(19) 대한민국특허청(KR)
(12) 등록특허공보(B1)

(45) 공고일자 2014년03월07일
(11) 등록번호 10-1371784
(24) 등록일자 2014년02월28일

(51) 국제특허분류(Int. Cl.)
G06F 21/44 (2013.01) *G06F 9/06* (2006.01)
(21) 출원번호 10-2012-7018367
(22) 출원일자(국제) 2010년10월27일
심사청구일자 2012년07월13일
(85) 번역문제출일자 2012년07월13일
(65) 공개번호 10-2012-0093439
(43) 공개일자 2012년08월22일
(86) 국제출원번호 PCT/US2010/054312
(87) 국제공개번호 WO 2011/084210
국제공개일자 2011년07월14일
(30) 우선권주장
12/639,616 2009년12월16일 미국(US)
(56) 선행기술조사문헌
US20090290712 A1
US20040003273 A1
US20060015749 A1
US20050066191 A1
전체 청구항 수 : 총 27 항

(73) 특허권자
인텔 코오퍼레이션
미합중국 캘리포니아 95052 산타클라라 미션 칼리지 블러바드 2200
(72) 발명자
스미쓰, 네드
미국 97006 오레곤주 비버튼 사우쓰웨스트 델타 드라이브 375
샨보그, 베드브야스
미국 97229 오레곤주 포트랜드 노쓰웨스트 헤닝거 레인 13686
(뒷면에 계속)
(74) 대리인
백만기, 양영준

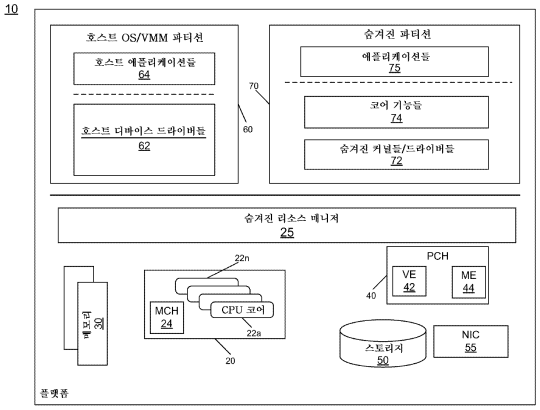
심사관 : 구대성

(54) 발명의 명칭 **숨겨진 실행 환경에서 무결성 검증 및 인증 제공**

(57) 요 약

일 실시예에서, 프로세서는 시스템 소프트웨어에 보이지 않는 숨겨진 환경에서 실행하기 위해 HRM(hidden resource manager)을 생성 및 실행하기 위한 프로세서 명령들을 포함하는 마이크로코드 스토리지를 포함한다. 프로세서는 숨겨진 환경의 적어도 하나의 커널 코드 모듈의 측정 및 적어도 하나의 커널 코드 모듈의 검증의 상태를 포함하는 보안 정보를 저장하기 위한 확장 레지스터를 더 포함할 수 있다. 다른 실시예들이 기술 및 청구된다.

대 표 도



(72) 발명자

쿠마, 아빈드

미국 97006 오레곤주 비버튼 노쓰웨스트 퍼시픽 그로브 드라이브 291

고엘, 푸루쇼탐

미국 97006 오레곤주 비버튼 노쓰웨스트 아리조나 드라이브 16984

특허청구의 범위

청구항 1

숨겨진 실행 환경에서 무결성 검증 및 인증을 제공하기 위한 장치로서,

명령들을 실행하기 위한 프로세서를 포함하며,

상기 프로세서는 상기 프로세서의 ISA(instruction set architecture)의 명령들을 실행하기 위한 제1 프로세서 명령들을 포함하는 제1 마이크로코드 스토리지 및 적어도 하나의 코어, 시스템 소프트웨어에 보이지 않는 숨겨진 환경에서 실행하기 위해 HRM(hidden resource manager)를 생성 및 실행하기 위한 제2 프로세서 명령들을 포함하는 제2 마이크로코드 스토리지, 및 상기 숨겨진 환경의 적어도 하나의 커널 코드 모듈의 측정 및 상기 적어도 하나의 커널 코드 모듈의 검증의 상태를 포함하는 보안 정보를 저장하기 위한 확장 레지스터를 포함하는 장치.

청구항 2

제1항에 있어서,

상기 확장 레지스터는 상기 숨겨진 환경에서 실행될 적어도 하나의 애플리케이션에 대한 론치(launch) 제어 정책의 해시를 더 저장하는 장치.

청구항 3

제1항에 있어서,

상기 프로세서는 상기 프로세서 내에서만 상기 숨겨진 환경에서 인증을 실행하는 장치.

청구항 4

제3항에 있어서,

상기 인증은 상기 프로세서에 연결된 임의의 에이전트와의 통신 없이 실행되는 장치.

청구항 5

제1항에 있어서,

상기 프로세서는 CMAC(cipher message authentication code) 계산을 실행하고 ICV(integrity check value) 어레이의 엔트리에 상기 CMAC 계산을 저장하기 위한 가속 장치(accelerator)를 더 포함하는 장치.

청구항 6

제5항에 있어서,

상기 가속 장치는 상기 프로세서에 연결된 메모리의 숨겨진 파티션의 복수의 페이지들 각각에 대해 상기 CMAC 계산을 실행하는 장치.

청구항 7

제1항에 있어서,

상기 프로세서는 제조자를 식별하기 위해 상기 프로세서의 상기 제조자에 의해 저장된 제1 식별자 및 상기 프로세서를 포함하는 시스템의 소유자에 의해 저장된 제2 식별자를 포함하고, 상기 제2 식별자는 제어 가능한 양의 고유성(uniqueness)을 가진 장치.

청구항 8

제1항에 있어서,

상기 프로세서는 상기 프로세서의 퓨즈 세팅에 기초하여 개인 키를 생성하는 장치.

청구항 9

숨겨진 실행 환경에서 무결성 검증 및 인증을 제공하기 위한 방법으로서,

제1 개인 키를 사용해서 시스템의 시스템 메모리의 숨겨진 파티션의 복수의 페이지의 CMAC(cipher message authentication code)를 생성하고 ICV(integrity check value) 어레이의 대응 엔트리에 상기 CMAC들 각각을 저장하는 단계;

제2 개인 키를 사용해서 암호화된 블로브에 상기 제1 개인 키 및 카운터의 값을 암호화하는 단계;

상기 시스템을 제1 전력 상태로부터, 상기 시스템 메모리가 상기 시스템의 프로세서에 의해 보호되지 않는 저 전력 상태로 전이하는 단계; 및

상기 시스템이 상기 저 전력 상태로부터 상기 제1 전력 상태로 전이될 때 상기 암호화된 블로브 및 상기 ICV 어레이를 사용해서 상기 숨겨진 파티션을 타당성 검사(validate)하는 단계

를 포함하는 방법.

청구항 10

제9항에 있어서,

상기 제2 개인 키는 숨겨진 환경에서 실행중인 프로세서의 마이크로코드에 의해 스토리지 키로부터 유도되는 방법.

청구항 11

제9항에 있어서,

상기 제1 개인 키를 사용해서 상기 ICV 어레이의 CMAC를 생성하고 상기 ICV 어레이의 CMAC를 제2 스토리지에 저장하는 단계를 더 포함하는 방법.

청구항 12

제9항에 있어서,

상기 시스템을 상기 저 전력 상태로 전이하기 전에 영구 메모리에 상기 카운터 값을 저장하는 단계, 및 상기 저 전력 상태로부터 복귀한 후에 상기 저장된 카운터 값을 증가시키는 단계를 더 포함하고, 상기 프로세서가 상기 저 전력 상태일 때 상기 시스템 메모리는 셀프-리프레시 모드(self-refresh mode)인 방법.

청구항 13

제9항에 있어서,

상기 시스템을 상기 저 전력 상태로 전이하기 전에 상기 카운터 값을 증가시키고 상기 증가된 카운터 값을 비휘발성 스토리지에 저장하는 단계를 더 포함하는 방법.

청구항 14

제13항에 있어서,

상기 숨겨진 파티션을 타당성 검사하는 단계는

상기 제2 개인 키를 사용해서 상기 암호화된 블로브를 해독해서 상기 제1 개인 키 및 상기 카운터 값을 획득하는 단계 - 상기 제2 개인 키는 프로세서의 마이크로코드에 의해 생성됨 - ;

상기 카운터 값을 상기 비휘발성 스토리지에 저장된 상기 증가된 카운터 값과 비교하는 단계; 및

상기 ICV 어레이의 CMAC 및 상기 페이지들의 CMAC를 검증하는 단계

를 포함하는 방법.

청구항 15

제14항에 있어서,

상기 비교가 미리결정된 임계값보다 더 큰 차이를 야기하면, 상기 타당성 검사의 실패를 나타내는 단계를 더 포함하는 방법.

청구항 16

제14항에 있어서,

상기 ICV 어레이의 계산된 CMAC가 상기 ICV 어레이의 저장된 CMAC 값과 매치하지 않거나 또는 상기 페이지들 중 적어도 한 페이지에 대해 계산된 CMAC가 상기 ICV 어레이에 저장된 대응 CMAC와 매치하지 않으면, 상기 타당성 검사의 실패를 나타내는 단계를 더 포함하는 방법.

청구항 17

숨겨진 실행 환경에서 무결성 검증 및 인증을 제공하기 위한 명령들을 포함하는 머신 액세스 가능 저장 매체를 포함하는 물품으로서,

상기 명령들은 실행될 때 시스템으로 하여금,

프로세서의 마이크로코드로 구현된 HRM(hidden resource manager)을 사용해서 실행되는 상기 시스템의 숨겨진 환경을 인증하기 위해 검증 장치(verifier)로부터 인증 요청 및 논스(nonce)를 수신하고 - 상기 숨겨진 환경은 시스템 소프트웨어에 보이지 않음 - ;

인터커넥트를 통해 상기 프로세서에 연결된 에이전트와의 통신 없이 상기 마이크로코드를 통해 상기 프로세서에서 직접 상기 인증 요청에 응답하여 서명된 인증 레코드를 생성하며;

상기 검증 장치에 상기 서명된 인증 레코드를 제공하게 하는 물품.

청구항 18

제17항에 있어서,

상기 숨겨진 환경의 커널에서 상기 인증 요청을 수신하고 상기 인증 요청과 연관된 애플리케이션의 론치 히스토리에 액세스하며, 상기 론치 히스토리를 해싱하고 상기 논스 및 상기 해싱된 론치 히스토리를 상기 HRM에 제공하기 위한 명령들을 더 포함하는 물품.

