM1儲存控制代碼255:128(標籤) XMM2儲存控制代碼383:256(密文)
Dest(GPR)	如果且僅當如果沒有設置備份旗標時，才會設置Bit 0 設置Bit 4:1為KeySource(如果LoadIKey軟體指定為0，如果LoadIKey隨機為1，其他值保留) 設置Bit 63-5為0
XMM4-6	歸零或可選地保存「內部金鑰持有者」身份

【 0082 】 在某些實施例中，控制代碼產生指令(例如，用於正在轉換為密文的256位元金鑰)具有下表2中的格式。

表 2：控制代碼產生指令(例如，ENCODEKEY256指令)的例示性格式。

輸入	說明
XMM0	Input key(例如，[127:0])
XMM1	Input key(例如，[255:128])
Src(GPR)	Bit 23-0:控制代碼限制條件類型。 如果指定了保留編碼，則異常(例如，#GP) Bit 31-24是保留的，且如果非0則#GP。忽略上32位元(例如，忽略 REX.W、CS.D)
輸出	說明
XMM0- XMM2	控制代碼之輸出： XMM0儲存控制代碼127:0(AAD) XMM1儲存控制代碼255:128(標籤) XMM2儲存控制代碼383:256(密文[127:0]) XMM2儲存控制代碼511:384(密文[255:128])
Dest(GPR)	如果且僅當如果沒有設置備份旗標時，才會設置Bit 0 設置Bit 4:1為KeySource(如果LoadIKey軟體指定為0，如果LoadIKey隨機為1，其他值保留) 設置Bit 63-5為0
XMM4-6	歸零或可選地保存「內部金鑰持有者」身份

【0083】再次參照圖1，這裡討論的功能的控制和能力可以例如分別由控制暫存器114和能力暫存器116控制和描述。下面的表3討論了用於控制和功能的例示性位元。

控制代碼的使用可能是一組 KeyLocker(KL)功能的一部分。

表 3：控制和能力的例示性列舉。

CPUID位元	設置時	表示支援
KL_SUPPORTED	硬體支援KL	CR4.KL，基本LoadIKey指令，KL CPUID葉
ENC.DEC_KL_ENABLED	KL 完全啟用 (CR4.KLset，如果需 要則完成功能啟用)	負載內部金鑰{128,256}和加密 或解密指令 {ENC,DEC} {128,256}KL指令
KL Wide	HW 支援 WIDE*KL 指 令	加密或解密指令 {ENC,DEC}WIDE{128,256}KL 指令
IKeyBackup	系統支援IKeyBackup	KeyLocker MSR (IA32_COPY_LOCAL_TO_PLA TFORM、 IA32_COPY_PLATFORM_TO_ LOCAL、 IA32_COPY_STATUS、 IA32_IKEYBACKUP_STATUS)
NoBackup	系統支援NoBackup	LoadIKey具有設置之EAX[0]
Random LoadIKey	系統支援帶有隨機金 鑰的HW之LoadIKey	LoadIKey具有設置之EAX[1]
Restrictions Supported bitmap	指示支援哪些限制條 件	可以在來源暫存器中為 Load Internal Key {128,256}設置哪些 位元

【 0084 】 本文的實施例可以用作權限檢查的基礎架構。在某些實施例中，(例如，KeyLocker)控制代碼僅以兩種方式可用：(i)由控制代碼限制條件所允許並且在同一系統上；或(ii)由知道內部金鑰(IKey)的代理，例如，該代

理可以解密控制代碼以恢復原始金鑰。在某些實施例中，可以藉由適當的內部金鑰維護來防止(ii)。控制代碼限制條件可用於防止惡意軟體在別處使用控制代碼，例如，即使它設法竊取該控制代碼。

【0085】本文的實施例可以用作限制條件，以將控制代碼使用限制於一指定位址空間(enclave)。例如，如果設置了控制代碼中的「指定位址空間(enclave)限制」位元，則該控制代碼僅在其(例如，64位元)指定位址空間(enclave) ID(EID)與附加已驗證資料欄位位元(例如，127:64)匹配的指定位址空間(enclave)內為可用(例如，受完整性保護但未加密的位元)。這可以防止竊取控制代碼的攻擊者假設攻擊者沒有內部金鑰而在指定位址空間(enclave)外使用它。指定位址空間(enclave)可能需要硬體隨機內部金鑰(例如，任何軟體都不知道)。

【0086】本文的實施例可以用作限制條件，以將控制代碼使用限制於一處理。例如，如果設置了控制代碼中的「處理限制(process restricted)」位元，則該控制代碼僅在其(例如，64位元)處理ID與控制代碼位元(例如，位元[95:32])匹配的處理內為可用。在一個實施例中，新的IA32_PROCESS_ID MSR在位元63:0中保存處理ID。當處理ID需要較少的位元時(例如，只有48位元)，OS可以將附加的MSR位元用於其他用途(例如，控制代碼撤銷)。特定應用程式可以使用「處理限制」位元來確保系統上其他應用程式竊取的任何控制代碼都不能使用該控制代碼。

【0087】在虛擬化中，虛擬機監控器(例如，超管理器)經常暫停或遷移訪客(例如，虛擬機)。可能需要暫停訪客並在之後(例如，幾個月)(例如，在幾次重新啟動之後)恢復它。可能需要將訪客移動到另一個系統(遷移)。在特定實施例中，控制代碼使金鑰為短暫(**ephemeral**)，使其不能在重新啟動或在另一系統上使用。作為滿足上述要求的解決方案，內部金鑰可以是軟體指定的(例如，虛擬機監控記錄內部金鑰，例如，當訪客使用載入內部金鑰指令時)及/或虛擬機監控隱藏/防止使用載入內部金鑰指令(例如，使用**EAX[1]**或可以退出**VM**並用軟體(**SW**)指定的金鑰甚至使用主機的内部金鑰進行替換)。在一實施例中，在載入訪客金鑰之前，虛擬機監控器使用平台範圍**IKeyBackup**以將超管理器內部金鑰進行備份。

【0088】圖5描繪了根據本揭露的實施例的用於載入第一訪客金鑰的第一「載入內部金鑰」指令的執行，和用於載入第二訪客金鑰的第二「載入內部金鑰」指令的執行。主機502有兩個訪客，即訪客0 504和訪客506。描繪的訪客504包括虛擬機控制結構512，以及描繪的訪客506包括虛擬機控制結構514。如圖所示，訪客504執行載入內部金鑰指令以使用訪客0金鑰508載入主機502。在一個實施例中，在訪客504退出時，發出**MSR**命令以恢復主機內部金鑰而不是訪客0內部金鑰。如圖所示，訪客506執行載入內部金鑰指令以使用訪客1金鑰510載入主機502。在一個實施例中，在訪客506退出時，發出**MSR**命令以恢復主

機內部金鑰而不是訪客1內部金鑰。

【0089】在某些實施例中，控制暫存器（例如，CR4.KL(位元19)）用於防止舊有虛擬機監控器的訪客寫入內部金鑰（例如，Key Locker(金鑰鎖)）狀態。在一個實施例中，例如當CR4.KL為0時，控制暫存器，例如，利用對所有控制代碼/內部金鑰相關指令（例如，所有讀取或寫入內部金鑰指令）上的異常（例如，未定義(#UD)），將防止讀取/寫入內部金鑰。在某些實施例中，異常不影響內部金鑰值，它只是導致那些指令產生異常。在一個實施例中，虛擬機監控器將使用MSR點陣圖來保護未知MSR（因此如果使用了IKeyBackup）。在一個實施例中，由CPUID.KL_SUPPORTED(CPUID.(07H,0).ECX[23])列舉CR4.KL之存在。

【0090】本文的實施例可以用作限制條件，以將控制代碼限制於一訪客。例如，如果設置了控制代碼中的「虛擬機擴充(VMX)訪客」位元，則該控制代碼僅在其（例如，32位元）VM ID與控制代碼位元（例如，位元[127:96]）匹配的VMX訪客內為可用。此處的某些實施例允許將控制代碼鏈接到特定的VMX訪客。在一個實施例中，VM ID是虛擬機控制結構中的一個欄位，例如，不同於虛擬機控制結構指標和擴充頁表指標欄位，它們識別訪客但在遷移時改變並且是實體地址。例如，最好不要向訪客公開平台之實體地址。如果受害者和攻擊者訪客有時需要共享控制代碼並使用相同的內部金鑰，則可以使用用以限制控制代碼

於訪客的限制條件(restriction)。在一個實施例中，當受害者和攻擊者訪客使用不同的內部金鑰時，他們已經不能使用彼此的控制代碼。

【0091】圖6描繪根據本揭露實施例的運用控制代碼(handle)之解密指令的處理方法600。所描繪的方法600包括：提取單一指令，其具有指示加將執行解密操作的運算碼和用於識別控制代碼的第一輸入運算元以及用於識別該加密金鑰進行加密的資料之第二輸入運算元之欄位，該控制代碼包括加密金鑰的密文、完整性標籤和附加認證資料602；將該單一指令解碼成解碼的單一指令604；取回與該識別的輸入運算元相關之資料606；(可選地)將該解碼的單一指令排程以供執行608；執行解碼的單一指令，以執行認證標籤對照密文以及附加認證資料的第一檢查，以用於檢查對密文或附加認證資料之任何修改；執行核心之當前請求對照由控制代碼之附加認證資料所指定的一或多限制條件的第二檢查；只有當第一檢查指示沒有對密文或附加認證資料進行修改以及第二檢查指示沒有違反一或多限制條件時，將密文解密以產生加密金鑰；將使用該加密金鑰加密的該資料進行解密以產生未加密的資料；以及提供該未加密的資料作為單一指令之生成結果610，以及提交該執行指令之生成結果612。

【0092】本文的某些實施例涉及在輸入資料(包括原始資料)和指定限制條件中傳遞並創建控制代碼的指令。在一個實施例中，原始資料是加密金鑰。在一個實施例

中，原始資料是 AES-128 或 AES-256 加密金鑰。在一個實施例中，元資料(例如，附加認證資料)也被加密。在一個實施例中，元資料沒有加密而是明文的(但仍然受到完整性保護)。在一個實施例中，原始資料是用於簽署雜湊之金鑰。在一個實施例中，限制條件將控制代碼(例如，以及在控制代碼內加密/完整性保護的原始資料)的使用限制為：加密(例如，特定於作為金鑰的資料)，解密(例如，特定於作為金鑰的資料)，對 ring 0(OS/核心(kernel))及/或對特定處理。在一個實施例中，當以特定方式設置特定的 OS 控制狀態時，限制條件限制控制代碼的使用。在一個實施例中，只有當以特定方式設置 OS 控制狀態之特定部分時，運用遮罩以允許控制代碼之使用。在一個實施例中，限制條件在特定的可信執行環境之外限制控制代碼的使用(例如，防止控制代碼的使用)，例如但不限於特定的軟體防護擴充(Software Guard extension, SGX)指定位址空間(enclave)、特定的信任域(trust domain, TD)(例如，TDX 訪客)或系統管理模式(system management mode, SMM)。在一個實施例中，當沒有以特定方式設置特定部分之應用程式控制狀態時，限制條件限制控制代碼的使用，例如，以限制子處理(例如，僅當在特定保護金鑰中時，或僅在處理該特定銀行的資料時才使用特定銀行資料庫存取的控制代碼)。在一個實施例中，限制條件限制控制代碼在虛擬機監控器外的使用。在一個實施例中，限制條件限制控制代碼對特定指令指標的使用。在一個實施例

中，當撤回欄位指示其已經被撤回時，限制條件限制控制代碼的使用。在一個實施例中，組合多個限制條件(例如，僅加密且僅在特定處理內)。

【0093】上文中可以使用的例示性架構、系統等在下文中進行詳述。

【0094】一指令集可包括一或多指令格式。給定指令格式可界定各種欄位(例如，位元數量、位元位置)，以指定將執行之操作(例如，運算碼)以及要對其執行該操作的運算元及/或其他資料欄位(例如，遮罩)等等。透過指令模板(或子格式)的界定，可以進一步細分某些指令格式。舉例而言，給定指令格式之指令模板可以被界定成具有不同子集之指令格式欄位(該包括欄位典型上是在相同順序，但若包括較少欄位則至少若干者可具有不同位元位置)及/或被界定以使給定欄位以不同方式進行解譯。因此，ISA之各指令是使用給定指令格式來表示(並且，若經界定，則在該指令格式之指令模板中給定一者中)且包括用於指定操作與運算元之欄位。舉例而言，例示性加法指令具有特定運算碼及指令格式，其包括用以指定該運算碼之運算碼欄位以及用以選擇運算元之運算元欄位(來源1/目的地與來源2)；以及在指令串流中此加法指令之出現將在選擇特定運算元之運算元欄位中具有特定內容。已經釋出及/或發表被稱為先進向量擴充(AVX)(AVX1與AVX2)且使用向量擴充(VEX)編碼方案之一組SIMD擴充(例如，請見Intel® 64與IA-32架構軟體開發商手冊，2018年11月；且

請見Intel®架構指令集擴充程式化參考，2018年10月)。

例示性指令格式

【0095】本文所述指令之實施例可以不同格式實作。額外地，將在下文中詳述例示性系統、架構與管線。可在諸如系統、架構與管線上(但不限於此等詳述者)執行指令之實施例。

通用向量友善指令格式

【0096】一種向量友善指令格式是一種適用於向量指令之指令格式(例如，其有向量運算(vector operation)專用之特定欄位)。雖然描述之實施例其中透過向量友善指令格式而得以支援向量運算(vector operation)與純量運算(scalar operation)兩者，但替代實施例可僅使用向量運算之向量友善指令格式。

【0097】圖7A-7B是描繪根據本揭露實施例之通用向量友善指令格式與其指令模板的方塊圖。圖7A是描繪根據本揭露實施例之通用向量友善指令格式與其A類別(class A)指令模板的方塊圖；而圖7B是描繪根據本揭露實施例之通用向量友善指令格式與其B類別(class B)指令模板的方塊圖。明確而言，針對其之通用向量友善指令格式700是界定之A類別與B類別指令模板，兩者皆包括無記憶體存取705指令模板與記憶體存取720指令模板。在向量友善指令格式之上下文中，術語通用(generic)意旨指令格式沒有

被綁訂於任何特定指令集。

【0098】雖然本揭露實施例將描述成其中向量友善指令格式支援下述內容：具有32位元(4位元組)或64位元(8位元組)資料元素寬度(或尺寸)的64位元組向量運算元長度(或尺寸)(且因此，一個64位元組向量是由16個雙字尺寸的元素組成，或者替代地，由8個四字尺寸的元素組成)；具有16位元(2位元組)或8位元(1位元組)資料元素寬度(或尺寸)的64位元組向量運算元長度(或尺寸)；具有32位元(4位元組)，64位元(8位元組)，16位元(2位元組)或8位元(1位元組)資料元素寬度(或尺寸)的32位元組向量運算元長度(或尺寸)；以及具有32位元(4位元組)，64位元(8位元組)，16位元(2位元組)或8位元(1位元組)資料元素寬度(或尺寸)的16位元組向量運算元長度(或尺寸)；但替代實施例可支援具有更多、更少或不同之資料元素寬度(例如，128位元(16位元組)資料元素寬度)之更多、更少及/或不同之向量運算元尺寸(例如，256位元組向量運算元)。

【0099】圖7A中A類別指令模板包括：1)在無記憶體存取705指令模板中，其示出無記憶體存取、全捨入控制類型操作710指令模板，以及無記憶體存取、資料轉換類型操作715指令模板；以及2)在記憶體存取720指令模板中，其示出記憶體存取、時序725指令模板以及記憶體存取、非時序730指令模板。圖7B中B類別指令模板包括：1)在無記憶體存取705指令模板中，其示出無記憶體存取、寫入遮罩控制、部分捨入控制類型操作712指令模板，以

及無記憶體存取、寫入遮罩控制、`vsize`類型操作717指令模板；以及2)在記憶體存取720指令模板中，其示出記憶體存取、寫入遮罩控制727指令模板。

【0100】通用向量友善指令格式700包括以下所列欄位，順序為如圖7A-7B中所描繪者。

【0101】格式欄位740-在此欄位中的特定值(指令格式識別符值)獨特地識別向量友善指令格式，並因此識別出在指令串流中向量友善指令格式中的指令之出現。如此，在針對僅具有通用向量友善指令格式的指令集並不需要此欄位之情況下，此欄位是可選的。

【0102】基礎操作欄位742-其內容區分不同的基礎操作。

【0103】暫存器索引欄位744-其內容直接或透過位址產生來指定該來源與目的地運算元之位置，無論其在暫存器中或在記憶體中。這包括足夠數量之位元來從 $P \times Q$ (例如， 32×512 , 16×128 , 32×1024 , 64×1024)個暫存器檔案中選擇出 N 個暫存器。雖然在一實施例中， N 可高達三個來源與一個目的地暫存器，但替代實施例可支援更多或更少的來源與目的地暫存器(例如，可支援高達兩個來源，其之此等來源中一者也作用成目的地；可支援高達三個來源，其中此等來源中一者也作用成目的地；可支援高達兩個來源與一個目的地)。

【0104】修飾符欄位746-其內容區分在通用向量指令格式中指定記憶體存取之指令之出現；亦即，在無記憶體

存取705指令模板與記憶體存取720指令模板之間。記憶體存取操作讀取及/或寫入到記憶體階層(在若干情況下，使用暫存器中的值指定來源及/或目的地地址)，而非記憶體存取操作不會進行此操作(例如，來源與目的地是暫存器)。雖然在一實施例中，此欄位也在三種不同執行記憶體地址計算方式之間作選擇，但替代實施例可支援更多、更少或不同執行記憶體地址計算之方式。

【0105】擴充操作欄位750-其內容區別除了基礎操作以外將執行該等各式各樣不同操作中的何者操作。此欄位為上下文特定。在本揭露一實施例中，此欄位被劃分成類別欄位768、Alpha欄位752與Beta欄位754。擴充操作欄位750允許在單一指令中而非在2、3、或4個指令中執行共用群組之操作。

【0106】比例(scale)欄位760-其內容允許索引欄位之內容針對記憶體地址產生而進行縮放(例如，針對使用 $2^{\text{scale}} * \text{index} + \text{base}$ 之位址產生)。

【0107】位移(displacement)欄位762A-其內容是用作記憶體地址產生之部分(例如，針對使用 $2^{\text{scale}} * \text{index} + \text{base} + \text{displacement}$ 之位址產生)。

【0108】位移因子欄位762B(應註明，直接在位移因子欄位762B上並置的位移欄位762A表示使用了該一者或另一者)-其內容用作地址產生之部分；其指定將要經縮放記憶體存取(N)尺寸之位移因子-其中N是記憶體存取中位元組數(例如，針對使用 $2^{\text{scale}} * \text{index} + \text{base} + \text{scaled}$

displacement之位址產生)。忽略冗餘低階位元，並因此，位移因子欄位之內容被乘以記憶體運算元總尺寸(N)以為產生用以計算有效位址之最終位移。N之值是由處理器硬體在運行時間基於完整運算碼欄位774(文後描述)與資料調處欄位754C所判定。在位移欄位762A與位移因子欄位762B不是用於無記憶體存取705指令模板之情況下，位移欄位762A與位移因子欄位762B是可選的，及/或不同實施例可僅實作兩者中一者或都不實作。

【0109】資料元素寬度欄位764-其內容區別將使用該數量個資料元素寬度中何者資料元素寬度(在若干實施例中是用於所有指令；在其他實施例中僅用於該等指令中若干者)。在若支援一資料元素寬度及/或使用運算碼之若干態樣支援資料元素寬度則該欄位為不需要之情況下，此欄位是可選的。

【0110】寫入遮罩欄位770-其內容基於每個資料元素位置控制在該目的地向量運算元中的該資料元素位置是否反映基礎操作與擴充操作之結果。A類別指令模板支援合併寫入遮罩，而B類別指令模板支援合併與歸零寫入遮罩。當合併時，向量遮罩允許在任何操作(由基礎操作與擴充操作所指定)之執行期間保護目的地中任意組之元素免受更新；在另一實施例中，在其中相應遮罩位元具有0時，保存目的地各元素之舊有值。相較之下，當歸零時，向量遮罩允許在任何操作(由基礎操作與擴充操作所指定)之執行期間目的地中任意組之元素可以被歸零；在一實施

例中，在相應遮罩位元具有0值時，目的地元素被設置成0。一子集之此功能是能夠控制正在執行之操作的向量長度的能力(亦即，要調整元素之範圍，從第一到最終元素)；然而調整之元素並不必須是連續的。因此，寫入遮罩欄位770允許部分向量運算，包括載入、儲存、算數、邏輯等等。雖然本揭露實施例被描述其中寫入遮罩欄位770之內容從一數量個寫入遮罩暫存器中選擇出包含要使用寫入遮罩之一者(並且因此，寫入遮罩欄位770之內容間接識別將執行之遮罩)，替代實施例取而代之或附加允許寫入遮罩欄位770之內容可以直接指定將執行之遮罩。

【0111】立即數欄位772-其內容允許立即數之規範。在不支援立即數之通用向量友善格式的實作中此欄位是不存在，且不使用立即數之指令中不存在該欄位之情況下，此欄位是可選。

【0112】類別欄位768-其內容區別不同類別之指令。透過參照圖7A-B，此欄位之內容在A類別與B類別指令之間進行選擇。在圖7A-B中，圓角方形是用以指示在一欄位中存在之一特定值(例如，分別在圖7A-B中針對類別欄位768之A類別768A與B類別768B)。

A類別之指令模板

【0113】在A類別的無記憶體存取705指令模板之情況下，Alpha欄位752被解譯成RS欄位752A，其內容區別出在該等不同擴充操作類型中將執行何者擴充操作類型(例

如，捨入 752A.1 與資料轉換 752A.2 個別指定用於無記憶體存取、捨入類型操作 710 及無記憶體存取、資料轉換類型操作 715 指令模板)，而 Beta 欄位 754 區別出該等指定類型操作中將執行何者操作。在無記憶體存取 705 指令模板中，不存在比例欄位 760、位移欄位 762A、以及位移因子欄位 762B。

無記憶體存取指令模板-全捨入控制類型操作

【0114】在無記憶體存取全捨入控制類型操作 710 指令模板中，Beta 欄位 754 被解譯成捨入控制欄位 754A，其內容提供靜態捨入。雖然本揭露之所述實施例中，捨入控制欄位 754A 包括抑制全部浮點異常(suppress all floating-point exceptions, SAE)欄位 756 以及捨入操作控制欄位 758，替代實施例可將此等兩種概念編碼到同一欄位中或僅具有此等概念/欄位中一者或另一者(例如，可能僅具有捨入操作控制欄位 758)。

【0115】SAE 欄位 756-其內容區別出是否將異常事件回報停用；當 SAE 欄位 756 之內容指示啟用抑制時，給定指令並不回報任何類型之浮點異常旗標並且不引發任何浮點異常處理程序(handler)。

【0116】捨入操作控制欄位 758-其內容區別出一群組捨入操作中將執行何者捨入操作(例如，向上捨入，向下捨入，朝零捨入以及近似值捨入)。因此，捨入操作控制欄位 758 允許可基於每個指令改變捨入模式。在本揭露中

處理器包括用於指定捨入模式之控制暫存器的一實施例中，捨入操作控制欄位758之內容會置換該暫存器值。

無記憶體存取指令模板-資料轉換類型操作

【0117】在無記憶體存取資料轉換類型操作715指令模板中，Beta欄位754被解譯成資料轉換欄位754B，其內容區別出一數量個資料轉換中將執行何者資料轉換(例如，無資料轉換、拌和、廣播)。

【0118】在A類別的記憶體存取720指令模板之情況下，Alpha欄位752被解譯成驅逐提示欄位752B，其內容區別出在該等驅逐提示中將執行何者驅逐提示(在圖7A中，時序752B.1與非時序752B.2經個別指定用於記憶體存取、時序725指令模板及記憶體存取、非時序730指令模板)，而Beta欄位754被解譯成資料調處欄位754C，其內容區別出一數量個資料調處操作(亦已知為基元)中將執行何者資料調處操作(例如，無調處；廣播；來源之上轉換；以及目的地之下轉換)。記憶體存取720指令模板包括比例欄位760，並可選地包括位移欄位762A或位移因子欄位762B。

【0119】向量記憶體指令執行從記憶體之向量載入以及向量儲存到記憶體，並具有支援轉換(conversion)。與常規向量指令相同，向量記憶體指令以資料元素方面的方式從記憶體轉移資料/轉移資料到記憶體中，實際傳輸的元素的位置取決於被選作寫入遮罩的向量遮罩的內容。

記憶體存取指令模板-時序

【0120】時序資料是可能會盡快重用之資料，以受益自快取。然而，這是一種暗示，且不同處理器可以不同方式實作，包括完全忽略該暗示。

記憶體存取指令模板-非時序

【0121】非時序資料是不太可能會盡快重用之資料，以受益自第一層級快取且應給予快取驅逐之優先度。然而，這是一種暗示，且不同處理器可以不同方式實作，包括完全忽略該暗示。

B類別之指令模板

【0122】在B類別指令模板之情況下，Alpha欄位752被解譯成寫入遮罩控制(Z)欄位752C，其內容區別由寫入遮罩欄位770所控制之寫入遮罩應該是合併或歸零。

【0123】在B類別的無記憶體存取705指令模板之情況下，部分之Beta欄位754被解譯成RL欄位757A，其內容區別出在該等不同擴充操作類型中將執行何者擴充操作類型(例如，捨入757A.1與向量長度(VSIZE)757A.2個別指定用於無記憶體存取、寫入遮罩控制、部分捨入控制類型操作712指令模板及無記憶體存取、寫入遮罩控制、VSIZE類型操作717指令模板)，而剩餘的Beta欄位754區別出該等指定類型操作中將執行何者操作。在無記憶體存取705指令模板中，不存在比例欄位760、位移欄位762A、以及位

移因子欄位 762B。

【0124】在無記憶體存取、寫入遮罩控制、部分捨入控制類型操作 712 指令模板中，剩餘的 Beta 欄位 754 被解譯成捨入操作欄位 759A，且異常事件回報被停用（一給定指令並不回報任何類型之浮點異常旗標且不引發任何浮點異常處理程序(handler)）。

【0125】捨入操作欄位 759A-其與捨入操作控制欄位 758 相同，其內容區別出一群組捨入操作中將執行何者捨入操作（例如，向上捨入，向下捨入，朝零捨入以及近似值捨入）。因此，捨入操作欄位 759A 允許可基於每個指令改變捨入模式。在本揭露中處理器包括用於指定捨入模式之控制暫存器的一實施例中，捨入操作欄位 759A 之內容會置換該暫存器值。

【0126】在無記憶體存取、寫入遮罩控制、VSIZE 類型操作 717 指令模板中，剩餘的 Beta 欄位 754 被解譯成向量長度欄位 759B，其內容區別出一數量個資料向量長度中何者將被予以執行（例如，128、256 或 512 位元組）。

【0127】在 B 類別記憶體存取 720 指令模板之情況中，部分之 Beta 欄位 754 被解譯成廣播欄位 757B，其內容區別出是否將執行廣播類型資料調處操作，而剩餘 Beta 欄位 754 被解譯成向量長度欄位 759B。記憶體存取 720 指令模板包括比例欄位 760，並可選地包括位移欄位 762A 或位移因子欄位 762B。

【0128】關於通用向量友善指令格式 700，示出包括

格式欄位 740、基礎操作欄位 742、以及資料元素寬度欄位 764之一完整運算碼欄位 774。雖然示出一實施例其中完整運算碼欄位 774包括全部此等欄位，但在不支援該等全部欄位之實施例中，完整運算碼欄位 774包括此等欄位中少於全部者。完整運算碼欄位 774提供運算碼(opcode)。

【0129】擴充操作欄位 750、資料元素寬度欄位 764、以及寫入遮罩欄位 770允許在通用向量友善指令格式中以基於每個指令方式指定此等特徵。

【0130】寫入遮罩欄位與資料元素寬度欄位之組合建立了類型指令，其中其允許可基於不同資料元素寬度來應用遮罩。

【0131】在A類別與B類別中找到的各種指令模板是對不同情形有益的。在本發明若干實施例中，不同處理器或處理器中不同核心可僅支援A類別、或僅支援B類別、或支援兩種類別。舉例而言，目的旨在用於通用計算之高性能通用失序核心可僅支援B類別，主要針對圖形及/或科學(產量)計算之核心可僅支援A類別，以及旨在用於兩者之核心可支援兩者(理當而言，具有來自兩個類別的模板和指令的某種混合但不是來自兩個類別的所有模板和指令的某種混合的一核心在本發明的範圍內)。另外，單一處理器可包括多個核心，其全支援相同類別或其中不同核心支援不同類別。舉例而言，在具有分離的圖形與通用核心之處理器中，主要用於圖形及/或科學計算的圖形核心之一可能僅支持A類別，而一或多個通用核心可能是高性能

通用核心，其具有旨在用於通用計算的失序執行和暫存器更名，該高性能通用核心可能僅支援B類別。另一不具有分離圖形核心之處理器可包括支援A類別與B類別兩者之一或多通用有序或失序核心。理當而言，在本揭露之不同實施例中，出自一類別之特徵也可實作在其他類別中。以高階語言撰寫之程式可以被置放於(例如即時編譯或靜態編譯)於各式各樣不同可執行形式中，包括：1)僅具有目標處理器支援的用於執行的(一或多)類別的指令之形式；或2)具有使用所有類別的指令的不同組合編寫的替代常式並具有控制流碼之形式，該控制流程碼基於當前正在執行該碼的處理器所支援的指令來選擇要執行的常式。

例示性特定向量友善指令格式

【0132】圖8是描繪根據本揭露實施例之例示性特定向量友善指令格式的方塊圖。圖8示出特定向量友善指令格式800，其就其指定欄位之位置、尺寸、解譯與順序與及此等欄位中若干者之值方面而言是特定。特定向量友善指令格式800可用以擴充x86指令集，並因此該欄位中若干者與用於現存x86指令集與其擴充(例如，AVX)中者相似或相同。此格式維持與現有x86指令集擴充的前綴編碼欄位、真實運算碼位元組欄位、MOD R/M欄位、SIB欄位、位移欄位及立即數欄位保持一致。描繪圖8的欄位映射到圖7之欄位。

【0133】應瞭解，雖然本揭露之實施例在通用向量友

善指令格式700之上下文中參照特定向量友善指令格式800所描述以為了說明性目的，但本揭露不應受限於特定向量友善指令格式800，除非另有主張。舉例而言，通用向量友善指令格式700有考慮到針對各種欄位的各式各樣可能尺寸，而特定向量友善指令格式800被示出具有特定尺寸之欄位。透過特定實例之方式，雖然資料元素寬度欄位764在特定向量友善指令格式800中被描述成為一位元欄位，但本揭露並不受限於此(亦即，通用向量友善指令格式700有考慮到其他尺寸之資料元素寬度欄位764)。

【0134】通用向量友善指令格式700包括以下所列欄位，順序為如圖8A中所描繪者。

【0135】EVEX前綴(第0-3位元組)802-是編碼的於4位元組形成。

【0136】格式欄位740(EVEX第0位元組，位元[7:0])-第一位元組(EVEX第0位元組)是格式欄位740且其包含0x62(在本揭露一實施例中，該獨特值用於區別向量友善指令格式)。

【0137】第二～第四位元組(EVEX第1-3位元組)包括提供特定能力之一數量的位元欄位。

【0138】REX欄位805(EVEX第1位元組，位元[7-5])-由EVEX.R位元欄位(EVEX第1位元組，位元[7]-R)、EVEX.X位元欄位(EVEX第1位元組，位元[6]-X)與757BEX第1位元組，位元[5]-B構成。EVEX.R、EVEX.X、與EVEX.B位元欄位提供與相應VEX位元欄位相同功能，並

且是使用 1 補碼形式編碼的，亦即，ZMM0 被編碼成 1111B，ZMM15 被編碼成 0000B。指令之其他欄位以技藝中熟知方式將暫存器索引之較低的三位元編碼(rrr、xxx、及 bbb)，因此透過加上 EVEX.R、EVEX.X、及 EVEX.B 可形成 Rrrr、Xxxx、及 Bbbb。

【0139】 REX'欄位 710-這是 REX'欄位 710 之第一部分且是用以編碼擴充 32 暫存器組之較高的 16 個或較低的 16 個暫存器之 EVEX.R' 位元欄位 (EVEX 第 1 位元組，位元 [4]-R')。在本揭露一實施例中，此位元(連帶下述其他者)以反轉格式儲存以(在已知 x86 32 位元模式中)區別自 BOUND 指令(其真實運算碼位元組是 62)，但不接受在 MOD R/M 欄位(下述)中，MOD 欄位之值是 11；本揭露之替代實施例不儲存此與其他下示位元於反轉格式(inverted format)中。1 之值用以編碼較低的 16 個暫存器。換言之，R'Rrrr 是透過結合 EVEX.R'、EVEX.R 與來自其他欄位之其他 RRR 所形成的。

【0140】 運算碼映射欄位 815 (EVEX 第 1 位元組，位元 [3:0]-mmmm)-其內容將暗示之引導運算碼位元組 (0F、0F 38 或 0F 3) 進行編碼。

【0141】 資料元素寬度欄位 764 (EVEX 第 2 位元組，位元 [7]-W) — 是由標記 EVEX.W 所表示。EVEX.W 是用以界定資料類型之粒度(尺寸)(無論在 32 位元資料元素或 64 位元資料元素中)。

【0142】 EVEX.vvvv 820 (EVEX 第 2 位元組，位元

[6:3]-vvvv)-EVEX.vvvv之角色可包括：1)EVEX.vvvv對第一來源暫存器運算元進行編碼，以反轉(1之補碼)形式指定，並且對於具有2個或更多來源運算元的指令有效；2)EVEX.vvvv對目的地暫存器運算元進行編碼，對於某些向量移位，以1之補碼形式指定；或3)EVEX.vvvv不編碼任何運算元，該欄位是保留欄位且應包含1111b。因此，EVEX.vvvv欄位820將以反轉(1之補數)形式儲存之第一來源暫存器指示符之4個低階位元編碼。取決於指令，附加不同的EVEX位元欄位用以擴充指示符尺寸到32個暫存器。

【0143】EVEX.U 768類別欄位(EVEX第2位元組，位元[2]-U)-若EVEX.U=0，則其指示A類別或EVEX.U0；若EVEX.U=1，則其指示B類別或EVEX.U1。

【0144】前綴編碼欄位825(EVEX第2位元組，位元[1:0]-pp)-提供針對基礎操作欄位之附加位元。除了在EVEX前綴格式中提供對舊有SSE指令之支援，其也具有緊密SIMD前綴之優勢(取代需要用以表示SIMD前綴之位元組，EVEX前綴僅需要2個位元)。在一實施例中，為了支援使用舊有格式且EVEX前綴格式兩者中SIMD前綴(66H、F2H、F3H)之舊有SSE指令，此等舊有SIMD前綴被編碼到SIMD前綴編碼欄位中；且在被提供到解碼器之PLA之前，在運行時間被擴充到舊有SIMD前綴中(以令PLA可執行此等舊有指令之舊有與EVEX格式，而無須修正)。雖然較新指令可使用EVEX前綴編碼欄位之內容直接用作運算碼擴

充，但特定實施例會以相似方式擴充以為了一致性，但允許透過此等舊有 SIMD 前綴以指定不同之意義。替代實施例可重新設計 PLA 以支援 2 位元之 SIMD 前綴編碼，並且因此不需要該擴充。

【0145】 Alpha 欄位 752 (EVEX 第 3 位元組，位元 [7]-EH)；亦已知是 EVEX.EH、EVEX.rs、EVEX.RL、EVEX.write mask control、及 EVEX.N；也被描繪成 α) - 如先前所述，此欄位為上下文特定。

【0146】 Beta 欄位 754 (EVEX 第 3 位元組，位元 [6:4]-SSS)；亦已知是 EVEX.s₂₋₀、EVEX.r₂₋₀、EVEX.rr1、EVEX.LL0、及 EVEX.LLB；也被描繪成 $\beta\beta\beta$) - 如先前所述，此欄位為上下文特定。

【0147】 REX' 欄位 710 - 這是 REX' 欄位之剩餘部分且是可用以編碼擴充 32 暫存器組之較高的 16 個或較低的 16 個暫存器之 EVEX.V' 位元欄位 (EVEX 第 3 位元組，位元 [3]-V')。此位元以位元反轉 (inverted) 格式來進行儲存。1 之值用以編碼較低的 16 個暫存器。換言之，V'VVVV 是透過結合 EVEX.V'、EVEX.vvvv 所形成的。

【0148】 寫入遮罩欄位 770 (EVEX 第 3 位元組，位元 [2:0]-kkk) - 如先前所述，其內容指定在寫入遮罩暫存器中暫存器之索引。在本揭露一實施例中，特定值 EVEX.kkk=000 具有一特殊行為，其暗示沒有針對特定指令使用寫入遮罩 (此可以各式各樣方式實作，包括對所有者使用固線之寫入遮罩或使用旁路遮罩硬體之硬體)。

【0149】真實運算碼欄位830(第4位元組)已知為運算碼位元組。在此欄位中指定部分之運算碼。

【0150】MOD R/M欄位840(第5位元組)包括MOD欄位842、Reg欄位844及R/M欄位846。如先前所述，MOD欄位842之內容區別出記憶體存取與非記憶體存取操作。Reg欄位844之角色可概括成兩種情況：編碼目的地暫存器運算元或來源暫存器運算元；或被視為運算碼擴充並沒有被用以編碼任何指令運算元。R/M欄位846之角色可包括：將參照記憶體位址之指令運算元進行編碼；或將目的地暫存器運算元或來源暫存器運算元進行編碼。

【0151】SIB(Scale, Index, Base Byte(比例、變址、基址))位元組(第6位元組)-該比例欄位750之內容是用於記憶體位址產生。SIB.xxx 854與SIB.bbb 856-此等欄位之內容先前已在暫存器索引Xxxx和Bbbb中提及。

【0152】位移欄位762A(第7-10位元組)-當MOD欄位842包含10時，第7-10位元組為位移欄位762A，且其作用與舊有32位元位移(disp32)相同且以位元組粒度運作。

【0153】位移因子欄位762B(第7位元組)-當MOD欄位842包含01時，第7位元組為位移因子欄位762B。此欄位之位置與舊有x86指令集8位元位移(disp8)之欄位相同，其以位元組粒度運作。由於disp8是符號擴充的，其僅在-128與127位元組偏移之間定址；就64位元組快取線而言，disp8使用可僅被設置成四個有用值-128、-64、0與64之8個位元；由於通常需要較大的範圍，故使用disp32；然而，

`disp32` 需要 4 個位元組。相較於 `disp8` 與 `disp32`，位移因子欄位 762B 是 `disp8` 重新解釋；當使用位移因子欄位 762B 時，實際位移是由位移因子欄位的內容乘以記憶體運算元存取尺寸 (N) 確定。此類型之位移被稱為 `disp8*N`。這降低平均指令長度 (單一位元組用於位移，但具有更大的範圍)。此種壓縮位移假定有效位移是記憶體存取粒度之倍數，且因此，冗餘的位址偏移之低階位元並不需要被編碼。換言之，位移因子欄位 762B 替代掉舊有 x86 指令集 8 位元位移。因此，位移因子欄位 762B 以與 x86 指令集 8 位元位移相同的方式編碼 (因此，對 ModRM/SIB 編碼規則中沒有變化)，唯一的例外是 `disp8` 被超載為 `disp8 * N`。換言之，對編碼規則或編碼長度中無改變，僅對由硬體進行之位移值解譯方面有改變 (其需要縮放位移該記憶體運算元之尺寸，以獲得位元組方面之位址偏移)。立即數欄位 772 如前述所述般操作。

完整運算碼欄位

【0154】圖 8B 是描繪根據本揭露一實施例且構成完整運算碼欄位 774 之特定向量友善指令格式 800 的欄位之方塊圖。更明確地，該完整運算碼欄位 774 包括格式欄位 740、基礎操作欄位 742、以及資料元素寬度 (W) 欄位 764。基礎操作欄位 742 包括前綴編碼欄位 825、運算碼映射欄位 815、以及真實運算碼欄位 830。

暫存器索引欄位

【0155】圖8C是描繪根據本揭露一實施例且構成暫存器索引欄位744之特定向量友善指令格式800的欄位之方塊圖。更明確地，暫存器索引欄位744包括 REX 欄位 805、REX' 欄位 810、MODR/M.reg 欄位 844、MODR/M.r/m 欄位 846、VVVV 欄位 820、xxx 欄位 854、以及 bbb 欄位 856。

擴充操作欄位

【0156】圖8D是描繪根據本揭露一實施例且構成擴充操作欄位750之特定向量友善指令格式800的欄位之方塊圖。當類別(U)欄位768包含0時，其表示EVEX.U0(A類別768A)；當其包含1時，其表示EVEX.U1(B類別768B)。當U=0且MOD欄位842包含11(表示無記憶體存取操作)時，Alpha欄位752(EVEX第3位元組，位元[7]-EH)被解譯成rs欄位752A。當rs欄位752A包含1(捨入752A.1)時，Beta欄位754(EVEX第3位元組，位元[6:4]-SSS)被解釋成捨入控制欄位754A。捨入控制欄位754A包括一位元之SAE欄位756以及兩位元之捨入操作欄位758。當rs欄位752A包含0(資料轉換752A.2)時，Beta欄位754(EVEX第3位元組，位元[6:4]-SSS)被解釋成三位元之資料轉換欄位754B。當U=0且MOD欄位842包含00、01或10(表示記憶體存取操作)時，Alpha欄位752(EVEX第3位元組，位元[7]-EH)被解譯成驅逐提示(EH)欄位752B且Beta欄位754(EVEX第3位元組，位元[6:4]-SSS)被解譯成三位元之資料調處欄位