청구항 19

제18항에 있어서,

상기 HRM을 사용해서, 상기 시스템의 소유자 식별자 - 상기 소유자 식별자는 상기 시스템의 소유자에 의해 생성됨 -, 론치 제어 정책의 측정, 상기 숨겨진 환경의 적어도 하나의 커널의 측정을 포함하는 상기 인증 레코드를 생성하고, 개인 키로 상기 인증 레코드에 서명하기 위한 명령들을 더 포함하는 물품.

청구항 20

제19항에 있어서,

상기 서명된 인증 레코드를 상기 검증 장치에 송신하고 상기 개인 키에 대응하는 공개 키를 사용해서 상기 서명된 인증 레코드를 검증하기 위한 명령들을 더 포함하는 물품.

청구항 21

제20항에 있어서,

상기 서명된 인증 레코드에 저장된 상기 논스를 상기 검증 장치에 의해 제공된 논스와 비교해서 상기 서명된 인증 레코드를 검증하고, 상기 애플리케이션이 애플리케이션들의 화이트 리스트에 포함되어 있는 지를 결정하기 위한 명령들을 더 포함하는 물품.

청구항 22

제17항에 있어서,

상기 마이크로코드의 실행을 통해 상기 숨겨진 환경을 론치하고;

상기 마이크로코드를 사용해서 제1 커널에 대한 제1 매니페스트(manifest)에 기초하여 상기 제1 커널을 검증하고, 상기 제1 커널이 검증되면, 상기 프로세서의 확장 레지스터를 상기 제1 커널의 측정으로 갱신하며;

상기 마이크로코드를 사용해서 제2 커널에 대한 제2 매니페스트에 기초하여 상기 제2 커널을 검증하고, 상기 제2 커널이 검증되면, 상기 확장 레지스터를 상기 제2 커널의 측정으로 갱신하기 위한 명령들을 더 포함하는 물품.

청구항 23

숨겨진 실행 환경에서 무결성 검증 및 인증을 제공하기 위한 시스템으로서,

명령들을 실행하기 위한 프로세서 - 상기 프로세서는 시스템 소프트웨어에 보이지 않는 숨겨진 환경에서 실행하기 위해 HRM(hidden resource manager)을 생성 및 실행하기 위한 프로세서 명령들을 포함하는 마이크로코드 스토리지 및 적어도 하나의 코어, 및 상기 숨겨진 환경의 적어도 하나의 커널 코드 모듈의 측정 및 상기 적어도 하나의 커널 코드 모듈의 검증의 상태를 포함하는 보안 정보를 저장하기 위한 확장 레지스터를 포함하고, 상기 HRM은 제1 개인 키를 사용해서 숨겨진 메모리 파티션의 복수의 페이지들 각각에 대해 CMAC(cipher message authentication code)를 생성하고 ICV(integrity check value) 어레이의 대응 엔트리에 상기 CMAC들 각각을 저장하며, 제2 개인 키를 사용해서 암호화된 블로브에 상기 제1 개인 키 및 카운터의 값을 암호화하고, 시스템이 저 전력 상태로부터 전이될 때, 상기 암호화된 블로브 및 상기 ICV 어레이의 상기 엔트리들 각각을 사용해서 상기 숨겨진 환경에서 사용될 상기 HRM에 의해 할당된 상기 숨겨진 메모리 파티션의 페이지들을 타당성 검사함 - ; 및

상기 프로세서에 연결되고 상기 숨겨진 메모리 파티션을 포함하는 시스템 메모리 - 상기 시스템 메모리는 상기 프로세서가 상기 저 전력 상태일 때 셀프-리프레시 상태에 있음 -

를 포함하는 시스템.

청구항 24

제23항에 있어서,

상기 프로세서는 상기 프로세서에 연결된 임의의 에이전트와의 통신 없이 상기 프로세서 내에서만 상기 숨겨진 환경에서 인증을 실행하는 시스템.

청구항 25

제23항에 있어서,

상기 프로세서는 CMAC 계산들을 실행하고 상기 ICV 어레이의 엔트리에 상기 CMAC들 각각을 저장하기 위한 가속 장치를 더 포함하는 시스템.

청구항 26

제23항에 있어서,

상기 프로세서는 제조자를 식별하기 위해 상기 프로세서의 상기 제조자에 의해 저장된 제1 식별자 및 상기 프로세서를 포함하는 시스템의 소유자에 의해 저장된 제2 식별자를 포함하고, 상기 제2 식별자는 제어 가능한 양의 고유성을 가진 시스템.

청구항 27

제23항에 있어서,

상기 HRM은 게스트 운영 체제에 액세스 가능한 상기 시스템 메모리의 메모리 파티션의 페이지들의 타당성 검사를 가능케 하는 시스템.

명 세 서

배 경 기 술

[0001] 컴퓨터 시스템들은 하나의 또는 그 이상의 프로세서들, 메모리, 입력/출력 장치들 등을 포함하는 하드웨어 소자들의 집합으로 형성된다. 또한, 운영 체제(OS), 가상 머신 모니터(VMM) 등의 시스템 소프트웨어를 포함해서, 상이한 타입들의 소프트웨어가 시스템 내에 존재할 수 있다. 또한, 응용 소프트웨어는 워드 프로세싱, 이메일, 게임들 등의 사용자가 바라는 특정 태스크들을 실행하기 위해 제공될 수 있다. 소프트웨어 및 기본 프로세서 하드웨어 간의 통신을 가능케 하기 위해, 소프트웨어의 명령들은 프로세서에 저장된 마이크로코드를 사용해서 구현될 수 있는 마이크로-명령들(micro-instructions)(uops) 등의 더 작은 명령들로 변환될 수 있다.

[0002] 일반적으로, 폭넓은 메모리는 OS 또는 VMM 등의 시스템 소프트웨어에 가시적이다. 그러나, 안전한 동작들 등을 제공하기 위해, 숨겨진 파티션(hidden partition)이 리소스 매니저를 사용해서 실행할 수 있는 코드를 포함하는 메모리 내에 제공될 수 있으며, 이는 프로세서 마이크로코드로 구현될 수 있으며 OS/VMM으로부터 숨겨질 수 있다. 그러나, 각종 보안 및 인증(attestation) 문제들이 숨겨진 리소스들을 제공할 때 발생하며, 복잡성을 증가시킨다.

[0003] 일례로서, 시스템이 특정 저 전력 상태들, 예를 들어, 고급 구성 및 파워 인터페이스(ACPI)(Advanced Configuration and Power Interface(ACPI)), Rev. 2.0 표준(2003년 4월 25일)에 따른 소위 중단(suspended)(S3) 상태에 있을 때, 시스템 메모리는 셀프-리프레시 상태(self-refresh state)이다. 그러나, 프로세서는 전원이 나간 상태이고, 따라서, 메모리에서 유효한 보호가 없다. 따라서, 당사자는, 시스템이 S3 상태일 때 메모리의 콘텐츠를 변경할 수 있으며, 메모리가 재개(S0 상태)시 검증되지 않으면, 이는 시스템 메모리의 숨겨진 메모리 파티션에서 코드 인젝션 공격들(code injection attacks)을 야기할 수 있다. S3으로 들어가기 전에 메모리를 해싱하고 S3으로부터 S0으로 재개할 때 콘텐츠를 검증해서 이러한 코드 인젝션 공격들을 방지할 수 있지만, 이는, 메모리 콘텐츠를 해싱하고 차후에 검증하기 위해 실행될 추가 작업으로 인해 S0 및 S3 간의 전이에 필요한 시간의 양을 연장하는 것을 포함해서, 단점들을 가진다.

도면의 간단한 설명

[0004] 도 1은 본 발명의 일 실시예에 따른 한 플랫폼의 블록도이다.

도 2는 본 발명의 일 실시예에 따른 무결성 검사 값(ICV) 어레이의 블록도이다.

도 3은 본 발명의 일 실시예에 따른 숨겨진 메모리 무결성 계산 및 보호의 개요의 블록도이다.

도 4는 본 발명의 일 실시예에 따라 저 전력 상태로 전이하기 위한 방법의 흐름도이다.

도 5는 저 전력 상태로부터 정상 전력 모드에서의 실행을 재개하기 위한 방법의 흐름도이다.

도 6은 본 발명의 일 실시예에 따른 한 검증 방법의 흐름도이다.

도 7은 본 발명의 일 실시예에 따른 한 하이 레벨 론치 방법의 흐름도이다.

도 8은 본 발명의 일 실시예에 따른 론치 오퍼레이션들의 흐름도이다.

도 9는 본 발명의 일 실시예에 따른 한 인증 방법의 흐름도이다.

도 10은 본 발명의 일 실시예에 따른 한 프로세서의 블록도이다.

발명을 실시하기 위한 구체적인 내용

[0005] 실시예들은, 저 전력 상태로부터 빠져나와 정상 동작 상태로 될 때 사용될 수 있는, 숨겨진 메모리 파티션의 무결성의 효율적인 검증을 제공할 수 있다. 또한, 실시예들은, 숨겨진 환경에서 실행될 각종 소프트웨어가 실행 전에 검증될 수 있도록 그 환경을 인증하는데 사용될 수 있다. 이러한 인증은 인터커넥트를 통해 다른 컴포넌트들에 액세스할 필요 없이 프로세서 내에서 완전히 구현될 수 있어서, 인터커넥트 및 다른 컴포넌트들과의 상호 작용을 통해 속도를 증가시키고 복잡성 및 보안 위협 잠재력을 감소시킨다.

[0006] 호스트 운영 체제(OS)/가상 머신 모니터(VMM) 및 숨겨진 파티션 간의 메모리 분리를 제공하기 위해, 프로세서는 숨겨진 메모리에 대한 호스트 OS/VMM 생성 액세스들을 효과적으로 차단하는 하나의 또는 그 이상의 하드웨어 레인지 레지스터들을 사용할 수 있다. 유사하게, 숨겨진 파티션으로부터 호스트 OS/VMM 메모리에 대한 액세스들을 제어하기 위해, 페이지 테이블이 숨겨진 파티션 아래에 존재할 수 있다. 숨겨진 환경의 오퍼레이션을 덜 잘

이해하기 위해, 이러한 환경을 제공하는 플랫폼의 개요를 제공하는 것이 유익하다.

[0007] 이제 도 1을 참조하면, 본 발명의 일 실시예에 따른 한 플랫폼의 블록도가 도시되어 있다. 도 1에 도시된 바와 같이, 플랫폼(10)은 서버 컴퓨터, 데스크탑 컴퓨터, 랩탑 컴퓨터, 노트북 컴퓨터 등의 임의의 타입의 컴퓨터 시스템일 수 있다. 플랫폼은 요청된 오퍼레이션들을 실행하도록 함께 동작하는 각종 하드웨어 및 소프트웨어를 포함한다. 도시된 바와 같이, 플랫폼 하드웨어는, 일 실시예에서, 복수의 프로세서 코어들(22_a-22_n)을 포함하는 멀티코어 프로세서일 수 있는 프로세서(20)를 포함한다. 더 후술되는 바와 같이, 각각의 코어(22)는 숨겨진 환경에서 실행을 위해 마이크로코드를 포함할 수 있으며, 이 환경은 OS 또는 VMM 등의 다른 시스템 소프트웨어로부터 숨겨진다. 더 도시된 바와 같이, 프로세서(20)는, 일 실시예에서 동적 랜덤 액세스 메모리(DRAM)일 수 있는 시스템 메모리(30)와 통신하기 위해 메모리 컨트롤러 허브(MCH)(24)를 포함한다.

[0008] 플랫폼(10)의 추가 하드웨어는, 스토리지(50), 예를 들어, 디스크 드라이브 등의 대용량 스토리지, 광 또는 다른 비휘발성 스토리지 등의 각종 주변 장치들에 대한 제어 기능들을 제공할 수 있는 가상화 엔진(42)(VE) 및 관리 엔진(ME)(44)을 포함할 수 있는 주변 컨트롤러 허브(PCH)(40)를 포함할 수 있다. 또한, 네트워크 인터페이스 컨트롤러(NIC)(55)는 유선 네트워크, 예를 들어, 근거리 통신망(LAN), 무선 LAN(WLAN) 등의 무선 네트워크, 또는 셀룰러 네트워크와 같은 광역 무선 네트워크 등의 네트워크의 다른 에이전트들 및 플랫폼(10) 간의 통신을 가능케 할 수 있다. 도 1의 실시예에서 이러한 특정 하드웨어로 도시되었지만, 본 발명의 범위는 이와 관련하여 한정되지 않음을 이해하라.

[0009] 도 1에 더 도시된 바와 같이, 각종 소프트웨어가 또한 존재한다. 먼저, 숨겨진 리소스 매니저(HRM)(25)가 도시되어 있다. 도시된 바와 같이, 이 소프트웨어 층은 기본 프로세서 하드웨어 및 숨겨진 파티션(70) 간의 인터페이스로서 동작할 수 있다. 일 실시예에서, HRM(25)은 프로세서의 마이크로코드 스토리지에 저장된 프로세서 마이크로코드로서 구현될 수 있다. 이 마이크로코드 및 스토리지는 ISA(instruction set architecture)의 사용자 레벨 명령들에 대응하기 위한 프로세서 명령들을 제공하는데 사용되는 종래의 마이크로코드 및 마이크로코드 스토리지와 별개의 것일 수 있음을 주지하라. 그러나, 일부 실시예들에서, 숨겨진 마이크로코드 및 종래의 마이크로코드는 싱글 마이크로코드 스토리지의 상이한 파티션들에 저장될 수 있다.

[0010] 리소스 매니저는 숨겨진 파티션(70)의 코드에 서비스들을 제공하도록 실행될 수 있다. 도시된 바와 같이, 각종 코드는 이 숨겨진 파티션에 존재할 수 있으며, 다른 시스템 소프트웨어, 즉, OS 및 VMM으로부터 분할 및 숨겨진, 예를 들어, 시스템 메모리(30)의 한 파티션에 저장될 수 있다. 숨겨진 파티션(70)은, 플랫폼의 각종 장치들과 인터페이스하기 위한 드라이버들 뿐만 아니라 커널 서비스들을 제공할 수 있는, 하나의 또는 그 이상의 숨겨진 커널들 및 드라이버들(72)을 포함하는, 각종 코드를 포함한다. 숨겨진 실행 환경에서 실행될 수 있는 하나의 또는 그 이상의 애플리케이션들(75) 외에, 추가의 코어 기능 코드(74)가 존재할 수 있다. 본 발명의 범위가 이와 관련하여 한정되지 않더라도, 숨겨진 실행 환경은 범용 OS 보다 더 높은 무결성을 제공하는 공식적인 보안 모델을 따르도록 구현된 바와 같은 오퍼레이션들을 위한 것일 수 있다. 또한, 플랫폼(10)은, 각종 호스트 디바이스 드라이버들(62) 및 호스트 애플리케이션들(64)을 포함할 수 있는, 호스트 OS/VMM 파티션(60)을 포함하는 종래의 소프트웨어를 포함한다. 이러한 코드는 시스템 메모리(30)의 다른 파티션에 저장될 수 있다. HRM(25)은 OS/VMM 파티션에 시스템 하드웨어의 추상화(abstraction)를 제공할 수 있다.

[0011] 실시예들은 S3에서 S0으로의 전이 및 S0에서 S3으로의 전이를 실행하는데 소비될 필요가 있는 시간의 양을 감소시키기 위해 숨겨진 환경에서 실행하는 프로세서 마이크로코드에 의해 구현될 수 있는 메커니즘들을 제공할 수 있다. 시스템을 S3으로 전이하면, 프로세서는 전원이 꺼지게 되고 메모리는 셀프-리프레시 상태가 된다. 이 때에, 메모리에 대한 보호가 없으며, 각종 실행 시간 보호들은 작동하지 않는다. 따라서, 숨겨진 메모리를 포함하는 메모리는 상이한 보안 위협들에 취약할 수 있다. 상태들 S1 및 S2는 프로세서 컴포넌트들에(예를 들어, 코어 및 언코어(uncore)에만) 적용되고 시스템 메모리를 수반하지 않음을 주지하라. 추가의 저 전력 상태들 S4 및 S5는 시스템 메모리로의 전력을 차단해서 모든 콘텐츠들이 손실된다.

[0012] 이하의 리스트가 철저하지 않지만, 위협들은 다수의 상이한 형태들을 취할 수 있음을 알 것이다. 예를 들어, 이러한 한 위협은 공격자(attacker)가 S3 전력 상태 전이들 동안에 메모리 이미지를 변경/대체할 수 있는 악의적인 코드 인젝션일 수 있다. 다른 위협들은 해시 알고리즘들을 사용하는데, 공격자는 유효한 페이지의 해시 서명을 가진 공격 메모리 페이지를 찾기 위해 암호 해시 알고리즘들의 약점들을 활용할 수 있으며, 그 후 공격 페이지가 메모리에 주입된다. BORE(break once run everywhere) 공격은, 공격 페이지를 획득한 공격자가 다수의 유사하게 구성된 시스템들에 공격 페이지를 주입할 수 있는 공격이다. 재생 공격(replay attack)은 공격자가 모든 메모리 페이지들 및 무결성 검사 값들을 보관하고 후에 현재 메모리 페이지들 및 무결성 값들을 대체하

는 경우 발생할 수 있다. 페이지 교환(a page swap) 공격은, 공격자가 동일한 암호 메시지 인증 코드(CMAC) 값들을 가진 페이지들을 교환할 수 있을 때, 발생한다.

[0013] 특정 위협들은 실행을 허용하기 전에 셀프-리프레시 상태로 메모리 이미지를 HRM이 측정(예를 들어, 해시 및 검증)하게 함으로써 처리될 수 있다. 일 실시예에서, 본 방식은 무결성 검사 값(ICV) 어레이에 값들이 기록되는 페이지 단위 무결성 검사를 실행한다. 해시는 HRM 마이크로코드에 의해 실행된다. 일 실시예에서, ICV 어레이의 각각의 엔트리는 이하의 정보를 포함한다: 코멘트들에 대한 네트워크 작업 그룹 요청(Network Working Group Request for Commnets(RFC) 4493)(2006년 6월)에 따른 AES-CMAC(Advanced Encryption Standard-Cipher-based Message Authentication Code) 오퍼레이션 또는 페이지의 안전한 해시 알고리즘(SHA) 해시일 수 있는, 무결성 검사 값; 최종 ICV 계산이 실행된 이래 페이지가 변경되었는 지의 여부를 나타내는 참/거짓일 수 있는, ICV 유효 지시자; 및 다이렉트 메모리 액세스(DMA)를 사용하는 장치들에 의해 페이지가 액세스될 수 있는지의 여부를 나타내는 참/거짓일 수 있는, DMA 페이지 지시자.

[0014] 일 실시예에서, HRM은 배경 태스크를 사용해서 숨겨진 페이지들을 정기적으로 해시할 수 있다. 숨겨진 환경으로의 각각의 전이 전에, 배경 태스크는 메모리의 한 고정된 집합의 페이지들을 해시하도록 호출될 수 있다. 예를 들어, 숨겨진 이미지가 64MB 크기이면, 무결성 보호할 16384개의 페이지들이 있다. 배경 태스크는 순차 순환 대기 방식(round robin manner)으로 16K 페이지들에 걸쳐 실행할 수 있으며, 각각의 실행시에 16 페이지들의 해시를 실행한다. 따라서, 배경 태스크는 숨겨진 실행 환경(HEE)에 할당된 모든 16K 페이지들의 해시를 실행하기 위해 1K 회 호출될 필요가 있다. 페이지들을 해시하는데 걸리는 시간은 HEE에 기인하며 실행 크레디트로부터 공제된다.

[0015] ICV 계산은 상이한 해시 알고리즘들에 따를 수 있다. 예를 들어, SHA1 해시 대신, SHA2 오퍼레이션 등의 더 강한 해시 알고리즘이 해시 알고리즘 활용을 처리하는데 사용될 수 있다. 그러나, SHA2 알고리즘들의 사용과 연관된 성능 페널티 및 메모리 오버헤드 비용이 있을 수 있다. SHA256은 대략 20 사이클/바이트를 요구하고, SHA512는 대략 12 사이클/바이트를 요구한다. 또한, SHA2 알고리즘들은 통상 꽤 큰 메시지 다이제스트를 생성한다. SHA256은 256-비트(32 바이트) 메시지 다이제스트(MD)를 가지며 SHA512는 64 바이트 MD를 가진다. 이 오버헤드는 상당한 추가 오버헤드인 메모리(16-64MB)의 페이지들의 수로 승산된다.

[0016] 따라서, 다수의 실시예들에서, AES-CMAC 오퍼레이션 등의 SHA2에 대한 대안이 사용될 수 있으며, 이 대안은 SHA 알고리즘보다 더 빠르다. 또한, CMAC는 제2 프리-이미지 저항을 가지기에 SHA2 함수 대신 사용될 수 있다.