754C。

【0157】當 $U=1$ 時，Alpha 欄位 752(EVEX 第 3 位元組，位元 [7]-EH) 被解譯成寫入遮罩控制 (Z) 欄位 752C。當 $U=1$ 且 MOD 欄位 842 包含 11 (表示無記憶體存取操作) 時，部分之 Beta 欄位 754(EVEX 第 3 位元組，位元 [4]-S₀) 被解譯成 RL 欄位 757A；當其包含 1 (捨入 757A.1) 時，剩餘的 Beta 欄位 754(EVEX 第 3 位元組，位元 [6-5]-S₂₋₁) 被解譯成捨入操作欄位 759A，而當 RL 欄位 757A 包含 0 (VSIZE 757.A2) 時，剩餘的 Beta 欄位 754(EVEX 第 3 位元組，位元 [6-5]-S₂₋₁) 被解譯成向量長度欄位 759B(EVEX 第 3 位元組，位元 [6-5]-L₁₋₀)。當 $U=1$ 且 MOD 欄位 842 包含 00、01 或 10 (表示記憶體存取操作) 時，Beta 欄位 754(EVEX 第 3 位元組，位元 [6:4]-SSS) 被解譯成向量長度欄位 759B(EVEX 第 3 位元組，位元 [6-5]-L₁₋₀) 以及廣播欄位 757B(EVEX 第 3 位元組，位元 [4]-B)。

例示性暫存器架構

【0158】圖 9 是根據本揭露一實施例的暫存器架構 900 之方塊圖。在所描繪實施例中，有 32 個向量暫存器 910 是 512 位元寬；此等暫存器被稱作 zmm0 到 zmm31。較低 16 個 zmm 暫存器中較低階的 256 位元被覆加在暫存器 ymm0-16 上。較低 16 個 zmm 暫存器中較低階的 128 位元 (該 ymm 暫存器之較低階 128 位元) 被覆加在暫存器 xmm0-15 上。特定向量友善指令格式 800 對此等覆加暫存器檔案進行操作，如下表所示。

可調整向量長度	類別	操作	暫存器
不包括向量長度欄位759B之指令模板	A(圖7A; U=0)	710, 715, 725, 730	zmm 暫存器(向量長度是64位元組)
	B(圖7B; U=1)	712	zmm 暫存器(向量長度是64位元組)
包括向量長度欄位759B之指令模板	B(圖7B; U=1)	717, 727	zmm、ymm或xmm暫存器(向量長度是64位元組、32位元組或16位元組)，其取決於向量長度欄位759B

【0159】換言之，向量長度欄位759B在最大長度與一或多其他較短長度之間進行選擇，其中各個此等較短長度為先前長度之一半長度；且無向量長度欄位759B之指令模板對最大向量長度操作。此外，在一實施例中，特定向量友善指令格式800之B類別指令模板對緊縮或純量單一/雙重精準浮點資料與緊縮或純量整數資料進行操作。純量運算為對zmm/ymm/xmm暫存器中最低階資料元素位置執行之操作，較高階資料元素位置不是保留其在指令之前之相同態樣就是被歸零，取決於實施例而有以上不同。

【0160】寫入遮罩暫存器915-在所描繪實施例中，有8個寫入遮罩暫存器(k0到k7)，各尺寸為64位元。在替代實施例中，寫入遮罩暫存器915尺寸為16位元。如先前所述，在本揭露一實施例中，向量遮罩暫存器k0並不能被用作寫入遮罩；當通常會指示k0之編碼被用於寫入遮罩時，其選擇0xFFFF之固線寫入遮罩，並有效地停用針對該指令之寫入遮罩。

【0161】通用暫存器925-在描繪實施例中，除了現存

x86定址模式外，有使用十六個64位元通用暫存器以定址記憶體運算元。透過名稱RAX、RBX、RCX、RDX、RBP、RSI、RDI、RSP以及R8到R15，參照此等暫存器。

【0162】純量浮點堆疊暫存器檔案(x87堆疊)945，其上混淆(alias)有MMX緊縮整數平(flat)暫存器檔案950-在描繪實施例中，x87堆疊是一八個元件堆疊，透過使用x87指令集擴充，用以對32/64/80位元之浮點資料執行純量浮點運算；而MMX暫存器是用以對64位元之緊縮整數資料進行操作，同時也用以保存針對在MMX與XMM暫存器之間執行若干操作之運算元。

【0163】本揭露替代實施例可使用更寬或更窄之暫存器。額外地，本揭露替代實施例可使用更多、更少或不同暫存器檔案與暫存器。

例示性核心架構、處理器、及電腦架構

【0164】針對不同目的，可以不同方式且在不同處理器中實作處理器核心。舉例而言，此類核心之實作包括：1)目標用於通用計算的通用依序核心；2)目標用於通用計算的高效能通用失序核心；3)目標主要用於圖形及/或科學(通量)計算的特殊用途核心。不同處理器之實作可包括：1)CPU包括目標用於通用計算的一或多通用依序核心及/或目標用於通用計算的一或多通用失序核心；及2)協同處理器包括目標主要用於圖形及/或科學(通量)的一或多特殊用途核心。此類不同處理器導致不同電腦系統架構，

其可包括：1)在與CPU不同的晶片上的共處理器；2)在與CPU同一封裝中的不同晶粒上的共處理器；3)在與CPU同一晶粒上的共處理器(在該情況中，此類共處理器有時被稱作特殊用途邏輯，諸如整合圖形及/或科學(通量)邏輯、或特殊用途核心)；及4)包括在與所述CPU同一晶粒上的系統單晶片(有時被稱作應用核心或應用處理器)、上述共處理器、及附加功能性。接下來說明例示性核心架構，並隨後接著例示性處理器及電腦架構之說明。

例示性核心架構

依序及失序核心方塊圖

【0165】圖10A係說明根據本揭露實施例的例示性依序管線及例示性暫存器更名、失序發出/執行管線兩者的方塊圖。圖10B係說明根據本揭露實施例將被包括於處理器中的例示性依序架構核心之實施例及例示性暫存器更名、失序發出/執行架構核心兩者的方塊圖。圖10A-B中的實線框說明依序管線及依序核心，而虛線框之選擇性添加說明暫存器更名、失序發出/執行管線及核心。有鑒於依序態樣為失序態樣之子集，將討論失序態樣。

【0166】在圖10A中，處理器管線1000包括提取級1002、長度解碼級1004、解碼級1006、分配級1008、更名級1010、排程(亦被稱為調度(dispatch)或發出(issue))級1012、暫存器讀取/記憶體讀取級1014、執行級1016、寫回/記憶體寫入級1018、異常處理級1022、及提交級

1024。

【0167】圖10B顯示處理器核心1090包括耦接到執行引擎單元1050之前端單元1030，且兩者皆耦接到記憶體單元1070。核心1090可以是精簡指令集計算(RISC)核心、複雜指令集計算(CISC)核心、極長指令字(VLIW)核心、或混合或替代核心類型。仍作為另一選項，核心1090可係特殊用途核心，舉例而言，諸如網路或通訊核心、壓縮引擎、共處理器核心、通用計算圖形處理單元(GPGPU)核心、或圖形核心等。

【0168】前端單元1030包括耦接到指令快取單元1034之分支預測單元1032，該指令快取單元1034耦接到指令轉譯旁視緩衝器(TLB)1036，該TLB 1036耦接到指令提取單元1038，該指令提取單元1038耦接到解碼單元1040。解碼單元1040(或解碼器或解碼器單元)可解碼指令(例如，巨集指令)，並產生一或多微操作、微碼進入點、微指令、其他指令、或其他控制訊號(其解碼自、或其另外反映、或導出自原始指令)作為輸出。可使用各種機制實作解碼單元1040。適當機制之實例包括，但未限於，查找表、硬體實作、可編程邏輯陣列(PLA)、微碼唯讀記憶體(ROM)等。在一實施例中，核心1090包括微碼ROM或針對特定巨指令儲存微碼之其他媒體(例如，在解碼單元1040中或另外在前端單元1030之中)。解碼單元1040係耦接到執行引擎單元1050中的更名/分配器單元1052。

【0169】執行引擎單元1050包括耦接到退役單元1054

及一組一或多個排程器單元 1056 的更名 / 分配器單元 1052。排程器單元 1056 代表任何數量之不同排程器，包括保留站、中央指令窗等。排程器單元 1056 係耦接到實體暫存器檔案單元 1058。實體暫存器檔案單元 1058 之各者代表一或多實體暫存器檔案，其不同者儲存一或多不同之資料類型，諸如純量整數、純量浮點、緊縮整數、緊縮浮點、向量整數、向量浮點、狀態(例如，其係下一個將被執行指令的位址之指令指標)等。在一實施例中，實體暫存器檔案單元 1058 包含向量暫存器單元、寫入遮罩暫存器單元及純量暫存器單元。此些暫存器單元可提供架構向量暫存器、向量遮罩暫存器、及通用暫存器。實體暫存器檔案單元 1058 係由退役單元 1054 重疊，以說明各種其中暫存器更名及失序執行可被實作的方法(例如，使用重排序緩衝器及退役暫存器檔案；使用未來檔案、歷史緩衝器、及退役暫存器檔案；使用暫存器對照表及暫存器池等)。退役單元 1054 及實體暫存器檔案單元 1058 係耦接到執行叢集 1060。執行叢集 1060 包括一組一或多個執行單元 1062 及一組一或多個記憶體存取單元 1064。執行單元 1062 可對各種類型之資料(例如，純量浮點、緊縮整數、緊縮浮點、向量整數、向量浮點)執行各種操作(例如，移位、加法、減法、乘法)。雖然若干實施例可包括數種專用於特定功能或功能組之執行單元，但其他實施例可包括都執行所有功能之僅一個執行單元或多個執行單元。排程器單元 1056、實體暫存器檔案單元 1058、及執行叢集 1060 被顯示為可能

係複數個，此係因為特定實施例建立獨立管線用於特定類型之資料/操作(例如，其各具有其自身排程器單元、實體暫存器檔案單元、及/或執行叢集之純量整數管線、純量浮點/緊縮整數/緊縮浮點/向量整數/向量浮點管線、及/或記憶體存取管線-且在獨立記憶體存取管線之情形中，特定實施例被實作，其中僅此管線之執行叢集具有記憶體存取單元1064)。應理解在使用獨立管線處，此類管線之一或多者可係失序發出/執行且剩餘者係依序。

【0170】該集合之記憶體存取單元1064是耦接到記憶體單元1070，該記憶體單元1070可包括資料TLB單元1072，該資料TLB單元1072耦接到資料快取單元1074，而該資料快取單元1074耦接到第2階(L2)快取單元1076。在一例示性實施例中，記憶體存取單元1064可包括載入單元、儲存位址單元、及儲存資料單元，其各者係耦接到記憶體單元1070中的資料TLB單元1072。指令快取單元1034可進一步耦接到記憶體單元1070中的第2階(L2)快取單元1076。L2快取單元1076係耦接到一或多其他層級之快取並最後耦接到主記憶體。

【0171】在某些實施例中，包括預取電路1078以預取資料，例如用以預測存取位址並將這些位址的資料帶入一個或多個快取(例如，從記憶體1080)。在一個實施例中，預取電路1078是圖3B中的預取電路的一個實例。

【0172】作為實例，例示性暫存器更名、失序發出/執行核心架構可如下實作管線1000：1)指令提取1038執行

提取及長度解碼級 1002 及 1004；2) 解碼單元 1040 執行解碼級 1006；3) 更名/分配器單元 1052 執行分配級 1008 及更名級 1010；4) 排程器單元 1056 執行排程級 1012；5) 實體暫存器檔案單元 1058 及記憶體單元 1070 執行暫存器讀取/記憶體讀取級 1014；執行叢集 1060 執行執行級 1016；6) 記憶體單元 1070 及實體暫存器檔案單元 1058 執行寫回/記憶體寫入級 1018；7) 各種單元可涉及異常處理級 1022；及 8) 退役單元 1054 及實體暫存器檔案單元 1058 執行提交級 1024。

【0173】核心 1090 可支援一或多指令集(例如，x86 指令集(具有較新版本已加入之若干擴充)；加州桑尼維爾的 MIPS Technologies 之 MIPS 指令集；加州桑尼維爾的 ARM Holdings 之 ARM 指令集(具有諸如 NEON 之可選附加擴充))，包括本文所述之(一或多)指令。在一實施例中，核心 1090 包括邏輯用以支援緊縮資料指令集擴充(例如，AVX1、AVX2)，從而允許被許多多媒體應用使用之該等操作將被執行使用緊縮資料。

【0174】應瞭解，核心可支援多執行緒(執行二或更多平行組之操作或執行緒)，且可以各種方法來達成上述操作，該等方法包括時間切割多執行緒、同時多執行緒(其中單一實體核心提供邏輯核心給實體核心正同時執行多執行緒的執行緒之各者)、或其組合(例如，時間切割提取及解碼及之後的同時多執行緒，諸如在 Intel®Hyper-Threading 技術中一般)。

【0175】儘管暫存器更名在失序執行之上下文中被論

及，但應瞭解暫存器更名可用於依序架構中。雖然所說明之該處理器的實施例亦包括獨立指令及資料快取單元 1034/1074 及共享 L2 快取單元 1076，但替代實施例可具有單一內部快取用於指令及資料兩者，舉例而言，諸如第 1 階 (L1) 內部快取、或多階之內部快取。在若干實施例中，系統可包括內部快取及外部快取之組合，該外部快取在核心及 / 或處理器之外部。替代地，所有的快取可在核心及 / 或處理器之外部。

特定例示性依序核心架構

【0176】圖 11A-B 說明更特定例示性依序核心架構之方塊圖，其核心為晶片中許多邏輯方塊之一 (包括同一類型及 / 或不同類型之其他核心)。邏輯方塊透過高頻寬互連網路 (例如，環狀網路) 與若干固定功能邏輯、記憶體 I/O 介面、及其他必要 I/O 邏輯 (取決於應用而定) 通訊。

【0177】圖 11A 係根據本揭露實施例之連同其到晶粒上互連網路 1102 之連接且具有其第 2 階 (L2) 快取之本地子集 1104 的單一處理器核心的方塊圖。在一實施例中，指令解碼單元 1100 支援具有緊縮資料指令集擴充的 x86 指令集。L1 快取 1106 允許到快取記憶體之低潛時存取進到純量及向量單元中。然而在一實施例中 (為簡化設計)，純量單元 1108 及向量單元 1110 使用獨立暫存器集 (個別地，純量暫存器 1112 及向量暫存器 1114)，且在彼者之間傳送的資料被寫入到記憶體並接著自第 1 階 (L1) 快取 1106 讀取回

來，本揭露替代實施例可使用不同方法(例如，使用單一暫存器集或包括允許資料在不被寫入與讀取回來的前提下在兩個暫存器檔案之間傳送的通訊路徑)。

【0178】L2快取之本地子集1104為總體L2快取之部分，其被分成獨立本地子集係每處理器核心有一個。各處理器核心具有到L2快取1104之其本身本地子集的直接存取路徑。藉由處理器核心讀取之資料被儲存在其L2快取子集1104中，且可被快速地存取，其平行於其他處理器核心存取其自身本地L2快取子集。藉由處理器核心寫入之資料被儲存在其L2快取子集1104中，且若有必要則自其他子集清除。環狀網路確保針對共享資料之一致性。環狀網路為雙向以允許諸如處理器核心、L2快取及其他邏輯方塊之代理(agent)用以與晶片之中的彼此通訊。各環狀資料路徑係每方向1012位元寬。

【0179】圖11B係根據本揭露實施例的圖11A中處理器核心的部分的展開圖。圖11B包括L1快取1106之L1資料快取1106A部分，以及包括更多相關於向量單元1110及向量暫存器1114之細節。明確地，向量單元1110係16位元寬的向量處理單元(VPU)(參見16位元寬ALU 1128)，其執行整數、單精度浮點、及雙精度符點指令之一或多者。VPU支援使用拌和單元1120將暫存器輸入拌和、使用數字轉換單元1122A-B之數字轉換、及使用複製單元1124對記憶體輸入之複製。寫入遮罩暫存器1126允許判定(predicate)結果向量寫入。

【0180】圖12係根據本揭露實施例可能具有多於一個核心、可能具有整合記憶體控制器、及可能具有整合圖形的處理器1200的方塊圖。圖12之實線框說明具有單一核心1202A、系統代理1210、一組一或多個匯流排控制器單元1216之處理器1200，而虛線框之可選添加性說明具有多核心1202A-N、系統代理單元1210中的一組一或多個整合記憶體控制器單元1214、及特殊用途邏輯1208之替代處理器1200。

【0181】因此，處理器1200之不同實作包括：1)具有係整合圖形及/或科學(通量)邏輯的特殊用途邏輯1208之CPU(其可包括一或多核心)，且核心1202A-N係一或多通用核心(例如，通用依序核心、通用失序核心、或兩者之組合)；2)具有係大數量主要目標用於圖形及/或科學(通量)的特殊用途核心的核心1202A-N之共處理器；及3)具有係大數量通用依序核心的核心1202A-N之共處理器。因此，處理器1200可係通用處理器、共處理器或特殊用途處理器，舉例而言，諸如網路或通訊處理器、壓縮引擎、圖形處理器、GPGPU(通用圖形處理單元)、高通量許多整合核心(MIC)共處理器(包括30或更多核心)、嵌入式處理器或相似者。該處理器可被實作在一或多晶片上。使用舉例而言諸如BiCMOS、CMOS、或NMOS之數種製程技術之任意者，處理器1200可係一或多基板之一部分及/或可被實作在一或多基板上。

【0182】記憶體階層包括核心內之一或多層級之快

取、一組或一或多共享快取單元 1206、及耦接到整合記憶體控制器單元 1214 之集合的外部記憶體(未示出)。該組共享快取單元 1206 可包括一或多中階快取，諸如第 2 階(L2)、第 3 階(L3)、第 4 階(L4)、或其他層級之快取、最後階快取(LLC)、及/或其組合。雖然在一實施例中，環狀為基互連單元 1212 將整合圖形邏輯 1208、該組共享快取單元 1206、及系統代理單元 1210/整合記憶體控制器單元 1214 互連，替代實施例可使用任意數量之已知技術來將此類單元互連。在一實施例中，於一或多共享快取單元 1206 及核心 1202-A-N 之間保持一致性。

【0183】在若干實施例中，核心 1202A-N 之一或多者能夠執行多執行緒。系統代理 1210 包括該些協調及操作核心 1202A-N 之組件。系統代理單元 1210 可例如包括功率控制單元(PCU)及顯示器單元。PCU 可係或可包括用於調節核心 1202A-N 和整合圖形邏輯 1208 之功率狀態所需之邏輯及組件。顯示器單元係用於驅動一或多外部連接顯示器。

【0184】就架構指令集而言核心 1202A-N 可係同質或異質；亦即，核心 1202A-N 之二或更多者可能能夠執行相同指令集，而它者可能僅能夠執行該指令集之子集或不同指令集。

例示性電腦架構

【0185】圖 13-16 是例示性電腦架構之方塊圖。針對下述者在該技術領域中已知的其他系統設計及組態亦適

用，下述者包括：膝上型電腦、桌上型電腦、手持PC、個人數位助理、工程工作站、伺服器、網路裝置、網路集線器、開關、嵌入式處理器、數位訊號處理器(DSP)、圖形裝置、視訊遊戲裝置、機上盒、微控制器、手機、可攜式媒體播放器、手持裝置、及各種其他電子裝置。普遍而言，能夠結合本文揭示之處理器及/或其他執行邏輯的廣大種類之系統或電子裝置亦屬適當。

【0186】現在參照圖13，所顯示為依據本揭露一實施例的系統1300之方塊圖。系統1300可包括一或多處理器1310、1315，其耦接到控制器集線器1320。在一實施例中，控制器集線器1320包括圖形記憶體控制器集線器(GMCH)1390及輸入/輸出集線器(IOH)1350(其可能在獨立晶片上)；GMCH 1390包括記憶體及圖形控制器，且與其耦接到記憶體1340及共處理器1345；IOH 1350將輸入/輸出(I/O)裝置1360耦接到GMCH 1390。替代地，記憶體及圖形控制器之一或兩者係整合於處理器(如本文所述)之中，記憶體1340及共處理器1345係直接地耦接到處理器1310，且控制器集線器1320與IOH 1350在單一晶片中。記憶體1340可以包括例如加密/解密碼1340A，例如以儲存在執行時使處理器執行本揭露的任何方法的碼。

【0187】附加處理器1315之可選性係以虛線標記於圖13中。各處理器1310、1315可包括本文所述處理核心之一或多者，且可能係某版本之處理器1200。

【0188】記憶體1340可舉例而言係動態隨機存取記憶

體 (DRAM)、相變記憶體 (PCM)、或兩者之組合。針對至少一實施例，控制器集線器 1320 經由諸如例如前端匯流排 (FSB) 之多落點匯流排、諸如 QuickPath 互連 (QPI) 之點對點介面、或類似連接 1395 而與處理器 1310、1315 通訊。

【0189】在一實施例中，共處理器 1345 係特殊用途處理器，舉例而言，諸如高通量 MIC 處理器、網路或通訊處理器、壓縮引擎、圖形處理器、GPGPU、嵌入式處理器或相似者。在一實施例中，控制器集線器 1320 可包括整合圖形加速器。

【0190】在包括架構、微架構、熱、功耗特徵等類似者之優點度量頻譜方面，實體資源 1310、1315 之間可具有各種不同之差異。

【0191】在一實施例中，處理器 1310 執行控制一般類型資料處理操作之指令。嵌入於指令之內的可能係共處理器指令。處理器 1310 認出此些共處理器指令為應由附接共處理器 1345 所執行之類型。據此，處理器 1310 對共處理器匯流排或其他互連發出此些共處理器指令 (或代表共處理器指令之控制訊號) 到共處理器 1345。共處理器 1345 接收並執行所接收到的共處理器指令。

【0192】現在參照圖 14，所顯示為依據本揭露實施例的第一較特定例示性系統 1400 之方塊圖。如圖 14 所示，多處理器系統 1400 係點對點互連系統，且包括經由點對點互連 1450 耦接之第一處理器 1470 及第二處理器 1480。處理器 1470 及 1480 之各者可能係某版本之處理器 1200。在本揭露

一實施例中，處理器1470及1480個別係處理器1310及1315，而共處理器1438係共處理器1345。在另一實施例中，處理器1470及1480個別係處理器1310及共處理器1345。

【0193】處理器1470及1480被顯示為個別包括整合記憶體控制器(IMC)單元1472及1482。處理器1470亦包括匯流排控制器單元點對點(P-P)介面1476及1478作為其一部分；相似地，第二處理器1480包括P-P介面1486及1488。處理器1470、1480可使用點對點(P-P)介面電路1478、1488經由點對點(P-P)介面1450來交換資訊。如圖14所示，IMC 1472及1482將處理器耦接到個別記憶體，亦即記憶體1432及記憶體1434，其可係本地附接到個別處理器的主記憶體之部分。

【0194】處理器1470、1480可各使用點對點介面電路1476、1494、1486、1498經由獨立P-P介面1452、1454來與晶片組1490交換資訊。晶片組1490可選地經由高效能介面1439而與共處理器1438交換資訊。在一實施例中，共處理器1438係特殊用途處理器，舉例而言，諸如高通量MIC處理器、網路或通訊處理器、壓縮引擎、圖形處理器、GPGPU、嵌入式處理器或相似者。

【0195】共享快取(未示出)可被包括在任一處理器中或在兩處理器外部，仍經由P-P互連而與處理器連接，以在假設處理器被置於低功率模式之情況下，可使得任一或兩個處理器之本地快取資訊被儲存在共享快取中。

【0196】晶片組1490可經由介面1496而被耦接到第一

匯流排 1416。在一實施例中，第一匯流排 1416 可係週邊組件互連 (PCI) 匯流排，或諸如 PCI 快速匯流排或其他第三世代 I/O 互連匯流排之匯流排，雖然本揭示之範圍並未如此限制。

【0197】如於圖 14 中所顯示，各種 I/O 裝置 1414 可連同匯流排橋接器 1418 而被耦接到第一匯流排 1416，該匯流排橋接器 1418 可將第一匯流排 1416 耦接到第二匯流排 1420。在一實施例中，一或多附加處理器 1415 耦接到第一匯流排 1416，該附加處理器諸如共處理器、高通量 MIC 處理器、GPGPU、加速器 (例如，諸如圖形加速器或數位訊號處理 (DSP) 單元)、現場可編程閘極陣列、或任何其他處理器。在一實施例中，第二匯流排 1420 可係低接腳數 (LPC) 匯流排。在一實施例中，可被耦接到第二匯流排 1420 之各種裝置包括，例如，鍵盤及 / 或滑鼠 1422、通訊裝置 1427 及可能包括指令 / 碼及資料 1430 的諸如硬碟機或其他大量儲存裝置之儲存器單元 1428。進一步地，音訊 I/O 1424 可被耦接到第二匯流排 1420。應注意其它架構係可行。例如，取代圖 14 之點對點架構，一系統可實作多落點匯流排或其他此類架構。

【0198】現在參照圖 15，所顯示為依據本揭露實施例的第二較特定例示性系統 1500 之方塊圖。圖 14 及 15 中相似元件具有相似參考編號，且圖 14 之特定態樣已自圖 15 省略以為了避免模糊圖 15 之其他態樣。

【0199】圖 15 說明處理器 1470、1480 可包括整合記憶

體及個別具有 I/O 控制邏輯 (control logic, 「CL」) 1472 及 1482。因此, CL 1472、1482 包括整合記憶體控制器單元且包括 I/O 控制邏輯。圖 15 不僅描繪記憶體 1432、1434 耦接到 CL 1472、1482, 且描繪 I/O 裝置 1514 亦耦接到控制邏輯 1472、1482。舊有 I/O 裝置 1515 係耦接到晶片組 1490。

【0200】現在參照圖 16, 所顯示為依據本揭露實施例的 SoC 1600 之方塊圖。圖 12 中相似元件具有相似參考編號。另外, 虛線框是在更先進 SoC 上之可選特徵。在圖 16 中, 互連單元 1602 係耦接到: 應用處理器 1610, 其包括一或多核心 202A-N 之集合及共享快取單元 1206; 系統代理單元 1210; 匯流排控制器單元 1216; 整合記憶體控制器單元 1214; 一或多個共處理器 1620 之集合, 其可包括整合圖形邏輯、影像處理器、音訊處理器、及視訊處理器; 靜態隨機存取記憶體 (SRAM) 單元 1630; 直接記憶體存取 (DMA) 單元 1632; 及用於耦接到一或多外部顯示器之顯示器單元 1640。在一實施例中, 共處理器 1620 係特殊用途處理器, 舉例而言, 諸如網路或通訊處理器、壓縮引擎、GPGPU、高通量 MIC 處理器、嵌入式處理器或相似者。

【0201】本文揭示之實施例(例如, 機制)可實作於硬體、軟體、韌體、或此類實作方法之組合之中。本揭露實施例可被實作為電腦程式或在可編程系統上執行的程式碼, 該可編程系統包含至少一處理器、儲存系統(包括揮發性及非揮發性記憶體及/或儲存元件)、至少一輸入裝置、及至少一輸出裝置。

【0202】程式碼，諸如圖14所說明的碼1430，可應用於輸入指令以執行本文所述之功能並產生輸出資訊。可以已知方式將該輸出資訊應用於一或多輸出裝置。針對本應用之目的，處理系統包括具有處理器之任意系統，舉例而言，諸如數位訊號處理器(DSP)、微控制器、特殊應用積體電路(ASIC)、或微處理器。

【0203】程式碼可以高階程序或物件導向程式語言來實作，以與處理系統通訊。如有需求，則程式碼亦可以組合或機器語言來實作。事實上，本文所述機制的範疇並未限制於任何特定程式語言。在任何情況中，該語言係編譯或解譯語言。

【0204】至少一實施例之一或多態樣可藉由儲存在記者可讀取媒體上之代表性指令來實作，其代表處理器之中的各種邏輯，當其被機器讀取時促使該機器製造邏輯以執行本文所述之技術。此類表示法，已知為「IP核心」，可被儲存在有形、機器可讀取媒體上，並被供應給各種顧客或製造工廠以載入於生產該邏輯或處理器的製造機器之中。

【0205】未限制地，此類機器可讀取儲存媒體可包括由機器或裝置所生產或形成製品之非暫態、有形配置，包括諸如硬碟、包括軟碟、光碟、光碟唯讀記憶體(CD-ROM)、光碟可覆寫記憶體(CD-RW)、及磁光碟之任何其他類型之碟片的儲存媒體；諸如唯讀記憶體(ROM)、諸如動態隨機存取記憶體(DRAM)、靜態隨機存取記憶體

(SRAM)之隨機存取記憶體(RAM)、可抹除可編程唯讀記憶體(EPROM)、快閃記憶體、電性可抹除可編程唯讀記憶體(EEPROM)、相變記憶體(PCM)之半導體裝置、磁或光卡、或適用於儲存電子指令之任何其他類型的媒體。

【0206】據此，本揭露實施例亦包括包含指令或包含設計資料(諸如硬體描述語言(HDL))的非暫態、有形機器可讀取媒體，該指令或設計資料界定本文所述之結構、電路、設備、處理器及/或系統特徵。此類實施例亦被參照為程式產品。

仿真(包括二進制轉譯、碼變形等)

【0207】在一些情況中，指令轉換器可被使用以將指令從源指令集轉換到目標指令集。舉例而言，指令轉換器可將指令轉譯(例如，使用靜態二進制轉譯、包括動態編譯之動態二進制轉譯)、變形、仿真、或另外轉換成將被核心處理之一或多其他指令。指令轉換器可被實作於軟體、硬體、韌體、或其組合中。指令轉換器可係在處理器上、不在處理器上、或部分在處理器上且部分不在處理器上。

【0208】圖17係對比根據本揭露實施例之使用軟體指令轉換器以將源指令集中的二進制指令轉換成目標指令集中的二進制指令的方塊圖。在所說明實施例中，指令轉換器係軟體指令轉換器，雖然替代地指令轉換器可被實作於軟體、韌體、硬體、或其各種組合中。圖17顯示為高階語

言 1702 之程式可使用 x86 編譯器 1704 來編譯，以產生 x86 二進制碼 1706，其可由具有至少一個 x86 指令集核心之處理器 1716 原生地執行。具有至少一個 x86 指令集核心之處理器 1716 代表能夠執行實質上與具有至少一個 x86 指令集核心之 Intel® 處理器所能執行相同功能之任何處理器，藉由相容地執行或另外處理 (1) Intel® x86 指令集核心之指令集之實質部分或 (2) 目標要在具有至少一個 x86 指令集核心之 Intel® 處理器上運行的應用或其他軟體之目標碼版本，以為了達到與具有至少一個 x86 指令集核心之 Intel® 處理器所能達到實質上相同之結果。x86 編譯器 1704 代表可操作以產生 x86 二進制碼 1706 (例如，目標碼) 之編譯器，該碼可在具有或不具有附加鏈接處理之前提下在具有至少一個 x86 指令集核心之處理器 1716 上執行。相似地，圖 17 顯示為高階語言 1702 之程式可使用替代指令集編譯器 1708 來編譯，以產生替代指令集二進制碼 1710，其可由不具有至少一個 x86 指令集核心之處理器 1714 原生地執行 (例如，處理器具有執行加州桑尼維爾 MIPS Technologies 的 MIPS 指令集及 / 或執行加州桑尼維爾 ARM Holdings 的 ARM 指令集之核心)。指令轉換器 1712 被使用以將 x86 二進制碼 1706 轉換成可由不具有 x86 指令集核心之處理器 1714 原生地執行之碼。此經轉換的碼不大可能與替代指令集二進制碼 1710 相同，因為能夠執行此操作之指令轉換器係難以製造；然而，經轉換的碼將完成一般操作，且由來自替代指令集之指令組成。因此，指令轉換器 1712 代表軟體、韌體、硬

體、或其組合，其透過仿真、模擬、或其他處理允許處理器或不具有 x86 指令集處理器或核心之其他電子裝置來執行 x86 二進制碼 1706。

進一步實施例描述

【0209】下面提供對上述實施例及/或其他實施例的進一步描述。在這些描述中，如在上面的描述中，諸如架構、特徵、指令、操作、暫存器、實作等的名稱的細節(例如，Key Locker)可以不受限制地使用，例如，它們可以被包括在描述中不將那些描述的其他細節限制於這些實施例，及/或不將使用第一名稱(例如，ENCODEKEY128、ENCODEKEY256)的描述的細節和使用第二名稱(例如，KEYTRANSFORM)的描述的細節限制到不同的實施例。

用於保護加密金鑰的實施例

【0210】描述了用於在執行基於主機的加密時保護對稱加密金鑰的系統和方法。在一個實施例中，一種電腦實作的方法包括從安全區域中取回至少一個真實金鑰並利用處理器執行金鑰轉換指令，以基於接收到至少一個真實金鑰來產生至少一個轉換後的金鑰。至少一個轉換後的金鑰是至少一個回合金鑰(round key)的加密版本，該回合金鑰由處理器使用至少一個真實金鑰加密。處理器能夠將至少一轉換後金鑰解密，並將該至少一回合金鑰加密。資料在為基於主機的加密創建的那一刻就被加密。

【0211】本發明的實施例可以減輕許多竊取底層金鑰的硬體和軟體攻擊情境。對稱金鑰保護的設計提出了增強對稱金鑰保護的建議。例如，該設計包括對AES-NI實作的增強，使AES-NI能夠用於AES演算法的高效實現，並具有額外的好處，即底層金鑰不會暴露在記憶體中，因此減輕了一些使用AES-NI代替專用硬體加密加速器之風險。應該注意的是，對AES-NI的提議增強是本設計的例示性實作，並且它對實作對稱加密操作(或加密操作的部分，如多回合)的任何指令集都具有類似的能力。

【0212】為了在存在軟體漏洞時減少金鑰材料的丟失，本設計包括一種使用加密形式的AES回合金鑰的方法，該回合金鑰被傳遞給AESNI指令，而不是傳給真正的回合金鑰。這些回合金鑰由CPU使用在電源初始化時隨機推衍的對話金鑰或每個CPU(或系統單晶片(SOC))的唯一金鑰通過基於融合的推衍金鑰在內部推衍和持久化進行加密。還可以使用融合(fuse)、暫存器轉移語言(RTL)中的隱藏金鑰以及可能的實體上不可複製功能的組合來推衍出每個CPU/SOC的推衍金鑰。

【0213】下面描述了潛在的實作，並且可以在平台上實現任何一種或多於一種的任何組合。

【0214】圖18描繪根據一個實施例在執行基於主機的加密時保護對稱加密金鑰的電腦實現的方法之一實施例的流程圖。該方法由處理邏輯執行，處理邏輯可以包括硬體(電路、專用邏輯等)、軟體(例如在通用電腦系統或專用機

器或裝置上運行)或兩者的組合。在一個實施例中，該方法由與這裡討論的基於主機的平台相關聯的處理邏輯執行。

【0215】在區塊1802，處理邏輯從安全區域取回真實金鑰(例如，從可信平台模組(TPM)解封真實金鑰)。在區塊1804處，處理邏輯執行輔助金鑰產生指令(例如，AESKEYGENASSIST)，其協助金鑰產生(例如，AES回合金鑰產生、完整金鑰)。在區塊1806，處理邏輯執行金鑰轉換指令，該指令接收金鑰作為輸入並產生轉換後金鑰作為輸出(例如，加密的回合金鑰、加密的完整金鑰)。在區塊1808處，在早期平台啟動的多個階段期間將轉換後的金鑰儲存在記憶體中。轉換後的金鑰僅對當前啟動週期和平台有效。在區塊1810，從系統記憶體中抹除金鑰(例如，回合金鑰、完整金鑰)。

【0216】當主機平台從重置啟動時，碼路徑的數量會增加並積累作業系統的編碼基底(code base)、裝置驅動器和其他大量包含當前或未來潛在漏洞的碼。因此，系統有被利用(exploitation)之漏洞的可能性隨著T的變化而增長，其中T等於執行的軟體碼之量，並與重置後的時間量相關。如果關鍵金鑰可以轉換成只能在平台CPU核心上用於此特定啟動的形式，那麼丟失轉換後金鑰不會透露未轉換的金鑰，並且若金鑰被攻擊者竊取則被盜的轉換後金鑰在下次重新啟動後也變得毫無價值。這僅適用於使用隨機數加密這些金鑰的情況，而不是使用在製造時隨機產生並

融合或永久組態到CPU中的金鑰，這些金鑰在每次啟動時都是相同的。

【0217】圖19描繪示出根據本一實施例創建轉換後金鑰之機制的方塊圖。該機制包括在區塊1902從TPM解封真實金鑰，執行協助AES金鑰產生(例如，AES回合金鑰產生)的輔助金鑰產生指令(例如，AESKEYGENASSIST)，以及引入新指令(例如，KEYTRANSFORM)將把區塊1906的真實回合金鑰(例如，真實回合金鑰1、2等)作為輸入並將加密版本的金鑰(例如，TransRound金鑰1、TransRound金鑰2、等等)儲存於記憶體位置1908。用於加密的實際金鑰僅對CPU可用，並且是不可提取。這使得可以在早期啟動的多個階段(包括啟動載入處理執行或OS啟動過程的早期階段)將加密金鑰放置在記憶體中。加密後的金鑰僅對當前啟動週期和平台有效。在另一個實施例中，為加密提供單個金鑰，並且在若干回合之後CPU自動建立回合金鑰並將其儲存。

【0218】假設真正的金鑰被密封在IP中，並且IP只會在啟動過程的這些早期階段釋放它們，例如透過密封平台的可信平台模組(TPM)上的平台組態暫存器。除了硬體偽隨機數(pseudo-random)產生器之外，TPM還提供用於安全產生密碼金鑰及其使用限制的設施。它還包括遠端證明和密封儲存器等功能。在一個實施例中，由於回合金鑰是固定的128位元金鑰並且與AES金鑰尺寸和模式要求無關，所以回合金鑰被轉換而不是完整的本地AES金鑰被轉換。

在一個實施例中，完整金鑰由新指令轉換。一旦執行金鑰轉換指令將加密形式的金鑰儲存在記憶體中，真正的回合金鑰應該從系統記憶體中刪除。「keytransform」指令可以在任何特權級別運行，儘管它通常在CPL0中最有用。

【0219】加密的回合金鑰可以儲存在內部表中。可以設計指令以更容易地在內部表中查找加密的回合金鑰(例如，將回合號傳入並使用回合號在表中查找)。

【0220】在一個實施例中，CPU具有用於加密金鑰的有限儲存並且軟體指定希望使用哪個金鑰的索引。索引的一個優點是因為沒有快取(即，在表中命中或不運作)，該設計可能包括內部性能最佳化機制，比如只允許查找解密的回合金鑰，其條件為當線性位址請求解密的指令與之前記錄的內容相匹配時。在這種情況下，加密指令將讀取「將我的回合金鑰X儲存到表的條目3中」。

【0221】可以創建替代實作，其中內部加密金鑰保存在CPU中。這樣做的好處是，任何應用程式都可以將金鑰綁定到平台，而無需依賴早期啟動碼。最初安裝應用程式時，應用程式將為平台產生轉換後的金鑰，然後不再需要真正的金鑰。這種方法可以防止金鑰在另一個平台上可用，但不能防止被竊取的轉換後金鑰被不良(rogue)的替代惡意程式軟體堆疊使用。

【0222】圖20描繪根據一實施例使用對稱加密金鑰進行加密的電腦實作方法的一實施例之流程圖。該方法由處理邏輯執行，處理邏輯可以包括硬體(電路、專用邏輯、

CPU等)、軟體(例如在通用電腦系統或專用機器或裝置上運行)或兩者的組合。在一個實施例中，該方法是由CPU執行。

【0223】在一個實施例中，為了使用轉換的回合金鑰，將實作四個AES-NI指令(例如，AESENC、AESENCLAST、AESDEC和AESDELAST)的新變體。新指令可以稱為AESENCTR、AESENCLASTTR、AESDECTR和AESDELASTTR。這些變體中的每一個都將與其當前實作相同地操作，只是這些變體將接受轉換的回合金鑰代替本機回合金鑰。例如，在區塊2002，具有變體(AESENCTR xmm1, xmm2/m12)的處理邏輯基於來自[xmm2/m12]的轉換回合金鑰對[xmm1]執行一回合AES加密。在內部，處理邏輯(例如，CPU)在區塊2004處將轉換後的回合金鑰解密為原生回合金鑰，然後在區塊2006處執行其當前實作(例如，AESENC)。

【0224】當AES用於加密或解密一個資料塊時，它將為每16位元組的區塊迭代該等回合金鑰。整個區塊的該組回合金鑰在整個加密或解密操作中保持不變。此特性啟用映射的積極內部快取，該映射將轉換後的回合金鑰映射到其相應的真實回合金鑰。轉換後的回合金鑰的解密只需要在快取未命中時發生。例如，假設設計需要對4K資料塊執行AES-256加密操作，該操作需要14個回合金鑰。在一個實施例中，回合金鑰轉換快取可以容納所有14個金鑰並且一次僅運行一個使用轉換範式(transform paradigm)的AES

操作。作為基準，普通的AES-NI加密需要對AESENC進行 $256 * 14 = 3484$ 次呼叫(例如， $4K/16B = 256$ 且針對256位元AES的14回合)。相比之下，如果假設微架構實作使用AESDEC在快取未命中時解密回合金鑰，那麼將額外呼叫140次AESDEC。每個128位元回合金鑰都使用AES-128加密，這需要10回合，因此 $14 * 10 = 140$ 總共3724次AESENC/AESDEC呼叫，這大約在處理過的第一個區塊上為7%的負擔(overhead)(加上轉換快取查找之負擔)。需要注意的是，資料集中的所有後續4K區塊都不會導致快取未命中，並將返回到最初的3484次呼叫。因此，該負擔僅包括轉換金鑰之快取查找。

【0225】在另一個實施例中，使用表或其他結構代替快取。CPU可能想要儲存從它開始時使用的隨機數/融合所產生的回合金鑰，而不是儲存實際的隨機數/融合。當需要加密或解密軟體指定的回合金鑰時，擁有這些基於硬體的回合金鑰會顯著加快速度。如果不存在基於硬體的回合金鑰，則不需要原始隨機數或融合。隨機數可以與其他東西(例如，CPU上的融合或隱藏值)組合，以防有人能夠擾亂隨機數。以類似的方式，融合可以與其他東西組合。