[0017] 이제 도 2를 참조하면, 본 발명의 일 실시예에 따른 ICV 어레이의 블록도가 도시되어 있다. 도 2에 도시된 바와 같이, 숨겨진 실행 환경은 메모리의 숨겨진 파티션에 정보(즉, 코드 및 데이터)를 저장할 수 있다. 구체적으로 말해서, 도 2에 도시된 바와 같이, 메모리(110)의 복수의 페이지들(110_1-110_n)은 숨겨진 실행 환경을 위해 분할될 수 있다. 숨겨진 환경의 배경에서의 동작 중에, HRM은 이러한 페이지들 각각의 페이지의 해시들을 실행하고 해시 값들을 ICV 어레이(120)에 저장할 수 있다. 도시된 바와 같이, ICV 어레이(120)는 숨겨진 파티션(110)의 한 페이지에 각각 대응하는 복수의 엔트리들(125_a-125_n)을 포함할 수 있다. 각각의 엔트리는, 상술된 바와 같이, 유효 지시자(122) 및 DMA 지시자(124)와 함께, CMAC 또는 해시 값을 저장하기 위해 ICV 필드(126)를 포함한다. 일 실시예에서, 구현들이 다른 로케이션들에 이 ICV 어레이를 저장할 수 있더라도, ICV 어레이(120)는 또한 숨겨진 파티션(110)의 일부분에 저장될 수 있다. ICV 어레이(120)의 해시는 또한 HRM에만 알려진 암호 키를 사용해서 계산 및 암호화될 수 있다. 이러한 방법으로, 오직 HRM 만이 ICV 어레이를 변경할 수 있다.

[0018] S0에서 S3으로 전이할 때, HRM은 숨겨진 메모리 페이지들에 대한 ICV 값들을 생성할 수 있다. 따라서, S0에서 S3으로 전이할 때, HRM은, 일 실시예에서, 상술된 지시자들과 함께, 각각의 메모리 페이지에 대한 CMAC 값들을 계산함으로써 동적으로 ICV 어레이를 구성한다. ICV 어레이는 사용되지 않은 메모리 페이지에 저장될 수 있다. 그 후, 숨겨진 메모리는 본 발명의 일 실시예에 따라 무결성 계산들 및 보호를 사용해서 보호될 수 있다.

[0019] 이제 도 3을 참조하면, 본 발명의 일 실시예에 따른 숨겨진 메모리 무결성 계산 및 보호의 개요의 블록도가 도시되어 있다. 도 3에 도시된 바와 같이, 숨겨진 파티션(110)은 복수의 메모리 페이지들 P1 - Pn을 포함할 수 있으며, 각각의 페이지는 페이지 테이블 엔트리(105)에 존재하는 정보를 사용해서 액세스될 수 있다. 이러한 페이지 테이블 엔트리들은 변환 색인 버퍼(TLB; translation lookaside buffer) 또는 다른 구조로 이 엔트리들을 캐시할 수 있는 프로세서 내에 존재할 수 있으며, 또는 엔트리들은 숨겨진 파티션 자체 내에 존재할 수 있다. 숨겨진 파티션에 대한 페이지 테이블 엔트리들은 개별 페이지 테이블에 저장될 수 있음을 주지하라.

[0020] 무결성 계산을 실행하기 위해, 각각의 메모리 페이지가, 일 실시예에서, 프로세서 코어, 언코어, 통합된 입력/

출력 컨트롤러, 주변 컨트롤러, 메모리 컨트롤러, 또는 보안 코프로세서에 존재하는 가속 장치 또는 다른 특별 하드웨어를 사용해서 구현될 수 있는, AES-CMAC 함수(150)에 제공될 수 있다. CMAC 계산의 일부로서 모든 플랫폼 마다 고유한 개인 키 K1을 사용해서, 공격자가 각각의 플랫폼에 대해 K1을 쉽게 추측할 수 없기에, 다른 플랫폼으로부터의 CMAC 값은 목표 플랫폼을 공격하는데 사용될 수 없다.

[0021] 일 실시예에서, 이 키는 128 비트들일 수 있으며, 각각의 플랫폼에 유일할 수 있다. 일 실시예에서, AES-CMAC 계산은 AES 알고리즘을 구현하기 위해 사용자 레벨 ISA 명령들을 실행할 수 있는 코어에 포함된 AES 가속 장치를 사용할 수 있다. AES 가속 장치에 의해, AES-CMAC는 일 실시예에서 대략 3.5 사이클/바이트로 작동할 수 있어서, 페이지 무결성 보호를 위한 효율적이고 안전한 방법을 제공한다. 각각의 페이지에 대한 결과 CMAC 값은 ICV 어레이(120)에 저장될 수 있다. 일 실시예에서, 페이지 당 메모리 오버헤드는 CMAC 당 64 비트이다. 고순위 비트들은 128-비트 K1 값으로부터 야기된 128-비트 CMAC로부터 선택된다. 또한, 전체 ICV 어레이(120)에 대한 무결성 계산은 유사하게 함수(160)를 사용해서 동일한 키 K1을 사용해서 실행될 수 있으며, 따라서 스토리지(165)에 어레이 CMAC 값(CMACA)을 저장한다. 일 실시예에서, 이 CMACA 값은 64 비트일 수 있다.

[0022] 이러한 무결성 값들이 계산된 예시를 식별하기 위해, 코멘트들 3602를 위한 네트워크 작업 그룹 요청(Network Working Group Request for Commnets 3602)(2003년 9월)에 따른 AES-암호 블록 체이닝(CBC) 암호 함수(170)가, 카운터(180)에 저장된, 카운터 C1의 값 및 키 값 K1에 대해 암호화를 실행하도록 동작하는, 스토리지(175)에 저장된, 제2 개인 키 K2를 사용해서 암호화된 블로브(190)를 생성하도록 동작할 수 있다. 일 실시예에서, K2는 HRM 스토리지 키(HRM에만 알려진 대칭 키)로부터 유도된다. 차례로, HRM 스토리지 키는 프로세서 또는 칩셋의 한 집합의 퓨즈들로부터 획득될 수 있다. 각종 실시예들에서, 카운터는 단조 카운터(monotonic counter)일 수 있다.

[0023] 상이한 구현들에서, 단조 카운터 리소스는 수개의 기술들을 사용해서 구현될 수 있다. 일례들로서, 카운터는 TPM 또는 ME에 액세스함으로써 실현될 수 있다. 본 일례에서, HRM은 카운터 리소스들에 대한 액세스를 보장하기 위해 TPM의 소유권을 가질 수 있으며, 또는 HRM은, 예를 들어, 프로세서에 존재하는 영구 레지스터에 액세스할 수 있다. 다른 일례로서, HRM은 ME와 신뢰 관계를 설정함으로써(예를 들어, 키 교환을 실행함으로써) 숨겨진 스토리지 영역(예를 들어, 플래시 또는 다른 비휘발성 메모리)에 액세스할 수 있다. 또 다른 일례로서, HRM은, 예를 들어, 스토리지 장치와 신뢰 관계를 설정함으로써(예를 들어, 키 교환을 실행함으로써), 하드 디스크 드라이브 또는 고체 상태 드라이브(HDD/SSD) 등의 스토리지 장치에 액세스할 수 있다.

[0024] 결과 블로브(190)는 저장되고, 차후에 한 동작 상태로 다시 전이한 후에 숨겨진 파티션의 메모리가 온전한 상태로 유지되는 지를 결정하기 위해 액세스될 수 있다. C1은 블로브에 포함됨에 이어 재개 카운터 값 C2로 증가한다. 그 후, C2는, 재생 공격이 검출될 수 있음을 보장하기 위해, 예를 들어, 판독-증가-기록 원자 조작(a read-increment-write atomic operation)을 사용해서, 영구 메모리에 기록될 수 있다. 다른 실시예들에서, S3 상태로부터 재개 후에 증가가 발생할 수 있다.

[0025] 일부 실시예들에서, 페이지 교환 공격은 CMAC의 페이지 메타데이터로부터 취해진 페이지 인덱스 값을 포함함으로써 처리될 수 있음을 주지하라. 페이지 인덱스 값들은 공격자들을 좌절시키기 위해 하드웨어 보호들에 의존하는 HRM에 의해 결정된다. HRM의 프로그램 버그들은 1 보다 많은 페이지 CMAC에 포함되는 복제 페이지 인덱스 값들을 야기할 수 있다. 대안 기술은 제2 단조 카운터를 더 신뢰할만한 고유한 식별자로서 사용할 수 있다.

[0026] 이제 도 4를 참조하면, 본 발명의 일 실시예에 따라 저 전력 상태로 전이하기 위한 한 방법의 흐름도가 도시되어 있다. 도 4에 도시된 바와 같이, 방법(200)은 저 전력 상태로 전이(예를 들어, S0에서 S3으로 전이)하라는 명령을 수신하는 것으로 시작될 수 있다(단계 210). 숨겨진 환경의 마이크로코드를 사용해서, AES-CMAC 값들은 개인 키 K1을 사용해서 숨겨진 메모리의 각각의 페이지에 대해 계산될 수 있다(단계 220). 이 CMAC 값들은 ICV 어레이에 저장될 수 있다(단계 230). 이러한 모든 계산들이 완료될 때, AES-CMAC 값은, 동일한 키 K1을 사용해서, 전체 어레이에 대해 계산될 수 있다(단계 240). 이 CMAC 값은 다른 스토리지에 저장될 수 있다(단계 250).

[0027] 여전히 도 4를 참조하면, 단조 카운터 값 C1 및 K1은 메모리에 기록될 수 있다(단계 260). 일례로서, 이 값들은 캐시 로케이션 등의 임시 스토리지 로케이션에 또는 숨겨진 파티션 내에 저장될 수 있다. 이 메모리는 그 후 개인 키 K2를 사용해서 암호화되어서 암호화된 블로브를 생성할 수 있다(단계 270). 이 블로브는 그 후 S3 영구 메모리에 저장될 수 있다(단계 280). 일례로서, 블로브는 S3 모드 동안에 셀프-리프레시 상태로 유지되는 숨겨진 파티션의 미리결정된 영역에 저장될 수 있다. 도 4의 실시예에서 이 특정 구현으로 도시되지만, 본 발명의 범위는 이와 관련하여 한정되지 않는다.

[0028] 이제 도 5를 참조하면, 저 전력 상태로부터 정상 전력 모드에서의 실행을 재개하기 위한 한 방법의 흐름도가 도시되어 있다. 도시된 바와 같이, 방법(300)은 HRM 재개 통지를 수신함으로써 시작될 수 있다(단계 310). S3 재개의 경우, HRM은, 예를 들어, 기본 입력/출력 시스템(BIOS)을 통해, 숨겨진 이미지를 재개하라는 통지를 수신할 수 있다(단계 310). HRM은 숨겨진 환경에 대한 컨테이너를 생성하고 그 후 숨겨진 메모리 내에 유효 ICV 어레이가 있는 지를 결정한다. 그렇다면, ICV 어레이의 타당성이 ICV 어레이(CMACA)의 CMAC 및 각각의 페이지의 CMAC를 검증함으로써 결정될 수 있다(단계 320). ICV 어레이 무결성이 온전하지 않으면, HRM은 정상 리셋 경로로서 숨겨진 초기화를 계속한다. 즉, 메모리의 이미지는 폐기되고, 예를 들어, 비휘발성 스토리지로부터 프레시 로드 코드 로드가 실행된다(단계 325).

[0029] ICV 어레이의 무결성이 타협되지 않으면, HRM은 모든 HEE 페이지들을 숨겨진 페이지 테이블들에 존재하지 않는 것으로 표시하며(단계 330), 숨겨진 부팅 로더 엔트리 포인트로 제어를 이동한다(단계 335). 부팅 로더가 실행을 개시함에 따라, 코드 및 데이터 페이지 액세스들(단계 340)은 HRM으로 페이지 테이블 폴트들을 생성한다(단계 345). ICV 어레이에 페이지에 대한 유효 ICV가 있으면(단계 350), HRM은 폴트 페이지에서 ICV를 계산하고 ICV 어레이의 ICV에 대해 매칭하여 페이지의 무결성을 검증한다(단계 355). 검증되면(단계 370), 페이지는 (페이지에 대한 변화들을 추적하기 위해) 판독 전용 페이지로서 숨겨진 페이지 테이블로 매핑된다(단계 375). 그 후, 페이지에 대한 액세스가 허용될 수 있다(단계 380).

[0030] 여전히 도 5를 참조하면, 단계 350에서, 대응 ICV 어레이 엔트리에 대해 (예를 들어, 유효 지시자에 의해 표시된 바와 같이) 페이지에 대한 유효 ICV가 없다고 결정되면, 제어는 단계 360으로 넘어가고, 페이지는 초기화 코드로 로딩될 수 있다. 또한, 도시된 바와 같이, 유효 ICV 엔트리가 존재하지만, 계산된 ICV가 저장된 대응 ICV 값과 매칭하지 않으면, 단계 370에서의 검증은 실패하고, 보안 위반이 보고될 수 있다(단계 390). 도 5의 실시예에서 이 특정 구현으로 도시되지만, 본 발명의 범위는 이와 관련하여 한정되지 않음을 이해하라. 예를 들어, 일부 구현들에서, (예를 들어, 시스템 메모리의 숨겨지지 않은 메모리 파티션의) 게스트 OS에 액세스될 수 있는 메모리 페이지들의 타당성 검사가 실행될 수 있으며, HRM이 이러한 타당성 검사를 가능케 한다.

[0031] 이제 도 6을 참조하면, 본 발명의 일 실시예에 따른 한 검증 방법의 흐름도가 도시되어 있다. 도 6에 도시된 바와 같이, 방법(400)은 S3 상태로부터 재개할 때 ICV 어레이를 검증할 때 HRM에 의해 구현될 수 있다. 도시된 바와 같이, 방법(400)은 ICV 무결성을 결정하라는 요청을 수신함으로써 개시될 수 있다(단계 410). 따라서, HRM은 HRM의 스토리지 키로부터 암호화 키 K2를 유도하고(단계 420), 그에 맞춰 K2를 사용해서 블로브를 해독하여 키 K1 및 카운터 값 C1을 획득할 수 있다(단계 430). 또한, S3으로부터 재개하기 전에 또는 재개할 때 재개 카운터 값 C2를 증가시키기 위해 전력 상태 전이가 야기될 수 있음을 주지하라. 재개 카운터 값 C2는 재개할 때 영구 스토리지로부터 획득될 수 있다(단계 440).

[0032] 단계 450에서, HRM은 C1 값이 미리결정된 임계값(예를 들어, 1) 보다 더 많이 C2 값과 상이한 지를 결정할 수 있다. 만약 그렇다면, 검증은 실패하고 보안 위반이 보고될 수 있다(단계 460). 비교가 예상대로 이면, 제어는 단계 470으로 넘어가서, ICV 어레이에 대한 CMAC 값 및 숨겨진 실행 환경 페이지들 각각에 대해 CMAC 값들이 검증될 수 있다(단계 470). 계산된 값들이 저장된 값들과 매치하면, 검증은 단계 475에서 결정된 바와 같이 성공적일 수 있으며, 성공적인 검증의 보고가 이루어질 수 있다(단계 480). 그렇지 않다면, 검증은 실패하고 제어는 단계 460으로 넘어간다.

[0033] AES-CMAC 및 CMAC 값들(CMACA)의 어레이의 AES-CMAC에 기초한 메모리 페이지들의 해싱을 실행함으로써, 실시예들은 제2 프리-이미지 공격들을 방지하기 위한 더 효율적인 기술을 실현할 수 있다. 또한, CMACA 값의 암호화가 포함된 단조 카운터 값을 사용해서, 공격자가 이전 전력 상태 전이로부터의 페이지들로 모든 메모리 페이지들을 대체하는 것을 방지할 수 있다. 또한, 페이지 CMAC 계산에서 페이지 인덱스 값을 포함함으로써, 동일한 CMAC 값을 가진 상이한 어드레스 로케이션들에서 발생하는 동일한 페이지들의 잘못된 대체가 방지될 수 있다.

[0034] 실시예들은, 사용자의 개인 목표들에 민감하면서, 외부 엔티티들에게 해당 환경을 인증해 주기 위해 마이크로프로세서의 마이크로코드를 사용해서 예시된 숨겨진 관리 환경을 또한 가능케 할 수 있다. 또한, 환경의 무결성 측정들이 마이크로코드에 의해 기록되어서, 차후에 원격 또는 외부 엔티티에 의해 사용 및 검증될 수 있다.

[0035] 이러한 인증 활동을 실현하기 위해, 마이크로코드는 인증의 기능들이 외부 위협들로부터 보호되는 신뢰 근원(a root-of-trust)을 생성할 수 있다. 프로세서의 확장 레지스터가 무결성 측정들을 저장하는데 사용될 수 있다. 그 후, 개인 키(예를 들어, 소위 EPID)가 확장 레지스터의 콘텐츠를 디지털 방식으로 서명하는데 사용될 수 있다. 차례로, 대응 공개 키는 EPID 개인 키의 서명을 암호로 검증할 수 있다. 일부 실시예들에서, OwnerID 값

이 또한 플랫폼의 프라이버시 및 보안 정책을 결정할 책임이 있는 엔티티에 의해 제공될 수 있다.

[0036] 비대칭 코드 서명 키가 실행/개시시 숨겨진 환경으로 로딩되는 펌웨어의 무결성을 검증하는데 사용될 수 있다. 차례로, 무결성 매니페스트는 수용할 만한 무결성 측정들의 화이트 리스트를 포함할 수 있다. 이 매니페스트는 숨겨진 환경으로 론치(launch)될 수 있는 모든 애플리케이션들을 나타내는 론치 제어 정책을 보호할 수 있다.

[0037] 일반적으로, 안전한 숨겨진 환경으로 부팅하는 동작의 2 단계들이 발생할 수 있다: 1) 론치 단계; 및 2) 인증 단계. 론치 단계에서, 펌웨어는 HRM에 의해 메인 메모리로부터 숨겨진 메모리 페이지들에 로딩된다. 무결성 매니페스트는 비대칭 코드 서명 키를 사용해서 먼저 검증된다. 그 후, 로딩될 모듈은 매니페스트에 포함된 화이트 리스트(예를 들어, 해시 값들)를 사용해서 검증된다.

[0038] 코드 측정들은, 그 값이 불신 코드에 의해 갱신 또는 리셋될 수 없도록, HRM에 의해서만 액세스될 수 있는 확장 레지스터에 기록된다. 후술되는 바와 같이, 일 실시예에서, 검증될 제1 객체는, 숨겨진 환경의 지속적인 커널(커널-1)을 론치하는데 사용되는, 커널-0, 부팅 로더라고 하는, 숨겨진 환경의 일시적인 커널이다. 커널-0은 HRM으로부터 코드 서명 키를 수신하고 커널-1에 대한 코드 무결성 측정들 및 검증들을 실행한다. 측정들은 HRM에 제공되어 확장 레지스터에 저장된다. 그 후, 커널-1은 커널-0에 의해 제공되는 론치 제어 정책을 사용해서 애플리케이션 코드 무결성을 검증한다. 애플리케이션(들)의 무결성 측정들은 론치 히스토리라고 하는 로그 파일에 기록될 수 있다. 인증 단계의 세부 사항들은 더 후술된다.

[0039] 이제 도 7을 참조하면, 본 발명의 일 실시예에 따른 한 하이 레벨 론치 방법의 흐름도가 도시되어 있다. 도 7에 도시된 바와 같이, 방법(500)은 시스템 리셋으로 시작될 수 있다(단계 510). HRM 코드는 일시적인 커널을 검증하기 위해 실행될 수 있다(단계 520). 일시적인 커널이 검증되면(단계 530), 제어는 단계 540으로 넘어가서, 일시적인 커널의 측정이 프로세서의 확장 레지스터에 저장될 수 있다(단계 540). 다른 경우, 검증이 실패하면, 실패가 보고될 수 있다(단계 535).

[0040] 도시된 바와 같이, 검증이 유효한 일시적인 커널을 나타낼 때, 일시적인 커널은 론치되고(단계 550) 숨겨진 커널을 검증하기 위해 실행될 수 있다(단계 560). 이 숨겨진 커널은 숨겨진 실행 환경에 대한 지속적인 커널일 수 있다. 따라서, 단계 565에서, 숨겨진 커널이 검증되었는 지가 결정될 수 있다. 그렇다면, 숨겨진 커널의 측정은 확장 레지스터에 저장될 수 있으며(단계 575), 실행될 하나의 또는 그 이상의 애플리케이션들이 숨겨진 커널을 사용해서 검증될 수 있다(단계 580). 다른 경우, 숨겨진 커널이 검증되지 않으면, 제어는 단계 570으로 넘어가서, 실패가 보고될 수 있다. 도 7의 실시예에서 이 특정 구현으로 도시되지만, 본 발명의 범위는 이와 관련하여 한정되지 않는다.