【0226】目前，對稱加密可以發生在加速器(或IP區塊)中，在這種情況下可以保護金鑰。在可能更適合某些工作量的當前軟體方法中，此處描述的攻擊不存在金鑰保護特性。

【0227】在加密或解密演算法的當前軟體或AES-NI

實作中，作業系統或應用程式中的安全漏洞使攻擊者能夠竊取加密金鑰和明文資料。即使加密資料在如下所述的事件期間被洩露或被竊取，提供增強的金鑰保護也具有重大價值。當透過漏洞之利用為未知時，這種增強的保護可最大限度地減少漏洞的影響。作業系統中已識別漏洞的挑戰之一是並不總是清楚是否發生了透過漏洞之利用。當組織識別漏洞之存在時，可以採取糾正措施 (correction action)，例如部屬軟體修補(patch)以防止漏洞被利用。與未知漏洞利用相關的一個問題是，組織不一定知道惡意攻擊者使用了哪些平台(如果有使用得化)。除了能夠使用解密實作來存取資料之外，作業系統漏洞還可以提供對其中對稱金鑰儲存在當前軟體實作中的記憶體之存取。因此，若組織欲確保攻擊者無法使用可能被竊取之金鑰，則組織將需要改變在受影響的所有平台上之加密金鑰並且將每個有漏洞的裝置上的所有資料重新加密。如果金鑰被竊取，則這些金鑰不僅可用於漏洞發生時存在的資料，還可用於存取將使用相同金鑰加密的未來資料。如果加密能力的實作方式使得在有漏洞期間僅令解密實作易受攻擊，但金鑰受到保護，則可以在不撤銷金鑰的情況下修補系統，並且無需重新加密所有當前資料。當將通用軟體應用於PC客戶端、手機、平板電腦和消費電子裝置等大型裝置時，這是一個巨大的好處，因為OS漏洞將影響非常多的使用者群，而底層對稱金鑰的保護將大幅降低將平台恢復到已知安全狀態之成本與努力。

【0228】提供增強的金鑰保護具有重大價值，因為解密可以移動到另一個平台或軟體堆疊。獲取金鑰使攻擊者能夠在安全性漏洞系統(compromised system)之外的輔助實作中實施解密，並且不限於使用安全性漏洞的實作作為監控或強制將資料從加密轉換為明文的唯一手段。例如，考慮經典的「冷啟動」攻擊。在這種情境下，一旦獲得金鑰，攻擊者就可以將機器重新啟動到OS並存取/解密平台上的任何資料。如果漏洞是允許存取解密實作的軟體漏洞，則攻擊者可以監控資料轉換，但對於攻擊者期望轉換的資料將需要強制將資料通過安全性漏洞實作。這在許多情境下是不實際的特別考慮到巨量資料時，並且無從得知將存取之有價值資料是位於何處。可以證明這一點的一種情境是系統在啟動後的某個時間點識別出漏洞。在其中漏洞可以竊取金鑰的系統中，在未來任何時間點都有機會可竊取資料。在僅對加密功能的存取有安全性漏洞的情況下，資料丟失將限於可以通過在有漏洞利用和漏洞利用偵測之間的安全性漏洞實作所傳遞的資料量。

【0229】提供增強的金鑰保護具有重大價值，因為對稱金鑰之重新加密與撤銷是昂貴且通常是不實際的。AES中使用的對稱金鑰通常用作最低級別的加密，以保護金鑰所保護的每個資料塊的靜態(儲存)和動態(轉移中)資料。由於存取每個加密的資料塊都需要金鑰，因此撤銷有安全性漏洞的金鑰是具有挑戰性，因為每個已加密的資料塊都需要是可存取的，以便可以使用新的加密材料(keying

material)進行重新加密。這不僅包括受保護系統上的有效資料，還包括可能已保存在其他媒體(例如備份或離線儲存器)中的任何其他副本。金鑰被竊取將導致對所有使用金鑰加密的資料進行攻擊，包括在裝置上無法再存取的資料。

【0230】本設計提供了創建加密回合金鑰(或真正的金鑰)的概念，只有平台的應用處理器才能將其轉換為真實回合(或真正)金鑰。本設計提供了在基於主機的軟體加密操作中使用加密的平台綁定/啟動唯一金鑰、金鑰密封機制(例如TPM)和早期平台啟動之間的交互以在主機執行有限碼時將金鑰移動到轉換狀態以減少受信任的編碼基底(code base)和攻擊區域，以及一種將轉換後的金鑰有效率快取的機制，從而令藉由加密所有回合金鑰所造成負擔最小化。

【0231】在一實施例中，系統包括處理器，該處理器具有對至少一真實金鑰之存取。記憶體經耦接至處理器。記憶體儲存至少一轉換後金鑰，處理器組態用以執行金鑰轉換指令，以根據至少一真實金鑰產生至少一轉換後金鑰。記憶體可以包括快取，用於儲存映射以將轉換後的金鑰映射到基於至少一個真實金鑰的相應回合金鑰。該處理器還被組態以當發生快取未命中時，將轉換後的金鑰進行解密。儲存在快取中的映射引入了轉換金鑰快取查找的有限負擔。在一個實施例中，記憶體具有用於轉換後金鑰的有限儲存以及處理器執行指令以指定用於選擇轉換後金鑰

的索引。記憶體可包括一表或其他結構(取代快取)以儲存映射轉換後金鑰至相應真實金鑰之映射。

【0232】在替代實施例中，處理器被組態以接收單一金鑰並自動產生由處理器儲存和存取的回合金鑰。處理器被組態以回應於產生至少一個回合金鑰以執行金鑰轉換指令以產生至少一個轉換後金鑰。

【0233】在一個實施例中，處理器包括處理單元以從安全區域取回至少一個真實金鑰並基於該至少一個真實金鑰產生至少一個回合金鑰。記憶體經耦接至處理單元。處理單元被組態以基於至少一個回合金鑰，執行金鑰轉換指令以產生至少一個轉換後金鑰。快取儲存該至少一個回合金鑰與該至少一個轉換後金鑰。至少一個轉換後金鑰是由處理器單元使用可以在電源初始化時隨機推衍的對話金鑰(session key)所加密的至少一個加密版本的回合金鑰。處理單元(以及無其他機器)能夠將至少一轉換後金鑰解密，並將該至少一回合金鑰加密。在另一實施例中，至少一個轉換後金鑰是由處理器單元使用處理單元的唯一金鑰(unique key)所加密的至少一個加密版本的回合金鑰，該唯一金鑰是透過推衍金鑰所推衍出並在內部持續的。推衍金鑰可藉由融合以獲得。

用於載入與虛擬化密碼金鑰之實施例

【0234】許多用於使用、轉移和儲存私人或機密資訊的技術都涉及密碼金鑰。因此，可能需要使用本發明的實

施例來為密碼金鑰提供保護並減輕資訊處理系統和裝置受到攻擊的脆弱性。

【0235】圖21描繪系統2100，其為包括本發明實施例用於載入與虛擬化密碼金鑰之資訊處理系統。系統2100可以代表任何類型的資訊處理系統，例如伺服器、桌上型電腦、可攜式電腦、機上盒、諸如平板電腦或智慧型電話的手持裝置、或嵌入式控制系統。系統2100包括處理器2110、系統記憶體2120、圖形處理器2130、週邊控制代理器2140以及資訊儲存裝置2150。實作本發明的系統可以包括任意數量的這些組件中的每一個以及任何其他組件或其他元件，例如週邊裝置和輸入/輸出裝置。本實施例或任何系統實施例中的任何或所有組件或其他元件可以透過任何數量的匯流排、點對點或其他有線或無線介面或連接以相互連接、耦接或以其他方式相互通訊，除非另有規定。系統2100的任何組件或其他部分，無論是否在圖21中示出或未在圖21中示出，都可以整合或以其他方式包括在單個晶片(系統單晶片或SOC)、晶粒、基板、或封裝上或中。

【0236】系統記憶體2120可以是動態隨機存取記憶體(DRAM)或處理器2110可讀的任何其他類型的媒體。圖形處理器2130可以包括用於處理顯示器2132的圖形資料的任何處理器或其他組件。週邊控制代理器2140可以表示任何組件，諸如晶片組組件，包括或透過該組件，週邊、輸入/輸出(I/O)或其他組件或裝置，例如裝置2142(例如，觸控螢幕、鍵盤、麥克風、揚聲器、其他音訊裝置、相機、視

訊或其他媒體裝置、網路配接器、運動或其他感測器、用於全球定位或其他資訊的接收器等)及/或資訊儲存裝置2150，可以連接或耦接至處理器2110。資訊儲存裝置2150可以包括任何類型的持久性或非揮發性記憶體或儲存裝置，例如快閃記憶體及/或固態、磁碟或光碟驅動器。

【0237】處理器2110可以代表一個或多個處理器或處理器核心，該處理器或處理器核心整合在單個基板上或封裝在單個封裝內，每個處理器或處理器核心可以包括任意組合之多個執行緒及/或多個執行核。表示為處理器2110或在處理器2110中表示的每個處理器可以是任何類型的處理器，包括通用微處理器，例如Intel® Core®處理器家族或來自Intel®公司或其他公司的其他處理器家族中的處理器、特殊用途處理器或微控制器，或資訊處理系統中的任何其他裝置或組件，其中可以實現本發明的實施例。處理器2110可以被架構和設計為根據任何指令集架構進行操作，無論有無受到微碼控制。

【0238】可以使用嵌入在硬體、微碼、韌體及/或其他結構中的電路及/或邏輯的任意組合(如下所述配置或根據任何方式配置)，在處理器(例如處理器2110)中實作對根據本發明實施例的載入和虛擬化密碼金鑰的支援，並且在圖21中表示為金鑰管理硬體及/或邏輯2160，其可以包括支援LOADKEY操作的載入金鑰硬體/邏輯2162、支援SAVEKEY操作的保存金鑰硬體/邏輯2164、支持RESTOREKEY指令的恢復金鑰硬體/邏輯2166、本地金鑰

暫存器 2168 和備份金鑰暫存器 2170，每個都在下面描述。

【0239】圖 22 描繪處理器 2200，其可在圖 21 中表示為處理器 2110 的實施例或在圖 21 中表現為處理器 2110 的多核心處理器實施例之執行核心。處理器 2200 可以包括儲存器單元 2210、指令單元 2220、執行單元 2230 和控制單元 2240，其中圖 21 的金鑰管理硬體/邏輯 2160 的電路、結構和功能分佈在處理器 2200 的每個單元中，如下所述。處理器 2200 還可以包括圖 21 中未示出的任何其他電路、結構或邏輯。

【0240】儲存器單元 2210 可以包括可用於處理器 2200 內的任何目的任何類型的儲存器的任何組合；例如，它可以包括使用任何記憶體或儲存器技術實現的任意數量的可讀、可寫及/或可讀寫暫存器、緩衝器及/或快取，其中用以儲存能力資訊、組態資訊、控制資訊、狀態資訊、性能資訊、指令、資料和可在處理器 2200 的操作中使用的任何其他資訊，以及包括可用於存取此類儲存器及/或導致或支援與存取此類儲存器相關聯的各種操作及/或組態的電路。

【0241】儲存器單元 2210 可包括一或多個金鑰來源暫存器 2212、本地金鑰暫存器 2214 及/或備份金鑰暫存器 2216。金鑰來源暫存器 2212 可以是任何尺寸的資料暫存器或其他暫存器或儲存位置，其中在 LOADKEY 操作中可自其載入密碼金鑰、要用作密碼金鑰的值、可從其推衍出密碼金鑰的值或此種金鑰或值之一部分。請注意，雖然金鑰

來源暫存器 2212 是可由軟體讀取，但其與 LOADKEY 操作相關的內容可能受到一或多已知安全技術的保護(例如，藉由將內容載入到其中，執行 LOADKEY 操作，然後藉由或在安全啟動過程、經過身份驗證的碼模組、安全或受信任的執行環境等中清除其全部)。

【0242】本地金鑰暫存器 2214 可以是任何尺寸的登記或其他儲存位置，可以在 LOADKEY 操作中將密碼金鑰或要用作密碼金鑰的值載入到其中。本地金鑰暫存器 2214 可以由硬體保護，使得其內容不會被發現或洩露。例如，根據如下所述的本發明的實施例，可以限制對本地金鑰暫存器 2214 的存取，使得其內容可以被處理器 2200 內的硬體使用(例如，被密碼硬體(cryptographic hardware)用作包裝或其他金鑰)，但僅在與 SAVEKEY 操作相關的情況下被複製或移動，或在與其他提供有限的可存取性的技術(例如，藉由密碼安全引擎、其他特權硬體代理器或實體、或安全或可信執行環境內的韌體或軟體)相關的情況下將其維護。換句話說，本地金鑰暫存器 2214 不能透過使用讀取指令或任何其他軟體可使用以使其內容對處理器 2200 外為公開可讀的指令進行存取。

【0243】備份金鑰暫存器 2216 可以是任何尺寸的登記或其他儲存位置，可以將密碼金鑰或要用作密碼金鑰的值儲存到其中。在本發明的一個實施例中，備份金鑰暫存器 2216 的內容可以跨電源循環持續或保留，例如，在待機、睡眠、暫停、休眠及/或其他電源管理狀態期間。例如，

當處理器 2200 中的其他電路(例如本地金鑰暫存器 2214)可能被斷電時，備份金鑰暫存器 2216 可以從處理器 2200 的封裝上的一個或多個引腳或其他電連接器接收電源。替代地，備份金鑰暫存器 2216 可以在與處理器 2200 相同的晶粒上或相同的封裝中的非揮發性(例如，快閃)記憶體或儲存器中實現。替代地，備份金鑰暫存器 2216 可以在與處理器 2200 相同的晶粒上或相同的封裝中的揮發性記憶體或儲存器中實現，但具有當處理器 2200 或處理器 2200 之部分可能被斷電時，允許其內容被安全(例如，使用加密、認證、完整性檢查及/或重播(replay)保護技術)移動到或複製到其他非揮發性儲存器或將其通電之安全備份機制。各種其他實施例在本發明範疇下為可行的。注意，使用任何這樣的技術在電源循環期間保持備份金鑰暫存器 2216 的內容不需要專用於備份金鑰暫存器 2216；換句話說，為了相同的目的，也可以使用相同的技術來保留其他資訊及/或處理器狀態。

【0244】備份金鑰暫存器 2216 可以由硬體保護，使得其內容不會被發現或洩露。例如，根據如下所述的本發明的實施例，可以限制對備份金鑰暫存器 2216 的存取，使得其內容可以在與 RESTOREKEY 操作相關的情況下被複製或移動，或在與其他提供有限的可存取性的技術(例如，藉由密碼安全引擎、其他特權硬體代理器或實體、或安全或可信執行環境內的韌體或軟體)相關的情況下將其維護。換句話說，備份金鑰暫存器 2216 不能透過使用讀取指令或

任何其他軟體可使用以使其內容對處理器 2200 外為公開可讀的指令進行存取。

【0245】儲存器單元 2210 還可以包括控制暫存器 2218，其可以表示一個或多個暫存器、其他儲存位置、或者暫存器或儲存位置的一部分，其控制及/或組態資訊可以被程式化到這些暫存器或儲存位置中。控制暫存器 2218 可以表示根據現有處理器或指令集架構指定或定義的一個或多個控制暫存器、添加到現有處理器或指令集架構的一個或多個新控制暫存器、或現有或新暫存器中的一個或多個位元/欄位。在各種實施例中，控制暫存器 2218 可以包括一個或多個位元以啟用本發明的一個或多個特徵。例如，控制暫存器 2218 可以包括 `LOADKEY_ENABLE` 位元、`SAVEKEY_ENABLE` 位元及/或 `RESTOREKEY` 啟用位元，其中的每一個可以分別被設置為啟用 `LOADKEY` 操作、`SAVEKEY` 操作及/或 `RESTOREKEY` 操作，及/或可以設置為啟用兩個或多個操作的組合的啟用位元，其中當相應的啟用位元未設置時嘗試使用此等操作將會導致異常。

【0246】在各種實施例中，控制暫存器 2218、不同的控制暫存器或虛擬機控制結構 (`virtual machine control structure`，`VMCS`) 可以包括位元或欄位，該位元或欄位用以使與虛擬機 (`VM`) 退出相關的 `SAVEKEY` 操作會自動發生及/或使與 `VM` 進入 (`VM entry`) 相關的 `RESTOREKEY` 操作會自動發生，如下所述。

【0247】指令單元 2220 可以包括任何電路、邏輯、結

構及/或其他硬體，例如指令解碼器，以獲取、接收、解碼、解譯、排程及/或處理將由處理器2200執行的指令。在本發明的範圍內可以使用任何指令格式；例如，一指令可以包括一個運算碼和一個或多個運算元，該運算碼可以被解碼為一個或多個微指令或微操作以供執行單元2230執行。運算元或其他參數可以隱含地、直接地、間接地或根據任何其他方法與指令相關聯。

【0248】在一個實施例中，指令單元2220可以接收LOADKEY指令、SAVEKEY指令及/或RESTOREKEY指令，下面描述處理器2200的執行。注意，提供這些指令的名稱僅僅是為了方便，並且本發明的實施例可以包括具有任何期望名稱的這些指令。在各種實施例中，LOADKEY、SAVEKEY和RESTOREKEY指令可以是指令單元230接收到的不同指令，它們可以是相同指令的葉，或者它們可以是具有不同運算元的相同指令。注意，描述了本發明的各種實施例並且更多的實施例是可能的，包括每個指令的參數和處理器對每個指令的回應的各種差異。還請注意，處理器對這些指令的執行及/或回應，如下所述，也可以或替代地執行(以下稱為LOADKEY、SAVEKEY或RESTOREKEY操作)以回應於命令、模型特定、或其他暫存器的位元或與其他事件有關(例如，寫入或設置(藉由軟體、韌體或硬體)；VM進入或退出(無論是由指令、中斷、異常還是其他事件觸發；等)，其中一些特徵將在下面描述。本發明的實施例可包括此等變體之任

一或多者。

【0249】在一個實施例中，LOADKEY指令可以包括一個或多個來源運算元以指定要用作密碼金鑰的來源、將用作密碼金鑰的其他值的來源、或可以從其推衍出密碼金鑰的其他值的來源。如果來源運算元未指定有效來源，則藉由引起異常，各種實施例可以對任何組或子集之暫存器提供一處理器用作來源。在圖22的實施例中，為了方便起見，示出了金鑰來源暫存器2212，並且可以是任意數量的有效來源暫存器之一。

【0250】在其他實施例中，將與LOADKEY操作相關聯(如下文所述)而載入之密碼金鑰(或將用作密碼金鑰之其他值或從其可推衍出密碼金鑰之其他值)可被另一來源提供(例如，隨機數產生器、融合陣列、物理上無法複製函數等)，在該情況下，可使用LOADKEY指令而無來源運算元，或具有不包括來源運算元欄位之格式。注意，本發明的實施例可以為處理器提供使用預設金鑰來源(例如，硬體)，該預設金鑰來源(例如，硬體)可以使用LOADKEY指令被置換(例如，藉由軟體)。

【0251】在一實施例中，指令單元2220還可或替代地接收一個或多個指令以存取一個或多個暫存器，例如本地金鑰暫存器2214及/或備份金鑰暫存器2216，其中對暫存器的存取可導致資料移入或移出暫存器，從或到另一個暫存器，例如金鑰來源暫存器2212、本地金鑰暫存器2214及/或備份金鑰暫存器2216，而任一暫存器的內容都不能被

軟體存取。這種指令可以是處理器的指令集架構中的指令，其可用於存取一個或多個其他暫存器或其他儲存位置。例如，這樣的指令可以是讀取、寫入或移動指令，其具有一個或多個運算元以指定要讀取、寫入或移入或移出內容或複製內容至其的暫存器或其他儲存位置。當這樣的運算元指定除了本地金鑰暫存器 2214 或備份金鑰暫存器 2216 之外的暫存器時，這樣的暫存器的內容可以被發出指令的軟體存取。然而，當這樣的運算元指定本地金鑰暫存器 2214 或備份金鑰暫存器 2216 時，執行單元 2230 中的硬體可以提供要在金鑰來源暫存器 2212、本地金鑰暫存器 2214 及 / 或備份金鑰暫存器 2216 之間移動的內容，其中該內容由發出指令之軟體或任何其他軟體是不可讀或無法觀察的 (在 LOADKEY、SAVEKEY 及 / 或 RESTOREKEY 操作期間或與其相關)。

【0252】執行單元 2230 可以包括任何電路、邏輯、結構及 / 或其他硬體，例如算數單元、邏輯單元、浮點單元、移位器等，以處理資料和執行指令、微指令及 / 或微操作。執行單元 2230 可表示任何一個或實體上或邏輯上不同之執行單元。執行單元 2230 可以包括用於執行 LOADKEY 指令 / 操作的載入金鑰硬體 2232、用於執行 SAVEKEY 指令 / 操作的保存金鑰硬體 2234 和用於執行 RESTOREKEY 指令 / 操作的恢復金鑰硬體 2236，各者將如下文進一步描述。儘管在圖 22 中被示為單獨的元件，載入金鑰硬體 2232、保存金鑰硬體 2234 及恢復金鑰硬體 2236 可

以包括也用於執行其他指令及/或操作的電路。

【0253】執行單元2230也可包括密碼硬體2238，以實作任一或多密碼演算法。在一個實施例中，本地金鑰暫存器2214可以固線至或以其他方式連接到密碼硬體2238，使得本地金鑰暫存器2214的內容可以用作這種演算法中的密碼金鑰，而不會將本地金鑰暫存器2214的內容暴露給軟體。

【0254】在一個實施例中，載入金鑰硬體2232可以回應於LOADKEY指令執行LOADKEY操作，其中LOADKEY操作包括將金鑰值載入、移動或複製到處理器2200中的私有內部儲存位置，從該位置，處理器2200中的硬體可以存取在任何特殊用途密碼硬體或通用硬體中用作密碼或其他金鑰的內容，用這些硬體或通用硬體可以執行密碼或其他基於金鑰的演算法，但軟體不能從中讀取內容。例如，回應於LOADKEY指令，載入金鑰硬體2232可以將來源暫存器的內容載入到本地金鑰暫存器2214中以供密碼硬體2236使用。

【0255】載入金鑰運算之各種實施例在本發明範疇下為可行的。在各種實施例中，金鑰值可以是包裝金鑰、對話金鑰、任何其他類型的金鑰、或可以從中推衍任何此類金鑰的任何值。在一個實施例中，金鑰值可以從一個或多個由LOADKEY指令的來源運算元指定的暫存器載入。在其他實施例中，金鑰值可以是任何其他軟體指定的值，例如，來自處理器內的另一個儲存位置或來自系統記憶體中

的指定位置的值。在其他實施例中，金鑰值可以由隨機數產生器、融合陣列、物理上無法複製函數或一些其他來源提供。本發明的實施例可以為處理器提供使用預設金鑰來源(例如，硬體)，該預設金鑰來源(例如，硬體)可以在LOADKEY操作中被置換(例如，藉由軟體)。

【0256】在一個實施例中，載入金鑰硬體2232還可以或替代地回應於任何LOADKEY命令執行LOADKEY操作，其中LOADKEY命令可以包括一個或多個如上所述的暫存器存取指令，例如寫入本地金鑰暫存器2214，或由處理器或指令集架構指定的任何其他指令或命令，該指令用於引起LOADKEY操作。

【0257】在一個實施例中，保存金鑰硬體2234可以回應於SAVEKEY指令執行SAVEKEY操作，其中SAVEKEY操作包括將核心範圍或邏輯處理器範圍金鑰的內容移動或複製到封裝範圍的金鑰儲存器。例如，回應於SAVEKEY指令，保存金鑰硬體2234可以將本地金鑰暫存器2214的內容移動/複製到備份金鑰暫存器2216。

【0258】在一個實施例中，保存金鑰硬體2234還可以或替代地回應於任何SAVEKEY命令執行SAVEKEY操作，其中SAVEKEY命令可以包括一個或多個如上所述的暫存器存取指令，例如寫入備份金鑰暫存器2216，或由處理器或指令集架構指定的任何其他指令或命令，該指令用於引起SAVEKEY操作。

【0259】在一個實施例中，恢復金鑰硬體2236可以回

應於 **RESTOREKEY** 指令執行 **RESTOREKEY** 操作，其中 **RESTOREKEY** 操作包括將封裝範圍的金鑰儲存器之內容移動或複製到核心範圍或邏輯處理器範圍金鑰儲存器，再移動或複製到封裝範圍的金鑰儲存器。例如，回應於 **RESTOREKEY** 指令，恢復金鑰硬體 2236 可以將備份金鑰暫存器 2216 的內容移動/複製到本地金鑰暫存器 2214。

【0260】在一個實施例中，恢復金鑰硬體 2236 還可以或替代地回應於任何 **RESTOREKEY** 命令執行 **RESTOREKEY** 操作，其中 **RESTOREKEY** 命令可以包括一個或多個如上所述的暫存器存取指令，例如從備份金鑰暫存器 2216 讀取，或由處理器或指令集架構指定的任何其他指令或命令，該指令用於引起 **RESTOREKEY** 操作。

【0261】在各種實施例中，保存金鑰和恢復金鑰命令和操作可以以不同方式與指令相關聯。例如，第一指令的單次執行可能僅導致保存金鑰命令和操作，而第二指令的單次執行可能僅導致恢復金鑰命令和操作。或者，第三指令的單次執行可能導致保存金鑰命令和操作以及其他命令及/或操作，例如處理器狀態的其他元素的保存，並且第四指令的單次執行可能導致恢復金鑰命令和操作以及其他命令及/或操作，例如處理器狀態的其他元素的恢復。各種實施例可以包括任何數量和種類之指令和指令格式，其可以獨立地或與任何數量的其他命令及/或操作相關聯地導致保存金鑰和恢復金鑰之命令和操作。

【0262】保存金鑰硬體 2234 和恢復金鑰硬體 2236 可以

回應於保存金鑰和恢復金鑰命令執行保存金鑰和恢復金鑰操作，使得在本地金鑰暫存器 2214 和備份金鑰暫存器 2216 之間移動/複製的資料無法被軟體讀取。

【0263】控制單元 2240 可以包括任何微碼、韌體、電路、邏輯、結構及/或硬體以控制處理器 2200 的單元和其他元件的操作以及控制在處理器 2200 內的資料轉移、資料轉移到處理器 2200 內以及資料轉移到處理器 2200 外。控制單元 2240 可以例如藉由使處理器 2200 使用執行單元 2230 及/或任何其他資源來使處理器 2200 執行或參與本發明的方法實施例的執行，例如下面描述的方法實施例，以執行由指令單元 2220 接收到的指令和由指令單元 2220 接收到的指令所推衍出的微指令或微操作。藉由執行 2230 對指令的執行可以基於儲存在儲存器單元 2210 中的控制及/或組態資訊而變化，其中這種控制及/或組態資訊可以儲存在或程式化到諸如控制暫存器 2218 的控制/組態暫存器中。

【0264】在一個實施例中，控制單元 2240 可以包括金鑰管理控制硬體/邏輯 2242 以控制載入金鑰硬體 2232、保存金鑰硬體 2234、恢復金鑰硬體 2236、處理器 2200 中支援以下本發明實施例的金鑰管理技術的任何其他硬體。

【0265】在一個實施例中，金鑰管理控制硬體/邏輯 2242 可以回應於在虛擬機 (VM) 中執行 LOADKEY 指令的嘗試以導致或可組態以導致虛擬機退出。因此，虛擬機監控 (VMM) 可能會攔截 LOADKEY 指令，以便管理訪客軟體對加密金鑰的使用。

【0266】例如，在核心或邏輯處理器上的VM中運行的訪客可能會嘗試使用LOADKEY指令將第二個密碼金鑰載入到與核心或邏輯處理器相應的本地金鑰暫存器中(其中第一個密碼金鑰已被保留供VMM使用；例如，參見下面圖24的描述)。作為回應，金鑰管理控制硬體/邏輯2242可以導致VM退出，允許VMM攔截LOADKEY指令。然後，VMM可以從來源暫存器複製第二個密碼金鑰，將其載入到本地金鑰暫存器中(使用其本身的LOADKEY指令或透過VM進入，包括將第二個密碼金鑰從VMCS中儲存的訪客狀態載入本地金鑰暫存器中)，以及如下文所述接續地使用其(例如，上下文切換、遷移等)。

【0267】由於多種原因中的任何一原因，可能需要這種方法。在一個實施例中，它可以規定為訪客和主機使用單獨的金鑰(例如，供訪客使用的第二金鑰和供主機金鑰使用的第一金鑰)，及/或為每個訪客使用單獨的金鑰。在一個實施例中，它可以促進上下文切換，其中VMM將訪客從一核心移動到另一核心或從一邏輯處理器移動到另一邏輯處理器，因為它為每個訪客提供自己的金鑰，該金鑰不特定於任何特定核心或邏輯處理器，並且VMM可以從一核心移動到另一核心或一邏輯處理器到另一邏輯處理器。在一個實施例中，它可以提供在VMM之間訪客的遷移；例如，第一個VMM可以將特定於訪客的金鑰載入到訪客的本地金鑰暫存器中，然後，可以將相同的特定於訪客的金鑰(例如，在VMCS中)提供給訪客遷移到的第二個

VMM。

【0268】其他本發明實施例之使用為可行的。例如，VMM可以攔截訪客的LOADKEY指令(例如，將第二個金鑰載入到本地金鑰暫存器中以便在VM內使用)，然後將第三個金鑰載入或導致將第三個金鑰載入到本地金鑰暫存器中以便在VM內使用(而不是第二個金鑰)。

【0269】儘管圖22示出了與本發明的實施例相關聯的各種硬體元件中的每一個的單個實例，但是各種實施例可以包括任意數量的這樣的元件。舉例而言，圖23描繪在多核心處理器2300中本發明之實施例。多核心處理器2300可包括任何數量的執行核心，每個核心具有其自己的儲存器單元、指令單元、執行單元、控制單元及/或任何其他單元，及/或此種單元(諸如第二級別快取)之任何電路、結構、邏輯及/或硬體，可在多核心處理器2300之「非核心」區域共享。

【0270】多核心處理器2300包括核心2310、核心2320、核心2330和核心2340，每個核心可分別包括金鑰管理單元2312、2322、2332及2342、分別包括金鑰來源暫存器2314、2324、2334及2344以及分別包括本地金鑰暫存器2316、2326、2336及2346。金鑰管理單元2312、2322、2332和2342中的每一個可以表示及/或包括上述硬體及/或控制邏輯以支援本發明實施例的金鑰管理技術。

【0271】多核心處理器2300還可以包括一個或多個備份金鑰暫存器2350，其可以由與核心2310、2320、2330及

/或 2340 中的任一個或所有核心分開的電源平面供電(使得可在不從備份金鑰暫存器 2350 移除電源的情況下可從核心 2310、2320、2330 及 / 或 2340 中任一個或所有者移除電源)，或者可以是非揮發性的，或者可以將其內容安全地複製並儲存在別處。因此，本發明的實施例提供藉由各種電源管理週期及 / 或可以從一個或多個處理器核心移除電源的狀態來維護本地密碼金鑰的值。例如，系統軟體，例如作業系統，可以與進入電源管理狀態(其中將從核心移除電源)相關，使用保存金鑰命令將本地金鑰暫存器 2316、2326、2336 及 / 或 2346 中的任何一個的內容保存在備份金鑰暫存器 2350 中。然後，與恢復核心的電源相關，系統軟體可以使用恢復金鑰命令以將內容從備份金鑰暫存器 2350 恢復到本地金鑰暫存器 2316、2326、2336 及 / 或 2346 中的任何一個。在一個實施例中，備份金鑰暫存器 2350 可以藉由重置以被清除。

【0272】此外，根據本發明實施例的 SAVEKEY 及 RESTOREKEY 操作的使用可以藉由備份金鑰暫存器 2350 提供一個或多個金鑰在一個或多個核心 2310、2320、2330 及 / 或 2340 之間移動及 / 或共享，無需複製到 DRAM 或系統記憶體。

【0273】根據本發明各種實施例之金鑰管理硬體之使用可以是被期望的，以提供根據本發明各種方法實施例的各種密碼金鑰管理技術。

【0274】一個實施例可以包括在不允許軟體讀取金鑰

的情況下藉由軟體使用密碼金鑰。例如，主機或系統軟體可以使用LOADKEY命令將金鑰載入到核心之本地金鑰暫存器中，以供在該核心上運行的訪客或應用程式軟體使用。

【0275】一實施例可包括由軟體進行密碼金鑰之產生或修改，以提供客製化金鑰產生。例如，軟體可以使用任何所需的金鑰產生方法，單獨或結合任何根金鑰、金鑰產生指令及/或硬體提供的隨機數產生支援，以產生自定義金鑰(custom key)或修改(例如，對硬體或硬體產生金鑰提供隨機性)硬體或硬體產生金鑰，然後使用LOADKEY命令將自定義金鑰載入到本地金鑰暫存器中。

【0276】一個實施例可以包括將密碼金鑰從一個執行核心、邏輯處理器或硬體執行緒遷移到不同的執行核心、邏輯處理器或硬體執行緒。例如，VMM可以使用保存金鑰命令來保存與在第一個核心上運行的訪客的VM退出相關的訪客金鑰，然後在其他核心上重新啟動訪客之前使用恢復金鑰命令將相同的金鑰恢復到不同的核心。本發明的實施例提供在不允許VMM讀取金鑰的情況下執行這樣的上下文切換。

【0277】一個實施例可以包括單獨的特定於訪客金鑰和單獨的主機金鑰的使用和管理。舉例而言，圖24描繪用於載入與虛擬化密碼金鑰之方法2400。儘管本發明的方法實施例在這方面不受限制，但可以參考圖21、22和23的元素來幫助描述圖24的方法實施例。方法2400的各個部分可

以由硬體、韌體、軟體及/或系統或裝置的使用者來執行。

【0278】在方法2400的方塊2410中，啟動資訊處理系統。在方塊2412中，諸如安全的基本輸入/輸出系統(BIOS)之類的可信系統韌體可以在安全啟動過程期間導致將第一密碼金鑰載入到處理器核心的本地金鑰儲存位置。藉由使用融合、金屬連接(metal tie-up或tie-down)或使用任何暫存器轉移語言技術，可以直接從程式化在或嵌入在處理器中的硬體、真實或根金鑰(例如，使用處理器的專用金鑰產生或轉換指令)、或從由可信平台模組提供的金鑰，複製或推衍出第一密碼金鑰。第一金鑰的產生、推衍或轉換可以包括使用硬體隨機數產生器、實體上不可複製的功能(unclonable function)或其他混淆技術來注入隨機性。本地金鑰儲存位置可由處理器的密碼硬體(cryptographic hardware)存取，但不能由軟體讀取。

【0279】在方塊2420中，處理器核心的控制可以轉移到VMM。在方塊2422中，VMM可以在處理器核心上執行指令，包括使用第一金鑰作為主機金鑰以加密及/或解密主機特定資訊。在方塊2424中，VMM可以使用SAVEKEY命令將第一金鑰從本地金鑰儲存位置複製到處理器非核心中的備份金鑰儲存位置，而無法讀取金鑰的值。在方塊2426中，為準備VM進入，VMM可以清除或替換本地金鑰儲存位置的內容以防止訪客使用該第一金鑰。在方塊2428中，VMM可以在處理器核心上的VM中開始或恢復訪客的

執行。應注意，方塊 2426 可以結合方塊 2428 自動執行。

【0280】在方塊 2430 中，訪客可以執行一個或多個金鑰產生、轉換或推衍(derivation)指令或操作以在金鑰來源暫存器中產生第二密碼金鑰。在方塊 2432 中，訪客可以嘗試 LOADKEY 指令以將第二金鑰從金鑰來源暫存器載入到本地金鑰儲存位置以用於加密及/或解密訪客特定資訊。在方塊 2434 中，LOADKEY 嘗試可能導致 VM 退出到 VMM。

【0281】在方塊 2440 中，VMM 可以從嘗試的 LOADKEY 指令的來源運算元所指定的金鑰來源暫存器中讀取第二金鑰的值。在框 2442 中，VMM 可以將第二金鑰的值儲存在 VMCS 的訪客金鑰欄位或其他記憶體位置中(其可以由任何已知技術保護，例如但不限於存取控制機制，該存取控制機制包括使用擴充或巢套頁表)以用作訪客特定的金鑰。在方塊 2444 中，VMM 可以使用 LOADKEY 指令將訪客特定的金鑰從金鑰來源暫存器載入到本地金鑰儲存位置。在方塊 2446 中，VMM 可以在 VM 中恢復訪客的執行。注意，方塊 2446 可以包括從 VMCS 的訪客金鑰欄位自動載入訪客特定的金鑰，在這種情況下可以省略方塊 2444。在方塊 2448 中，訪客可以在 VM 中執行，包括使用來自本地金鑰儲存位置的訪客特定的金鑰來加密及/或解密資訊，而無法讀取該金鑰的值。

【0282】在方塊 2450 中，可以發生第一電源管理事件，作為該事件的結果是系統將進入電源管理狀態，在該

狀態中，電源將從支援訪客正在其中執行的 VM 的處理器核心移除。在方塊 2452 中，第一電源管理事件導致 VM 退出 VMM。在方塊 2454 中，處理器核心被斷電(但備份金鑰儲存位置或儲存 VMCS 或訪客特定的金鑰之記憶體並沒有被斷電)。

【0283】在方塊 2460，第二電源管理事件可發生，其結果會導致處理器核心被恢復通電。在方塊 2462，處理器核心被恢復通電。在方塊 2464 中，VMM 恢復在處理器核心上執行。在方塊 2466 中，VMM 使用恢復金鑰命令將主機金鑰從備份金鑰儲存位置複製到本地金鑰儲存位置，而無法讀取金鑰的值。在方塊 2468 中，VMM 可以在處理器核心上執行指令，包括使用主機金鑰以加密及/或解密主機特定資訊。

【0284】在方塊 2470 中，為準備 VM 進入，VMM 可以將訪客特定的金鑰的值從其記憶體位置複製到金鑰源極暫存器。在方塊 2472 中，VMM 可以使用 LOADKEY 指令將訪客特定的金鑰從金鑰來源暫存器載入到本地金鑰儲存位置中。在方塊 2474 中，VMM 可以在 VM 中恢復訪客的執行。注意，方塊 2474 可以包括從 VMCS 的訪客金鑰欄位自動載入訪客特定的金鑰，在這種情況下可以省略方塊 2470 與 2472。在方塊 2476 中，訪客可以在 VM 中執行，包括使用來自本地金鑰儲存位置的訪客特定的金鑰來加密及/或解密資訊，而無法讀取該金鑰的值。

軟體程式化介面實施例

【0285】下面描述了與根據實施例及/或實作所包括細節(其若干或所有者被包括在實施例中)之某個特徵相關的指令集擴充的軟體程式化介面(可稱為 **Key Locker**(金鑰鎖)，被描述為對特定指令集架構(**Intel Architecture**或**IA**)的擴充，為方便起見，但不限制其範圍)。根據需要對細節的任何描述意味著在特定實現中可能需要這樣的細節，但並不意味著在本發明的所有或任何其他實施例中需要任何這樣的細節。此外，任何特定名稱的使用，例如 **Key Locker**及/或指令、暫存器等的名稱，並不將細節和描述限制為所描述的實作及/或實施例，換言之，細節和描述可或者可能不會在本說明書中其他地方描述的實施例中使用及/或實現。

【0286】**Key Locker**藉由將**AES**金鑰轉換為「控制代碼」，提供了一種使用**AES**金鑰加密和解密資料的機制，而無需存取原始金鑰值。這些控制代碼可用於執行與原始**AES**金鑰相同的加密和解密操作，但它們僅適用於當前系統，並且僅在它們被撤銷之前有效。如果軟體撤銷**Key Locker**控制代碼(例如，在重新啟動時)，則不能再使用任何以前的控制代碼。

【0287】大多數攻擊者無法竊取實際的**AES**金鑰，除非在軟體請求創建金鑰控制代碼的短時間內。一旦創建了金鑰控制代碼，就可以從記憶體中抹除包裝在這些控制代碼中的原始金鑰。

【0288】如果OS選擇在每次重啟時撤銷控制代碼的策略，則使得可能被盜竊的控制代碼在重啟後對攻擊者不再有用。

【0289】可以創建的金鑰控制代碼數量並沒有任意限制。內部包裝金鑰用於創建控制代碼，每個控制代碼本質上都是底層AES金鑰的加密形式。內部包裝金鑰可以由特權軟體創建及載入，也可以由CPU隨機產生，因此它永遠不會透露給軟體。

【0290】在許多平台上，軟體可以備份當前的內部包裝金鑰也可以恢復它。這可以使OS能夠跨S3(睡眠)和S4(休眠)系統睡眠狀態保存和恢復金鑰，並提供一種在整個平台上分發內部包裝金鑰而不將其放入記憶體的方法。

【0291】軟體密碼庫可能能夠使用Key Locker指令，而無需從根本上改變它們的API，這為軟體提供一種簡單的方法來提高其AES金鑰的安全性，而無需直接新增對Key Locker指令的支援。

基本指令與用途

【0292】Key Locker包含三種類型的指令：

【0293】用以從AES金鑰(ENCODEKEY128以及ENCODEKEY256)創建控制代碼的指令。

【0294】使用控制代碼執行AES加密或解密的指令(AESDEC128KL、AESDEC256KL、AESDECWIDE128KL、AESDECWIDE256K、AESENC128KL、AESENC256KL、

AESENCWIDE128KL以及AESENCWIDE256KL)。

【0295】用以載入內部包裝金鑰(LOADIWKEY)的指令。

用以創建控制代碼的指令

【0296】Key Locker添加了兩條指令，用於獲取AES金鑰並創建控制代碼。他們還獲得有關請求如何使用控制代碼之何者限制條件之輸入。他們使用當前IWKey創建控制代碼。輸出控制代碼包含AES金鑰的加密版本以及元資料。整個控制代碼是完整性保護的，這樣控制代碼的任何修改(例如，編輯元資料)或以不同的IWKey使用控制代碼都將被偵測到。

【0297】ENCODEKEY128將128位元AES金鑰作為輸入並產生384位元控制代碼。

【0298】ENCODEKEY256將256位元AES金鑰作為輸入並產生512位元控制代碼。

【0299】許多軟體使用將希望在產生控制代碼後覆寫AES金鑰，以便使以後的漏洞僅限於被盜的控制代碼，而不是原始AES金鑰。

【0300】下面提供了有關用於創建控制代碼的演算法的更多詳細細節。

【0301】除了產生控制代碼之外，ENCODEKEY*指令還指示載入的IWKey的類型和零暫存器XMM4、XMM5和XMM6。未來對Key Locker的增強可能會為XMM4-6產生