[0041] 애플리케이션의 론치는 HRM에 이르기까지 각각의 선행 층의 성공적인 론치에 좌우되며, 본 발명의 일 실시예에 따른 론치 오퍼레이션들의 흐름(500')을 기술한, 도 8에 그 세부 사항들이 도시되어 있다. 도시된 바와 같이, HRM 층(510')은, 마이크로코드(및 따라서 HRM)가 실행되는 패치 리셋 메커니즘(Patch-at-Reset mechanism)의 일부로서 암시적으로 론치된다. HRM 코드 자체는 제조의 일부로서 서명된다. 제조자-발행 RSA(Rivest Shamir Adelman) 2048 또는 더 큰 공개 키인 코드 서명 키(HPK)가 코드 이미지에 포함된다. HEE 커널-0 코드는 리셋 후에 HRM에 의해 로딩된다. 커널-0 및 커널-1 이미지들은 비휘발성 스토리지에 위치할 수 있다. 이미지들은 실행 가능 코드 및 코드의 무결성 해시를 포함하는 서명된 매니페스트를 포함한다.

[0042] HRM이 커널-0 코드를 로딩할 때, 매니페스트의 해시 값에 비교되는 커널-0 코드의 해시(예를 들어, AES-CMAC 해시)를 계산한다. 매니페스트 서명은 HPK 코드-서명 키를 사용해서 검증된다. 커널-0 이미지 무결성이 검증되면, 프로세서 내에 있을 수 있는 확장 레지스터(515')는 해시 값 및 성공적인 검증을 나타내는 플래그(예를 들어, K0_SUCCESS_BIT = TRUE)로 갱신될 수 있다. 다른 경우, 레지스터는 실패한 검증을 나타내도록 갱신된다(예를 들어, K0_SUCCESS_BIT = FALSE).

[0043] 플래그 비트가 설정되는 경우에만(예를 들어, K0_SUCCESS_BIT = TRUE), HRM이 커널-0을 론치하고 HPK 공개 키를 커널-0에 전달할 때, 커널-0 층(520')은 개시될 수 있다. 다른 경우, HRM은 확장 레지스터의 커널-0 또는 커널-1 사용을 허용하지 않는다. 일 실시예에서, 커널-0은 일시적인 커널이며, 그 목적은 커널-1을 론치하는 것이다. 커널-0은 커널-1 코드, 코드의 해시, 커널-1에서의 실행이 허용되는 애플리케이션들의 화이트 리스트를 포함하는 애플리케이션 론치 제어 정책(AppLCP), 및 HPK의 서명을 포함하는 커널-1 매니페스트를 로딩한다. AppLCP는 해싱되어 확장 레지스터(515')로 확장된다. AppLCP 포맷은 확장 레지스터에서 확장된 해시에 포함된 버전 정보를 포함할 수 있다. 커널-0이 커널-1 코드를 로딩할 때, 커널-1 매니페스트의 해시 값에 비교되는 커널-1 코드의 해시(예를 들어, AES-CMAC 해시)를 계산한다. 매니페스트 서명은 HPK 코드-서명 키를 사용해서 검

증된다. 일 실시예에서, 모든 커널-1 코어 서비스들 및 공유 라이브러리들은 커널-1 무결성 측정에 포함된다.

[0044] 커널-1 이미지 무결성이 검증되면, 확장 레지스터(515')는 해시 값 및 성공적인 검증을 나타내는 플래그(예를 들어, K1_SUCCESS_BIT = TRUE)로 갱신될 수 있다. 다른 경우, 레지스터는 실패한 검증을 나타내도록 설정된 플래그로 갱신된다(예를 들어, K1_SUCCESS_BIT = FALSE). 커널-1은 AppLCP 무결성 해시를 계산한다. 해시는 갱신을 위해 확장 레지스터(515')에 제공된다. 이 때에, 커널-1 층(530')은 론치된다. AppLCP는 론치의 일부로서 커널-1에 전달된다. 커널-1은 론치될 각각의 애플리케이션에 대한 무결성 해시를 계산함으로써 안전하게 애플리케이션들을 로딩한다. AppLCP와 별개인 정책이, 어떤 애플리케이션들이 로딩될 것인지를 결정한다. 커널-1 계산 해시(AES-CMAC)가 AppLCP에 포함된 해시 값과 매치하면, 애플리케이션 무결성 검사는 성공이다. 애플리케이션 론치 결과들의 히스토리는 론치 히스토리(535')에 기록될 수 있다. 이 때에, 숨겨진 환경은 실행할 준비가 되어 있다.

[0045] 인증 단계에서, 검증 장치(verifier)는 환경 무결성의 증명을 획득하라는 요청을 개시할 수 있다. 검증 장치는 논스(nonce)(예를 들어, 논스-1)를 커널-1에 제공하며, 이는 인증 결과가 프레시임(즉, 이전 요청으로부터 재생되지 않음)을 보장한다. 그 후, 커널-1은 처리를 위해 논스-1과 함께 HRM에 전달되는 론치 히스토리의 해시를 구성할 수 있다.

[0046] HRM은 론치 히스토리 해시, 논스, 확장 레지스터 및 OwnerID(후술됨)를 메시지에 포함시키며, 메시지는 후에 개인 키에 의해 디지털 방식으로 서명된다. 서명된 인증 레코드는 커널-1로 복귀되고, 그 후 원래의 검증 장치에 발송된다. 검증 장치는 이전에 수신한 개인 키의 공개 카피를 사용해서 EPID 서명을 검사한다. 검증 장치는 숨겨진 환경과의 상호 작용들과 연관된 리스크를 추정하도록 설계된 인증 레코드를 해석하는 정책을 적용한다.

[0047] 따라서, 각종 실시예들에서, 인증은 마이크로프로세서 코어들과 별개인 이산 인증 컨트롤러를 필요로 하지 않고 프로세서에서 전적으로 제공될 수 있다. 이는 인증 컨트롤러와 코어를 결합하는 버스 아키텍처들 및 인터커넥트들을 신뢰할 필요를 방지한다. 또한 여전히, 온칩 프로세서 리소스가 숨겨진 관리 환경을 예시할 수 있기에, 실시예들은 숨겨진 환경과 인증을 결합할 수 있다.

[0048] 즉, 인증은 프로세서 코어(들)에서 실행되는 마이크로코드의 숨겨진 환경과 통합된다. 마이크로코드는 프로세서 마이크로코드와 유사한 신뢰 근원을 가지며, EPID, 확장 레지스터 및 코드 서명 키를 적절히 관리하는 것을 신뢰할 수 있다. 소정의 제조자의 제공된 하드웨어에 의해 구성되었기 때문에, EPID를 사용해서 수신된 인증에 서명해서, 인증 레코드가 믿을만하다고 검증 장치가 알 수 있게 한다. 사용자 생성 개인 키(예를 들어, OwnerID)가 사용되는 일 실시예에서, OwnerID를 인증 레코드에 포함시켜서, 플랫폼의 소유자가 OwnerID가 어떻게 고유한 지 또는 유일하지 않은 지를 결정함으로써 프라이버시 속성들을 제어할 수 있게 한다.

[0049] HEE 환경의 인증은 검증 장치의 인증 요청 메시지로 시작된다. 검증 장치는 커널-1과 상호 작용하는 기능을 가진 임의의 환경이다. 이제 도 9를 참조하면, 본 발명의 일 실시예에 따른 인증 방법(600)의 흐름도가 도시되어 있다. 도 9에 도시된 바와 같이, 검증 장치(610)는 인증이 의미 있는 트랜잭션에 고유한 논스(논스1)를 계산한다. 트랜잭션은 인증서 등록 프로세스와 같이 오래 지속되는 것(수년에 걸침)일 수 있으며, 또는 웹 페이지를 서비스하는 HTTP(hypertext trasfer protocol) 세션과 같이 순간적인 것(수초에 걸침)일 수 있다. 검증 장치는 논스1이 어떻게 구성되는 지를 완전히 제어한다. 논스1은 인증이 획득되는 트랜잭션에 대한 재생 공격들을 방지할 수 있다. 검증 장치(610)는 인증 요청의 일부로서 논스1을 커널-1(620)에 제공하며, 이는 어떤 애플리케이션들이 인증에 포함될 것인지를 지정한다.

[0050] 커널-1은 소정의 애플리케이션(예를 들어, App-X)에 대한 무결성 값 및 론치 상태를 획득하라고 론치 히스토리(625)에 질의한다. 이 정보는 그 후 논스1과 함께 HRM(630)에 발송된다. HRM은 커널-1에 의해 전달된 정보를 포함하는 인증 레코드 및 고객의 환경으로의 숨겨진 전개의 일부로서 제공된 OwnerID 등의 개인 키를 구성한다. 도시된 바와 같이, OwnerID(635)는 프로세서 내부 또는 외부의 HRM과 연관된 비휘발성 스토리지에 저장될 수 있다. 플랫폼이 먼저 환경에 로딩될 때, OwnerID가 형성될 수 있어서, 제조자에 의해 정의된 아이덴티티와 대조적으로 플랫폼의 소유자가 하드웨어에 대한 자신의 아이덴티티를 설정할 수 있게 한다. 따라서, 하드웨어의 소유자는 플랫폼의 이름을 결정할 수 있으며, 고유성을 얼마나 많이 또는 얼마나 적게 원하는 지를 결정할 수 있다. 따라서, 조직에 대한 개인 정책들에 기초하여, OwnerID는 결정된 양의 개인적으로 식별 가능한 정보를 개시할 수 있다. OwnerID는, 예를 들어, 언코어에서, 퓨즈들로부터 유도된 HWK(HRM wrapping key)를 사용해서 암호화된다. 래핑된 OwnerID는 그 후 커널-0, 커널-1, 애플리케이션들, 매니페스트들 및 애플리케이션 데이터와 함께 비휘발성 스토리지에 저장될 수 있다. AppLCP의 해시 뿐만 아니라 커널-1 및 커널-0의 무결성 측정들을 포함하는, 예를 들어, 언코어에 포함된 머신 특정 레지스터(MSR)일 수 있는 확장 레지스터(638)로부터의 정보는

EPID 개인 키를 사용해서 서명된 인증 레코드를 제공받는다.