非零值(例如，指示有關指定IWKey的更多資訊)。

控制代碼限制條件

【0302】各控制代碼包括AAD(附加認證資料)欄位，其為完整性受保護但不加密的。其被用於保存控制代碼之元資料，包括其限制條件。

【0303】當經由ENCODEKEY*指令之一創建Key Locker控制代碼時，SW可以藉由設置控制代碼的AAD欄位的指示位元來指定以下限制條件：

【0304】僅Ring 0(AAD的位元0)：控制代碼只有在CPL 0(監督者模式)下為可用；其在應用模式(CPL>0)下為不可用。

【0305】無加密(AAD的位元1)：控制代碼不能用於加密。

【0306】無解密(AAD的位元2)：控制代碼不能用於解密。

【0307】可以在單個控制代碼中設置多個限制條件位元。控制代碼限制條件失敗(包括AAD保留位元設置)將導致AES*KL指令設置RFLAGS.ZF並且不執行請求的加密或解密。

【0308】僅Ring 0的控制代碼可能對不供應用程式使用的OS金鑰有用。如果惡意應用程式有辦法竊取到這樣的控制代碼，它將無法在應用程式本身中使用它。請注意，Ring 0控制代碼可以在任何權限級別創建，儘管其僅

可用於 Ring 0 的加密 / 解密。

【0309】當協定的一側只需要創建訊息(帶加密)而協定的另一側只需要讀取訊息(帶解密)時，不解密和不加密的控制代碼可能成對使用。這將需要使用同時使用 AES 加密和解密的 AES 模式(例如，AES-CBC)，而不是僅使用加密的 AES 模式(例如，AES-CTR)。

用已使用控制代碼以執行 AES 加密或解密之指令

【0310】Key Locker 指令可以接受一個控制代碼和明文或密文，並對其進行加密 / 解密。表 4 示出不同 Key Locker 指令的詳細細節。

指令	AES 金鑰尺寸	加密或解密	單一128位元區塊對比八個128位元區塊
AESENC128KL	128 位元	加密	Non-wide(非寬)(單一128位元區塊)
AESENCWIDE128KL			Wide(寬)(八個128位元區塊)
AESDEC128KL		解密	Non-wide(非寬)(單一128位元區塊)
AESDECWIDE128KL			Wide(寬)(八個128位元區塊)
AESENC256KL	256位元	加密	Non-wide(非寬)(單一128位元區塊)
AESENCWIDE256KL			Wide(寬)(八個128位元區塊)
AESDEC256KL		解密	Non-wide(非寬)(單一128位元區塊)
AESDECWIDE256KL			Wide(寬)(八個128位元區塊)

表 4. Key Locker 指令

【0311】與執行非寬的 Key Locker指令的八次迭代相比，在8個128位元區塊上操作的「寬」指令在並行AES模式(如，AES-CTR)上具有更高的性能。

【0312】在每個Key Locker AES加密/解密操作之後，軟體應該檢查ZF以確保操作沒有失敗(例如，由於控制代碼損壞或限制條件失敗)。當指令失敗時，如果不進行此檢查可能會導致使用明文作為密文(或反之亦然)。

用以載入內部包裝金鑰的指令

【0313】內部包裝金鑰(Key Locker IWKey)用於在控制代碼和原始金鑰之間進行轉換。內部包裝金鑰是由LOADIWKEY指令撰寫的，並且應該對攻擊者保密，以防止他們手動解包控制代碼以獲得得原始金鑰。出於這個原因，雖然有「IWKeyBackup」MSR可用於備份內部包裝金鑰而不洩露其值(參見第4節)，但沒有讀取內部包裝金鑰的操作。

【0314】由於內部包裝金鑰被認為是系統狀態，所以LOADIWKEY只能在監督者模式(CPL 0)下執行。為了支援VM上下文切換和遷移，VMM還可以在訪客執行LOADIWKEY時導致VM退出，以獲得IWKey值。

【0315】建議在OS啟動的早期執行LOADIWKEY，以最大限度地減少軟體已載入的機會，攻擊者可以利用該機會查看LOADIWKEY的資料。

【0316】LOADIWKEY可以指定它載入的內部包裝金

鑰不能透過 **IWKeyBackup MSR** 進行備份。這可用於確保內部包裝金鑰不會因未來在備份或恢復機制中發現的任何缺陷而洩露；但會阻止需要這些 **MSR** 的使用(例如跨 **S3** 或 **S4** 睡眠狀態維護控制代碼)。

【0317】 **LOADIWKEY** 可以直接載入指定的引數，也可以請求硬體產生的隨機內部包裝金鑰。當 **LOADIWKEY** 請求硬體產生的隨機金鑰時，處理器讀取 384 位元的隨機資料(提供由 **RDSEED** 讀取的隨機資料的同一個晶片上的隨機數產生器)並將其與 **LOADIWKEY** 的引數進行 **XOR** 運算。因為資料經 **XOR** 運算處理，故軟體可以將它們自己的熵資料與由硬體隨機位元產生器提供的資料相結合。

【0318】 軟體無法讀取硬體產生的隨機內部包裝金鑰的值。這一事實限制了虛擬機監控(**VMM**)將虛擬機從一個處理器移動到另一個處理器的能力。由於此限制，**VMM** 在虛擬化 **CPUID** 指令時可能會選擇不列舉(**enumerate**)對硬體產生的隨機內部包裝金鑰的支援。

【0319】 可以透過 **MSR** 備份和恢復硬體隨機內部包裝金鑰(除非 **LOADIWKEY** 指定不允許此類備份)。**ENCODEKEY*** 指令的輸出指示用於創建控制代碼的內部包裝金鑰的特性，包括它是否來自晶片上硬體隨機數產生器以及它是否可以透過 **MSR** 進行備份。

IWKey

【0320】 **IWKey** 是 **Key Locker ENCODEKEY*** 和

AES*KL指令使用的內部包裝金鑰。它是邏輯處理器範圍的，並透過LOADIWKEY指令寫入。不能直接用軟體讀取。

【0321】內部包裝金鑰的指令當前包括：

【0322】IntegrityKey[127:0]-一個128位元完整性金鑰，用於檢查控制代碼是否未被篡改。

【0323】EncryptionKey[255:0]-一個256位元加密金鑰，用於包裝/解包以保護控制代碼所指示的金鑰的機密性。

【0324】KeySource[3:0]-唯一允許的值是0(具有軟體指定金鑰的AES GCM SIV包裝演算法)和1(具有硬體強制執行的隨機金鑰的AES GCM SIV包裝演算法)。

【0325】NoBackup旗標-設置後，此IWKey無法備份。

【0326】IWKeyBackup(如下所述)具有與IWKey相同的格式。

CR4.KL

【0327】Key Locker在CR4(位元19)中引入了一個新的“KL”位元，以防止在系統軟體未正確啟用Key Locker時使用它。這也可以防止舊VMM的訪客使用Key Locker。

【0328】CR4.KL的存在由CPUID.KL列舉(CPUID.(EAX=07H, ECX=0H).ECX.KL[bit 23])。

【0329】當CR4.KL為0時，CPUID.AESKLE(CPUID.

19H:EBX[0])也將為0。這將導致所有AES Key Locker指令產生#UD異常，包括AES*KL、ENCODEKEY128、ENCODEKEY256及LOADIWKEY指令。Key Locker IWKeyBackup MSR(稍後描述)不受CR4.KL值的影響，因為它們是由CPUID.KL而不是CPUID列舉的。

控制代碼格式

【0330】128位元AES金鑰的控制代碼尺寸為384位元，並具有格式如下：

- Handle[127:0]=AAD
- Handle[255:128]=Integrity Tag (完整性標籤)
- Handle[383:256]=Ciphertext (密文)

【0331】256位元AES金鑰的控制代碼尺寸為512位元，並具有格式如下：

- Handle[127:0]=AAD
- Handle[255:128]=Integrity Tag (完整性標籤)
- Handle[383:256]=Ciphertext[127:0] (密文)
- Handle[511:384]=Ciphertext[255:128] (密文)

【0332】兩種類型的控制代碼的AAD(附加身份驗證資料)格式具有以下格式：

- AAD[0]=如果CPL>0，控制代碼不可用
- AAD[1]=控制代碼不可用於加密
- AAD[2]=控制代碼不可用於解密
- AAD[23:3]=保留

- AAD[27:24]=金鑰類型。0表示AES-128控制代碼，以及1表示AES-256控制代碼。當前保留所有其他金鑰類型。
- AAD[127:28]=保留

Intel®交易同步延伸(Intel® TSX)操作

【0333】在某些實作中，Key Locker指令在Intel TSX交易內執行時會導致Intel TSX中止。

Key Locker支援之CPUID列舉

【0334】透過CPUID.KL列舉Key Locker的硬體支援：CPUID.(07H,0).ECX[23]=1。這指示處理器支援Key Locker功能，以及CPUID葉19H提供了有關Key Locker功能的更多資訊。

【0335】一個單獨的CPUID位元，CPUID.AESKLE：CPUID.19H.EBX[0]=1指示作業系統和系統韌體(如BIOS)已啟用Key Locker AES指令(啟用AES KL)。由於它是CPUID 19H的一部分，因此在查看如CPUID.AESKLE這樣的葉19H位元之前，軟體應首先判定有支援CPUID葉19H(藉由檢查CPUID.KL是否被列舉)。除非設置CR4.KL，否則該位元將為0。某些實作可能需要啟用Key Locker的系統韌體。如果設置了CR4.KL但未列舉AESKLE(讀取為0)，則可能是該實作需要系統韌體啟用Key Locker但其並未被執行。

【0336】在決定是否啟用 Key Locker時，作業系統應檢查 CPUID.KL。

【0337】期望使用 Key Locker保護 AES金鑰的軟體(例如，應用程式)應檢查 CPUID.KL和 CPUID.AESKLE位元是否都已設置。CPUID.KL將指示 CPUID.19H(以及因此指示 ASKLE)有效，AESKLE將指示 OS(以及，若有需求，指示系統韌體)已啟用 Key Locker。

【0338】要判定 Key Locker的其他功能，請使用表 5 中所示的定義。調用 CPUID為(適當地實例化暫存器和位元欄位)：CPUID.19H:REG[bit #]。

暫存器	位元位置	內容
EAX	0	支援僅有CPL0(CPL0-only)的KL限制條件。
EAX	1	支援無加密的KL限制條件。
EAX	2	支援無解密的KL限制條件。
EAX	31:3	保留。
EBX	0	AESKLE:當為1時，AES Key Locker指令完全啟用。
EBX	1	保留。
EBX	2	WIDE_KL:當為1時，支援AES寬Key Locker指令。
EBX	3	保留。
EBX	4	當為1時，平台支援IWKeyBackup MSR並備份內部包裝金鑰。
EBX	31:5	保留。
ECX	0	當為1時，支援LOADIWKEY的NoBackup參數。
ECX	1	當為1時，支援為1的KeySource編碼(內部包裝金鑰之隨機性)。
ECX	31:2	保留。
EDX	31:0	保留。

表 5. 葉 19H之 Key Locker CPUID 定義

【 0339 】

指 令

【 0340 】

AESDEC128KL

運算碼/指令 (Opcode/Instruction)	Op/En	64/32位元模 式支援	CPUID旗標	說明
F3 0F 38 DD !(11):rrr:bbb AESDEC128KL xmm, m384	A	V/V	AESKLE	使用由m384處 的控制代碼指 示之 128 位 元 AES 金鑰解密 xmm並將結果 儲存在xmm。

【 0341 】

指 令 運 算 元 編 碼

Op/En	Tuple (元組)	Operand 1 (運算元1)	Operand 2 (運算元2)	Operand 3 (運算元3)	Operand 4 (運算元4)
A	NA	ModRM:reg(r,w)	ModRM:r/m(r)	NA	NA

【 0342 】

說 明

【 0343 】 AESDEC128KL指令執行 10輪 AES以使用由
來自第二個運算元的控制代碼指示的 128位元金鑰解密第
一個運算元。它將結果儲存在第一個運算元中。

【 0344 】

操 作

```

AESDEC128KL (DEST, SRC) {
    Handle := UnalignedLoad of 384 bit (SRC); // 不保證載入是基元的(atomic) 。
    Illegal Handle = (HandleReservedBitSet (Handle) ||
        (Handle[0] AND (CPL > 0)) ||
        Handle [2] ||
        HandleKeyType (Handle) != HANDLE_KEY_TYPE_AES128);
    If (Illegal Handle) {
        RFLAGS.ZF := 1;
    } ELSE {
        (UnwrappedKey, Authentic) := UnwrapKeyAndAuthenticate384 (Handle[383:0],
        IWKey);
        If (Authentic == 0) {
            RFLAGS.ZF := 1;
        } ELSE {
            DEST := AES128Decrypt (DEST,
            UnwrappedKey);
            RFLAGS.ZF := 0;
        }
    }
    RFLAGS.OF, SF, AF, PF, CF := 0;
}

```

【 0345 】

受影響的旗標

【 0346 】 如果操作成功，則 ZF 設置為 0，如果操作因控制代碼衝突而失敗，則設置為 1。其他算數旗標 (OF、SF、AF、PF、CF) 清零。

【 0347 】

異常 (Exception)

- #UD If the LOCK prefix is used(若使用

LOCK 前綴)。

If CPUID.07H:ECX.KL [bit 23] = 0.

If CR4.KL = 0.

If CPUID.19H:EBX.AESKLE [bit 0] = 0.

If CR0.EM = 1.

If CR4.OSFXSR = 0.

- #NM If CR0.TS = 1.
- #PF If a page fault occurs(如果發生頁面錯誤)。
- #GP(0) If a memory operand effective address is outside the CS, DS, ES, FS, or GS segment limit(若記憶體運算元有效位址在 CS、DS、ES、FS 或 GS 片段限制之外)。

If the DS, ES, FS, or GS register is used to access memory and it contains a NULL segment selector(若 DS、ES、FS 或 GS 暫存器用於存取記憶體且它包含 NULL 片段選擇器)。
- #SS(0) If a memory operand effective address is outside the SS segment limit(若記憶體運算元有效位址在 SS 片段限制之外)。

【 0348 】

內在函數(Intrinsics)

```
• unsigned char _mm_aesdec128kl_u8(__m128i* odata,
__m128i idata, const void* h);
```

【 0349 】

AESDEC256KL

運算碼/指令 (Opcode/Instruction)	Op/En	64/32位元 模式支援	CPUID 旗標	說明
F3 0F 38 DF !(11):rrr:bbb AESDEC256KL xmm, m512	A	V/V	AESKLE	使用由m512處的控制代碼指示之256位元AES金鑰解密xmm並將結果儲存在xmm。

【 0350 】

指令 運算元 編碼

Op/En	Tuple (元組)	Operand 1 (運算元1)	Operand 2 (運算元2)	Operand 3 (運算元3)	Operand 4 (運算元4)
A	NA	ModRM:reg (r,w)	ModRM:r/m(r)	NA	NA

【 0351 】

說明

【 0352 】 AESDEC256KL指令執行14輪AES以使用由來自第二個運算元的控制代碼指示的256位元金鑰解密第一個運算元。它將結果儲存在第一個運算元中。

【 0353 】

操作

```

AESDEC256KL (DEST, SRC) {
    Handle := UnalignedLoad of 512 bit (SRC); // 不保證載入是基元的(atomic)。
    Illegal Handle = (HandleReservedBitSet (Handle) ||
        (Handle[0] AND (CPL > 0)) ||
        Handle [2] ||
        HandleKeyType (Handle) != HANDLE_KEY_TYPE_AES256);
    If (Illegal Handle) {
        RFLAGS.ZF := 1;
    } ELSE {
        (UnwrappedKey, Authentic) := UnwrapKeyAndAuthenticate512 (Handle[511:0],
        IWKey);
        If (Authentic == 0) {
            RFLAGS.ZF := 1;
        } ELSE {
            DEST := AES256Decrypt (DEST, UnwrappedKey);
            RFLAGS.ZF := 0;
        }
    }
    RFLAGS.OF, SF, AF, PF, CF := 0;
}

```

【 0354 】

受影響的旗標

【 0355 】 如果操作成功，則 ZF 設置為 0，如果操作因控制代碼衝突而失敗，則設置為 1。其他算數旗標 (OF、SF、AF、PF、CF) 清零。

【 0356 】

異常 (Exception)

- #UD If the LOCK prefix is used(若使用 LOCK 前綴)。

If CPUID.07H:ECX.KL[bit 23]=0.

If CR4.KL=0.

If CPUID.19H:EBX.AESKLE[bit 0]=0.

If CR0.EM=1.

If CR4.OSFXSR=0.

- #NM If CR0.TS=1.
- #GP(0) If a memory operand effective address is outside the CS, DS, ES, FS, or GS segment limit(若記憶體運算元有效位址在CS、DS、ES、FS或GS片段限制之外)。

If the DS, ES, FS, or GS register is used to access memory and it contains a NULL segment selector(若DS、ES、FS或GS暫存器用於存取記憶體且它包含NULL片段選擇器)。

- #SS(0) If a memory operand effective address is outside the SS segment limit(若記憶體運算元有效位址在SS片段限制之外)。

【 0357】

內在函數(Intrinsics)

```
• unsigned char _mm_aesdec256kl_u8(__m128i* odata,
__m128i idata, const void* h);
```

【 0358】

AESDECWIDE128KL

運算碼/指令 (Opcode/Instruction)	Op/En	64/32位元 模式支援	CPUID 旗標	說明
F3 0F 38 D8 !(11):001:bbb AESDECWIDE128KL m384, <XMM0-7>	A	V/V	AESKLE WIDE_KL	使用由 m384 處的控制代碼指示的128位元 AES 金鑰解密 XMM0-7，並將每個生成區塊儲存回其相應的暫存器。

【 0359】

指令運算元編碼

Op/En	Tuple(元組)	Operand 1(運算元1)	Operands 2-9(運算元2-9)
A	NA	ModRM:r/m(r)	Implicit XMM0-7(r, w)

【 0360】

說明

【 0361】 AESDECWIDE128KL指令執行十輪 AES以使用由來自第二個運算元的控制代碼指示的128位元金鑰解密 XMM0-7中八個區塊中各塊。它將 XMM0-7中的每個輸入區塊替換為其相應的解密區塊。

【 0362】

操作

```

AESDECWIDE128KL (SRC) {
    Handle := UnalignedLoad of 384 bit (SRC); // 不保證載入是基元的(atomic)。
    Illegal Handle = (HandleReservedBitSet (Handle) ||
        (Handle[0] AND (CPL > 0)) ||
        Handle [2] ||
        HandleKeyType (Handle) != HANDLE_KEY_TYPE_AES128);
    If (Illegal Handle) {
        RFLAGS.ZF := 1;
    } ELSE {
        (UnwrappedKey, Authentic) := UnwrapKeyAndAuthenticate384 (Handle[383:0],
        IWKey);
        If Authentic == 0 {
            RFLAGS.ZF := 1;
        } ELSE {
            XMM0 := AES128Decrypt (XMM0, UnwrappedKey);
            XMM1 := AES128Decrypt (XMM1, UnwrappedKey);
            XMM2 := AES128Decrypt (XMM2, UnwrappedKey);
            XMM3 := AES128Decrypt (XMM3, UnwrappedKey);
            XMM4 := AES128Decrypt (XMM4, UnwrappedKey);
            XMM5 := AES128Decrypt (XMM5, UnwrappedKey);
            XMM6 := AES128Decrypt (XMM6, UnwrappedKey);
            XMM7 := AES128Decrypt (XMM7, UnwrappedKey);
            RFLAGS.ZF := 0;
        }
    }
    RFLAGS.OF, SF, AF, PF, CF := 0;
}

```

【 0363 】

受影響的旗標

【 0364 】 如果操作成功，則 ZF 設置為 0，如果操作因控制代碼衝突而失敗，則設置為 1。其他算數旗標 (OF、

SF、AF、PF、CF)清零。

【 0365】

異常 (Exception)

- #UD If the LOCK prefix is used(若使用 LOCK 前綴)。
If CUID.07H:ECX.KL[bit 23]=0.
If CR4.KL=0.
If CUID.19H:EBX.AESKLE[bit 0]=0.
If CR0.EM=1.
If CR4.OSFXSR=0.
If CUID.19H:EBX.WIDE_KL[bit 2]=0.
- #NM If CR0.TS=1(若 CR0.TS=1)。
- #GP(0) If a memory operand effective address is outside the CS, DS, ES, FS, or GS segment limit(若記憶體運算元有效位址在 CS、DS、ES、FS或GS片段限制之外)。
If the DS, ES, FS, or GS register is used to access memory and it contains a NULL segment selector(若 DS、ES、FS或GS暫存器用於存取記憶體且它包含 NULL 片段選擇器)。
- #SS(0) If a memory operand effective address is outside the SS segment limit(若記憶體運算元有效

位址在SS片段限制之外)。

【 0366】

內在函數 (Intrinsics)

```
• unsigned char_mm_aesdecwide128kl_u8(__m128i odata
[8], const __m128i idata[8], const void* h);
```

【 0367】

AESDECWIDE256KL

運算碼/指令 (Opcode/Instruction)	Op/En	64/32位元 模式支援	CPUID 旗標	說明
F3 0F 38 D8 ! (11): 011:bbb AESDECWIDE256KL m512, <XMM0-7>	A	V/V	AESKLE WIDE_KL	使用由m512處的控制代 碼指示的256位元AES 金鑰解密XMM0-7，並 將每個生成區塊儲存回 其相應的暫存器。

【 0368】

指令運算元編碼

Op/En	Tuple(元組)	Operand 1(運算元1)	Operands 2-9(運算元2-9)
A	NA	ModRM:r/m(r)	Implicit XMM0-7(r, w)

【 0369】

說明

【 0370】 AESDECWIDE256KL指令執行14輪AES以使用由來自第二個運算元的控制代碼指示的256位元金鑰解密XMM0-7中八個區塊中各塊。它將XMM0-7中的每個輸入區塊替換為其相應的解密區塊。

【 0371】

操 作

```
AESDECWIDE256KL (SRC) {  
    Handle := UnalignedLoad of 512 bit (SRC); // 不保證載入是基元的(atomic)。  
    Illegal Handle = (HandleReservedBitSet (Handle) ||  
        (Handle[0] AND (CPL > 0)) ||  
        Handle [2] ||  
        HandleKeyType (Handle) != HANDLE_KEY_TYPE_AES256);  
    If (Illegal Handle) {  
        RFLAGS.ZF := 1;  
    } ELSE {  
        (UnwrappedKey, Authentic) := UnwrapKeyAndAuthenticate512 (Handle[511:0],  
IWKey);  
        If (Authentic == 0) {  
            RFLAGS.ZF := 1;  
        } ELSE {  
            XMM0 := AES256Decrypt (XMM0, UnwrappedKey);  
            XMM1 := AES256Decrypt (XMM1, UnwrappedKey);  
            XMM2 := AES256Decrypt (XMM2, UnwrappedKey);  
            XMM3 := AES256Decrypt (XMM3, UnwrappedKey);  
            XMM4 := AES256Decrypt (XMM4, UnwrappedKey);  
            XMM5 := AES256Decrypt (XMM5, UnwrappedKey);  
            XMM6 := AES256Decrypt (XMM6, UnwrappedKey);  
            XMM7 := AES256Decrypt (XMM7, UnwrappedKey);  
            RFLAGS.ZF := 0;  
        }  
    }  
    RFLAGS.OF, SF, AF, PF, CF := 0;  
}
```

【 0372 】

受影響的旗標

【 0373 】 如果操作成功，則ZF設置為0，如果操作因

控制代碼衝突而失敗，則設置為 1。其他算數旗標 (OF、SF、AF、PF、CF) 清零。

【 0374】

異常 (Exception)

- #UD If the LOCK prefix is used(若使用 LOCK 前綴)。
If CPUID.07H:ECX.KL[bit 23]=0.
If CR4.KL=0.
If CPUID.19H:EBX.AESKLE[bit 0]=0.
If CR0.EM=1.
If CR4.OSFXSR=0.
If CPUID.19H:EBX.WIDE_KL[bit 2]=0.
- #NM If CR0.TS=1.
- #GP(0) If a memory operand effective address is outside the CS, DS, ES, FS, or GS segment limit(若記憶體運算元有效位址在 CS、DS、ES、FS 或 GS 片段限制之外)。
If the DS, ES, FS, or GS register is used to access memory and it contains a NULL segment selector(若 DS、ES、FS 或 GS 暫存器用於存取記憶體且它包含 NULL 片段選擇器)。
- #SS(0) If a memory operand effective address is

outside the SS segment limit(若記憶體運算元有效位址在SS片段限制之外)。

【 0375】

內在函數 (Intrinsics)

```
• unsigned char_mm_aesdecwide256kl_u8(__m128i odata[8], const __m128i idata[8], const void* h);
```

【 0376】

AESENC128KL

運算碼/指令 (Opcode/Instruction)	Op/En	64/32位元 模式支援	CPUID 旗標	說明
F3 0F 38 DC !(11):rrr: bbb AESENC128KL xmm, m384	A	V/V	AESKLE	使用由m384處的控制代碼指示之128位元AES金鑰加密xmm並將結果儲存在xmm。

【 0377】

指令運算元編碼

Op/En	Tuple (元組)	Operand 1 (運算元1)	Operand 2 (運算元2)	Operand 3(運算元3)	Operand 4(運算元4)
A	NA	ModRM:reg(r,w)	ModRM:r/m(r)	NA	NA

【 0378】

說明

【 0379】 AESENC128KL指令執行十輪AES以使用由來自第二個運算元的控制代碼指示的128位元金鑰加密第一個運算元。它將結果儲存在第一個運算元中。

【 0380】

操 作

```
AESENC128KL (DEST, SRC) {  
    Handle := UnalignedLoad of 384 bit (SRC); // 不保證載入是基元的(atomic)。  
    Illegal Handle = (  
        HandleReservedBitSet (Handle) ||  
        (Handle[0] AND (CPL > 0)) ||  
        Handle [1] ||  
        HandleKeyType (Handle) != HANDLE_KEY_TYPE_AES128  
    );  
    If (Illegal Handle) {  
        RFLAGS.ZF := 1;  
    } ELSE {  
        (UnwrappedKey, Authentic) := UnwrapKeyAndAuthenticate384 (Handle[383:0],  
IWKey);  
        If (Authentic == 0) {  
            RFLAGS.ZF := 1;  
        } ELSE {  
            DEST := AES128Encrypt (DEST, UnwrappedKey);  
            RFLAGS.ZF := 0;  
        }  
    }  
    RFLAGS.OF, SF, AF, PF, CF := 0;  
}
```

【 0381 】

受影響的旗標

【 0382 】 如果操作成功，則ZF設置為0，如果操作因控制代碼衝突而失敗，則設置為1。其他算數旗標 (OF、SF、AF、PF、CF) 清零。

【 0383 】

異常 (Exception)

- **#UD** If the LOCK prefix is used(若使用 LOCK 前綴)。
If CPUID.07H:ECX.KL[bit 23]=0.
If CR4.KL=0.
If CPUID.19H:EBX.AESKLE[bit 0]=0.
If CR0.EM=1.
If CR4.OSFXSR=0.
- **#NM** If CR0.TS=1.
- **#GP(0)** If a memory operand effective address is outside the CS, DS, ES, FS, or GS segment limit(若記憶體運算元有效位址在 CS、DS、ES、FS 或 GS 片段限制之外)。
If the DS, ES, FS, or GS register is used to access memory and it contains a NULL segment selector(若 DS、ES、FS 或 GS 暫存器用於存取記憶體且它包含 NULL 片段選擇器)。
- **#SS(0)** If a memory operand effective address is outside the SS segment limit(若記憶體運算元有效位址在 SS 片段限制之外)。

【 0384 】

內在函數 (Intrinsics)

- `unsigned char _mm_aesenc128kl_u8(__m128i* odata,`

```
__m128i idata, const void* h);
```

【 0385】

AESENC256KL

運算碼/指令 (Opcode/Instruction)	Op/En	64/32位元 模式支援	CPUID 旗標	說明
F3 0F 38 DE !(11):rrr: bbb AESENC256KL xmm, m512	A	V/V	AESKLE	使用由m512處的控制代 碼指示之 256 位元 AES 金鑰加密xmm並將結果 儲存在xmm。

【 0386】

指令運算元編碼

Op/En	Tuple (元組)	Operand 1 (運算元1)	Operand 2 (運算元2)	Operand 3 (運算元3)	Operand 4 (運算元4)
A	NA	ModRM:reg(r,w)	ModRM:r/m(r)	NA	NA

【 0387】

說明

【 0388】 AESENC256KL指令執行 14 輪 AES 以使用由
來自第二個運算元的控制代碼指示的 256 位元金鑰加密第
一個運算元。它將結果儲存在第一個運算元中。

【 0389】

操作

```

AESENC256KL (DEST, SRC) {
    Handle := UnalignedLoad of 512 bit (SRC); // 不保證載入是基元的(atomic) 。
    Illegal Handle = (
        HandleReservedBitSet (Handle) ||
        (Handle[0] AND (CPL > 0)) ||
        Handle [1] ||
        HandleKeyType (Handle) != HANDLE_KEY_TYPE_AES256
    );
    If (Illegal Handle) {
        RFLAGS.ZF := 1;
    } ELSE {
        (UnwrappedKey, Authentic) := UnwrapKeyAndAuthenticate512 (Handle[511:0],
IWKey);
        If (Authentic == 0) {
            RFLAGS.ZF := 1;
        } ELSE {
            DEST := AES256Encrypt (DEST, UnwrappedKey);
            RFLAGS.ZF := 0;
        }
    }
    RFLAGS.OF, SF, AF, PF, CF := 0;
}

```

【 0390】

受影響的旗標

【 0391】 如果操作成功，則ZF設置為0，如果操作因控制代碼衝突而失敗，則設置為1。其他算數旗標 (OF、SF、AF、PF、CF) 清零。

【 0392】

異常 (Exception)

- #UD If the LOCK prefix is used(若使用 LOCK

前綴)。

If CPUID.07H:ECX.KL[bit 23]=0.

If CR4.KL=0.

If CPUID.19H:EBX.AESKLE[bit 0]=0.

If CR0.EM=1.

If CR4.OSFXSR=0.

- #NM If CR0.TS=1.
 - #GP(0) If a memory operand effective address is outside the CS, DS, ES, FS, or GS segment limit(若記憶體運算元有效位址在CS、DS、ES、FS或GS片段限制之外)。
- If the DS, ES, FS, or GS register is used to access memory and it contains a NULL segment selector(若DS、ES、FS或GS暫存器用於存取記憶體且它包含NULL片段選擇器)。
- #SS(0) If a memory operand effective address is outside the SS segment limit(若記憶體運算元有效位址在SS片段限制之外)。

【0393】

內在函數(Intrinsics)

```
• unsigned char _mm_aesenc256kl_u8(__m128i* odata,
__m128i idata, const void* h);
```

【 0394】

AESENCWIDE128KL

運算碼/指令 (Opcode/Instruction)	Op/En	64/32位元 模式支援	CPUID 旗標	說明
F3 0F 38 D8 !(11):000: bbb AESENCWIDE128KL m384, <XMM0-7>	A	V/V	AESKLE WIDE_KL	使用由 m384 處的控制代碼指示的 128 位元 AES 金鑰加密 XMM0-7，並將每個生成區塊儲存回其相應的暫存器。

【 0395】

指令運算元編碼

Op/En	Tuple(元組)	Operand 1(運算元1)	Operands 2-9(運算元2-9)
A	NA	ModRM:r/m(r)	Implicit XMM0-7(r, w)

【 0396】

說明

【 0397】 AESENCWIDE128KL指令執行十輪 AES 以使用由來自第二個運算元的控制代碼指示的 128 位元金鑰加密 XMM0-7 中八個區塊中各塊。它將 XMM0-7 中的每個輸入區塊替換為其相應的加密區塊。

【 0398】

操作

```
AESENCWIDE128KL (SRC) {  
    Handle := UnalignedLoad of 512 bit (SRC); // 不保證載入是基元的(atomic)。  
    Illegal Handle = (  
        HandleReservedBitSet (Handle) ||  
        (Handle[0] AND (CPL > 0)) ||  
    )
```

```

        Handle [1] ||
        HandleKeyType (Handle) != HANDLE_KEY_TYPE_AES128
    );
If (Illegal Handle) {
    RFLAGS.ZF := 1;
} ELSE {
    (UnwrappedKey, Authentic) := UnwrapKeyAndAuthenticate384 (Handle[383:0],
IWKey);
    If Authentic == 0 {
        RFLAGS.ZF := 1;
    } ELSE {
        XMM0 := AES128Encrypt (XMM0, UnwrappedKey);
        XMM1 := AES128Encrypt (XMM1, UnwrappedKey);
        XMM2 := AES128Encrypt (XMM2, UnwrappedKey);
        XMM3 := AES128Encrypt (XMM3, UnwrappedKey);
        XMM4 := AES128Encrypt (XMM4, UnwrappedKey);
        XMM5 := AES128Encrypt (XMM5, UnwrappedKey);
        XMM6 := AES128Encrypt (XMM6, UnwrappedKey);
        XMM7 := AES128Encrypt (XMM7, UnwrappedKey);
        RFLAGS.ZF := 0;
    }
}
RFLAGS.OF, SF, AF, PF, CF := 0;
}

```

【 0399 】

受影響的旗標

【 0400 】 如果操作成功，則 ZF 設置為 0，如果操作因控制代碼衝突而失敗，則設置為 1。其他算數旗標 (OF、SF、AF、PF、CF) 清零。

【 0401】

異常 (Exception)

- #UD If the LOCK prefix is used(若使用 LOCK 前綴)。
If CPUID.07H:ECX.KL[bit 23]=0.
If CR4.KL=0.
If CPUID.AESKLE=0。
If CR0.EM=1.
If CR4.OSFXSR=0.
If CPUID.19H:EBX.WIDE_KL[bit 2]=0.
- #NM If CR0.TS=1.
- #GP(0) If a memory operand effective address is outside the CS, DS, ES, FS, or GS segment limit(若記憶體運算元有效位址在 CS、DS、ES、FS 或 GS 片段限制之外)。
If the DS, ES, FS, or GS register is used to access memory and it contains a NULL segment selector(若 DS、ES、FS 或 GS 暫存器用於存取記憶體且它包含 NULL 片段選擇器)。
- #SS(0) If a memory operand effective address is outside the SS segment limit(若記憶體運算元有效位址在 SS 片段限制之外)。

【 0402 】

內 在 函 數 (Intrinsics)

```
• unsigned char_mm_aesencwide128kl_u8(__m128i odata[8], const __m128i idata[8], const void* h);
```

【 0403 】

AESENCWIDE256KL

運算碼/指令 (Opcode/Instruction)	Op/En	64/32位元 模式支援	CPUID 旗標	說明
F3 0F 38 D8 !(11):010: bbb AESENCWIDE256KL m512, <XMM0-7>	A	V/V	AESKLE WIDE_KL	使用由m512處的控制代碼指示的256位元AES金鑰加密XMM0-7，並將每個生成區塊儲存回其相應的暫存器。

【 0404 】

指 令 運 算 元 編 碼

Op/En	Tuple(元組)	Operand 1(運算元1)	Operands 2-9(運算元2-9)
A	NA	ModRM:r/m(r)	Implicit XMM0-7(r, w)

【 0405 】

說 明

【 0406 】 AESENCWIDE256KL指令執行14輪AES以使用由來自第二個運算元的控制代碼指示的256位元金鑰加密XMM0-7中八個區塊中各塊。它將XMM0-7中的每個輸入區塊替換為其相應的加密區塊。

【 0407 】

操 作

```
AESENCWIDE256KL (SRC) {  
    Handle := UnalignedLoad of 512 bit (SRC); // 不保證載入是基元的(atomic)。  
    Illegal Handle = (  
        HandleReservedBitSet (Handle) ||  
        (Handle[0] AND (CPL > 0)) ||  
        Handle [1] ||  
        HandleKeyType (Handle) != HANDLE_KEY_TYPE_AES256  
    );  
    If (Illegal Handle) {  
        RFLAGS.ZF := 1;  
    } ELSE {  
        (UnwrappedKey, Authentic) := UnwrapKeyAndAuthenticate512 (Handle[511:0],  
IWKey);  
        If (Authentic == 0) {  
            RFLAGS.ZF := 1;  
        } ELSE {  
            XMM0 := AES256Encrypt (XMM0, UnwrappedKey);  
            XMM1 := AES256Encrypt (XMM1, UnwrappedKey);  
            XMM2 := AES256Encrypt (XMM2, UnwrappedKey);  
            XMM3 := AES256Encrypt (XMM3, UnwrappedKey);  
            XMM4 := AES256Encrypt (XMM4, UnwrappedKey);  
            XMM5 := AES256Encrypt (XMM5, UnwrappedKey);  
            XMM6 := AES256Encrypt (XMM6, UnwrappedKey);  
            XMM7 := AES256Encrypt (XMM7, UnwrappedKey);  
            RFLAGS.ZF := 0;  
        }  
    }  
    RFLAGS.OF, SF, AF, PF, CF := 0;  
}
```

【 0408 】

受影響的旗標

【0409】如果操作成功，則ZF設置為0，如果操作因控制代碼衝突而失敗，則設置為1。其他算數旗標(OF、SF、AF、PF、CF)清零。

【0410】

異常(Exception)

- #UD If the LOCK prefix is used(若使用LOCK前綴)。
If CPUID.07H:ECX.KL[bit 23]=0.
If CR4.KL=0.
If CPUID.19H:EBX.AESKLE[bit 0]=0.
If CR0.EM=1.
If CR4.OSFXSR=0.
If CPUID.19H:EBX.WIDE_KL[bit 2]=0.
- #NM If CR0.TS=1.
- #GP(0) If a memory operand effective address is outside the CS, DS, ES, FS, or GS segment limit(若記憶體運算元有效位址在CS、DS、ES、FS或GS片段限制之外)。
If the DS, ES, FS, or GS register is used to access memory and it contains a NULL segment selector(若DS、ES、FS或GS暫存器用於存取記憶體且它包含NULL片段選擇器)。

- #SS(0)If a memory operand effective address is outside the SS segment limit(若記憶體運算元有效位址在SS片段限制之外)。

【 0411】

內 在 函 數 (Intrinsics)

- unsigned char _mm_aesdecwide256kl_u8(__m128i odata[8], const __m128i idata[8], const void* h);

【 0412】

ENCODEKEY128

運算碼/指令 (Opcode/Instruction)	Op/En	64/32位元 模式支援	CPUID 旗標	說明
F3 0F 38 FA 11:rrr: bbb ENCODEKEY128 r32, r32, <XMM0-2>, <XMM4-6>	A	V/V	AESKLE	將來自XMM0的128位元AES金鑰包裝到金鑰控制代碼中並將控制代碼輸出於XMM0-2中。

【 0413】

指 令 運 算 元 編 碼

Op/En	Tuple (元組)	Operand 1 (運算元1)	Operand 2 (運算元2)	Operand 3(運算元 3)	Operands 4-5(運算 元4-5)	Operands 6-8(運算 元6-8)
A	NA	ModRM:reg(w)	ModRM:r/m(r)	Implicit XMM0 (r, w)	Implicit XMM1- 2(w)	Implicit XMM4- 6(w)

【 0414】

說 明

【 0415】 ENCODEKEY128 指 令 將 來 自 隱 含 運 算 元

XMM0的128位元AES金鑰包裝到金鑰控制代碼中，然後將其儲存在隱含目的地運算元XMM0-2中。

【0416】顯式來源運算元指定控制代碼限制條件(如果有)。

【0417】顯式目的地運算元填充有關於金鑰來源及其屬性的資訊。XMM4到XMM6保留供將來使用，軟體不應依賴它們被歸零。

【0418】

操作

```

ENCODEKEY128 (DEST, SRC) {
    #GP (0) if a reserved bit in SRC[31:0] is set
    InputKey[127:0] := XMM0;
    KeyMetadata[2:0] = SRC[2:0];
    KeyMetadata[23:3] = 0; // 保留用於未來使用
    KeyMetadata[27:24] = 0; // KeyType is AES-128 (為0之值)
    KeyMetadata[127:28] = 0; // 保留用於未來使用
    // KeyMetadata是AAD輸入 · InputKey是WrapKey128的明文輸入
    Handle[383:0] := WrapKey128(InputKey[127:0], KeyMetadata[127:0], IWKey.Integrity
        Key[127:0], IWKey.Encryption Key[255:0]);
    DEST[0] := IWKey.NoBackup;
    DEST[4:1] := IWKey.KeySource[3:0];
    DEST[31:5] = 0;
    XMM0 := Handle[127:0]; // AAD
    XMM1 := Handle[255:128]; // 完整性標籤
    XMM2 := Handle[383:256]; // 密文
    XMM4 := 0; // 保留用於未來使用
    XMM5 := 0; // 保留用於未來使用
    XMM6 := 0; // 保留用於未來使用
    RFLAGS.OF, SF, ZF, AF, PF, CF := 0;
}

```

【 0419 】 SRC[31:3]當前保留以用於未來使用。若 CPUID.19H:EAX[2]為 0，則 SRC[2](其指示無解密限制條件)被保留。若 CPUID.19H:EAX[1]為 0，則 SRC[1](其指示無加密限制條件)被保留。若 CPUID.19H:EAX[0]為 0，則 SRC[0](其指示唯 CPL0限制條件)被保留。

【 0420 】

受影響的旗標

【 0421 】 所有算數旗標 (OF、SF、ZF、AF、PF、CF) 清零。儘管為當前定義的操作清除了它們，但未來的擴充可能會報告旗標中的資訊。

【 0422 】

異常 (Exception)

- #GP If reserved bit is set in source register value(若保留位元被設置於來源暫存器值中)。
- #UD If the LOCK prefix is used(若使用 LOCK 前綴)。

If CPUID.07H:ECX.KL[bit 23]=0.

If CR4.KL=0.

If CPUID.19H:EBX.AESKLE[bit 0]=0.

If CR0.EM=1.

If CR4.OSFXSR=0.
- #NM If CR0.TS=1.

【 0423 】

內在函數 (Intrinsics)

```
• unsigned int_mm_encodekey128_u32(unsigned int htype,
__m128i key, void* h);
```

【 0424 】

ENCODEKEY256

運算碼/指令 (Opcode/Instruction)	Op/En	64/32位元模 式支援	CPUID 旗標	說明
F3 0F 38 FB 11:rrr: bbb ENCODEKEY256 r32, r32 <XMM0-6>	A	V/V	AESKLE	將來自 XMM1:XMM0 的 256位元AES金鑰包裝到 金鑰控制代碼中並將其 儲存在XMM0-3中。

【 0425 】

指令 運算元 編碼

Op/En	Tuple (元組)	Operand 1 (運算元1)	Operand 2 (運算元2)	Operands 3-4 (運算元3-4)	Operands 5-9 (運算元5-9)
A	NA	ModRM:reg(w)	ModRM:r/m(r)	Implicit XMM0-1(r, w)	Implicit XMM2-6(w)

【 0426 】

說明

【 0427 】 ENCODEKEY256 指令將來自隱含運算元 XMM1:XMM0 的 256 位元 AES 金鑰包裝到金鑰控制代碼中，然後將其儲存在隱含目的地運算元 XMM0-3 中。

【 0428 】 顯式來源運算元是一個通用暫存器，並且指定應該在控制代碼中構建什麼控制代碼限制條件。

【 0429 】 顯式目的地運算元填充有關於金鑰來源及其

屬性的資訊。XMM4到XMM6保留供將來使用，軟體不應依賴它們被歸零。

【 0430 】

操作

```

ENCODEKEY256 (DEST, SRC) {
    #GP (0) if a reserved bit in SRC[31:0] is set
    InputKey[255:0] := XMM1:XMM0;
    KeyMetadata[2:0] = SRC[2:0];
    KeyMetadata[23:3] = 0; // 保留用於未來使用
    KeyMetadata[27:24] = 1; // KeyType is AES-256 (為1之值)
    KeyMetadata[127:28] = 0; // 保留用於未來使用
    // KeyMetadata是AAD 輸入 · InputKey是WrapKey128的明文輸入
    Handle[511:0] := WrapKey256(InputKey[255:0], KeyMetadata[127:0], IWKey.Integrity
        Key[127:0], IWKey.Encryption Key[255:0]);
    DEST[0] := IWKey.NoBackup;
    DEST[4:1] := IWKey.KeySource[3:0];
    DEST[31:5] = 0;
    XMM0 := Handle[127:0]; // AAD
    XMM1 := Handle[255:128]; // 標籤
    XMM2 := Handle[383:256]; // 密文[127:0]
    XMM3 := Handle[511:384]; // 密文[255:128]
    XMM4 := 0; // 保留用於未來使用
    XMM5 := 0; // 保留用於未來使用
    XMM6 := 0; Integrity// 保留用於未來使用
    RFLAGS.OF, SF, ZF, AF, PF, CF := 0;
}

```

【 0431 】 SRC[31:3]當前保留以用於未來使用。若 CPUID.19H:EAX[2]為0，則 SRC[2](其指示無解密限制條件)被保留。若 CPUID.19H:EAX[1]為0，則 SRC[1](其指示無加密限制條件)被保留。若 CPUID.19H:EAX[0]為0，則

SRC[0](其指示唯 CPL0 限制條件)被保留。

【 0432】

受影響的旗標

【 0433】 所有算數旗標 (OF、SF、ZF、AF、PF、CF) 清零。儘管為當前定義的操作清除了它們，但未來的擴充可能會報告旗標中的資訊。

【 0434】

異常 (Exception)

- #GP If reserved bit is set in source register value(若保留位元被設置於來源暫存器值中)。
- #UD If the LOCK prefix is used(若使用 LOCK 前綴)。
If CPUID.07H:ECX.KL[bit 23]=0.
If CR4.KL=0.
If CPUID.19H:EBX.AESKLE[bit 0]=0.
If CR0.EM=1.
If CR4.OSFXSR=0.
- #NM If CR0.TS=1.

【 0435】

內在函數 (Intrinsics)

• unsigned int_mm_encodekey256_u32(unsigned int htype, __m128i key_lo, __m128i key_hi, void* h);

【 0436】

LOADIWKEY

運算碼/指令 (Opcode/Instruction)	Op/En	64/32位元 模式支援	CPUID 旗標	說明
F3 0F 38 DC 11:rrr:bbb LOADIWKEY xmm1, xmm2, <EAX>, <XMM0>	A	V/V	AESKLE	從xmm1、xmm2和XMM0 載入內部包裝金鑰。

【 0437 】

指令運算元編碼

Op/En	Tuple(元組)	Operand 1 (運算元1)	Operand 2 (運算元2)	Operand 3(運算元3)	Operand 4(運算元4)
A	NA	ModRM:reg(r)	ModRM:r/m(r)	Implicit EAX(r)	Implicit XMM0(r)

【 0438 】

說明

【 0439 】 LOADIWKEY指令寫入Key Locker內部包裝金鑰，稱為IWKey。ENCODEKEY*指令使用此IWKey將金鑰包裝到控制代碼中。相反地，AESENC/DEC*指令使用IWKey從控制代碼中解開這些金鑰並驗證控制代碼完整性。出於安全原因，沒有指令允許軟體直接讀取IWKey值。

【 0440 】 IWKey包括兩個密碼金鑰以及元資料。這兩個密碼金鑰是從暫存器來源載入的，因此LOADIWKEY可以在金鑰不在記憶體中的情況下執行。

【 0441 】

金鑰輸入運算元為：

- 256位元加密金鑰是從兩個顯式運算元載入的。
- 128位元完整性金鑰是從隱含運算元XMM0載入的。

【0442】隱含運算元EAX指定KeySource以及是否允許備份金鑰：

- EAX[0]-當設置後，不允許將正在初始化的包裝金鑰備份到平台範圍的儲存器中。
- EAX[4:1]-這指定了KeySource，它是金鑰的類型。目前只支援兩種編碼。KeySource為0指示上述金鑰輸入運算元應直接儲存為內部包裝金鑰。KeySource為1的LOADIWKEY將來自晶載隨機數產生器的隨機數與源極暫存器(包括XMM0)進行XOR運算，因此執行LOADIWKEY的軟體不知道實際的IWKey加密和完整性金鑰。軟體可以選擇將附加的隨機資料放入來源暫存器，以便其他隨機資料的來源與硬體隨機數產生器提供的值進行結合。在KeySource為1的情況下執行LOADIWKEY後，軟體應始終檢查ZF，因為此操作可能會因無法從晶載隨機數產生器獲得足夠的全熵資料而失敗。0和1的KeySource都指定IWKey用於且僅用於AES-GCM-SIV演算法。CPUID.19H.ECX[1]列舉了對為1的KeySource之支援。保留所有其他KeySource編碼。

- EAX[31:5]-保留。

【0443】

操作

```

LOADIWKEY (SRC1, SRC2) {
  IF CPL > 0 { // LOADKWKEY 僅在ring 0為被允許(監督者模式)
    #GP (0);
  }
  IF "LOADIWKEY exiting" VM execution control set {
    VMexit;
  }
  IF EAX[4:1] > 1 { // 使用保留的KeySource 編碼
    #GP (0);
  }
  IF EAX[31:5] != 0 { // 在EAX中的保留位元被設置
    #GP (0);
  }
  IF EAX[0] AND (CPUID.19H.ECX[0] == 0) { // 這部分不支援 NoBackup
    #GP (0);
  }
  IF (EAX[4:1] == 1) AND (CPUID.19H.ECX[1] == 0) { // 這部分不支援為1
    之KeySource
    #GP (0);
  }
  IF (EAX[4:1] == 0) { // KeySource 為0。
    IWKey.Encryption Key[127:0] := SRC2[127:0];
    IWKey.Encryption Key[255:128] := SRC1[127:0];
    IWKey.IntegrityKey[127:0] := XMM0[127:0];
    IWKey.NoBackup = EAX [0];
    IWKey.KeySource = EAX [4:1];
    RFLAGS.ZF := 0;
  } ELSE { // KeySource 為1。有關隨機性的詳細說明，請參閱RDSEED定義
    IF HW_NRND_GEN.ready == 1 { // 收到來自RdSeed的全熵隨機
      資料
      IWKey.Encryption Key[127:0] := SRC2[127:0] XOR
      HW_NRND_GEN.data[127:0];
      IWKey.Encryption Key[255:128] := SRC1[127:0] XOR
      HW_NRND_GEN.data[255:128];
      IWKey.Encryption Key[255:0] := SRC2[127:0]:SRC1[127:0] XOR
      HW_NRND_GEN.data[255:0];
    }
  }
}

```

```

        IWKey.IntegrityKey[127:0] := XMM0[127:0] XOR
HW_NRND_GEN.data[383:256];
        IWKey.NoBackup = EAX [0];
        IWKey.KeySource = EAX [4:1];
        RFLAGS.ZF := 0;
    } ELSE { // 隨機資料不是從RdSeed返回。未載入IWKey

        RFLAGS.ZF := 1;
    }
}
RFLAGS.OF, SF, AF, PF, CF := 0;
}

```

【 0444 】

受影響的旗標

【 0445 】 如果操作成功，則ZF設置為0，如果操作因全熵隨機資料沒有從RdSeed接收到而失敗，則設置為1。其他算數旗標(OF、SF、AF、PF、CF)清零。

【 0446 】

異常(Exception)

- #GP If CPL>0.
 If EAX[4:1]>1.
 If EAX[31:5]!=0.
 If(EAX[0]==1)AND(CPUID.19H.
 ECX[0]==0).
 If(EAX[4:1]==1)AND(CPUID.19H.
 ECX[1]==0).
- #UD If the LOCK prefix is used(若使用 LOCK

前綴)。

If CPUID.07H:ECX.KL[bit 23]=0.

If CR4.KL=0.

If CR0.EM=1.

If CR4.OSFXSR=0.

- #NM If CR0.TS=1.

【 0447】

內在函數(Intrinsics)

- void _mm_loadiwkey(unsigned int ctl, __m128i intkey, __m128i enkey_lo, __m128i enkey_hi);

【 0448】

備份MSR

【 0449】

將內部包裝金鑰備份與恢復

【 0450】 當列舉IWKeyBackup支援時，邏輯處理器範圍內的IWKey可以複製到或從稱為IWKeyBackup的平台範圍狀態複製。將IWKey複製到IWKeyBackup被稱為「備份IWKey」，而將IWKeyBackup複製到IWKey被稱為「恢復IWKey」。

【 0451】 IWKeyBackup及其與IWKey之間的路徑受到保護，免受軟體和簡單硬體攻擊(例如，插座(socket)之間的物理匯流排上的加密和完整性保護)。這代表IWKeyBackup可用於以受保護的方式在平台中的邏輯處理器內分佈IWKey。一個邏輯處理器可以用一個秘密值寫入

IWKey，然後將該IWKey備份到IWKeyBackup。然後其他邏輯處理器可以將IWKeyBackup複製到它們自己的IWKey。

【0452】在支援這些狀態的平台上跨S3(睡眠)和S4(休眠)睡眠狀態維護IWKeyBackup。這允許OS使用IWKeyBackup跨S3和S4睡眠狀態備份平台的IWKey，以在這些睡眠狀態中使用該IWKey維護控制代碼。

【0453】IWKeyBackup也可以在某些系統上跨S5(關閉)和G3(機械關閉)狀態進行維護，但這在架構上沒有保證，因此軟體不應依賴於此。因為它可以被維護，需要確保所有控制代碼被真正撤銷(例如，在關閉系統電源之前)的軟體應該覆寫IWKeyBackup，以確保以後不能使用它來恢復使這些控制代碼工作所需的IWKey。

【0454】VMM也可以使用IWKeyBackup來維護主機のIWKey，以便它可以在執行訪客(在訪客のIWKey載入的情況下運行)後恢復。由於在當前處理器上複製到IWKeyBackup和從IWKeyBackup複製的高延遲，這不會頻繁執行(例如，在每個VM進入之前或每個VM退出之後)，並且這些MSR不應包含在VM進入或VM退出MSR載入區域。

【0455】備份或恢復IWKey涉及到幾個MSR，所有這些都由CPUID.19H:EBX[4]列舉：

- IA32_COPY_LOCAL_TO_PLATFORM(write-only, address D91H)

- IA32_COPY_PLATFORM_TO_LOCAL(write-only, address D92H)
- IA32_COPY_STATUS(read-only, address 990H)
- IA32_IWKEYBACKUP_STATUS(read-only, address 991H)

【0456】IA32_COPY_LOCAL_TO_PLATFORM是一個唯寫MSR，可用於發出命令以將資料從當前邏輯處理器複製到平台範圍狀態。它可用於將此邏輯處理器的IWKey複製到平台的IWKeyBackup暫存器。

【0457】IA32_COPY_PLATFORM_TO_LOCAL是一個唯寫MSR，可用於發出命令以將資料從平台範圍狀態複製到當前邏輯處理器。它可用於將平台的IWKeyBackup複製到此邏輯處理器的IWKey。

【0458】IA32_COPY_STATUS是唯讀邏輯處理器範圍的MSR，它指示從該邏輯處理器執行的對IA32_COPY_PLATFORM_TO_LOCAL或IA32_COPY_LOCAL_TO_PLATFORM的最新寫入是成功還是失敗。

【0459】IA32_IWKEYBACKUP_STATUS是唯讀MSR，其指示IWKeyBackup的屬性。

【0460】在某些系統上，可能需要啟用Key Locker的系統韌體才能使CPUID.19H:EBX[4]列舉為1。

【0461】IA32_COPY_LOCAL_TO_PLATFORM MSR

【0462】當WRMSR設置IA32_COPY_LOCAL_TO_PLATFORM[0]時，IA32_COPY_LOCAL_TO_PLATFORM

MSR支援將IWKey 內容複製到IWKeyBackup暫存器。此寫入可能會失敗，例如因為設置了IWKey.NoBackup，因此軟體應始終在寫入後檢查它是否成功。

【0463】軟體可以透過在WRMSR之後立即檢查IA32_COPY_STATUS[0]以設置IA32_COPY_LOCAL_TO_PLATFORM[0]來判定IWKey到IWKeyBackup的複製是否成功。IA32_COPY_STATUS[0]中的值為1表示寫入IWKeyBackup成功。

【0464】IWKeyBackup還包括完整性測量。因為來自正在寫入不同值的不同邏輯處理器的兩個同時寫入之緣故，可能會損壞IWKeyBackup。由於該讀取正與不同值的寫入同時發生之緣故，這也可能損壞IWKeyBackup讀取(恢復)。由於IWKeyBackup完整性測量，CPU可以偵測到IWKeyBackup讀取將是一個損壞的值，並且不會將讀取標記為有效(意旨IA32_COPY_STATUS[0]將為0)或在這種情況下修改IWKey。

【0465】在寫入IWKeyBackup後，平台將確保將其寫入持久儲存器，該儲存器為跨S3或S4睡眠狀態進行維護者。在持久儲存器中，它將被加密、完整性保護以及(若可用)重播保護。對尚未完成更新持久儲存器(因此尚未設置IA32_IWKEYBACKUP_STATUS[Backup/Restore Valid])的IWKeyBackup(IA32_COPY_LOCAL_TO_PLATFORM[0]設置)的較舊寫入可能會導致較新的寫入失敗。

【0466】儘管對IWKeyBackup的較新寫入可能會被尚

未更新持久儲存器的較舊寫入阻擋，但讀取並不會被阻擋。IWKeyBackup的讀取(使用IA32_COPY_PLATFORM_TO_LOCAL[0])可以在驗證寫入成功完成(IA32_COPY_STATUS[0]已設置)後立即啟動(例如，在另一個邏輯處理器上)。

暫存器 位址		架構MSR名稱/ 位元欄位	MSR/位元說明	註解
十六 進制	十 進制			
D91	3473	IA32_COPY_LOCAL_TO_PLATFORM	將本地狀態複製到平台狀態(W)	IF((CPUID.19H:EBX[4]==1)&&(CPUID.(07H,0).ECX[23]==1))
		0	IWKeyBackup-將IWKey複製到IWKeyBackup	IF((CPUID.19H:EBX[4]==1)&&(CPUID.(07H,0).ECX[23]==1))
		63:1	保留	

【 0467】

IA32_COPY_PLATFORM_TO_LOCAL MSR

【 0468】當WRMSR設置IA32_COPY_PLATFORM_TO_LOCAL[0]時，IA32_COPY_PLATFORM_TO_LOCAL MSR支援將IWKeyBackup內容(平台暫存器)複製到當前邏輯處理器的IWKey暫存器。此寫入可能會失敗，例如，如果在寫入IWKeyBackup的同時讀取IWKeyBackup，則軟體應始終在寫入後檢查它是否成功。軟體可以透過在WRMSR後檢查IA32_COPY_STATUS[0]設置IA32_COPY_PLATFORM_TO_LOCAL[0]來判斷IWKeyBackup複製到

IWKey是否成功；IA32_COPY_STATUS[0]中的值為1則表示寫入IWKey成功。

暫存器位址		架構MSR名稱/位元欄位	MSR/位元說明	註解
十六 進制	十 進制			
D92	3474	IA32_COPY_PLATFORM_TO_LOCAL	將平台狀態複製到本地狀態(W)	IF((CPUID.19H:EBX[4]=1)&&(CPUID.(EAX=07H, ECX=0H).ECX[23]==1))
		0	IWKeyBackup- 將IWKeyBackup 複製到IWKey	IF((CPUID.19H:EBX[4]=1)&&(CPUID.(EAX=07H, ECX=0H).ECX[23]==1))
		63:1	保留	

【 0469】

IA32_COPY_STATUS MSR

【 0470】位址 990H 處的唯讀 IA32_COPY_STATUS MSR 中的每一位元指示透過 IA32_PLATFORM_TO_LOCAL MSR 或 IA32_LOCAL_TO_PLATFORM MSR 中的相應位元執行的最新命令是成功(值為 1)還是不成功(值為 0)。重置值為 0。

【 0471】此 MSR 是邏輯處理器範圍的，並且可以在 MSR 寫入後立即讀取以進行備份或恢復操作。它由冷/熱重置清除，並且不受 INIT 影響。

暫存器位址		架構MSR名稱 /位元欄位	MSR/位元說明	註解
十六 進制	十 進制			
990	2448	IA32_COPY_ STATUS	最近期平台到本地或本地到平台複製的狀態	IF((CPUID.19H:EBX[4]=1)&&(CPUID.(07H,0).ECX[23]==1))
		0	IWKEY_COPY_SUCCESSFUL：最近期複製到或從IWKeyBackup複製的狀態	IF((CPUID.19H:EBX[4]=1)&&(CPUID.(07H,0).ECX[23]==1))
		63:1	保留	

【 0472】

IA32_IWKEYBACKUP_STATUS MSR

【 0473】 IA32_IWKEYBACKUP_STATUS MSR(唯 讀 ， 平台範圍)提供有關 Key Locker之IWKeyBackup暫存器的狀態的資訊。由任何邏輯處理器(設置IA32_COPY_STATUS[IWKEY_COPY_SUCCESSFUL]的處理器)將IWKey成功複製到IWKeyBackup時，它會被清除，並在稍後再次進行評估。它也由冷/熱重置清除，並且不受INIT影響。

暫存器位址		架構MSR名稱/ 位元欄位	MSR/位元說明	註解
十六 進制	十 進制			
991	2449	IA32_IWKEYBACKUP_STATUS	有關IWKeyBackup暫存器之資訊	IF((CPUID.19H:EBX[4]==1)&&(CPUID.(07H,0).ECX[23]==1))
		0	備份/恢復有效。在啟動對IWKeyBackup的寫入時清零，然後在IWKeyBackup的最新寫入已寫入跨 S3/S4 睡眠狀態持續的儲存器時進行設置。如果在發生 IWKeyBackup寫入與設置該位元之間進入S3/S4，則在S3/S4退出後可能無法恢復 IWKeyBackup。在S3/S4睡眠狀態退出(系統喚醒)期間，該位元被清除。當IWKeyBackup從持久儲存器中恢復時，它會再次設置，因此可以使用IA32_COPY_PLATFORM_TO_LOCAL MSR複製到IWKey。如果先前的寫入尚未設置此位元，則對IWKeyBackup的另一次寫入(經由IA32_COPY_LOCAL_TO_PLATFORM MSR)可能會失敗。	IF((CPUID.19H:EBX[4]==1)&&(CPUID.(07H,0).ECX[23]==1))
		1	保留	
		2	備份金鑰儲存器讀取/寫入錯誤。在設置備份/恢復有效之前更新。在備份或恢復金鑰時遇到錯誤時設置。	IF((CPUID.19H:EBX[4]==1)&&(CPUID.(07H,0).ECX[23]==1))
		3	消耗 IWKeyBackup。在平台消耗掉先前備份操作之後進行設置。這並不表示系統已準備好進行第二次IWKeyBackup寫入，因為之前的IWKeyBackup寫入可能仍需要將備份/恢復設置為有效。	IF((CPUID.19H:EBX[4]==1)&&(CPUID.(07H,0).ECX[23]==1))
		63:4	保留	

【 0474】

OS啟用 (OS Enabling)

【 0475】

OS啟動 (OS Boot)

【 0476】 在列舉 CPUID.KL 的處理器上，OS 啟動過程應設置 CR4.KL，然後使用隨機數作為輸入執行 LOADIWKEY。當列舉支援 HW 隨機化的 IWKey 和 IWKeyBackup 時 (CPUID.19H.ECX[1] 和 CPUID.19H.EBX[4] 皆已設置)，軟體可能希望 LOADIWKEY 指定 KeySource 為 1 以便使硬體隨機數與軟體指定的任何內容相結合 (例如，來自軟體定義池的隨機數)。當指定 KeySource 為 1 時，軟體應該在 LOADIWKEY 之後檢查 RFLAGS.ZF 是否為 0，如果 RFLAGS.ZF 為 1 (這表明缺乏全熵資料的可用性將導致 LOADIWKEY 不完整)，則重試 LOADIWKEY。如果重複的 LOADIWKEY 不成功 (每次都設置 RFLAGS.ZF)，則來自晶載隨機數產生器的全熵資料為不可用，軟體可能希望改用為 0 的 KeySource。

【 0477】 除非成功載入 LOADIWKEY，否則 OS 不應允許應用程式在設置 CR4.KL 的情況下執行，因為應用程式可能會在設置 CR4.KL 時假定 IWKey 已正確初始化。

【 0478】 OS 應確保每個邏輯處理器上的 IWKey 相同，以便應用程式之控制代碼工作而不管它在哪個邏輯處理器上執行。這可以藉由使用軟體指定的 IWKey (KeySource 為 0) 並將其透過記憶體傳遞到其他邏輯處理器來完成，這樣

每個處理器都可以使用相同的值執行 `LOADIWKEY`。或者，它可以藉由在一個邏輯處理器上執行 `LOADIWKEY` 來完成，然後使用 `IA32_COPY_LOCAL_TO_PLATFORM MSR` 將 `IWKey` 從該邏輯處理器複製到 `IWKeyBackup`，然後在每個其他邏輯處理器上，使用 `IA32_COPY_PLATFORM_TO_LOCAL MSR` 從 `IWKeyBackup` 複製到 `IWKey`。在成功備份(寫入)到 `IWKeyBackup` 完成(在寫入邏輯處理器上設置了 `IA32_COPY_STATUS[0]`)後，軟體可以立即從 `IWKeyBackup` 複製到 `IWKey`，而無需檢查 `IA32_IWKEYBACKUP_STATUS MSR`。對 `IWKeyBackup` 的第二次寫入可能需要等到前一次寫入完成並設置 `IA32_IWKEYBACKUP_STATUS[Backup/Restore Valid]`。大多數使用不需要在啟動後立即對 `IWKeyBackup` 進行第二次寫入，但是當系統快速進行核心軟重置或藉由覆寫所有 `IWKeys` 以及 `IWKeyBackup` 來撤銷控制代碼時可能會發生這種情況；特別是在人工測試環境中。`IWKeyBackup` 支援軟體指定的 `IWKey` (`KeySource` 為 0) 以及與來自晶載硬體隨機數產生器的資料組合的金鑰 (`KeySource` 為 1)，並透過 `CPUID.19H.EBX[4]` 列舉。

【0479】藉由在啟動過程的早期(在載入大多數軟體組件之前)載入 `IWKey` 並且不記錄 `IWKey` 值，稍後載入的軟體組件中的漏洞將無法單步系統查看正在載入的 `IWKey` 值(因為 `IWKey` 載入已經發生)。

【0480】在某些平台上，可能需要啟用系統韌體才能

使用 Key Locker。在此類平台上，如果系統韌體未正確啟用 Key Locker，則即使設置了 CR4.KL，CPUID.AESKLE 也可能保持為 0。如果系統韌體沒有正確啟用 Key Locker，則某些系統也可能不會列舉對 IWKeyBackup 的支援。

【0481】在支持 S3 和 S4 睡眠狀態的系統上，OS 可能希望藉由在啟動時設置 IA32_COPY_LOCAL_TO_PLATFORM[0] 來將 LoadIWKey 值複製到 IWKeyBackup。這使平台有更多時間將 IWKeyBackup 寫入持久儲存器（並設置 IA32_IWKEYBACKUP_STATUS[0]），然後再請求進入 S3 或 S4 睡眠狀態（以便使進入這些睡眠狀態不會延遲）。支援 S3 和 S4 睡眠狀態但不支援 IWKeyBackup 的平台可能需要找到一些其他方法來維護跨 S3/S4 睡眠狀態的 IWKey。如果沒有其他可用的方法，其可以選擇不啟用 Key Locker。

【0482】

OS 關閉 (OS Shutdown)

【0483】關閉系統 (S5 軟關閉或 G3 機械關閉) 涉及關閉所有應用程式，因此是撤銷控制代碼的好時機。雖然 OS 的下一次啟動通常會載入一個新的 IWKey 並因此撤銷這些控制代碼，但一些安全使用模型可能不希望相信 OS 的下一次啟動會執行該撤銷（例如，以防 OS 被惡意替換的情況）。

【0484】為了確保攻擊者即使在這種情況下也無法在關機後重用任何控制代碼，OS 可以在關閉系統電源之前覆寫 IWKey 和 IWKeyBackup。具體來說，它可以在每個邏

輯處理器上使用 `LOADIWKEY` 來覆寫 `IWKey` (例如，使用零或新的隨機值)，並且可以使用 `IA32_COPY_LOCAL_TO_PLATFORM[0]` 將覆寫的 `IWKey` 從那些邏輯處理器之一複製到 `IWKeyBackup`。請注意，如果先前對 `IWKeyBackup` 的寫入 (例如啟動寫入) 尚未完成且設置 `IA32_IWKEY_BACKUP_STATUS[Backup/Restore Valid]`，則此寫入至 `IWKeyBackup` 可能會失敗。在重置處理器之前，軟體可以藉由等待 `IA32_IWKEYBACKUP_STATUS[Backup/Restore Valid]` 再次設置來確保 `IWKeyBackup` 的新覆寫已經覆寫了其持久儲存的副本。

【 0485】

進入 S3 或 S4 系統睡眠狀態 (睡眠或休眠)

【 0486】 進入 S3 (睡眠) 或 S4 (休眠) 狀態將關閉所有處理器，從而導致丟失 `IWKey` (因為它在系統重新啟動時被重置清除)。為了維護 `IWKey` (以便在進入 S3 或 S4 之前創建的應用控制代碼在喚醒後繼續工作)，軟體應確保 `IWKeyBackup` 已寫入 (例如，在啟動時) 並設置 `IA32_IWKEYBACKUP_STATUS[Backup/Restore Valid]` 以便它跨 S3/S4 維護。如果寫入持久儲存器時出現錯誤，則將設置 `IA32_IWKEYBACKUP_STATUS[Backup key storage read/write error]`。

【 0487】

退出 S3 或 S4 系統睡眠狀態 (從睡眠或休眠中喚醒)

【 0488】 從 S3 或 S4 睡眠狀態喚醒時，OS 將希望恢復

之前的IWKey，以便在進入S3/S4狀態之前創建的應用控制代碼在從S3/S4喚醒後繼續工作。

【0489】如果之前的IWKeyBackup在進入S3/S4睡眠狀態之前完成了對持久儲存器的寫入，那麼恢復過程將在退出S3/S4睡眠狀態時自動開始。

【0490】OS應藉由等待IA32_IWKEYBACKUP_STATUS[Backup/Restore Valid]設置來檢查IWKeyBackup是否已準備好進行複製。當IWKeyBackup準備好時，可以在每個邏輯處理器上執行WRMSR以設置IA32_COPY_PLATFORM_TO_LOCAL[0]，以便在這些邏輯處理器上恢復IWKey。如果OS在S3/S4後無法正確恢復IWKey(例如，重試後未設置IA32_IWKEYBACKUP_STATUS[Backup/restore valid]或從IWKeyBackup恢復IWKey失敗)，則它可能希望記錄錯誤並清除CR4.KL或關閉系統。

【0491】

應用程式啟用

【0492】應用程式可以直接使用Key Locker指令，也可以使用允許選擇Key Locker來維護其AES金鑰的軟體庫。

【0493】軟體可以使用CPUID.AESKLE來判定系統和OS是否支援Key Locker。此CPUID位元僅在OS已設置CR4.KL時進行設置。任何已經這樣做的OS也應該將IWKey寫入一個隨機值。請注意，如果IWKey未初始化(因此其所有欄位都為0)，則輸入為0的ENCODEKEY128將創

建一個控制代碼，其完整性標籤為 0x8720849214a248ad_898940a278c095dc 且密文為 0xd3e9d22b334fb3c2_3382228c8474c308。AES GCM SIV 演算法的完整性檢查可能需要非零 IWKey 才能正確偵測控制代碼中的變化。

【0494】一旦應用程式獲得了它想要使用的 AES 金鑰 (例如，與另一個實體的金鑰協商結果或從 TPM 中解封的金鑰)，它應該使用 ENCODEKEY128 (用於 128 位元 AES 金鑰) 或 ENCODEKEY256 (用於 256 位元 AES 金鑰) 來創建控制代碼。控制代碼限制條件 (例如，不加密或不解密) 可以透過 SRC 暫存器進行指定。

【0495】連同控制代碼，ENCODEKEY128 和 ENCODEKEY256 還產生用於創建控制代碼的 IWKey 相關的資訊。例如，目的地暫存器的位元 0 指示是否禁止將 IWKey 寫入 IWKeyBackup。目的地暫存器的位元 4:1 中的值為 0 指示用於保護該控制代碼的 IWKey 是系統軟體指定的，值為 1 則表示 IWKey 是隨機的，並且因此不被任何軟體 (包括系統軟體) 知曉。

【0496】在其信任邊界內沒有系統軟體的 Intel SGX 指定位址空間 (enclave) 可以拒絕使用不使用晶載硬體隨機數產生器的 IWKey，以確保 Key Locker 控制代碼提供針對系統軟體攻擊者的深度防禦，這些攻擊者也獲得控制代碼；但這可能會限制 Key Locker 的使用。知道 IWKey 的攻擊者 (例如，將 IWKey 載入到攻擊者指定值的系統軟體攻擊者) 本身並不能讓他們推斷出控制代碼/金鑰，但同時擁有

控制代碼和IWKey的攻擊者可以藉由將控制代碼解封以推衍出原始AES金鑰。

【0497】當應用程式希望執行加密或解密指令時，它應該將控制代碼連同相應的明文或密文一起傳遞給AES*KL指令。執行並行AES操作(如AES-CTR模式)的軟體可能希望使用寬指令來並行加密/解密多個區塊。執行串行AES操作(如AES-CBC加密)的軟體應使用非寬指令。例如，使用256位元AES金鑰進行AES CBC加密的軟體應使用AESENC256KL。

【0498】如果控制代碼或IWKey損壞(例如，透過改變IWKey的惡意系統軟體)或違反限制條件，則AES*KL將失敗並設置RFLAGS.ZF。在這種情況下，目的地未修改，並因此保存明文(如果正在執行加密操作)或密文(如果正在執行解密操作)。因此，軟體應在每次執行AES*KL指令後仔細檢查ZF是否為0。

【0499】應用程式應該像通常保護金鑰一樣保護控制代碼；即使它不能遠端使用或在控制代碼撤銷後不能使用，它仍然可以被該系統上的敵手(adversary)使用，直到控制代碼被撤銷(由OS或VMM覆寫IWKey及其任何備份)。

【0500】

虛擬化支援

【0501】

虛擬化策略

【0502】VMM可能需要能夠進行上下文切換和遷移

訪客以及能夠將訪客保存到磁碟上並在以後恢復它的能力(例如，在主機重新啟動並因此撤銷其控制代碼之後)。

【0503】當訪客載入IWKey並記錄訪客期望的IWKey時，VMM可以藉由導致VM退出來執行此操作。然後，VMM可以在運行該訪客之前載入此IWKey(使用LOADIWKEY)。這可能意味著在對使用Key Locker(已設置CR4.KL)的訪客的每個上下文切換時執行LOADIWKEY。

【0504】請注意，VMM可能希望不支援向訪客列舉硬體產生的隨機IWKey(即，將CPUID.19H:ECX[1]列舉為0)，因為這樣的IWKey無法輕易進行上下文切換。訪客ENCODEKEY*將返回所使用的IWKey的類型(IWKey.KeySource)，並且因此會注意到VMM是否使用軟體指定的IWKey虛擬化硬體產生的隨機IWKey。

【0505】雖然使用虛擬化的系統可能需要保存訪客的IWKey以便它們可以在訪客上下文切換時進行載入，訪客的攻擊者不僅需要恢復控制代碼，還需要觀察VMM維護的IWKey值以便解開該控制代碼並恢復AES金鑰。對於攻擊者來說，這比簡單地竊取控制代碼更困難。

【0506】由於不能直接從處理器中讀取IWKey，因此載入IWKey後啟動的VMM無法判定該IWKey值。如果VMM也使用Key Locker(與訪客使用分開)，那麼每次載入訪客IWKey時都會覆寫VMM的IWKey。需要使用Key Locker的VMM可以將其自己的IWKey保存在記憶體/暫存器中(這會影響安全性，因為可以觀察任意VMM記憶體的

敵手 (adversary) 可能能夠竊取控制代碼和 IWKey 並要求 VMM 在第一次 IWKey 載入之前運行) 或在使用 VMM 控制代碼之前使用 IWKeyBackup 恢復自己的 IWKey 或不允許訪客使用 Key Locker (因此不需要載入訪客 IWKey)。初始實作可能需要大量時間來執行 IWKeyBackup 到 IWKey 的複製 (經由 MSR 寫入 IA32_COPY_PLATFORM_LOCAL[0])，因此在每個 VM 退出後重新載入 IWKey 可能會導致顯著的性能影響。因此，建議 VMM 僅在需要使用控制代碼之前恢復 IWKey，或者如果 Key Locker 被列舉給訪客，則甚至避免使用 Key Locker 用於 VMM 之用途 (這樣 VMM 使用 Key Locker，或者訪客使用 Key Locker，但不會兩者同時成立)。

【0507】VMM 可以使用 MSR 點陣圖來獲得訪客對 IA32_COPY_LOCAL_TO_PLATFORM、IA32_COPY_PLATFORM_TO_LOCAL、IA32_COPY_STATUS 及 IA32_IWKEYBACKUP_STATUS MSR 的使用。其可以透過記錄 IWKeyBackup 和 IWKey 的訪客期望的軟體進行虛擬化。這避免了訪客需要實際讀取或寫入實際平台 IWKeyBackup；訪客互動將會是與虛擬 IWKeyBackup 進行互動。

【0508】不知道 Key Locker 的 VMM 不應允許訪客設置 CR4.KL (因為 VMM 應該已經阻止訪客設置 VMM 不知道的 CR4 位元)。這將確保 Key Locker 指令在此類 VMM 的訪客上產生 #UD 異常。

【0509】

基於三級處理器的 VM 執行控制

【0510】定義了一個新的 64 位元控件向量(vector of control)來管理同步事件的處理，主要是管理由執行特定指令引起的事件。此向量被稱作基於三級處理器的 VM 執行控制。軟體可以使用 VMREAD 和 VMWRITE 指令使用編碼對 2034H/2035H 存取基於三級處理器的 VM 執行控制(tertiary processor-based VM-execution controls)。

【0511】基於主處理器的 VM 執行控制的第 17 位元被定義為「啟用三級控制」。其判定是否使用基於三級處理器的 VM 執行控制。如果該位元為 0，VM 進入和 VMX 非根操作就如同所有基於第三處理器的 VM 執行控制都是 0 一樣。僅支援基於主處理器的 VM 執行控制的第 17 位元設置為 0 的處理器不支援基於三級處理器的 VM 執行控制。因此，當且僅當 IA32_VMX_PROCBASED_CTL[49]=1 時，處理器才支援基於三級處理器的 VM 執行控制。如果 IA32_VMX_BASIC[55]=1，則 IA32_VMX_TRUE_PROCBASED_CTL MSR 存在並且與位元位置 63:32 中的 IA32_VMX_PROCBASED_CTL 相同。因此，如果 IA32_VMX_BASIC[55]=1，則當且僅當 IA32_VMX_TRUE_PROCBASED_CTL[49]=1 時，處理器才支援基於三級處理器的 VM 執行控制。

【0512】支援基於三級處理器的 VM 執行控制的處理器也支援 IA32_VMX_PROCBASED_CTL3 MSR(索引 492H)。此 MSR 列舉了這些控制允許為 1 的設置。具體來

說，當且僅當 MSR 的位元 X 設置為 1 時，VM 進入才允許基於三級處理器的 VM 執行控制的位元 X 設置為 1。如果 MSR 的位元 X 被清除為 0，如果控制 X 和「啟用三級控制」的主處理器的 VM 執行控制都為 1，則 VM 進入失敗。

【0513】

LOADIWKEY 虛擬機退出

【0514】為了讓 VMM 獲得訪客的 IWKey 值，支援 Key Locker 的處理器還支援基於三級處理器的 VM 執行控制的位元 0 中的新「LOADIWKEY 退出」VM 執行控制。如果處理器設置了 IA32_VMX_PROCBASED_CTL5 MSR 的第 49 位元和 IA32_VMX_PROCBASED_CTL53 MSR 的第 0 位元，則該處理器支援此控制為 1 的設置。

【0515】如果「啟用三級控制」的 VM 執行控制和「LOADIWKEY 退出」的 VM 執行控制都為 1，則在 VMX 非 root 操作中執行 LOADIWKEY 會導致 VM 退出。這樣的 VM 退出使用 45H(十進制 69)的基本退出原因，但沒有符合退出條件。指令長度儲存在 VM 退出指令長度欄位中，且指令運算元資訊儲存在 VM 退出指令資訊欄位中(詳見表 6)。

位元位置	內容
2:0	保留
6:3	Reg1: 0=XMM0 1=XMM1 ... 7=XMM7 8-15分別代表XMM8-XMM15(僅用於支援Intel® 64架構的處理器)
9:7	未界定
10	設置為1(reg格式)
14:11	未界定
17:15	未界定(Segreg)
27:18	未界定(IndexReg和BaseReg)
31:28	Reg2: 0=XMM0 1=XMM1 ... 7=XMM7 8-15分別代表XMM8-XMM15(僅用於支援Intel® 64架構的處理器)

表 6.用於 LOADIWKEY的 VM退出指令資訊欄位的格式

【 0516】

Key Locker性能

【 0517】 AES CBC加密等串行演算法模式應使用一次加密或解密單個區塊的指令(例如 AESENC128KL)，而 AES CTR或 AES GCM等並行演算法模式應使用一次加密或解密多個區塊的指令(例如， AESENCWIDE128KL)。

【 0518】 為了提高性能，處理器可以在「控制代碼快取」中儲存最近使用的控制代碼及其關聯的金鑰或當前載入的 IWKey的回合金鑰。出於安全原因，除非使用 Key

Locker指令，否則無法存取此控制代碼快取；軟體無法直接讀取金鑰或回合金鑰。控制代碼快取的尺寸可能因實作而異。

【0519】Key Locker安全特性

【0520】Key Locker旨在防止在金鑰被包裝後闖入系統的攻擊者能夠在撤銷後或在另一個系統上使用該包裝的金鑰，或違反指定的控制代碼限制條件(例如，在ring 3控制代碼是限於ring 0)。

【0521】當在可信執行環境(TEE)之外使用時，在金鑰被包裝之前闖入系統的攻擊者可能能夠在金鑰變成控制代碼之前觀察到金鑰(例如，透過單步執行並觀察ENCODEKEY*指令輸入)，或觀察用於保護Key Locker控制代碼的IWKey(例如，當KeySource為0時，透過單步執行並觀察LOADIWKEY輸入)。

【0522】當金鑰在啟動過程的早期變成控制代碼時，Key Locker在保護金鑰方面可能特別有效，因為載入的軟體越少，攻擊者為了提升權限或發現資訊洩露而瞄準的受害者就越少。但是，Key Locker也應該在以後使用時降低風險，因為可以顯著減少金鑰的公開時間(僅在產生控制代碼之前，而不是在金鑰的整個生命週期內)。

【0523】ENCODEKEY*指令從暫存器而不是從記憶體中獲取金鑰值，以支援希望避免將金鑰儲存到記憶體中直到該金鑰被包裝到Key Locker控制代碼中被保護為止的軟體。

【0524】Key Locker控制代碼包含完整性測量，以便偵測任何更改控制代碼的嘗試(例如，更改限制條件)。這些測量還偵測使用錯誤的金鑰尺寸或使用與創建控制代碼不同的IWKey。軟體應在每次操作後小心檢查ZF，以偵測這些情況。

【0525】Key Locker不會藉由覆寫IWKey或IWKeyBackup的更多特權系統軟體來阻止拒絕服務。有許多現有的方法可以讓特權較高的軟體拒絕對特權較低的軟體執行。

【0526】除非IWKey被撤銷，否則Key Locker不會阻止以控制代碼限制允許的方式使用包裝金鑰。撤銷Key Locker控制代碼涉及覆寫用於創建控制代碼的IWKey值。這可能包括覆寫其他邏輯處理器上的IWKey值(如果它們包含它)，以及覆寫IWKeyBackup中的IWKey值(如果IWKey備份於該處)並等待新的IWKeyBackup覆寫持久儲存器(這會導致設置IA32_IWKEYBACKUP_STATUS [Backup/Restore valid])。