[0051] 서명된 결과는 검증 장치(610)에 발송되어, EPID 서명이 EPID 공개 키를 사용해서 검증된다. 인증 레코드 논스 1은 재생 공격들을 검출하기 위해 원본과 비교된다. 검증 장치에 의해 소유된 플랫폼인 지를 결정하기 위해 고객 제공 OwnerID를 검사한다. App-X가 의도된 환경에서 실행중인 지를 검증하기 위해, 애플리케이션들, 커널-1, 및 커널-0 이미지들의 공개된 화이트 리스트를 참고할 수 있다. 일례로서, 고객의 환경 캐시로 전개된 플랫폼들에 할당된 OwnerID(들)는 네트워크 또는 기업 내에 일반적으로 전개된 검증 에이전트들이 질의할 수 있는 디렉토리 서비스에 보관된다. 또한, 검증 장치는, 디렉토리 서비스에 또한 보관된, 소유자 승인 커널들, 애플리케이션들 및 구성들의 화이트 리스트에 액세스할 수 있다.

[0052] 이제 도 10을 참조하면, 본 발명의 일 실시예에 따른 한 프로세서의 블록도가 도시되어 있다. 도 10에 도시된 바와 같이, 프로세서(700)는 멀티-스테이지 파이프라인 무순위 프로세서일 수 있다. 프로세서(700)는 본 발명의 일 실시예에 따른 숨겨진 실행 환경에서 사용되는 각종 피처들을 설명하기 위해 도 10에서 비교적 간소화된 그림으로 도시된다.

[0053] 도 10에 도시된 바와 같이, 프로세서(700)는, 실행될 매크로-명령들을 페치하고 프로세서에서 차후에 사용되도록 준비시키는데 사용될 수 있는 프론트 엔드 유닛들(710)을 포함한다. 예를 들어, 프론트 엔드 유닛들(710)은 페치 유닛(701), 명령 캐시(703), 및 명령 디코더(705)를 포함할 수 있다. 도시된 바와 같이, 프론트 엔드 유닛들(710)은 프로세서의 ISA의 프로세서 명령들을 저장하는 ISA 마이크로코드 스토어(706)를 더 포함할 수 있다. 또한, 별개의 숨겨진 마이크로코드 스토어(708)는 숨겨진 실행 환경에 대한 프로세서 명령들을 저장할 수 있다. 페치 유닛(701)은, 예를 들어, 메모리 또는 명령 캐시(703)로부터 매크로-명령들을 페치할 수 있으며, 프리미티브들, 즉, 프로세서에 의해 실행될 매크로-오퍼레이션들로 디코딩하기 위해 명령 디코더(705)에 제공할 수 있다.

[0054] 여전히 도 10을 참조하면, 마이크로-명령들을 수신해서 실행을 위해 준비시키는데 사용될 수 있는 OOO(out-of-order) 엔진(715)이 프론트 엔드 유닛들(710) 및 실행 유닛들(720) 사이에 연결된다. 더 구체적으로 말해서, OOO 엔진(715)은 레지스터 파일(730) 및 벡터 레지스터 파일(735) 등의 각종 레지스터 파일들 내의 스토리지 로케이션들에 논리 레지스터들의 개명을 제공할 뿐만 아니라, 마이크로-명령 흐름을 재정렬하고 실행에 필요한 각종 리소스들을 할당하기 위해 각종 버퍼들을 포함할 수 있다. 레지스터 파일(730)은 정수 및 부동 소수점 연산들을 위한 별개의 레지스터 파일들을 포함할 수 있다. 벡터 레지스터 파일(735)은 벡터 사이즈 유닛들, 예를 들어, 레지스터 당 256 또는 512 비트들을 위한 스토리지를 제공할 수 있다. 확장 레지스터(725) 및 비휘발성 스토리지(722) 등의 추가 스토리지들이 존재할 수 있다. 상술된 바와 같이, 확장 레지스터(725)는 숨겨진 실행 환경에서의 인증 동작들 중에 사용될 정보를 저장하는데 사용될 수 있다. 다른 정보 중에, 비휘발성 스토리지(722)는, 예를 들어, EPID 및 OwnerID를 포함하는 프로세서 식별자들 및 개인 키들을 저장할 수 있다.

[0055] 각종 리소스들은, 암호화 가속 장치 등의 다른 특별 하드웨어 중에, 예를 들어, 각종 정수, 부동 소수점, 및 SIMD(single instruction multiple data) 로직 유닛들을 포함하는, 실행 유닛들(720)에 존재할 수 있다. 결과들은 은퇴 로직, 즉, 레코더 버퍼(ROB)(740)에 제공될 수 있다. 더 구체적으로 말해서, ROB(740)는 실행되는 명령들과 연관된 정보를 수신하기 위해 각종 어레이들 및 로직을 포함할 수 있다. 이 정보는 그 후 명령들이 타당하게 은퇴할 수 있으며, 결과 데이터가 프로세서의 아키텍처 상태로 커미트(commit)될 수 있는 지를, 또는 명령들의 적합한 은퇴를 방지하는 하나의 또는 그 이상의 예외들이 발생했는 지를 결정하기 위해 ROB(740)에 의해 조사된다. 물론, ROB(740)는 은퇴와 연관된 다른 오퍼레이션들을 처리할 수 있다.

[0056] 도 10에 도시된 바와 같이, ROB(740)는, 일 실시예에서, 로우 레벨 캐시(예를 들어, L1 캐시)일 수 있는 캐시(750)에 연결되지만, 본 발명의 범위는 이와 관련하여 한정되지 않는다. 또한, 실행 유닛들(720)은 캐시(750)에 직접 연결될 수 있다. 캐시(750)로부터, 더 높은 레벨의 캐시들, 시스템 메모리 등과의 데이터 통신이 발생할 수 있다. 도 10의 실시예에서 하이 레벨로 도시되었지만, 본 발명의 범위는 이와 관련하여 한정되지 않음을 이해하라.

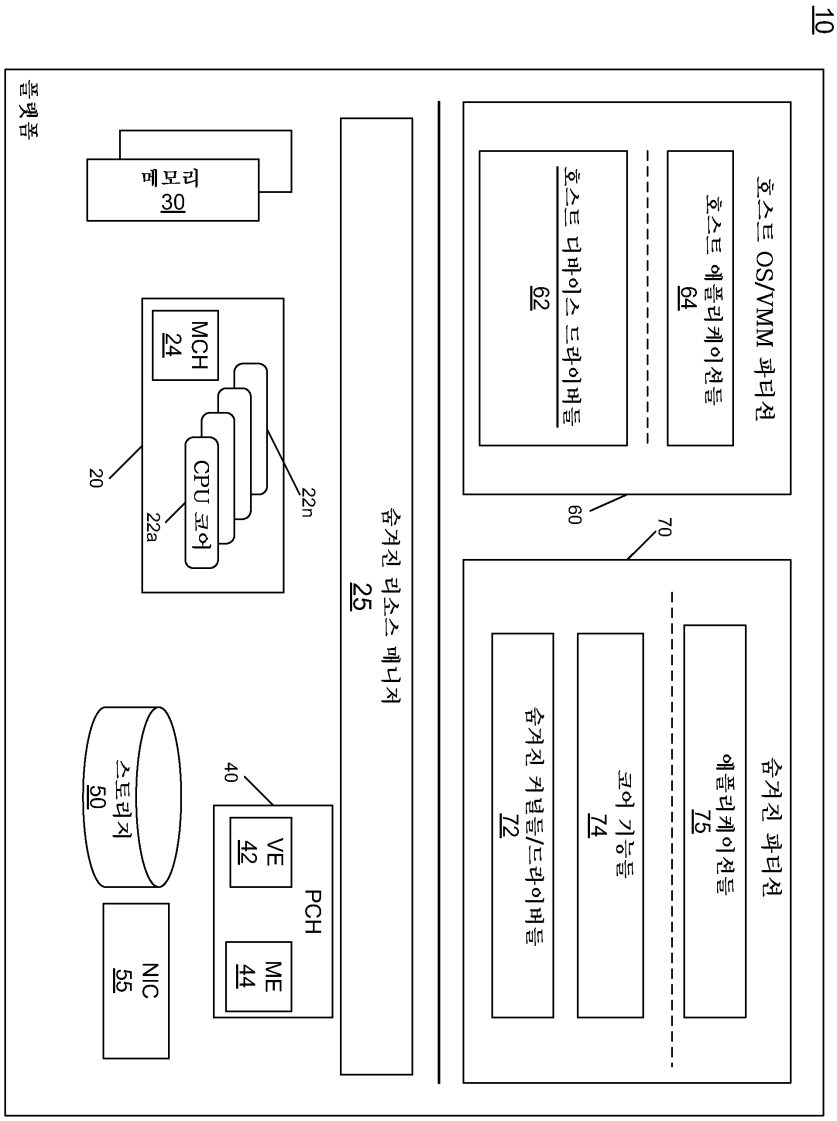
[0057] 실시예들은 코드로 구현될 수 있으며, 명령들을 실행하도록 시스템을 프로그래밍하는데 사용될 수 있는 명령들이 저장되어 있는 기억 매체에 저장될 수 있다. 기억 매체는, 플로피 디스크들, 광 디스크들, 광 디스크들, SSD들(solid state drives), CD-ROM들(compact disk read-only memories), CD-RW들(compact disk rewritables), 및 광자기 디스크들을 포함하는 임의의 타입의 디스크, ROM들(read-only memories), DRAM들(dynamic random access memories), SRAM들(static random access memories) 등의 RAM들(random access memories), EPROM들(erasable programmable read-only memories), EEPROM들(electrically erasable

programmable read-only memories), 자기 또는 광 카드들 등의 반도체 장치들, 또는 전자 명령들을 저장하기에 적합한 임의의 다른 타입의 매체를 포함할 수 있지만, 이들로 한정되지 않는다.