【0527】Key Locker針對簡單硬體攻擊提供了一定程度的保護，包括對外部匯流排的探測，但可能無法針對能夠物理探測CPU晶粒內的值或修改外部匯流排上的資料值的攻擊者提供足夠的保護。

【0528】如果攻擊者發現用於產生控制代碼的IWKey，他們可以解開控制代碼以獲得原始金鑰，然後攻擊者可以直接使用它進行加密和解密。Key Locker安全性

倚賴確保IWKey對潛在攻擊者而言為秘密的。

【 0529 】

以 TEE 使用 Key Locker

【 0530 】 可信執行環境(TEE)模式可以提供一個安全的地方來執行操作。這種TEE的一個實例是Intel SGX指定位址空間(enclave)。另一個實例是運行受信任服務的VMX訪客，這些服務與在該VMX訪客之外運行的其他不太受信任的碼隔離。

【 0531 】 Key Locker 控制代碼可以由 TEE 內的 ENCODEKEY*產生，以進一步保護被保護的金鑰。這提供了針對在控制代碼產生之前已經存在於系統上但尚未滲透到TEE的攻擊者的保護。此類攻擊者無法在將金鑰值轉換為控制代碼之前單步執行並觀察該金鑰值，因為該轉換過程受TEE保護。為了保護不控制IWKey(例如指定位址空間(enclave))的TEE免受知道IWKey的攻擊者的攻擊(因為控制代碼僅對不知道IWKey的攻擊者提供保護)，TEE可以在ENCODEKEY*之後檢查目的地暫存器並拒絕運行，除非IWKey的KeySource為1。請注意，此類TEE策略可能會阻止該軟體使用虛擬化，因為此時KeySource為1的IWKey無法完全虛擬化。

【 0532 】 更改TEE下IWKey的更高特權的實體將導致先前創建的控制代碼不適用於加密或解密，並設置RFLAGS.ZF以指示這一點。使用Key Locker應始終檢查ZF以偵測到這一點。儘管這可以用作TEE上的拒絕服務，但

已經存在許多技術，特權更高的實體(例如，OS或VMM)可以使用這些技術來拒絕向他們管理的特權較低的實體提供服務。

【0533】某些使用模型可以通過讓TEE產生金鑰並將其包裝到控制代碼來提供服務，但允許控制代碼本身在TEE之外使用。為了支持這樣的使用，ring 3上的應用程式可以創建僅限於在ring 0使用上運行的系統軟體的控制代碼。這些使用要求使用控制代碼的TEE和SW都使用相同的IWKey。如果沒有這樣的使用，則可以將單獨的IWKey用於TEE和軟體的其餘部分(例如，當TEE是單獨的VMX訪客時)。如果TEE的控制代碼被盜竊(例如，透過資訊攻擊)，這將防止TEE的控制代碼被TEE之外的軟體使用，前提是攻擊者也不能竊取用於TEE的IWKey值。

【0534】在首次啟動時信任IWKey，但在稍後呼叫時不信任的TEE(由於擔心稍後會使用已知的非隨機值覆蓋寫IWKey的權限提升攻擊)可能能夠產生一個控制代碼首次呼叫時的已知值。它可以稍後使用該控制代碼來確認在啟動早期使用的IWKey仍然是當前的IWKey。如果攻擊者能夠單步執行TEE，這可能不太有效，因為它可能僅在TEE檢查IWKey時才能恢復原始IWKey，而在其餘時間使用攻擊者已知的IWKey。

【0535】

其他軟體使用控制代碼的能力

【0536】Key Locker控制代碼鏈接到用於創建它們的

IWKey。這意味著竊取控制代碼而不是原始金鑰的攻擊者將無法使用原始金鑰(例如，使用控制代碼)進行加密或解密，除非他們可以使用被竊取的控制代碼及其相應的IWKey。

【0537】如果攻擊者無法發現IWKey值(例如，因為它在攻擊者能夠執行資訊洩露或特權升級之前被載入到處理器中)，那麼攻擊者將無法在另一個系統上設置該IWKey值或其是在IWKey被撤銷後；且因此將無法在另一個系統上或在IWKey撤銷後使用被盜的控制代碼。如果按照建議在重新啟動時執行IWKey撤銷，這可以防止在重新啟動後使用先前獲得的控制代碼。

【0538】如果VMM為不同的訪客使用不同的IWKey(預計這是正常情況)，那麼從一個訪客竊取的控制代碼將無法在不同攻擊者的訪客中使用，除非攻擊者還設法從VMM獲取IWKey。希望半虛擬化訪客共享控制代碼的VMM需要對兩者使用相同的IWKey。

【0539】還可以創建包含限制條件的控制代碼(例如，只能加密且不能解密的僅ring 0之控制代碼)。

【0540】僅Ring 0的控制代碼可能對不供應用程式使用的OS金鑰有用。如果惡意應用程式有辦法竊取到這樣的控制代碼，它將無法在應用程式本身中使用它。請注意，Ring 0控制代碼可以在任何權限級別創建，儘管其僅可用於CPL 0的加密/解密。

【0541】當協定的一側只需要創建訊息(帶加密)而協

定的另一側只需要讀取訊息(帶解密)時，不解密和不加密的控制代碼可能成對使用。這將需要使用同時使用AES加密和解密的AES模式(例如，AES-CBC)，而不是僅使用加密的AES模式(例如，AES-CTR)。

【0542】獲得受害者控制代碼並使用與受害者相同的IWKey的攻擊者可以以控制代碼的限制條件所允許的方式使用該控制代碼。預期的使用模型是所有應用程式都具有相同的IWKey，這可能允許使用被不同應用程式竊取的控制代碼，直到它們被撤銷(例如，透過重新啟動)。目前，限制條件不會限制特定應用程式的控制代碼使用。

【0543】

CCA/CPA-加密/解密即服務的限制

【0544】雖然使用Key Locker可以保護控制代碼內的金鑰值，但選擇密文攻擊(CCA)和選擇明文攻擊(CPA)可能會洩露資料模式(data pattern)。例如，AES ECB模式將始終將相同的明文值加密為相同的密文值。因此，知道給定密文區塊是從0的明文創建的攻擊者知道具有相同值的其他密文區塊也映射到0的明文，並且可能能夠在密文中找到模式。

【0545】AES的使用者應該注意他們選擇的演算法模式提供對CCA和CPA攻擊的必要抵抗。無論金鑰值是由Key Locker保護還是由其他機制(例如，使用硬體安全模組(HSM))保護，這都是屬實的。

【0546】

對旁通道之抵抗

【0547】為了提高性能，Key Locker指令可能會使用最近使用過的控制代碼的快取。雖然軟體無法直接觀察快取中的值，但使用快取內的控制代碼會比使用不在快取中的控制代碼更快。這意味著基於時間的側信道攻擊可能能夠觀察到同一處理器上的另一個實體是否正在使用完全相同的控制代碼。

【0548】此外，某些實作可能會根據控制代碼的完整性標籤的子集對快取條目進行部分查找，如果它們僅匹配Key Locker控制代碼快取中的控制代碼儲存子集，則可能導致控制代碼的延遲不同。只有在完整性標記的底部64位元內的位元才會以這種方式使用。AAD、密文和標籤的上64位元將不會用於部分查找。由於標籤底部64位元的某些子集可能用於部分查找，因此Key Locker指令的惡意使用者可能能夠使用基於時間的旁通道攻擊來透露其他Key Locker使用者的控制代碼位子集。因此只有在完整性標記的底部64位元內的位元才用於部分查找。能夠推斷出64位元完整性標籤的攻擊者將不知道關於完整性標籤的其餘部分或控制代碼的密文部分的資訊。這為256位元AES金鑰提供了320位元的安全容器，並為128位元AES金鑰提供了192位元的安全容器。攻擊者猜測控制代碼的未知位元比直接猜測整個AES金鑰本身更難。

【0549】雖然攻擊者無法使用金鑰快取來推斷其他控制代碼的足夠位元來猜測它們的值，但理論上攻擊者可以

使用金鑰快取來推斷處理器上其他 Key Locker 使用者的控制代碼使用模式。

【0550】

系統韌體啟用

【0551】

SMM 和 SMM 轉移監控器 (STM)

【0552】 Key Locker 的當前實作不允許在 SMM 中使用 AES Key Locker 指令。這透過 CPUID.AESKLE 在 SMM 中時為 0 來列舉到 SMM 軟體，包括 SMI 的預設處理和雙監控器處理。支援 SMM 的未來實作可能允許 CPUID.AESKLE 在 SMM 中為 1，以指示 AES*KL 和 ENCODEKEY* 指令可以在此模式下使用。

【0553】

功能組態

【0554】在支援 MSR_FEATURE_CONFIG (MSR 位址 13CH) 的部件上，如果 MSR_FEATURE_CONFIG[1:0] 的值為 11b，則 AES Key Locker 指令不可用。MSR_FEATURE_CONFIG[1:0] 值為 11b 也會導致 CPUID.AESKLE 為 0，從而導致所有 AES Key Locker 指令產生 #UD 異常。

附加實例

【0555】可以結合以下實例來描述所揭露技術的至少一些實施例：

實例 1. 一種硬體處理器，包含：

核心的解碼器，用以將單一指令解碼成解碼的單一指令，該單一指令包含控制代碼的第一輸入運算元與利用加密金鑰加密的資料的第二輸入運算元，該控制代碼包括加密金鑰的密文(例如，密碼金鑰或其他資料)、完整性標籤和附加認證資料；以及

該核心的執行單元，用以執行該解碼的單一指令，以：

執行該認證標籤對照該密文與該附加認證資料的第一檢查，以檢查對該密文或該附加認證資料之任何修改，

執行該核心的當前請求對照由該控制代碼的該附加認證資料所指定的一或多限制條件的第二檢查，

只有當該第一檢查指示沒有對該密文或該附加認證資料進行修改以及該第二檢查指示沒有違反該一或多限制條件時，將該密文解密以產生加密金鑰；{或在快取中查找原始資料(例如，明文)。此外，即使在檢查失敗時，實作也可以執行解密(或在快取中查找明文)，並且在“檢查失敗”的情況下根本不使用原始資料。或者它可以開始使用原始資料但不將結果返回給使用者(例如，甚至推測性地以防止推測性執行旁通道))，

將使用該加密金鑰進行加密的該資料進行解密以產生未加密的資料；以及

提供該未加密的資料作為該單一指令之結果。

實例 2.如實例 1所述的硬體處理器，其中執行單元在以下情況下執行解碼的單一指令以設置旗標(例如，設立

相應的旗標)：

該第一檢查指示對密文或附加認證資料的修改；或
該第二檢查指示該一或多個限制條件被違反。

實例 3.如實例 1所述的硬體處理器，其中，該解碼器將第二指令解碼為解碼的第二指令，以及該執行單元執行解碼的第二指令，以將用於解密密文的內部金鑰載入到核心的暫存器中。

實例 4.如實例 3所述的硬體處理器，其中，該執行單元是以執行該解碼的第二指令，以在暫存器中設置第一值以指示軟體透過執行該第二指令請求了一特定金鑰作為內部金鑰，以及在暫存器中設置第二值以指示軟體透過執行該第二指令請求了一隨機金鑰作為內部金鑰。

實例 5.如實例 3所述的硬體處理器，其中內部金鑰包含完整性金鑰和單獨的加密金鑰。

實例 6.如實例 1所述的硬體處理器，其中，該解碼器將第二指令解碼為解碼的第二指令，以及該執行單元執行解碼的第二指令，以從加密金鑰與控制代碼限制條件類型的輸入產生控制代碼。

實例 7.如實例 1所述的硬體處理器，其中該附加認證資料包含指示該控制代碼是否可用於加密的欄位。

實例 8.如實例 1所述的硬體處理器，其中該附加認證資料包含指示該控制代碼是否可用於解密的欄位。

實例 9.一種方法，包含：

使用硬體處理器的核心的解碼器，將單一指令解碼為

解碼的單一指令，該單一指令包含控制代碼的第一輸入運算元與利用加密金鑰加密的資料的第二輸入運算元，該控制代碼包括加密金鑰的密文、完整性標籤和附加認證資料；以及

使用該核心的執行單元，執行該解碼的單一指令，以：

執行該認證標籤對照該密文與該附加認證資料的第一檢查，以檢查對該密文或該附加認證資料之任何修改，

執行該核心的當前請求對照由該控制代碼的該附加認證資料所指定的一或多限制條件的第二檢查，

只有當該第一檢查指示沒有對該密文或該附加認證資料進行修改以及該第二檢查指示沒有違反該一或多限制條件時，將該密文解密以產生加密金鑰；

將使用該加密金鑰進行加密的該資料進行解密以產生未加密的資料；以及

提供該未加密的資料作為該單一指令之結果。

實例 10.如實例 9所述的方法，其中該執行該解碼的單一指令會設置一旗標，其條件為：

該第一檢查指示對密文或附加認證資料的修改；或
該第二檢查指示該一或多個限制條件被違反。

實例 11.如實例 9所述的方法，更包含：

用核心的解碼器將第二指令解碼為解碼的第二指令；
以及

使用該核心的該執行單元執行該解碼的第二指令以將

用以解密該密文之內部金鑰載入該核心的暫存器中。

實例 12.如實例 11所述的方法，其中，該執行單元是用以在暫存器中設置第一值以指示軟體透過執行該第二指令請求了一特定金鑰作為內部金鑰，以及在暫存器中設置第二值以指示軟體透過執行該第二指令請求了一隨機金鑰作為內部金鑰。

實例 13.如實例 11所述的方法，其中內部金鑰包含完整性金鑰和單獨的加密金鑰。

實例 14.如實例 9所述的方法，更包含：

用核心的解碼器將第二指令解碼為解碼的第二指令；
以及

使用該核心的該執行單元執行該解碼的第二指令，以從該加密金鑰與控制代碼限制條件類型的輸入產生該控制代碼。

實例 15.如實例 9所述的方法，其中該附加認證資料包含指示該控制代碼是否可用於加密的欄位。

實例 16.如實例 9所述的方法，其中該附加認證資料包含指示該控制代碼是否可用於解密的欄位。

實例 17.一種儲存程式碼的非暫態機器可讀媒體，當機器執行該程式碼時，該程式碼使機器執行一種方法，該方法包含：

使用硬體處理器的核心的解碼器，將單一指令解碼為解碼的單一指令，該單一指令包含控制代碼的第一輸入運算元與利用加密金鑰加密的資料的第二輸入運算元，該控

制代碼包括加密金鑰的密文、完整性標籤和附加認證資料；以及

使用該核心的執行單元，執行該解碼的單一指令，以：

執行該認證標籤對照該密文與該附加認證資料的第一檢查，以檢查對該密文或該附加認證資料之任何修改，

執行該核心的當前請求對照由該控制代碼的該附加認證資料所指定的一或多限制條件的第二檢查，

只有當該第一檢查指示沒有對該密文或該附加認證資料進行修改以及該第二檢查指示沒有違反該一或多限制條件時，將該密文解密以產生加密金鑰；

將使用該加密金鑰進行加密的該資料進行解密以產生未加密的資料；以及

提供該未加密的資料作為該單一指令之結果。

實例 18.如實例 17所述的非暫態機器可讀媒體，其中該執行該解碼的單一指令會設置一旗標，其條件為：

該第一檢查指示對密文或附加認證資料的修改；或

該第二檢查指示該一或多個限制條件被違反。

實例 19.如實例 17所述的非暫態機器可讀媒體，更包含：

用核心的解碼器將第二指令解碼為解碼的第二指令；以及

使用該核心的該執行單元執行該解碼的第二指令以將用以解密該密文之內部金鑰載入該核心的暫存器中。

實例 20.如實例 19所述的非暫態機器可讀媒體，其中，該執行單元是用以在暫存器中設置第一值以指示軟體透過執行該第二指令請求了一特定金鑰作為內部金鑰，以及在暫存器中設置第二值以指示軟體透過執行該第二指令請求了一隨機金鑰作為內部金鑰。

實例 21.如實例 19所述的非暫態機器可讀媒體，其中內部金鑰包含完整性金鑰和單獨的加密金鑰。

實例 22.如實例 17所述的非暫態機器可讀媒體，更包含：

用核心的解碼器將第二指令解碼為解碼的第二指令；
以及

使用該核心的該執行單元執行該解碼的第二指令，以從該加密金鑰與控制代碼限制條件類型的輸入產生該控制代碼。

實例 23.如實例 17所述的非暫態機器可讀媒體，其中該附加認證資料包含指示該控制代碼是否可用於加密的欄位。

實例 24.如實例 17所述的非暫態機器可讀媒體，其中該附加認證資料包含指示該控制代碼是否可用於解密的欄位。

實例 25.一種硬體處理器，包含：

核心的解碼器，用以將單一指令解碼成解碼的單一指令，該單一指令包含控制代碼的輸入運算元，該控制代碼包括原始資料的密文、完整性標籤和附加認證資料；以及

該核心的執行單元，用以執行該解碼的單一指令，
以：

執行該認證標籤對照該密文與該附加認證資料的第一
檢查，以檢查對該密文或該附加認證資料之任何修改，

執行該核心的當前請求對照由該控制代碼的該附加認
證資料所指定的一或多限制條件的第二檢查，

只有當該第一檢查指示沒有對該密文或該附加認證資
料進行修改時以及當該第二檢查指示沒有違反該一或多限
制條件時，將該密文解密以產生原始資料；

對原始資料進行操作，以產生結果資料，以及

提供該結果資料作為該單一指令之結果。

實例 26.如實例 25所述的硬體處理器，其中該附加認
證資料包含指示該控制代碼是否在作業系統外為不可用的
欄位。

實例 27.如實例 25所述的硬體處理器，其中該附加認
證資料包含指示該控制代碼是否在特定可信賴執行環境外
為不可用的欄位。

實例 28.如實例 25所述的硬體處理器，其中該附加認
證資料包含指示該控制代碼是否在以一特定方式設置一特
定部分之作業系統控制狀態時才可為可用的欄位。

實例 29.如實例 25所述的硬體處理器，其中該附加認
證資料包含指示該控制代碼是否可由虛擬機監控器使用但
不可由虛擬機使用的欄位。

實例 30.如實例 25所述的硬體處理器，其中該附加認

證資料包含指示該控制代碼是否可用於解密的欄位。

實例 31.一種設備，其包含：

解碼器，用以將單一指令解碼為解碼的單一指令，該單一指令具有用於指定加密資料的第一來源運算元和用於指定控制代碼的第二來源運算元，該控制代碼包括加密金鑰的密文、完整性標籤和附加認證資料；

執行單元，用以執行該解碼的單一指令，以：

執行該完整性標籤對照該密文與該附加認證資料的第一檢查，以檢查對該密文或該附加認證資料之任何修改，

執行當前請求對照由該控制代碼的該附加認證資料所指定的一或多限制條件的第二檢查，

只有當該第一檢查指示沒有對該密文或該附加認證資料進行修改以及該第二檢查指示沒有違反該一或多限制條件時，將該密文解密以產生加密金鑰；

使用該加密金鑰將該加密資料進行解密以產生未加密的資料；以及

提供該未加密的資料作為該單一指令之結果；及

快取，用以儲存該控制代碼，其中在該控制代碼的查找中僅使用該完整性標籤的一部分。

實例 32.如實例 31所述的設備，其中該完整性標籤的該部分是該完整性標籤的一半。

實例 33.如實例 31所述的設備，其中該完整性標籤是 128 位元，以及該完整性標籤的該部分是 64 位元。

實例 34.如實例 31或 32所述的設備，其中該完整性標

籤是該控制代碼之位元128至256。

實例35.如實例34所述的設備，其中該附加認證資料是該控制代碼之位元0至127。

實例36.如實例35所述的設備，其中該密文是該控制代碼之位元256至383。

實例37.如實例35所述的設備，其中該密文是該控制代碼之位元256至511。

實例38.如實例35所述的設備，其中該附加認證資料包括用以指示該控制代碼是否可用於加密的欄位。

實例39.如實例35所述的設備，其中該附加認證資料包括用以指示該控制代碼是否可用於解密的欄位。

實例40.如實例35所述的設備，其中該附加認證資料包括用以指示該控制代碼是否在一特權級別為可用的欄位。

實例41.如實例35所述的設備，其中該附加認證資料包括用以指示該密文的尺寸的欄位。

實例42.如實例41所述的設備，其中該附加認證資料包括用以指示該密文是128位元或256位元的欄位。

實例43.一種方法，包含：

使用硬體處理器的解碼器，將單一指令解碼為解碼的單一指令，該單一指令具有用於指定加密資料的第一來源運算元和用於指定控制代碼的第二來源運算元，該控制代碼包括加密金鑰的密文、完整性標籤和附加認證資料；

使用該處理器的執行單元，執行該解碼的單一指令，

以：

執行該完整性標籤對照該密文與該附加認證資料的第一檢查，以檢查對該密文或該附加認證資料之任何修改，

執行當前請求對照由該控制代碼的該附加認證資料所指定的一或多限制條件的第二檢查，

只有當該第一檢查指示沒有對該密文或該附加認證資料進行修改以及該第二檢查指示沒有違反該一或多限制條件時，將該密文解密以產生加密金鑰；

使用該加密金鑰將該加密資料進行解密以產生未加密的資料；

提供該未加密的資料作為該單一指令之結果，及

將該控制代碼儲存在該快取中；以及

根據該完整性標籤的一部分從該快取中查找該控制代碼。

實例 44.如實例 43所述的方法，其中該完整性標籤的該子集是該完整性標籤的一半。

實例 45.如實例 43所述的方法，其中該完整性標籤是 128 位元，以及該完整性標籤的該子集是 64 位元。

實例 46.如實例 43 或 44 所述的設備，其中該完整性標籤是該控制代碼之位元 128 至 256。

實例 47.如實例 46 所述的設備，其中該附加認證資料是該控制代碼之位元 0 至 127。

實例 48.如實例 47 所述的方法，其中該附加認證資料包括用以指示該密文是該控制代碼之位元 256 至 383 或位元

256至511的欄位。

實例 49.一種系統，其包含：

記憶體，用以儲存控制代碼；以及

處理器，包括：

快取；

解碼器，用以將單一指令解碼為解碼的單一指令，該單一指令具有用於指定加密資料的第一來源運算元和用於指定控制代碼的第二來源運算元，該控制代碼包括加密金鑰的密文、完整性標籤和附加認證資料；

執行單元，用以執行該解碼的單一指令，以：

從該記憶體讀取該控制代碼，

執行該完整性標籤對照該密文與該附加認證資料的第一檢查，以檢查對該密文或該附加認證資料之任何修改，

執行當前請求對照由該控制代碼的該附加認證資料所指定的一或多限制條件的第二檢查，

只有當該第一檢查指示沒有對該密文或該附加認證資料進行修改以及該第二檢查指示沒有違反該一或多限制條件時，將該密文解密以產生加密金鑰；

使用該加密金鑰將該加密資料進行解密以產生未加密的資料；以及

提供該未加密的資料作為該單一指令之結果；及

將該控制代碼儲存於該快取中，其中在該控制代碼的查找中僅使用該完整性標籤的一部分。

實例 50.如實例 49所述的系統，其中該完整性標籤的

該部分是該完整性標籤的一半。

實例 51.一種設備，其包含：

解碼器機構，用於將單一指令解碼為解碼的單一指令，該單一指令具有用於指定加密資料的第一來源運算元和用於指定控制代碼的第二來源運算元，該控制代碼包括加密金鑰的密文、完整性標籤和附加認證資料；

執行機構，用於執行該解碼的單一指令，以：

執行該完整性標籤對照該密文與該附加認證資料的第一檢查，以檢查對該密文或該附加認證資料之任何修改，

執行當前請求對照由該控制代碼的該附加認證資料所指定的一或多限制條件的第二檢查，

只有當該第一檢查指示沒有對該密文或該附加認證資料進行修改以及該第二檢查指示沒有違反該一或多限制條件時，將該密文解密以產生加密金鑰；

使用該加密金鑰將該加密資料進行解密以產生未加密的資料；以及

提供該未加密的資料作為該單一指令之結果；及

快取機構，用於儲存該控制代碼，其中在該控制代碼的查找中僅使用該完整性標籤的一部分。

實例 52.如實例 51所述的設備，其中該完整性標籤的該部分是該完整性標籤的一半。

實例 53.如實例 52所述的設備，其中該完整性標籤是 128 位元，以及該完整性標籤的該部分是 64 位元。

實例 54.如實例 53所述的設備，其中該完整性標籤是

該控制代碼之位元128至256。

實例55.如實例54所述的設備，其中該附加認證資料是該控制代碼之位元0至127。

【0556】在又一個實施例中，一種設備包括資料儲存裝置，該資料儲存裝置儲存在由硬體處理器執行時使該硬體處理器執行本文揭露的任何方法的碼。一種可以如詳細描述中所述的設備。一種可以如詳細描述中所述的方法。

【符號說明】

【0557】

100:硬體處理器

101:計算系統

102:記憶體

104(1)-(N):核心

106:暫存器

108:暫存器

110:片段暫存器

112:模型特定暫存器

114:控制暫存器

116:能力暫存器

118:內部金鑰暫存器

120:加密金鑰

122:完整性金鑰

124:加密金鑰

126:控制代碼
128:加密資料
128(1)-(X):區塊
130:作業系統碼
132:應用程式碼 1
134:應用程式碼 2
136:虛擬機監控器碼
200:硬體處理器
202:儲存器
204:加密/解密指令
206:解碼器
208:排程器電路
210:暫存器檔案/記憶體電路
212:執行電路
214:寫回電路
216:控制代碼
218:未加密資料
220:加密資料
300:硬體處理器
302:儲存器
304:內部金鑰指令
306:解碼器
308:排程器電路
310:暫存器檔案/記憶體電路

312:執行 電 路
314:寫 回 電 路
316:內 部 金 鑰 儲 存 器
400:硬 體 處 理 器
402:儲 存 器
404:指 令
406:解 碼 器
408:排 程 器 電 路
410:暫 存 器 檔 案 / 記 憶 體 電 路
412:執行 電 路
414:寫 回 電 路
416:控 制 代 碼
502:主 機
504:訪 客 0
506:訪 客 1
508:訪 客 0 金 鑰
510:訪 客 1 金 鑰
512:VMC
514:VMC
600:方 法
602:步 驟
604:步 驟
606:步 驟
608:步 驟

610:步驟

612:步驟

700:通用向量友善指令格式

705:無記憶體存取

710:無記憶體存取、全捨入控制類型操作

712:無記憶體存取、寫入遮罩控制、部分捨入控制類型操作

715:無記憶體存取、資料轉換類型操作

717:無記憶體存取、寫入遮罩控制、`vsize`類型操作

720:記憶體存取

725:記憶體存取、時序

727:記憶體存取、寫入遮罩控制

730:記憶體存取、非時序

740:格式欄位

742:基礎操作欄位

744:暫存器索引欄位

746:修飾符欄位

746A:無記憶體存取欄位

746B:記憶體存取欄位

750:擴充操作欄位

752:Alpha欄位

752A:RS欄位

752A.1:捨入

752A.2:資料轉換

752B:驅逐提示 (EH) 欄位

752B.1:時序

752B.2:非時序

752C:寫入遮罩控制欄位

754:Beta欄位

754A:捨入控制欄位

754B:資料轉換欄位

754C:資料調處欄位

756:抑制全部浮點例外 (SAE) 欄位

757A:RL欄位

757A.1:捨入

757A.2:向量長度

757B:廣播欄位

758:捨入操作控制欄位

759A:捨入操作欄位

759B:向量長度欄位

760:比例欄位

762A:位移欄位

762B:位移因子欄位

764:資料元素寬度欄位

768:類別欄位

768A:A類別

768B:B類別

770:寫入遮罩欄位

772:立即數欄位
774:完整運算碼欄位
800:特定向量友善指令格式
802:EVEX前綴
805:REX欄位
810:REX'欄位
810A:REX'
810B:REX'
815:運算碼映射欄位
820:VVVV欄位
825:前綴編碼欄位
830:真實運算碼欄位
840:MOD R/M欄位
842:MOD欄位
844:reg欄位
846:r/m欄位
850:SIB
852:比例欄位
854:XXX欄位
856:BBB欄位
900:暫存器架構
910:向量暫存器
915:寫入遮罩暫存器
925:通用暫存器

945:純量浮點推疊暫存器檔案
950:MMX緊縮整數平暫存器檔案
1000:管線
1002:提取級
1004:長度解碼級
1006:解碼級
1008:分配級
1010:更名級
1012:排程級
1014:暫存器讀取/記憶體讀取級
1016:執行級
1018:寫回/記憶體寫入級
1022:異常處理級
1024:提交級
1030:前端單元
1032:分支預測單元
1034:指令快取單元
1036:指令轉譯旁視緩衝器單元
1038:指令提取單元
1040:解碼單元
1050:執行引擎單元
1052:更名/分配器單元
1054:退役單元
1056:排程器單元

1058:實體暫存器檔案單元
1060:執行叢集
1062:執行單元
1064:記憶體存取單元
1070:記憶體單元
1072:資料TLB單元
1074:資料快取單元
1076:第2階(L2)快取單元
1078:預取電路
1080:記憶體
1090:核心
1100:指令解碼單元
1102:環狀網路
1104:第2階(L2)快取之本地子集
1106:L1快取
1106A:L1資料快取
1108:純量單元
1110:向量單元
1112:純量暫存器
1114:向量暫存器
1120:拌和單元
1122A:數字轉換單元
1122B:數字轉換單元
1124:複製單元

1126:寫入遮罩暫存器
1128:16位元寬向量 ALU
1200:處理器
1202A-N:核心
1204A-N:快取單元
1206:共享快取單元
1208:特殊用途邏輯
1210:系統代理單元
1212:環
1214:整合記憶體控制器單元
1216:匯流排控制器單元
1300:系統
1310:處理器
1315:處理器
1320:控制器集線器
1340:記憶體
1340A:加密/解密碼
1345:共處理器
1350:IOH
1360:輸入/輸出(I/O)裝置
1390:圖形記憶體控制器集線器(GMCH)
1395:連接
1400:多處理器系統
1414:I/O裝置

1415:處理器
1416:第一匯流排
1418:匯流排橋接器
1420:第二匯流排
1422:鍵盤及/或滑鼠
1424:音訊 I/O
1427:通訊裝置
1428:資料儲存器單元
1430:碼及資料
1432:記憶體
1434:記憶體
1438:共處理器
1439:高效能介面
1450:點對點互連
1452:P-P介面
1454:P-P介面
1470:處理器
1472:記憶體控制器(IMC)單元
1476:點對點介面電路
1478:點對點介面電路
1480:處理器
1482:記憶體控制器(IMC)單元
1486:點對點介面電路
1488:點對點介面電路

1490:晶片組
1492:高效能介面
1494:點對點介面電路
1496:介面
1498:點對點介面電路
1500:系統
1514:I/O裝置
1515:舊有 I/O 裝置
1600:SoC
1602:互連單元
1610:應用處理器
1620:共處理器
1630:靜態隨機存取記憶體 (SRAM) 單元
1632:直接記憶體存取 (DMA) 單元
1640:顯示器單元
1702:高階語言
1704:x86編譯器
1706:x86二進制碼
1708:指令集編譯器
1710:指令集二進制碼
1712:指令轉換器
1714:無 x86指令集核心之處理器
1716:具有至少一個 x86指令集核心之處理器
1802:區塊

1804:區塊
1806:區塊
1808:區塊
1810:區塊
1902:區塊
1904:區塊
1906:區塊
1908:記憶體位置
2002:區塊
2004:區塊
2006:區塊
2100:系統
2110:處理器
2120:系統記憶體
2130:圖形處理器
2132:顯示器
2140:週邊控制代理器
2142:裝置
2150:資訊儲存裝置
2160:金鑰管理硬體及/或邏輯
2162:載入金鑰硬體/邏輯
2164:保存金鑰硬體/邏輯
2166:恢復金鑰硬體/邏輯
2168:本地金鑰暫存器

2170:備份金鑰暫存器
2200:處理器
2210:儲存器單元
2212:金鑰來源暫存器
2214:本地金鑰暫存器
2216:備份金鑰暫存器
2218:控制暫存器
2220:指令單元
2230:執行單元
2232:載入金鑰硬體
2234:保存金鑰硬體
2236:恢復金鑰硬體
2238:密碼硬體
2240:控制單元
2242:金鑰管理控制
2300:多核心處理器
2310:核心
2312:金鑰管理單元
2314:金鑰來源暫存器
2316:本地金鑰暫存器
2320:核心
2322:金鑰管理單元
2324:金鑰來源暫存器
2326:本地金鑰暫存器

- 2330:核 心
- 2332:金 鑰 管 理 單 元
- 2334:金 鑰 來 源 暫 存 器
- 2336:本 地 金 鑰 暫 存 器
- 2340:核 心
- 2342:金 鑰 管 理 單 元
- 2344:金 鑰 來 源 暫 存 器
- 2346:本 地 金 鑰 暫 存 器
- 2350:備 份 金 鑰 暫 存 器
- 2400:方 法
- 2410:方 塊
- 2412:方 塊
- 2420:方 塊
- 2422:方 塊
- 2424:方 塊
- 2426:方 塊
- 2428:方 塊
- 2430:方 塊
- 2432:方 塊
- 2434:方 塊
- 2440:方 塊
- 2442:方 塊
- 2444:方 塊
- 2446:方 塊

- 2448:方 塊
- 2450:方 塊
- 2452:方 塊
- 2454:方 塊
- 2460:方 塊
- 2462:方 塊
- 2464:方 塊
- 2466:方 塊
- 2468:方 塊
- 2470:方 塊
- 2472:方 塊
- 2474:方 塊
- 2476:方 塊

【發明申請專利範圍】

【請求項1】一種用於創建、使用、及管理受保護的密碼金鑰的設備，其包含：

解碼器，用以將單一指令解碼為解碼的單一指令，該單一指令具有用於指定加密資料的第一來源運算元和用於指定控制代碼的第二來源運算元，該控制代碼包括加密金鑰的密文、完整性標籤和附加認證資料；

執行單元，用以執行該解碼的單一指令，以：

執行該完整性標籤對照該密文與該附加認證資料的第一檢查，以檢查對該密文或該附加認證資料之任何修改，

執行當前請求對照由該控制代碼的該附加認證資料所指定的一或多限制條件的第二檢查，

只有當該第一檢查指示沒有對該密文或該附加認證資料進行修改以及該第二檢查指示沒有違反該一或多限制條件時，將該密文解密以產生加密金鑰；

使用該加密金鑰將該加密資料進行解密以產生未加密的資料；以及

提供該未加密的資料作為該單一指令之結果；及

快取，用以儲存該控制代碼，其中在該控制代碼的查找中僅使用該完整性標籤的一部分；

其中該執行該解碼的單一指令會設置一旗標，其條件為：

該第一檢查指示對密文或附加認證資料的修改；或
該第二檢查指示該一或多個限制條件被違反。

【請求項2】如請求項1所述之設備，其中該完整性標籤的該部分是該完整性標籤的一半。

【請求項3】如請求項1所述之設備，其中該完整性標籤是128位元，以及該完整性標籤的該部分是64位元。

【請求項4】如請求項1或2所述之設備，其中該完整性標籤是該控制代碼之位元128至256。

【請求項5】如請求項4所述之設備，其中該附加認證資料是該控制代碼之位元0至127。

【請求項6】如請求項5所述之設備，其中該密文是該控制代碼之位元256至383。

【請求項7】如請求項5所述之設備，其中該密文是該控制代碼之位元256至511。

【請求項8】如請求項5所述之設備，其中該附加認證資料包括用以指示該控制代碼是否可用於加密的欄位。

【請求項9】如請求項5所述之設備，其中該附加認證資料包括用以指示該控制代碼是否可用於解密的欄位。

【請求項10】如請求項5所述之設備，其中該附加認證資料包括用以指示該控制代碼是否在一特權級別為可用的欄位。

【請求項11】如請求項5所述之設備，其中該附加認證資料包括用以指示該密文的尺寸的欄位。

【請求項12】如請求項11所述之設備，其中該附加認

證資料包括用以指示該密文是128位元或256位元的欄位。

【請求項13】一種用於創建、使用、及管理受保護的密碼金鑰的方法，包含：

使用硬體處理器的解碼器，將單一指令解碼為解碼的單一指令，該單一指令具有用於指定加密資料的第一來源運算元和用於指定控制代碼的第二來源運算元，該控制代碼包括加密金鑰的密文、完整性標籤和附加認證資料；

使用該處理器的執行單元，執行該解碼的單一指令，以：

執行該完整性標籤對照該密文與該附加認證資料的第一檢查，以檢查對該密文或該附加認證資料之任何修改，

執行當前請求對照由該控制代碼的該附加認證資料所指定的一或多限制條件的第二檢查，

只有當該第一檢查指示沒有對該密文或該附加認證資料進行修改以及該第二檢查指示沒有違反該一或多限制條件時，將該密文解密以產生加密金鑰；

使用該加密金鑰將該加密資料進行解密以產生未加密的資料；

提供該未加密的資料作為該單一指令之結果，及

將該控制代碼儲存在快取中；以及

根據該完整性標籤的一部分從該快取中查找該控制代碼；

其中該執行該解碼的單一指令會設置一旗標，其條件為：

該第一檢查指示對密文或附加認證資料的修改；或
該第二檢查指示該一或多個限制條件被違反。

【請求項14】如請求項13所述之方法，其中該完整性標籤的子集是該完整性標籤的一半。

【請求項15】如請求項14所述之方法，其中該完整性標籤是128位元，以及該完整性標籤的該子集是64位元。

【請求項16】如請求項13或14所述之方法，其中該完整性標籤是該控制代碼之位元128至256。

【請求項17】如請求項16所述之方法，其中該附加認證資料是該控制代碼之位元0至127。

【請求項18】如請求項17所述之方法，其中該附加認證資料包括用以指示該密文是該控制代碼之位元256至383或位元256至511的欄位。

【請求項19】一種用於創建、使用、及管理受保護的密碼金鑰的系統，其包含：

記憶體，用以儲存控制代碼；以及

處理器，包括：

快取；

解碼器，用以將單一指令解碼為解碼的單一指令，該單一指令具有用於指定加密資料的第一來源運算元和用於指定控制代碼的第二來源運算元，該控制代碼包括加密金鑰的密文、完整性標籤和附加

認證資料；

執行單元，用以執行該解碼的單一指令，以：

從該記憶體讀取該控制代碼，

執行該完整性標籤對照該密文與該附加認證資料的第一檢查，以檢查對該密文或該附加認證資料之任何修改，

執行當前請求對照由該控制代碼的該附加認證資料所指定的一或多限制條件的第二檢查，

只有當該第一檢查指示沒有對該密文或該附加認證資料進行修改以及該第二檢查指示沒有違反該一或多限制條件時，將該密文解密以產生加密金鑰；

使用該加密金鑰將該加密資料進行解密以產生未加密的資料；以及

提供該未加密的資料作為該單一指令之結果；及

將該控制代碼儲存於該快取中，其中在該控制代碼的查找中僅使用該完整性標籤的一部分；

其中該執行該解碼的單一指令會設置一旗標，其條件為：

該第一檢查指示對密文或附加認證資料的修改；或

該第二檢查指示該一或多個限制條件被違反。

【請求項20】如請求項19所述之系統，其中該完整性標籤的該部分是該完整性標籤的一半。

【請求項21】一種用於創建、使用、及管理受保護的密碼金鑰的設備，其包含：

解碼器機構，用於將單一指令解碼為解碼的單一指令，該單一指令具有用於指定加密資料的第一來源運算元和用於指定控制代碼的第二來源運算元，該控制代碼包括加密金鑰的密文、完整性標籤和附加認證資料；

執行機構，用於執行該解碼的單一指令，以：

執行該完整性標籤對照該密文與該附加認證資料的第一檢查，以檢查對該密文或該附加認證資料之任何修改，

執行當前請求對照由該控制代碼的該附加認證資料所指定的一或多限制條件的第二檢查，

只有當該第一檢查指示沒有對該密文或該附加認證資料進行修改以及該第二檢查指示沒有違反該一或多限制條件時，將該密文解密以產生加密金鑰；

使用該加密金鑰將該加密資料進行解密以產生未加密的資料；以及

提供該未加密的資料作為該單一指令之結果；及

快取機構，用於儲存該控制代碼，其中在該控制代碼的查找中僅使用該完整性標籤的一部分；

其中該執行該解碼的單一指令會設置一旗標，其條件為：

該第一檢查指示對密文或附加認證資料的修改；或
該第二檢查指示該一或多個限制條件被違反。

【請求項22】如請求項21所述之設備，其中該完整性

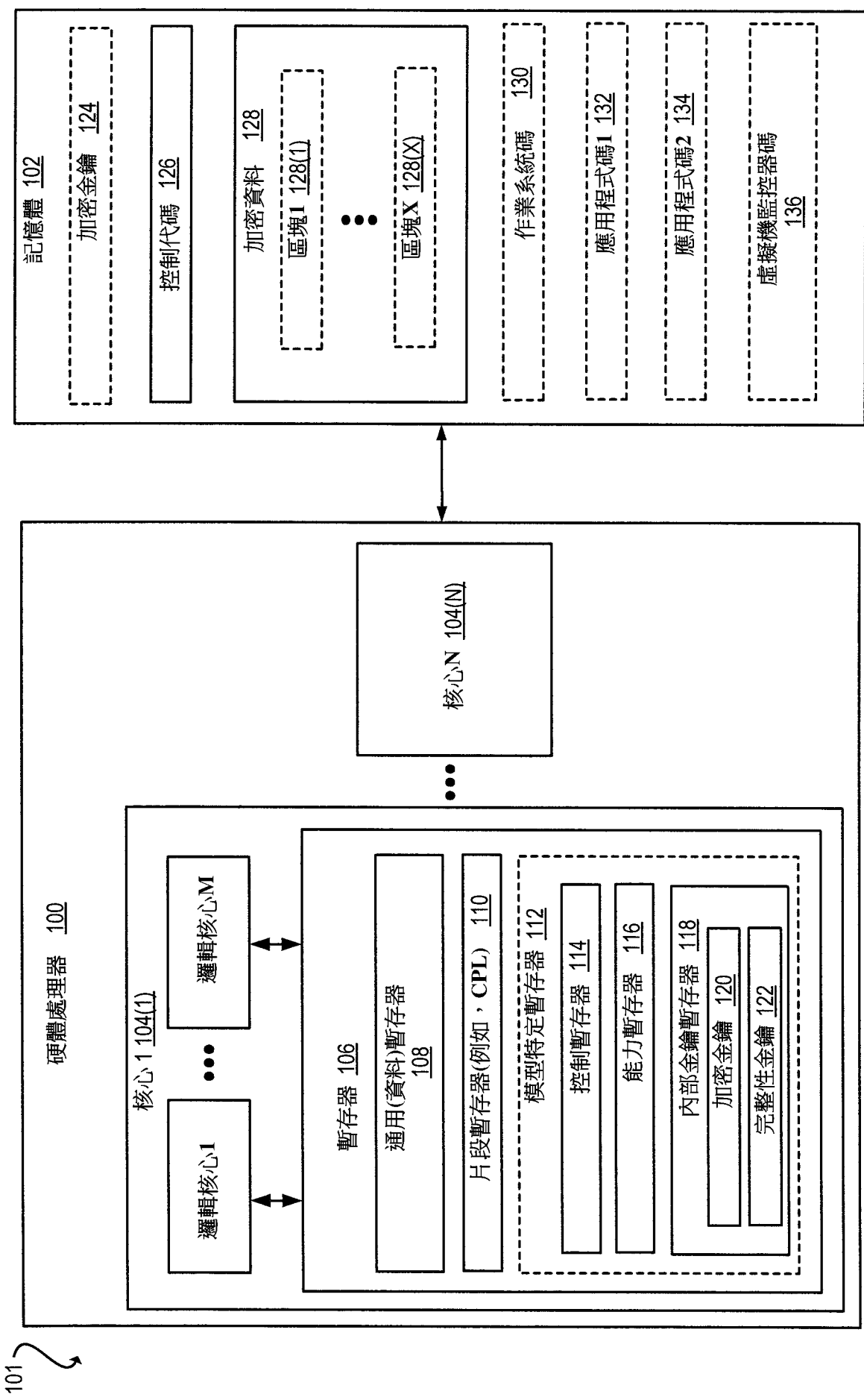
標籤的該部分是該完整性標籤的一半。

【請求項23】如請求項22所述之設備，其中該完整性標籤是128位元，以及該完整性標籤的該部分是64位元。

【請求項24】如請求項23所述之設備，其中該完整性標籤是該控制代碼之位元128至256。

【請求項25】如請求項24所述之設備，其中該附加認證資料是該控制代碼之位元0至127。

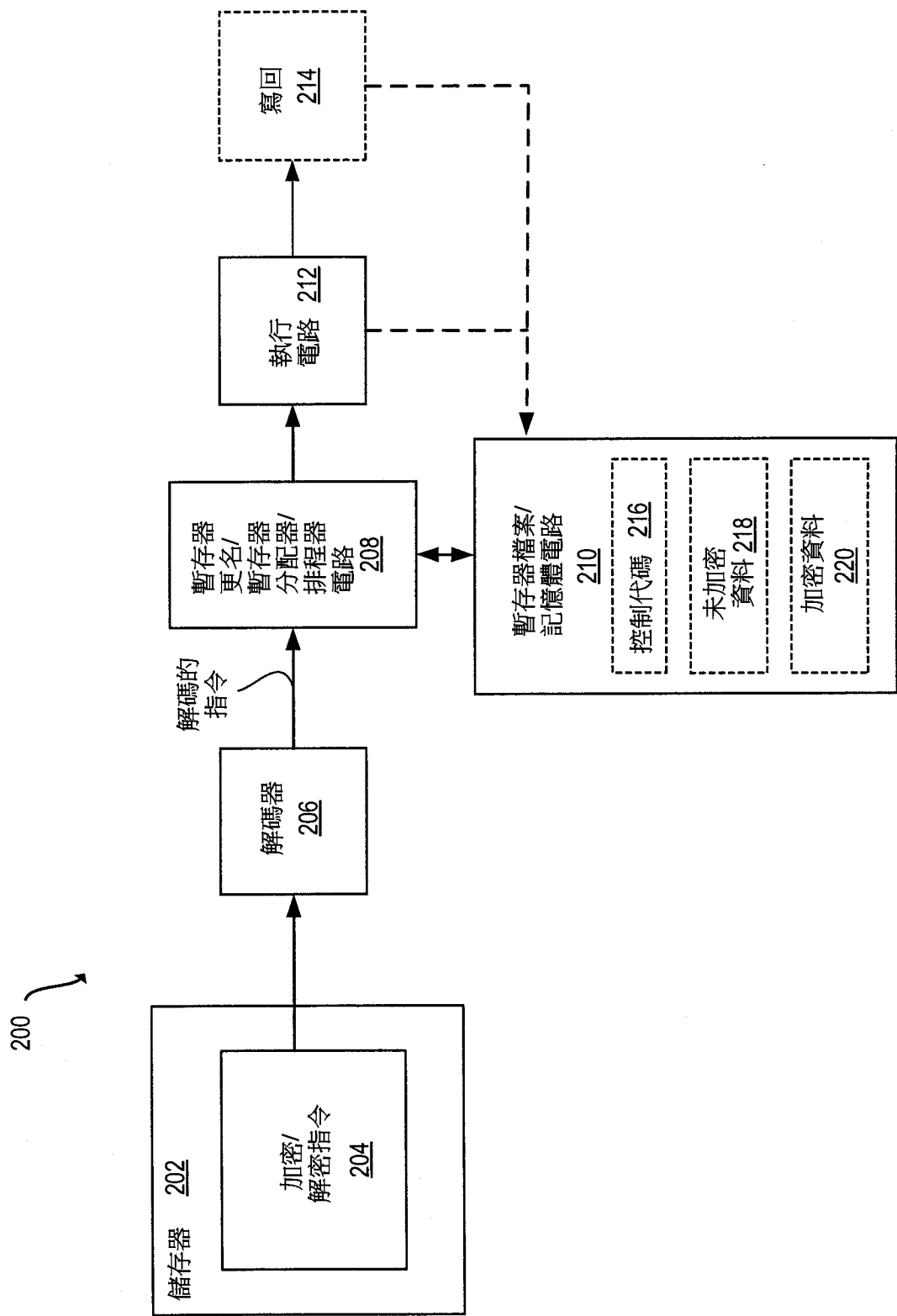
【發明圖式】



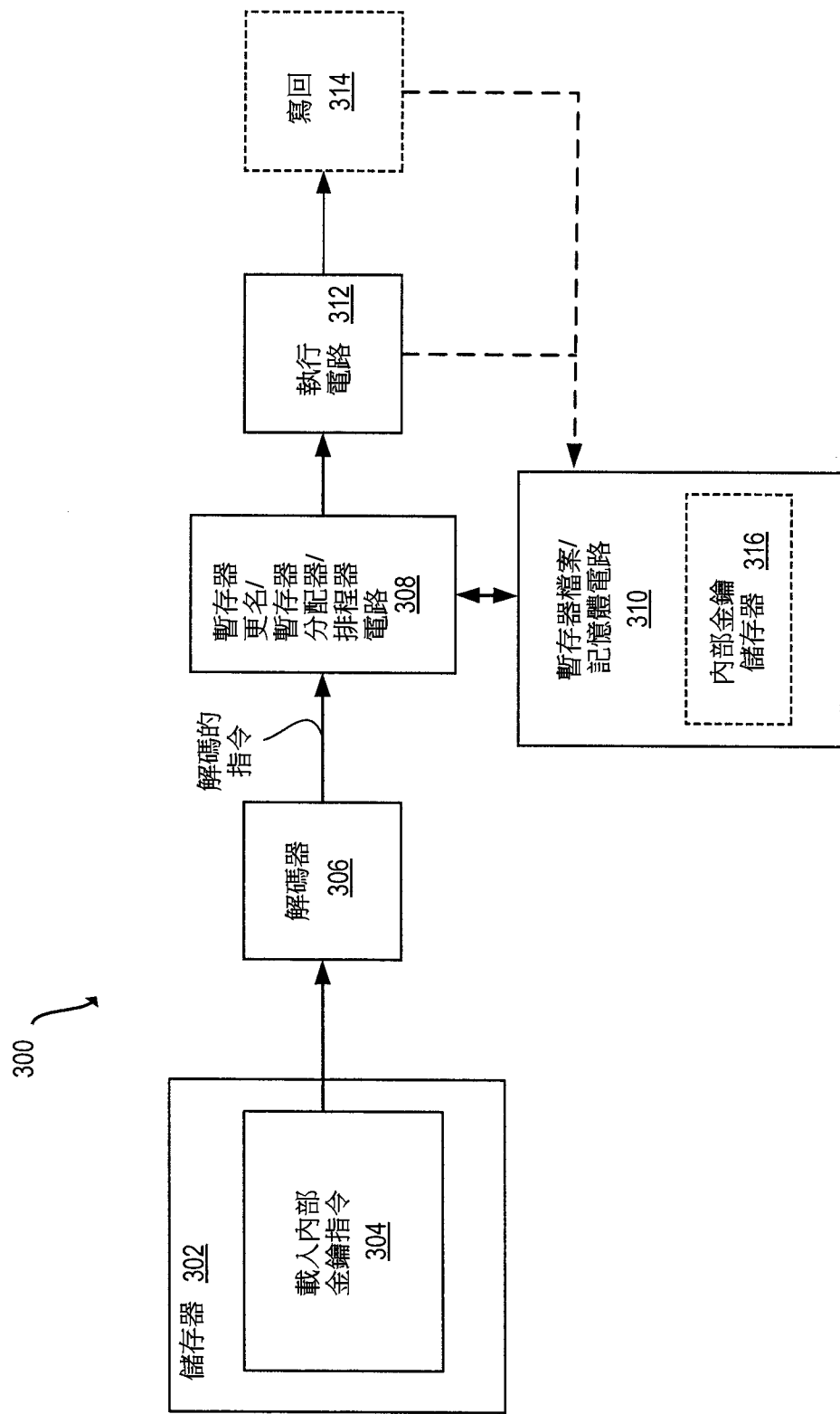
【圖 1】

101

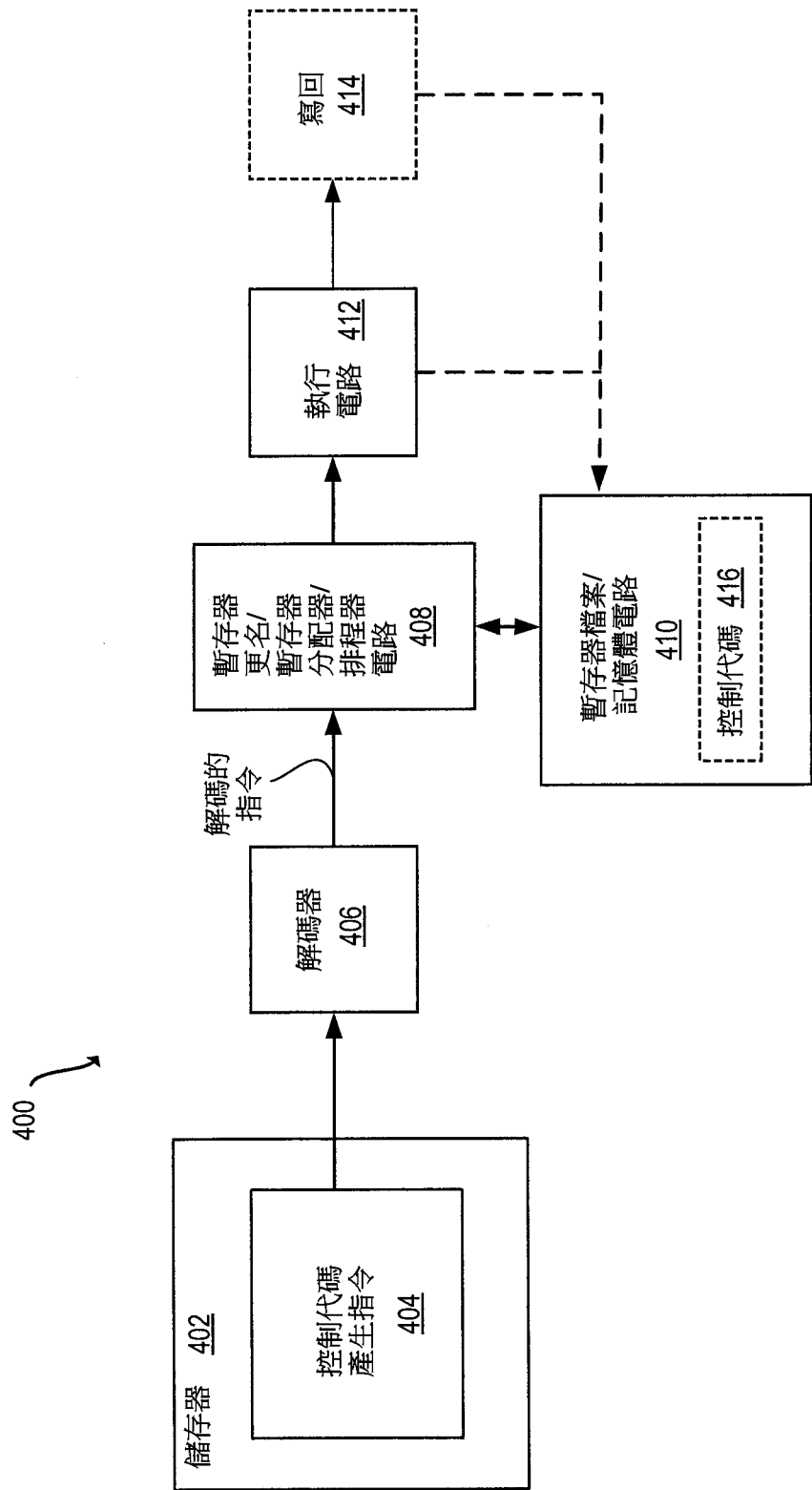
893479



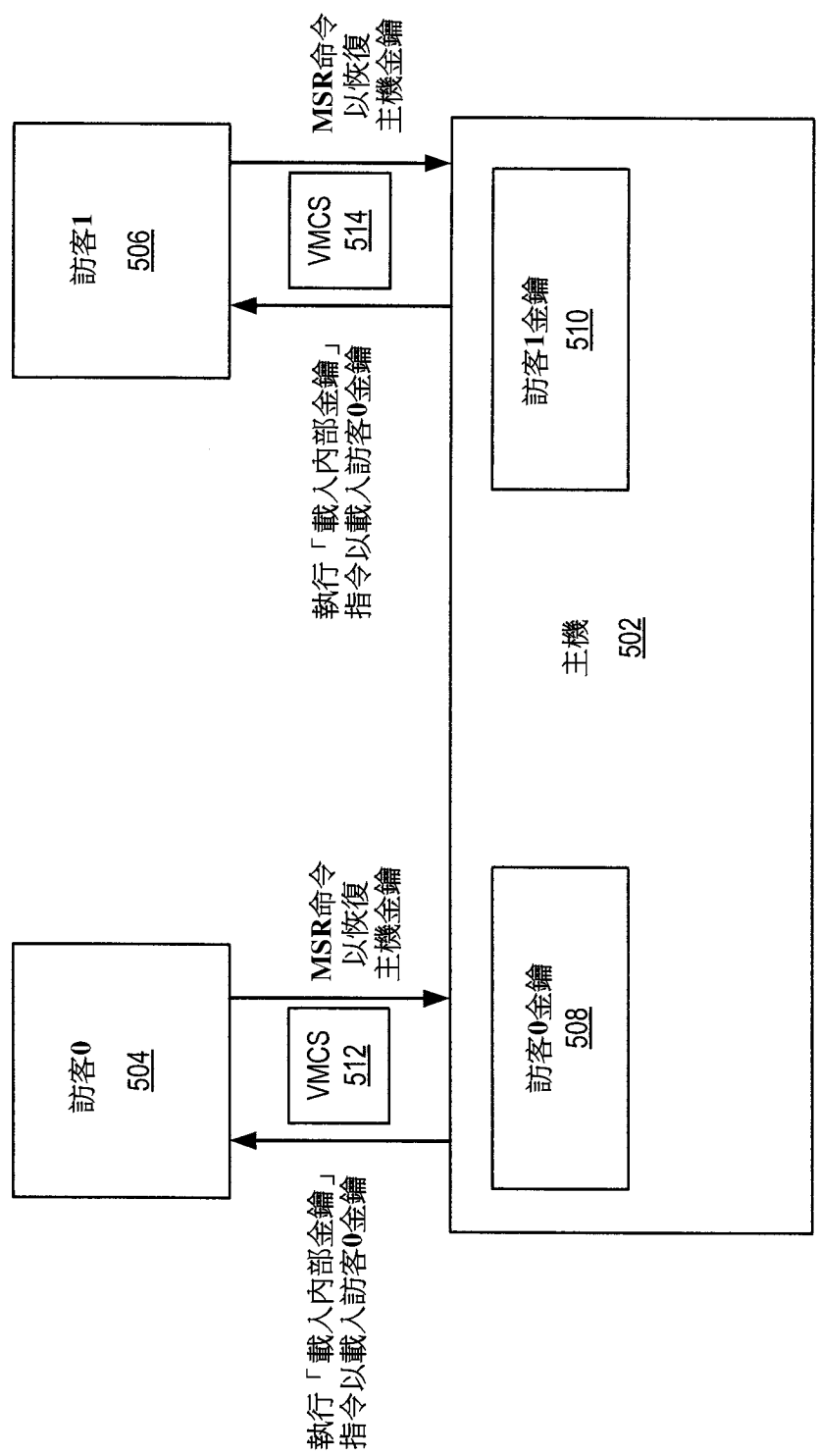
【圖 2】



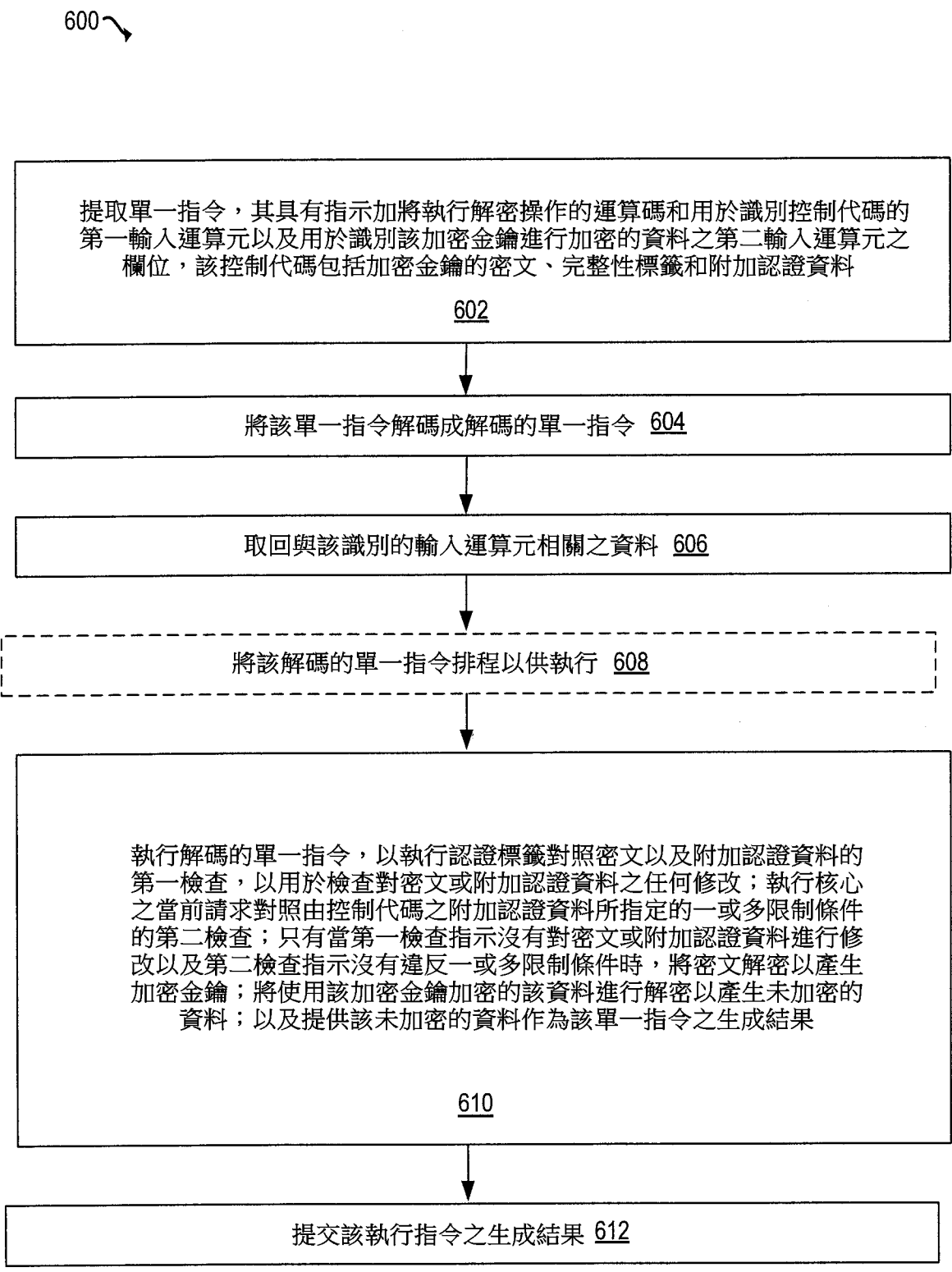
【圖 3】



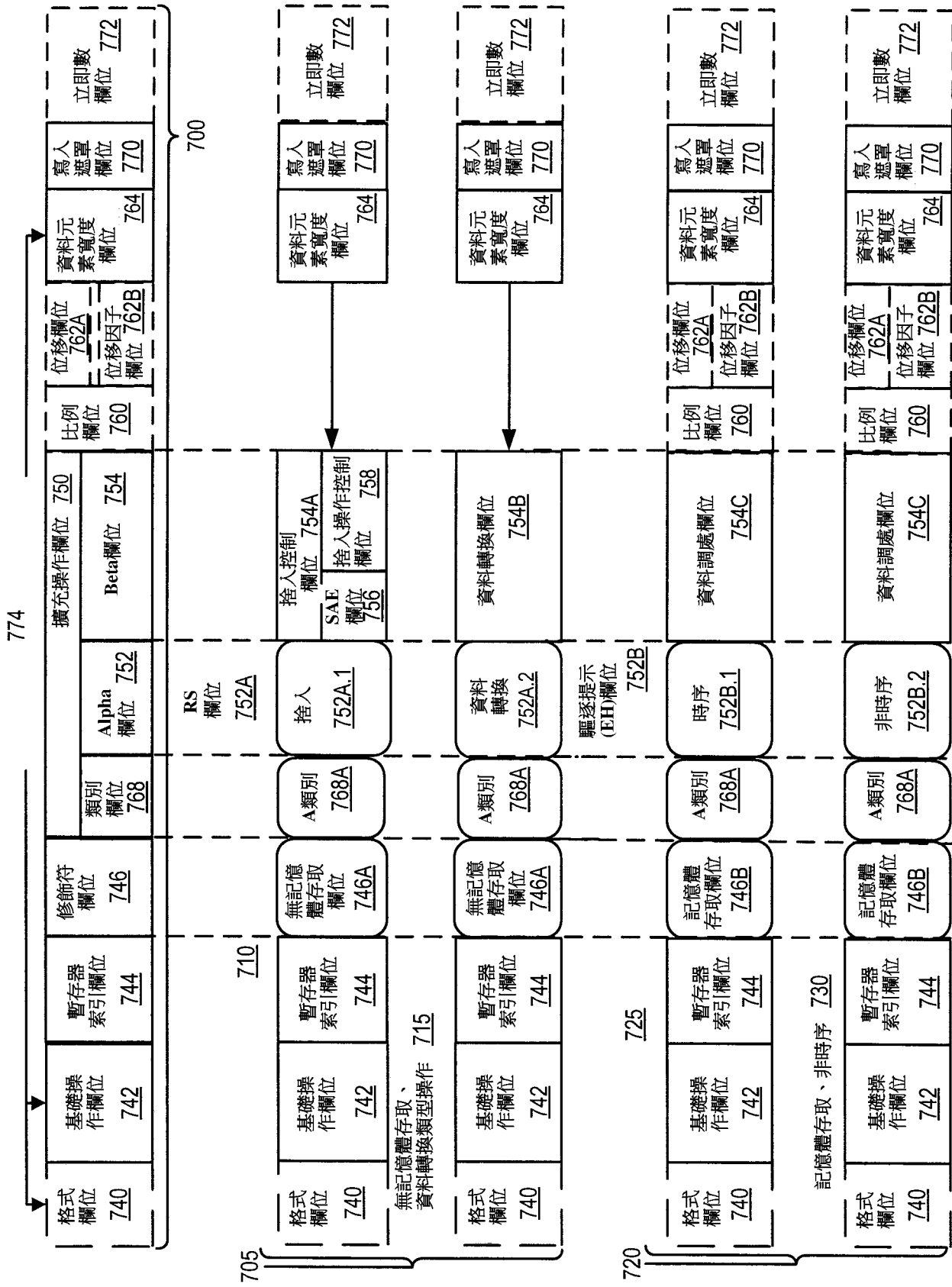
【圖 4】



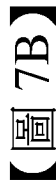
【圖 5】

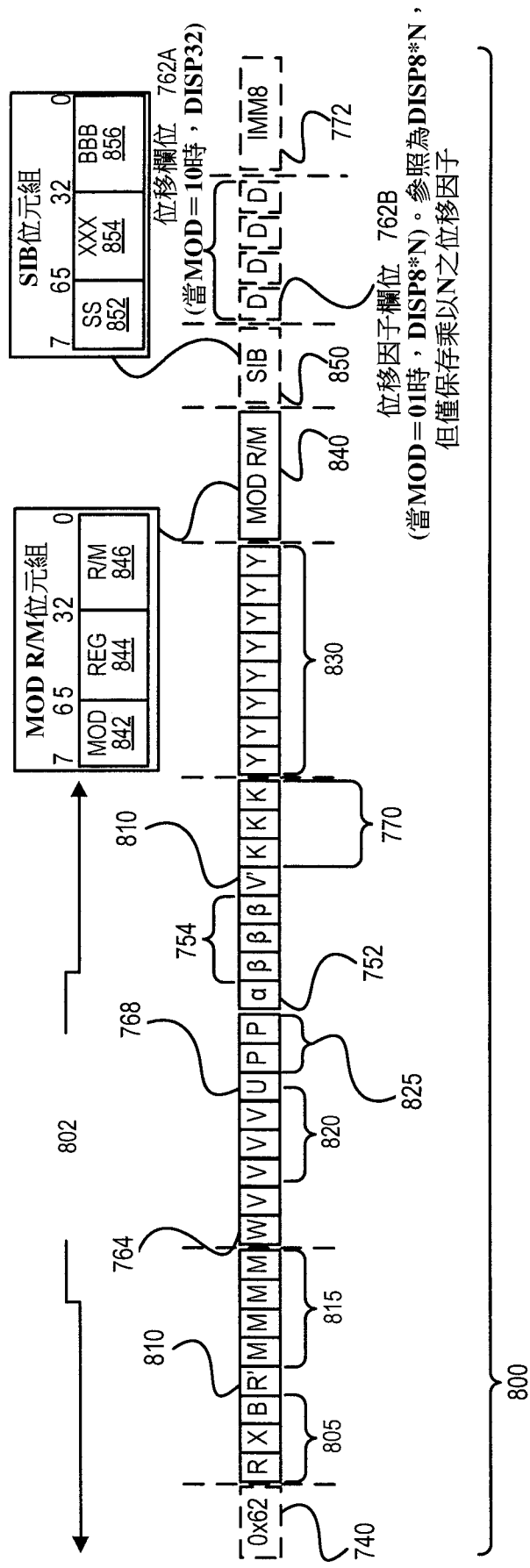


【圖 6】

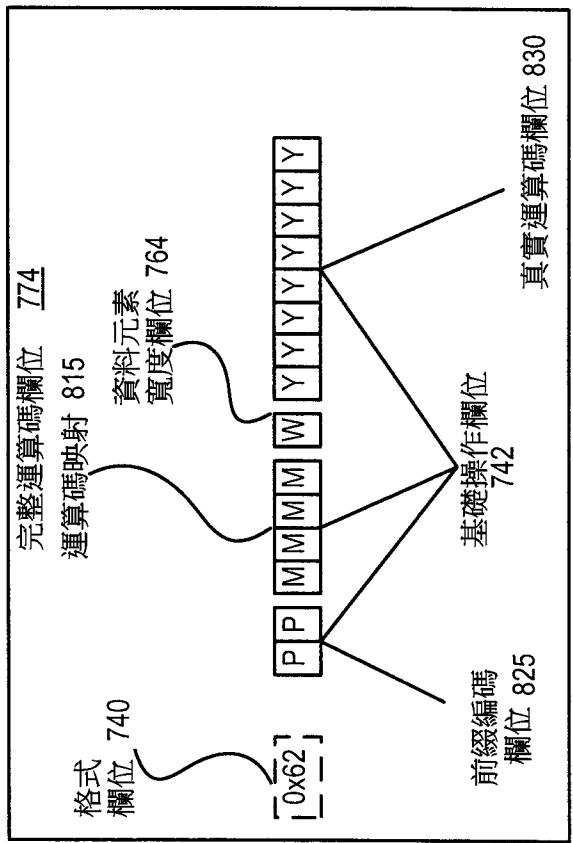


【圖 7A】

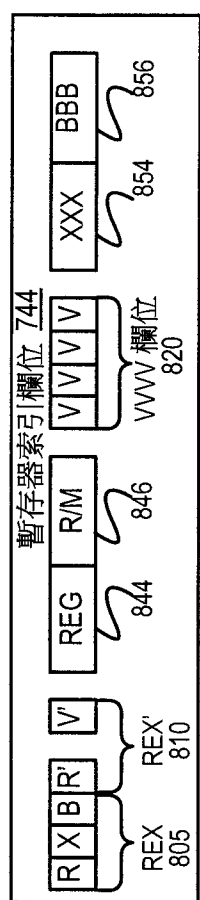




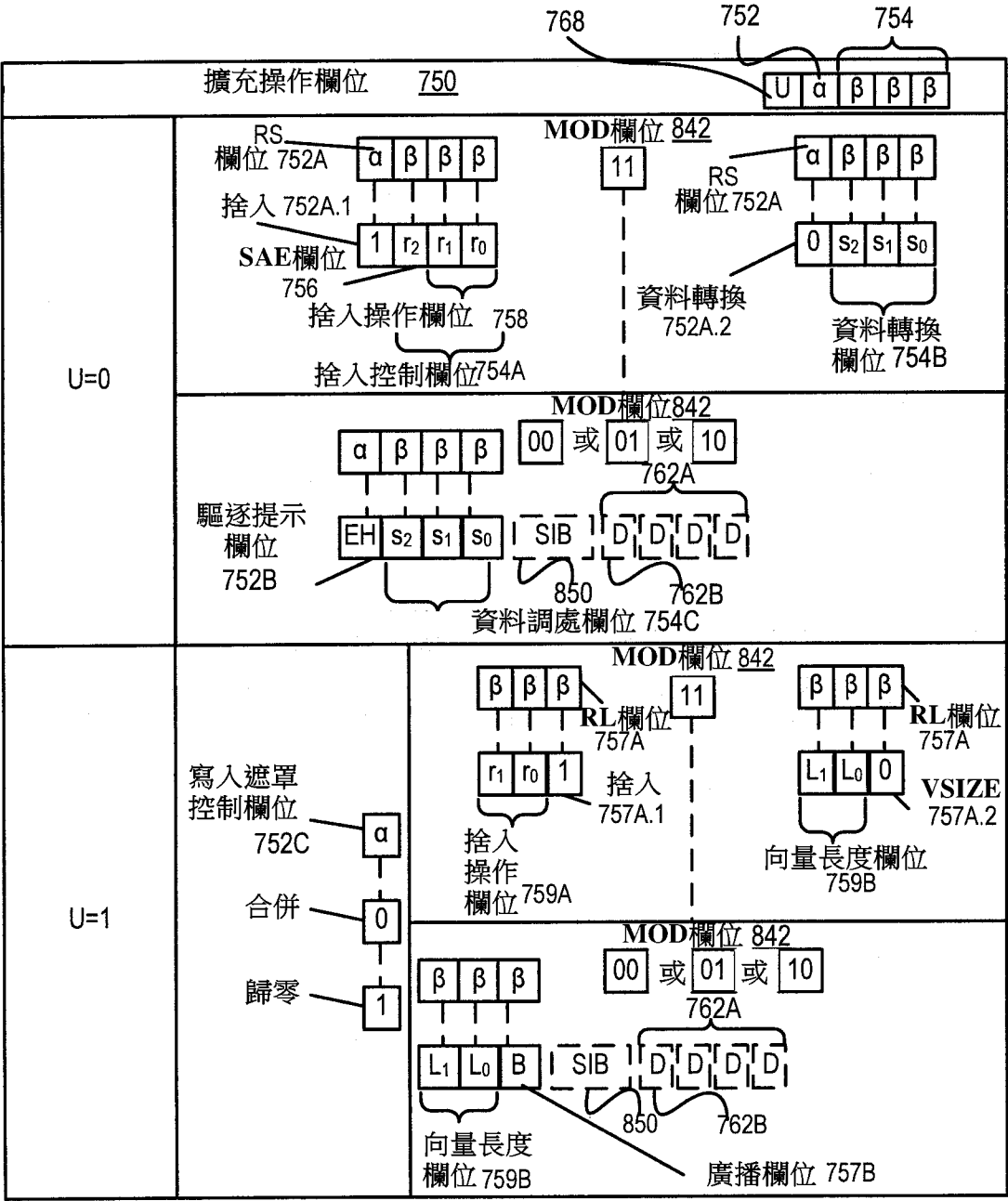
【圖 8A】



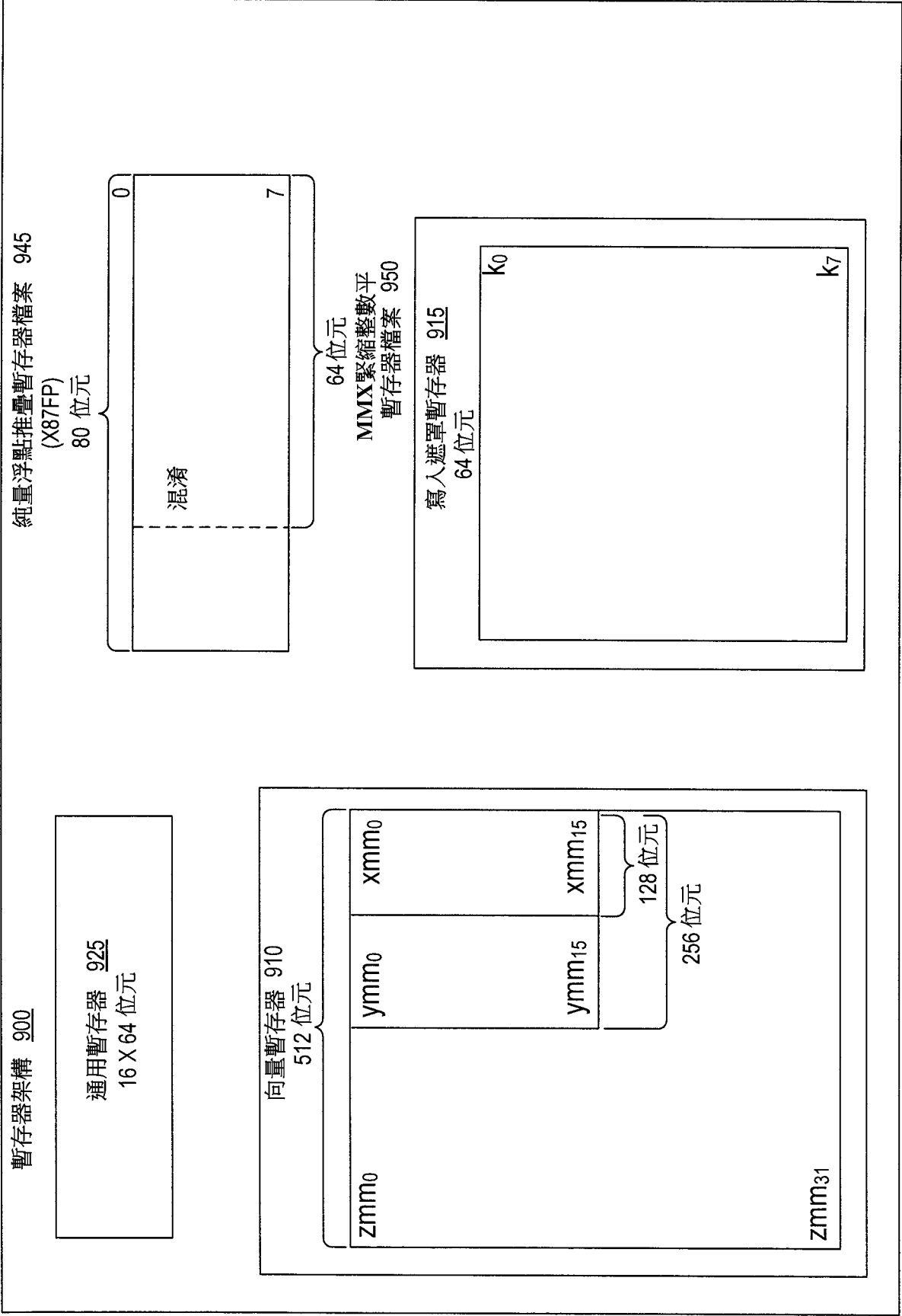
【圖 8B】



【圖 8C】

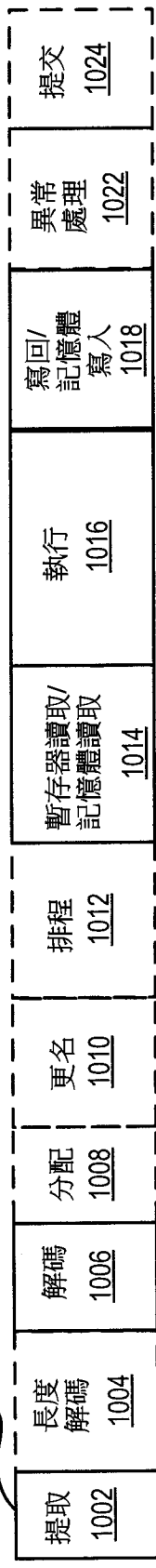


【圖 8D】

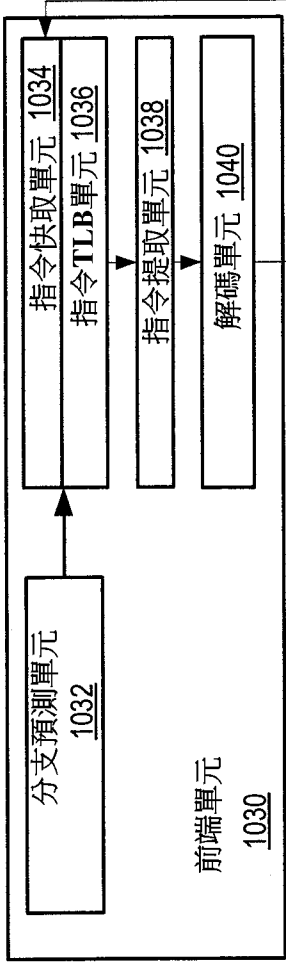


【圖 9】

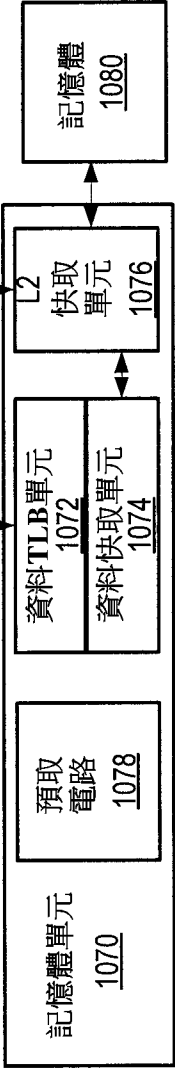
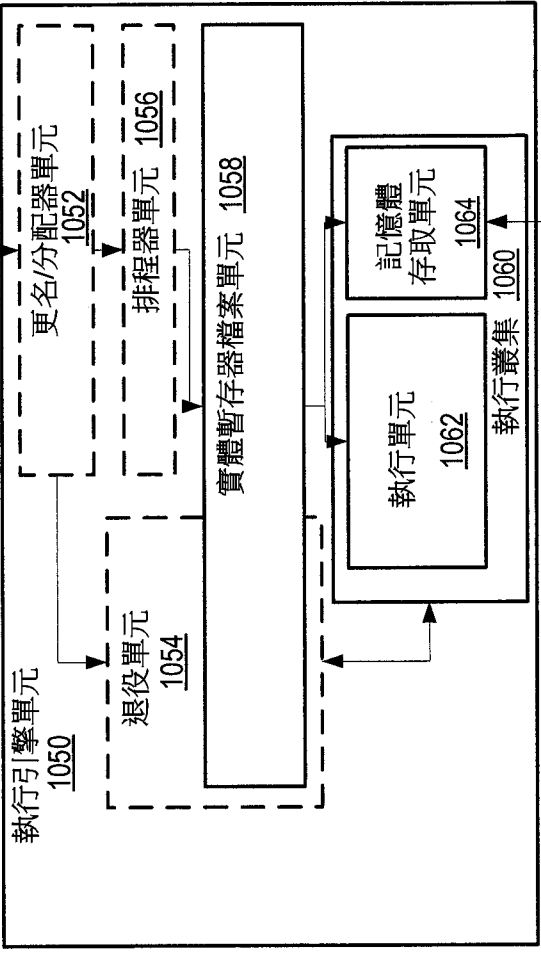
1000



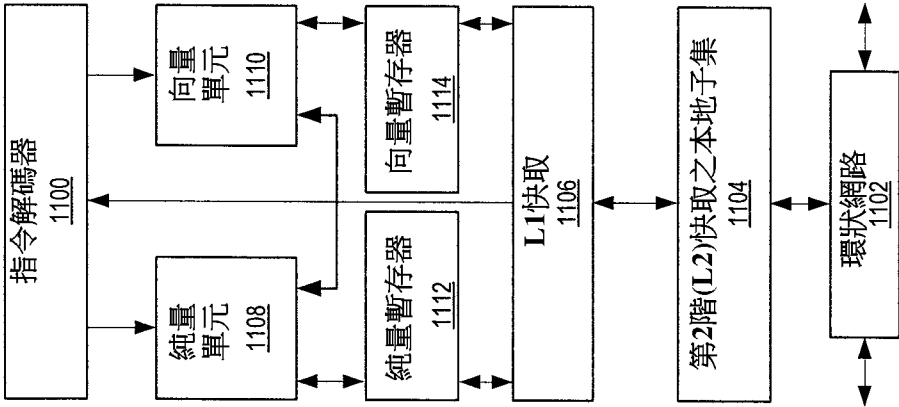
【圖 10A】



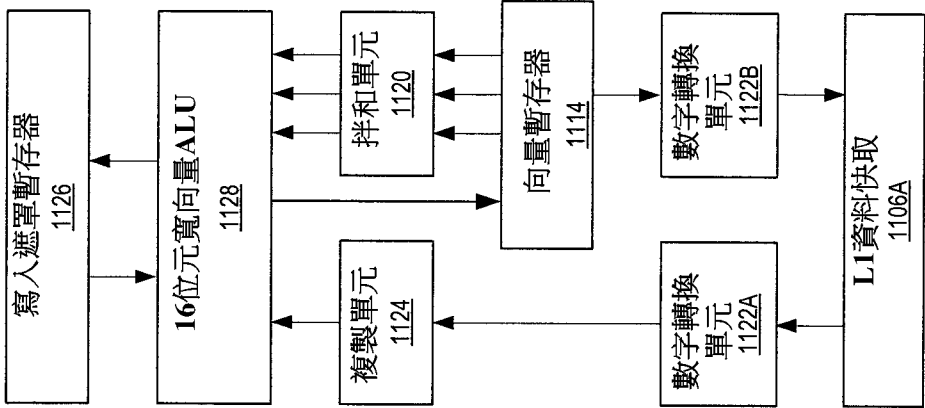
1090



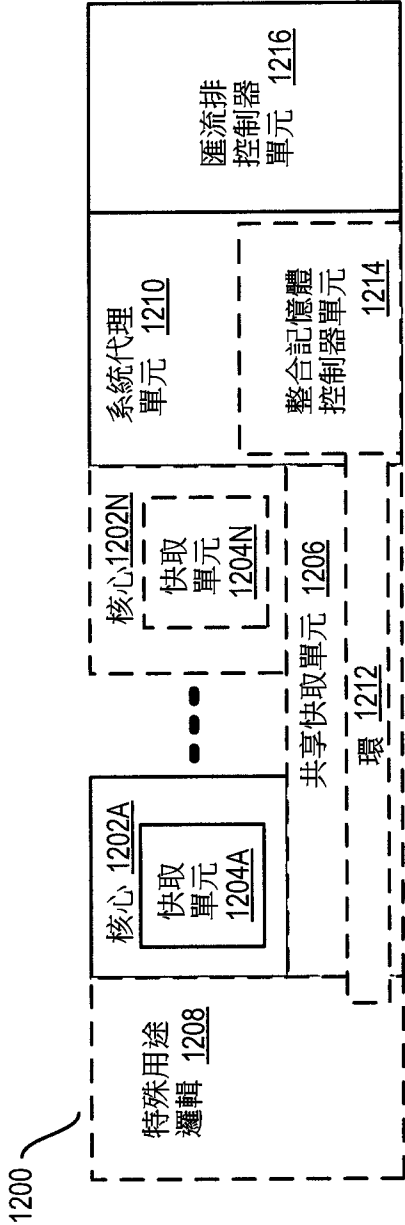
【圖 10B】



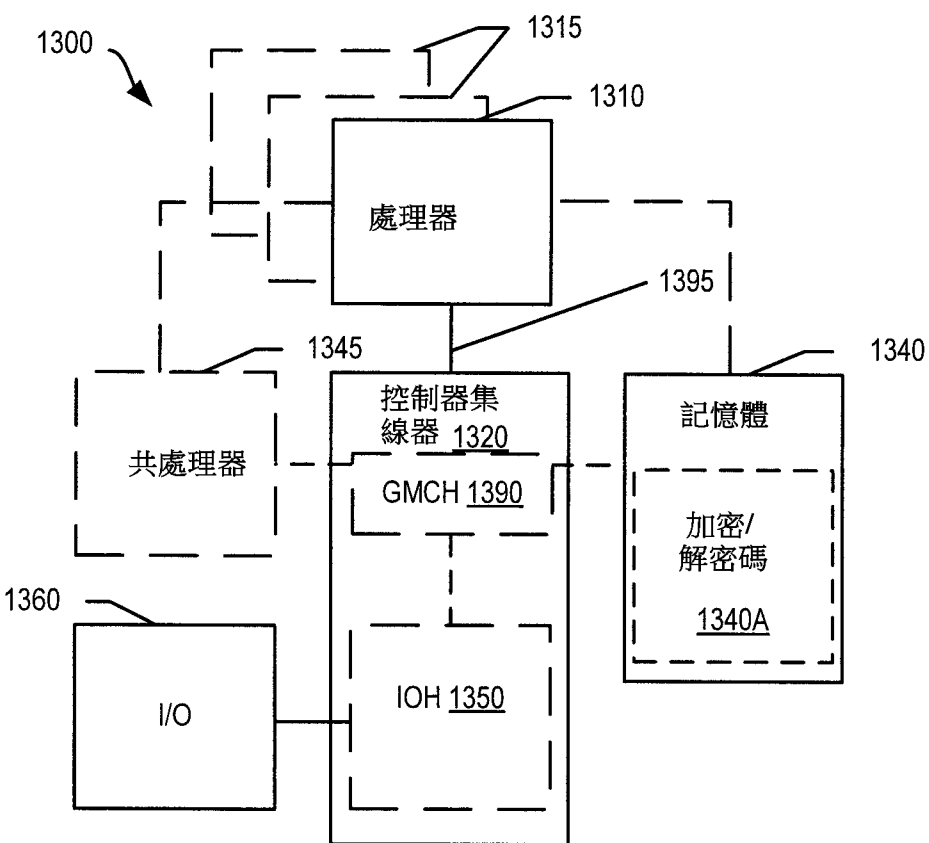
【圖 11A】



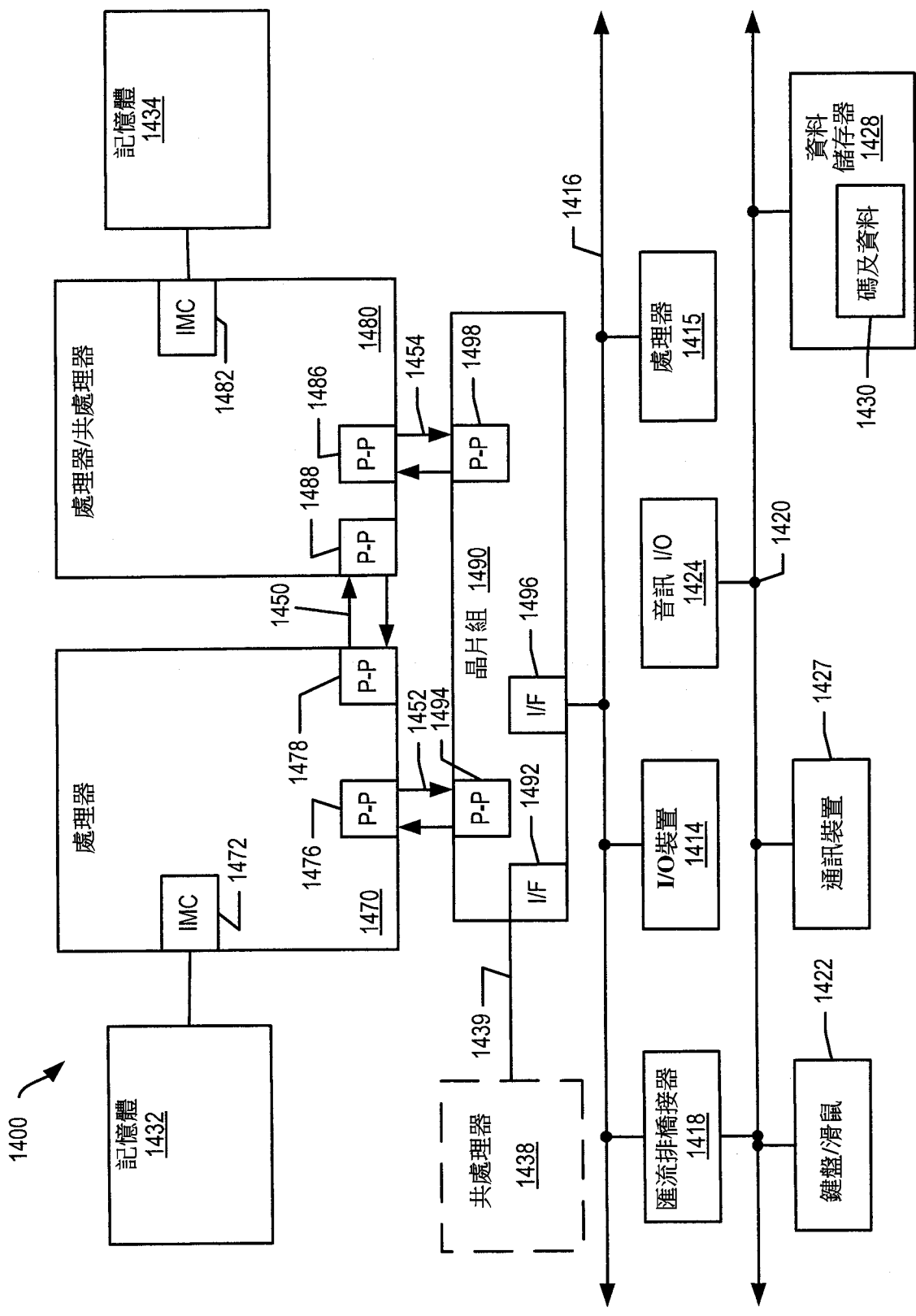
【圖 11B】



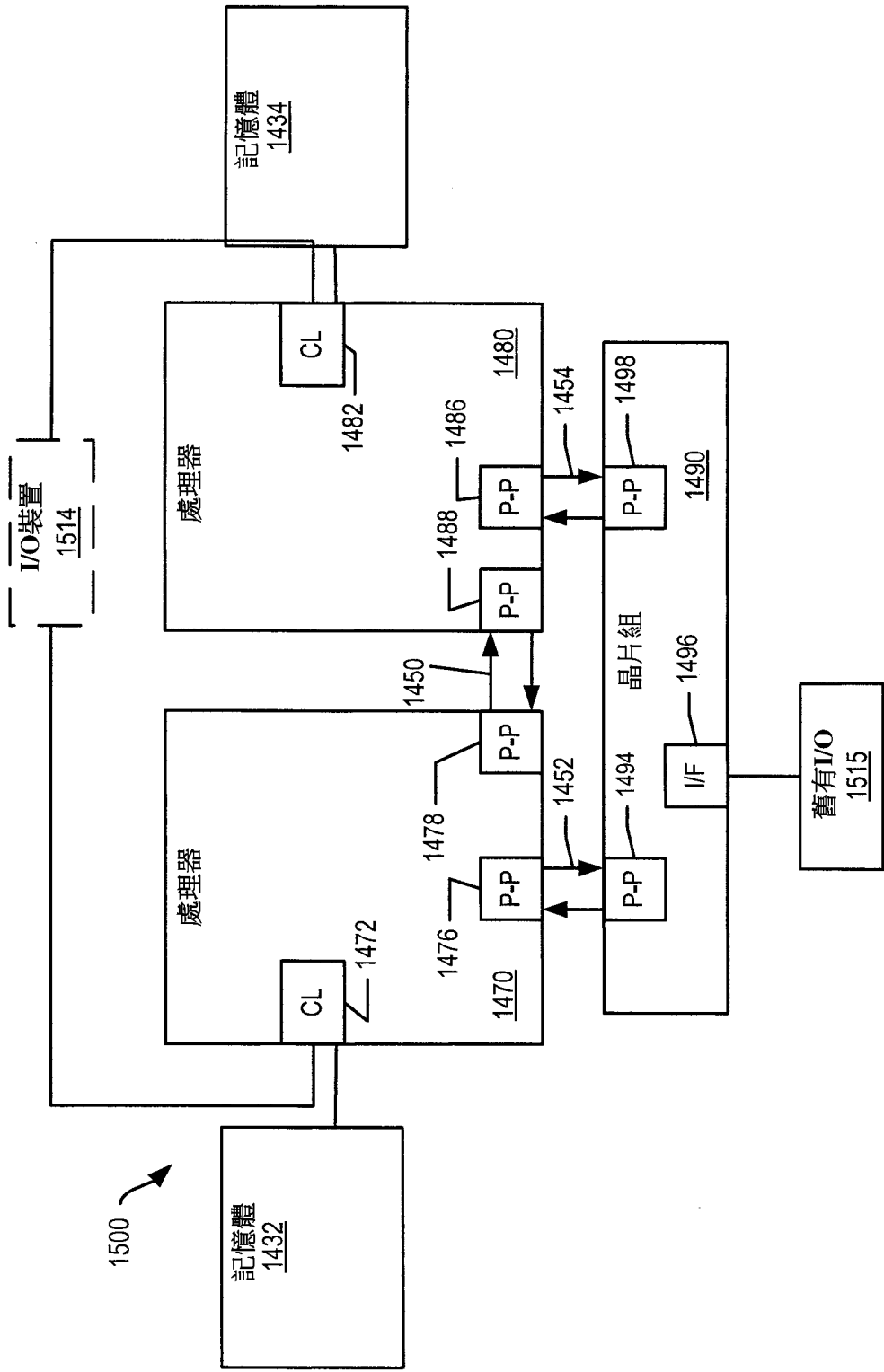
【圖 12】



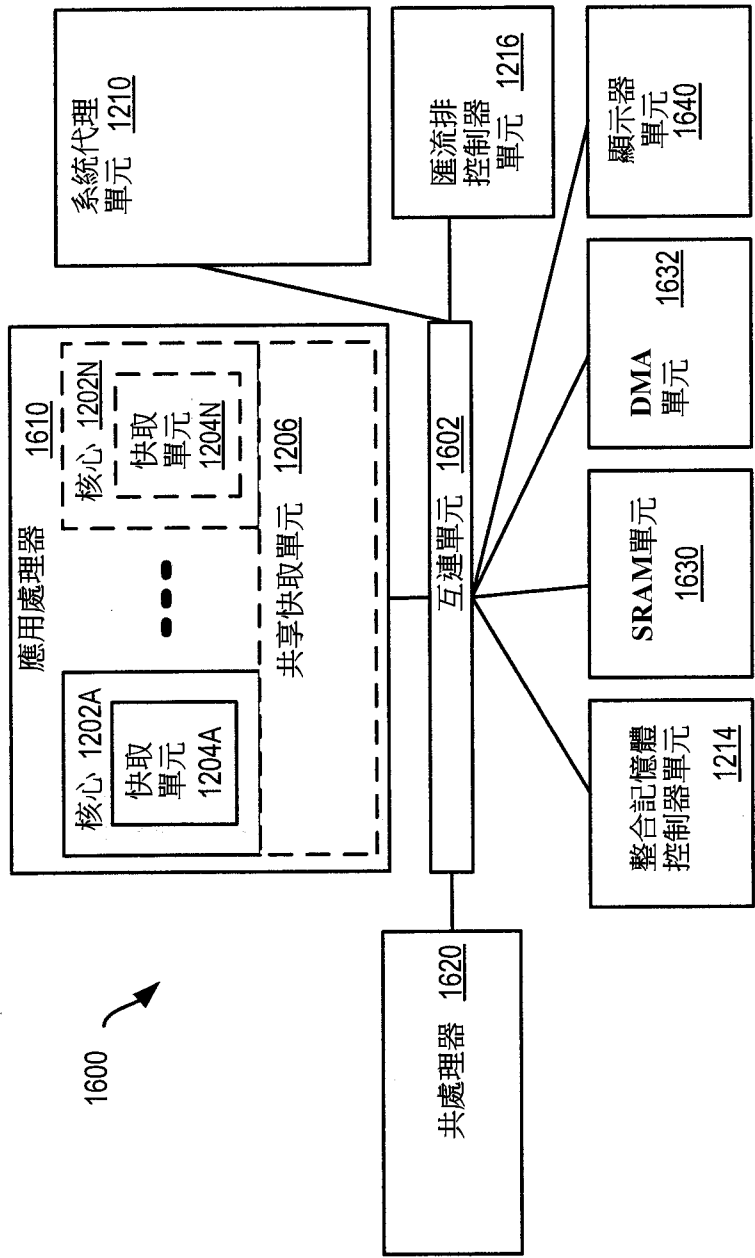
【圖 13】



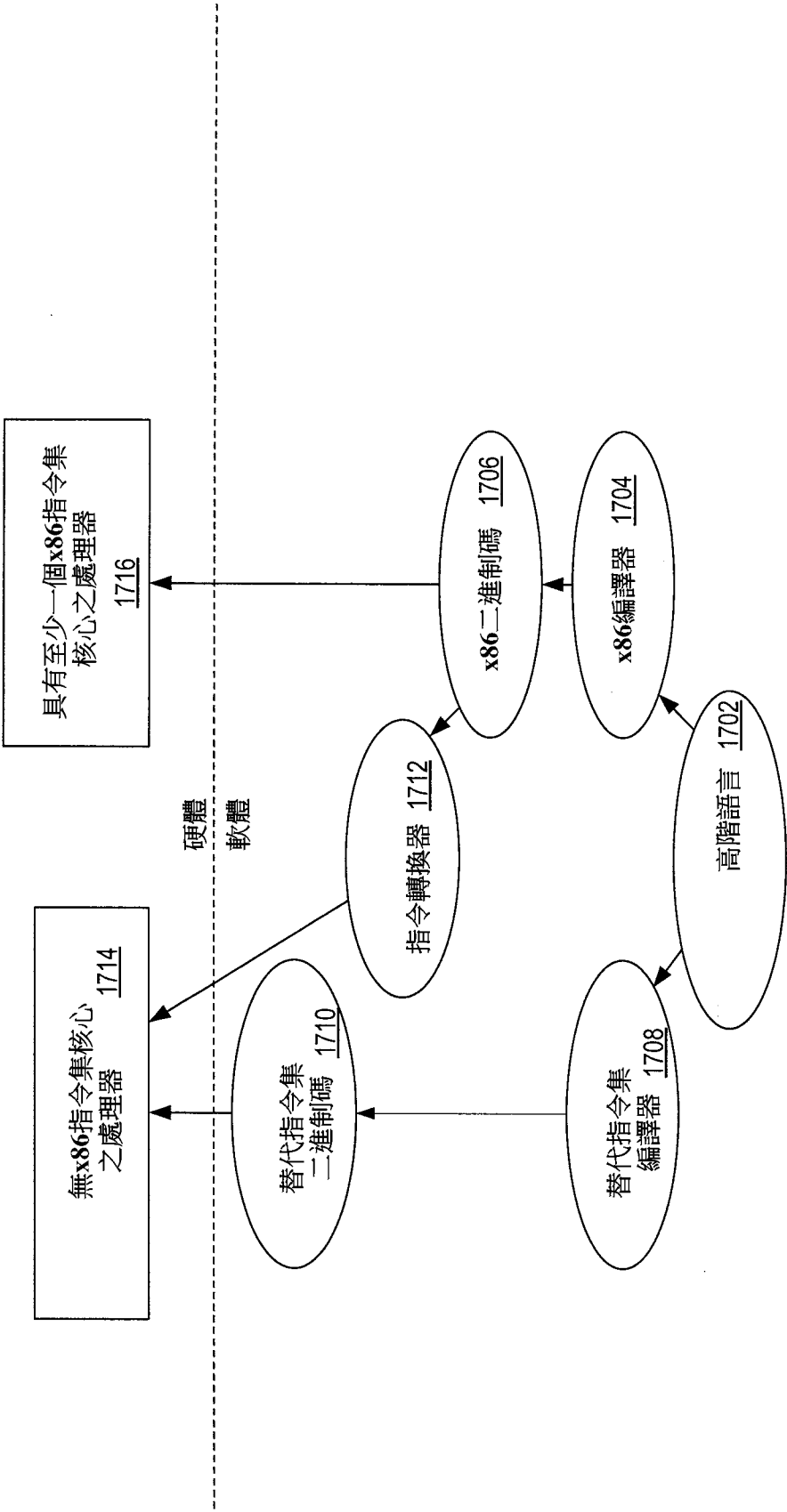
【圖 14】



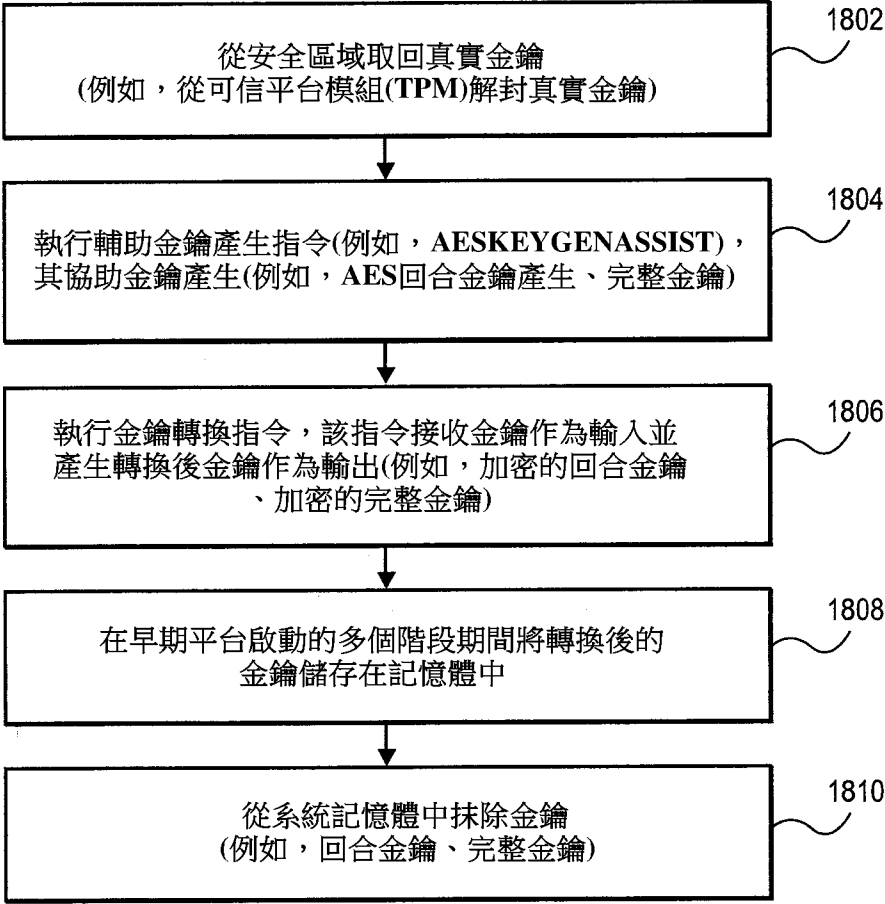
【圖 15】



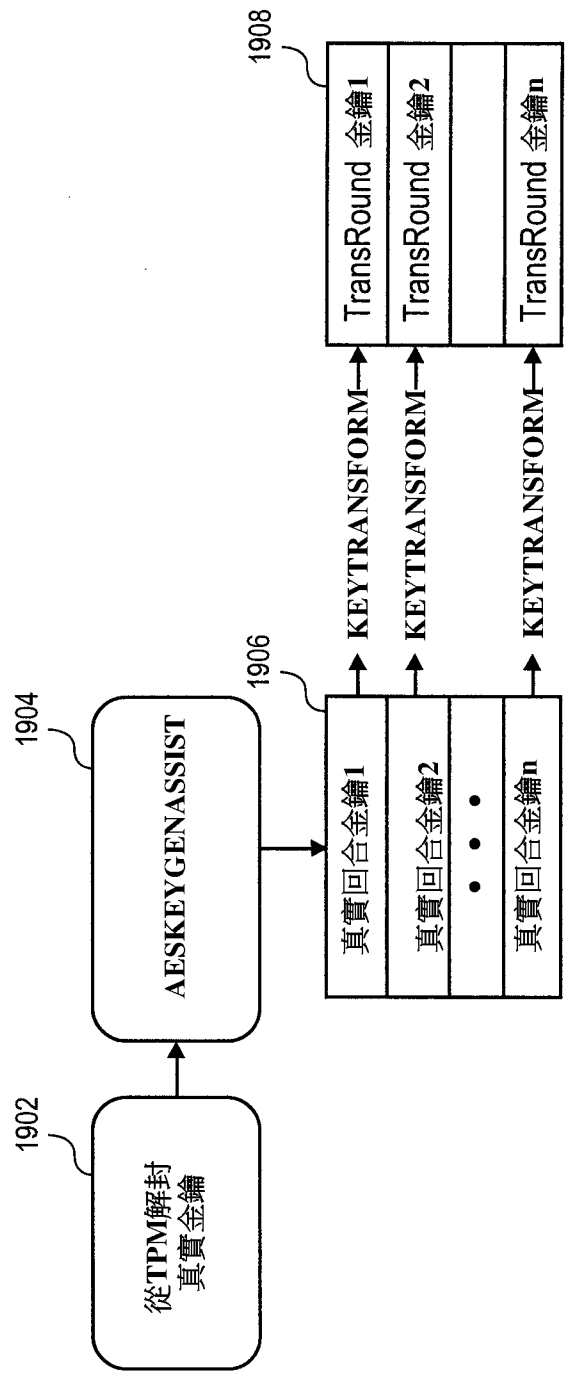
【圖 16】



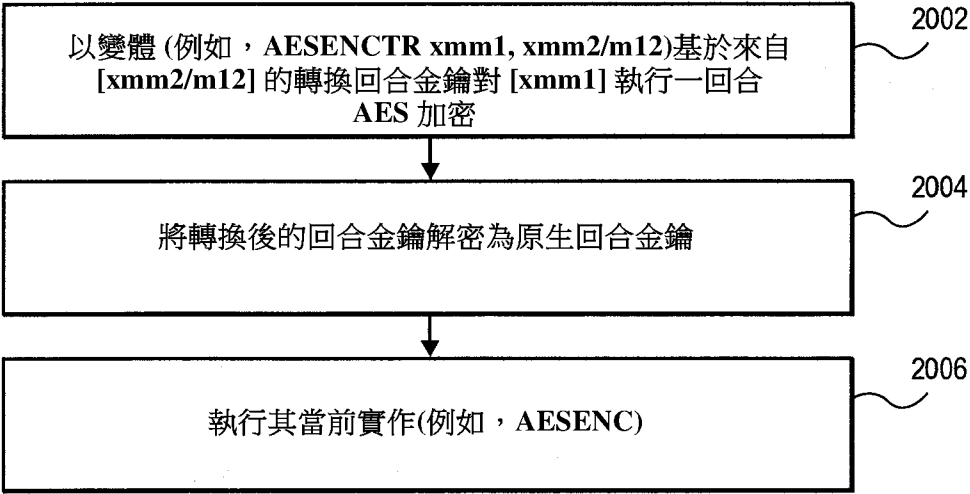
【圖 17】



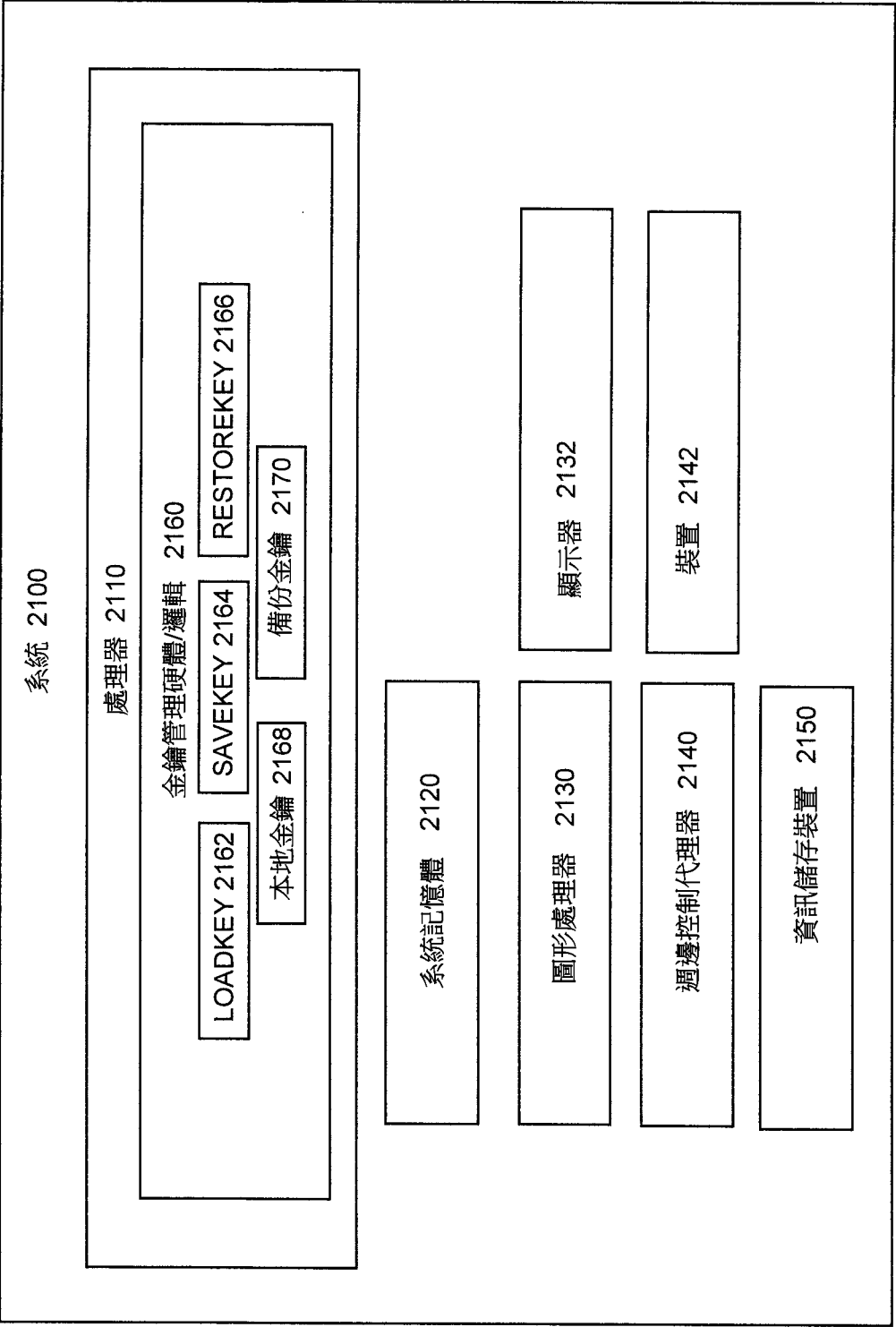
【圖 18】



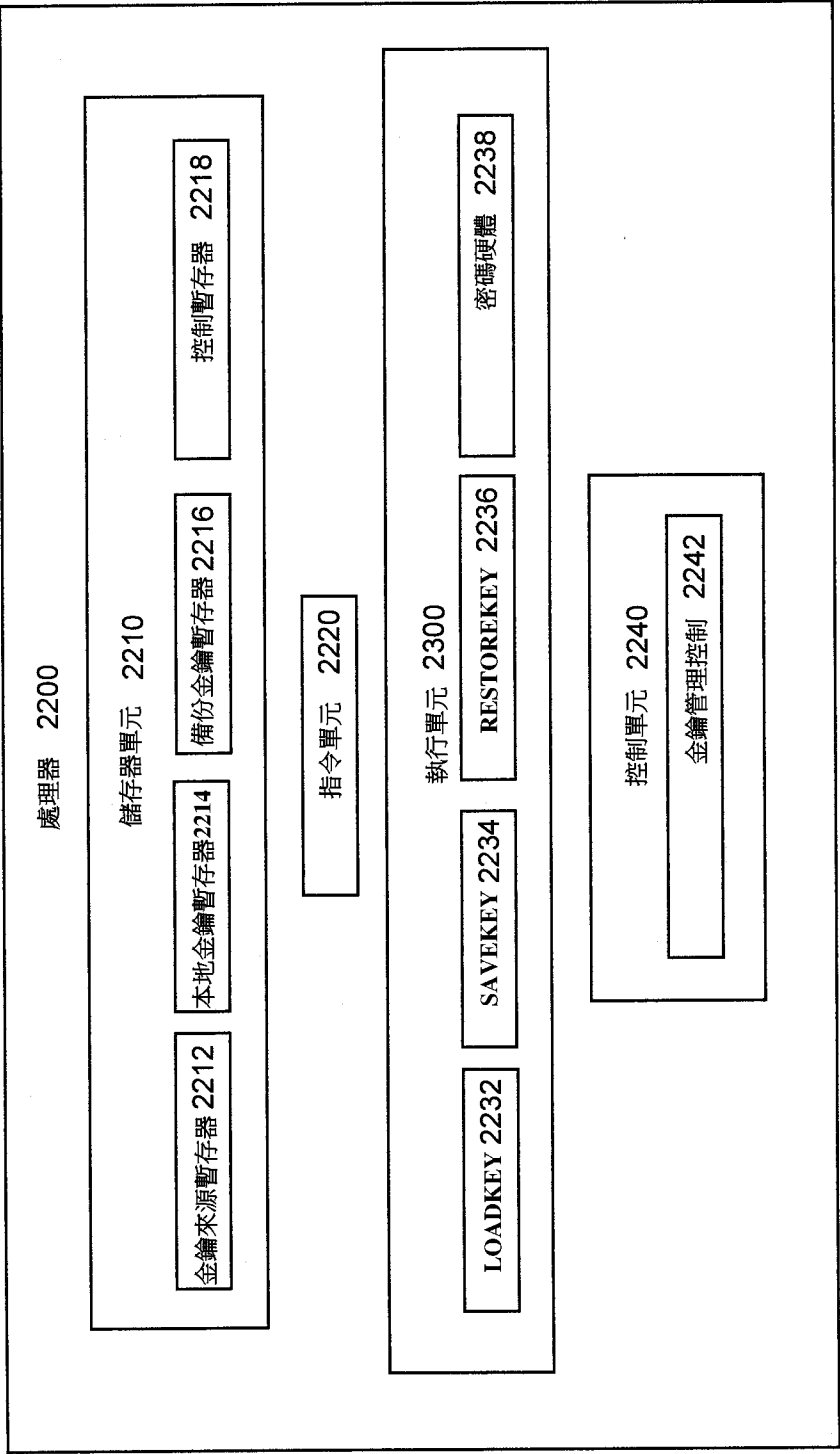
【圖 19】



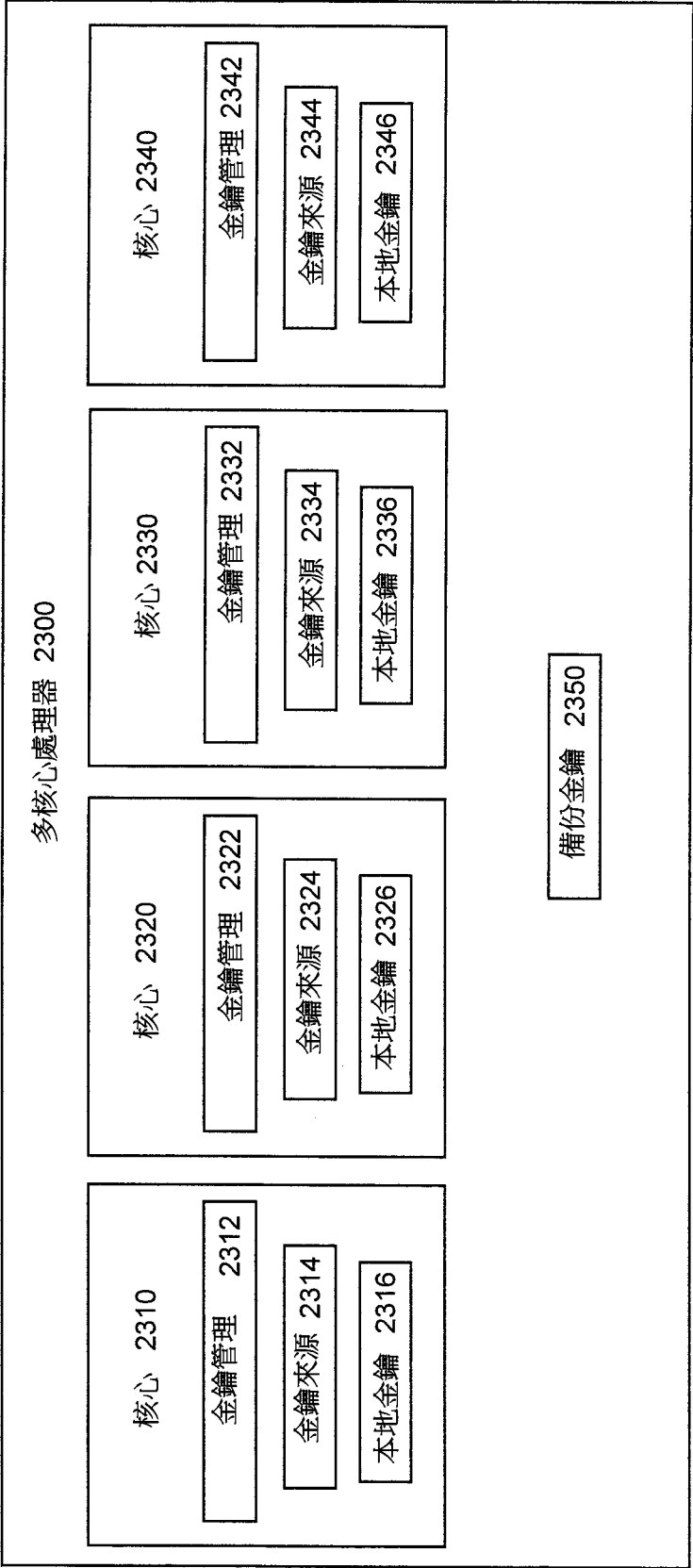
【圖 20】



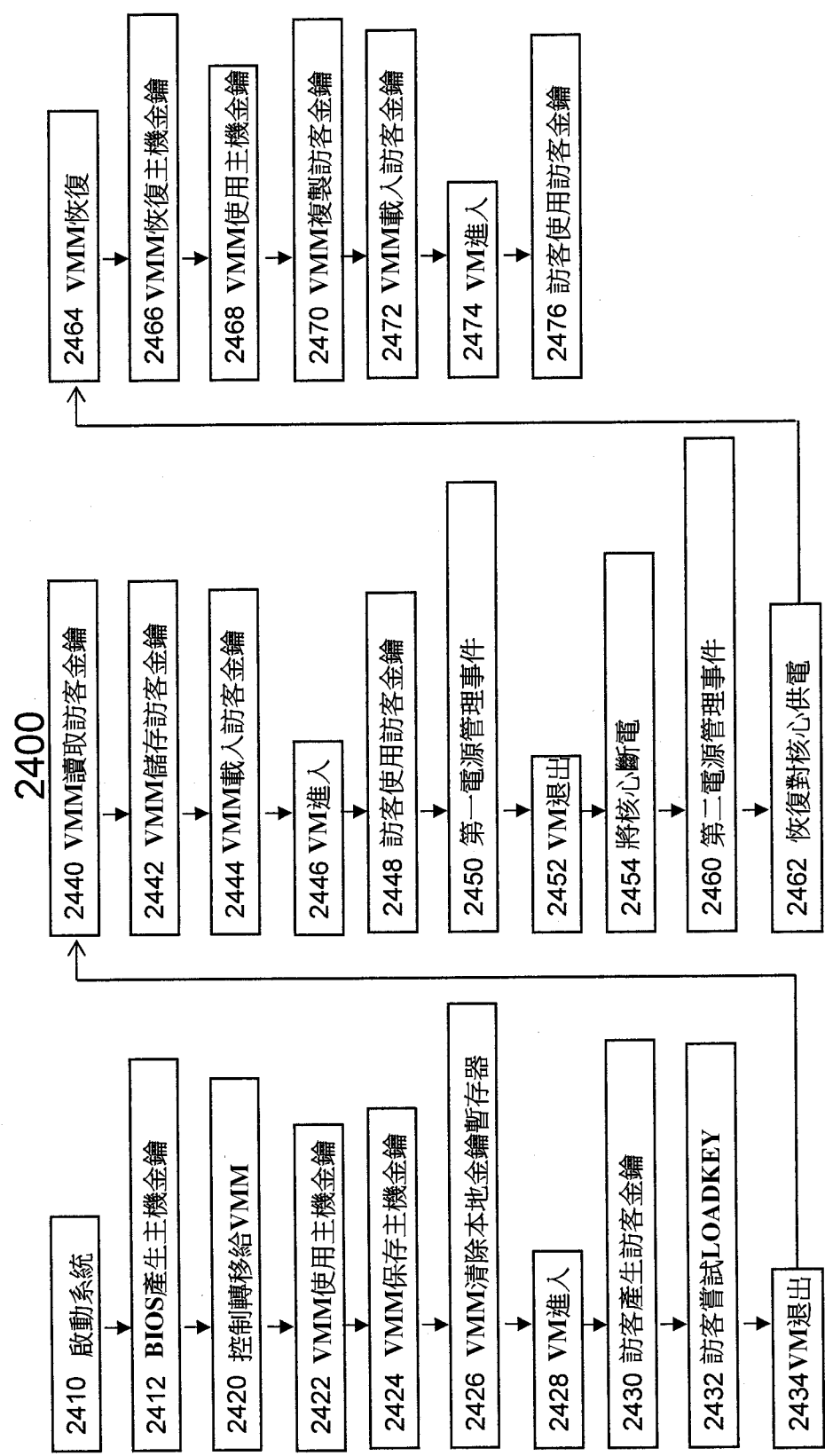
【圖 21】



【圖 22】



【圖 23】



【圖 24】