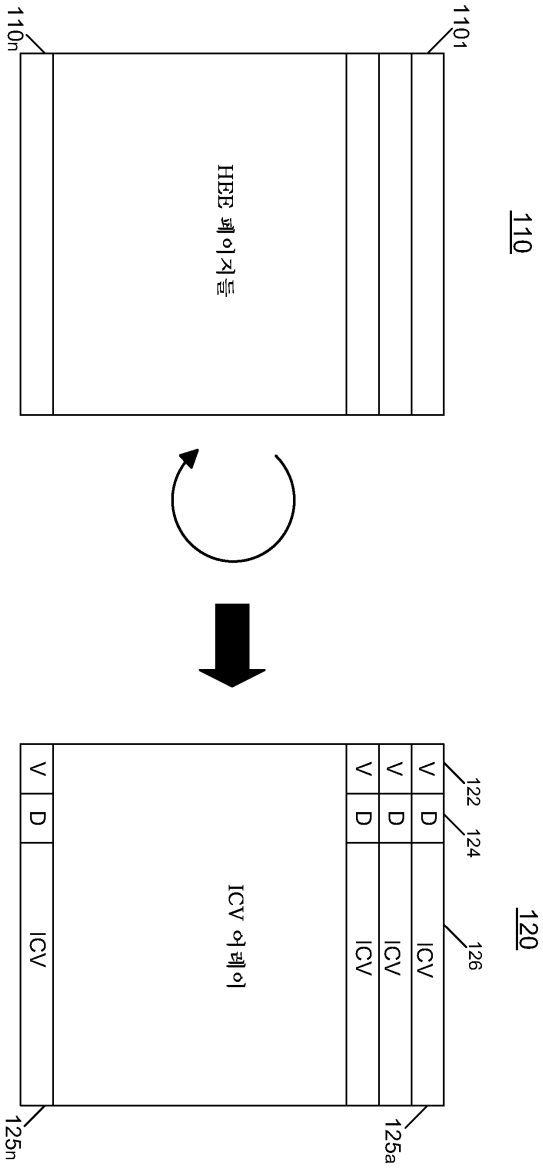
[0058] 본 발명이 제한된 수의 실시예들과 관련해서 기술되었지만, 당업자는 다수의 변경들 및 변형들을 알 것이다. 첨부된 청구항들은 본 발명의 정확한 원리 및 범위 내에 속한 모든 변경들 및 변형들을 포함한다고 의도된다.

도면

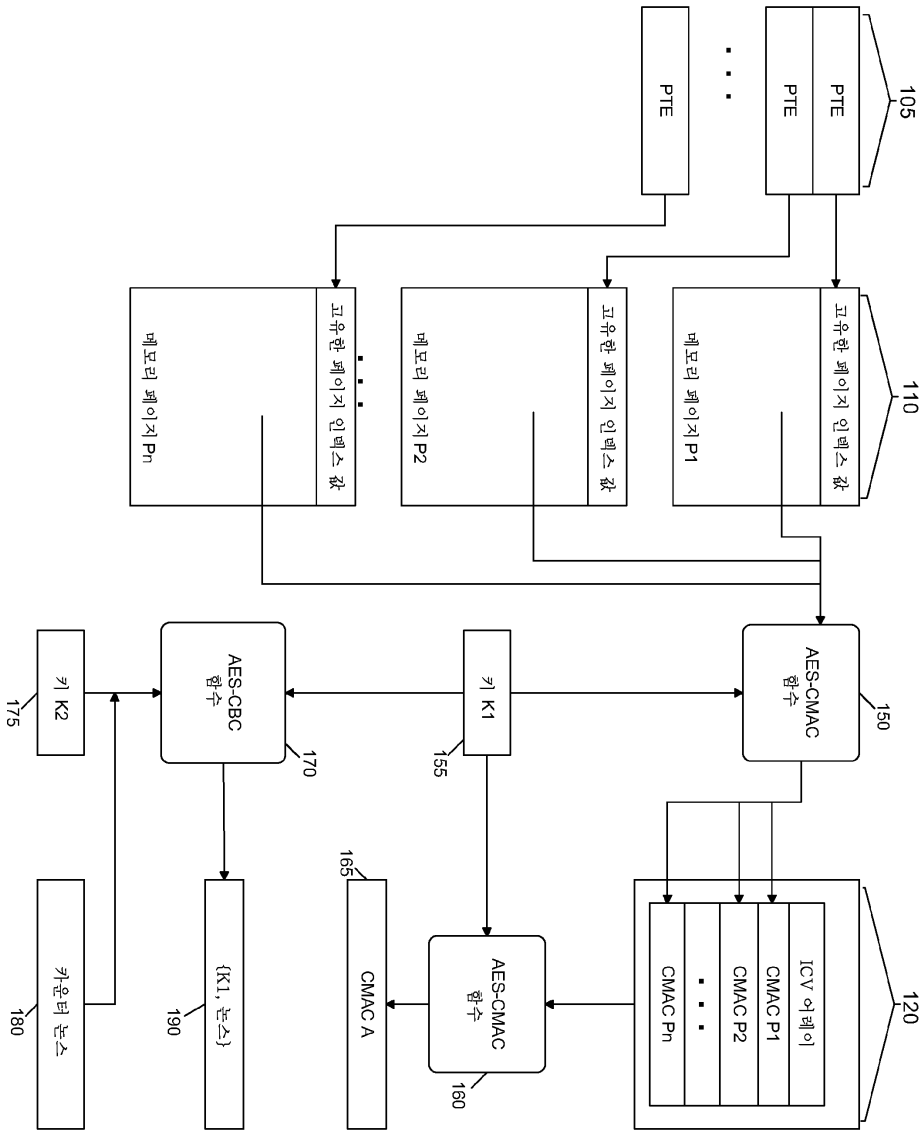
도면1



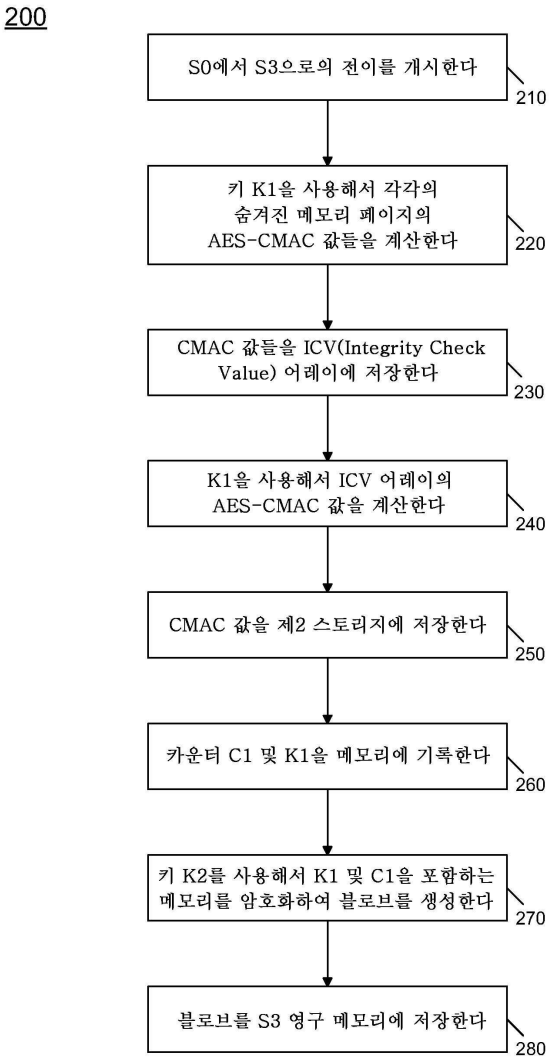
도면2



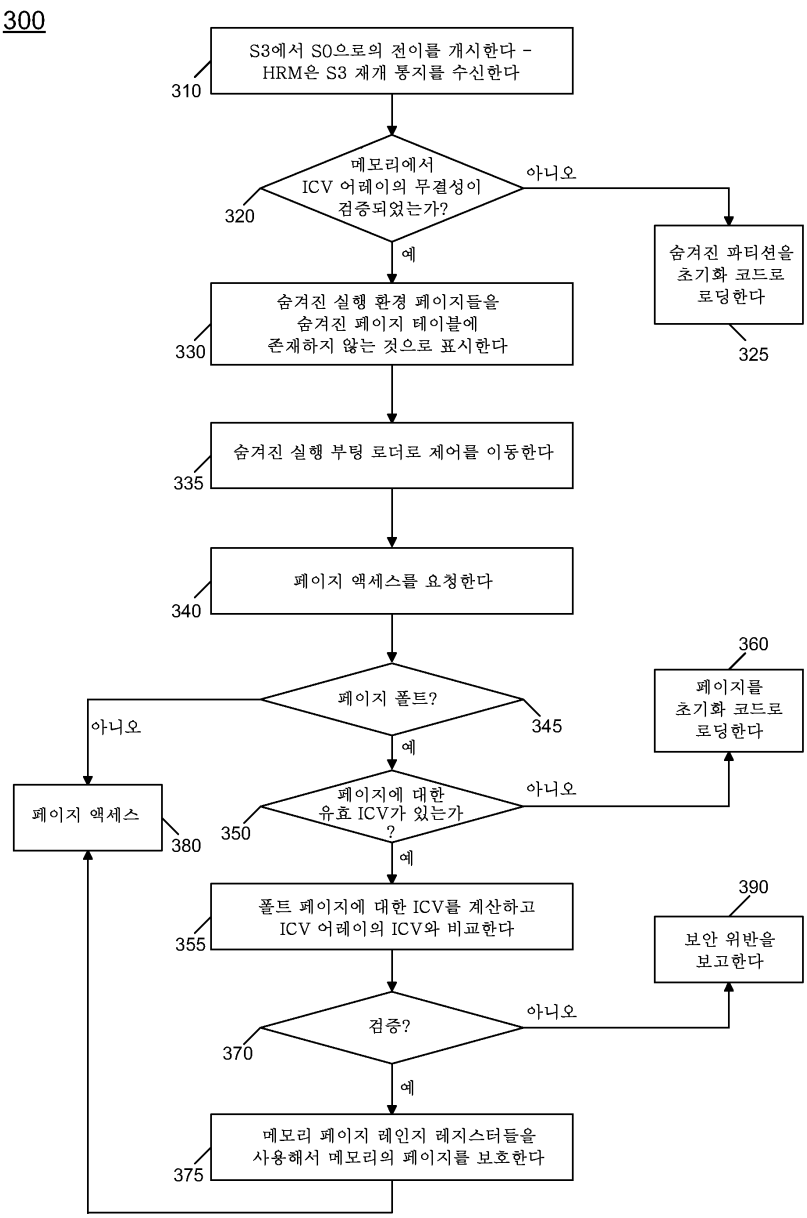
도면3


105
PTE
PTE
PTE
110
고유한 페이지 인덱스 값
메모리 페이지 P1
고유한 페이지 인덱스 값
메모리 페이지 P2
고유한 페이지 인덱스 값
메모리 페이지 Pn
150
AES-CMAC 함수
155
키 K1
170
AES-CBC 함수
175
키 K2
120
ICV 어레이
CMAC P1
CMAC P2
CMAC Pn
160
AES-CMAC 함수
165
CMAC A
180
카운터 논스
190
{K1, 논스}

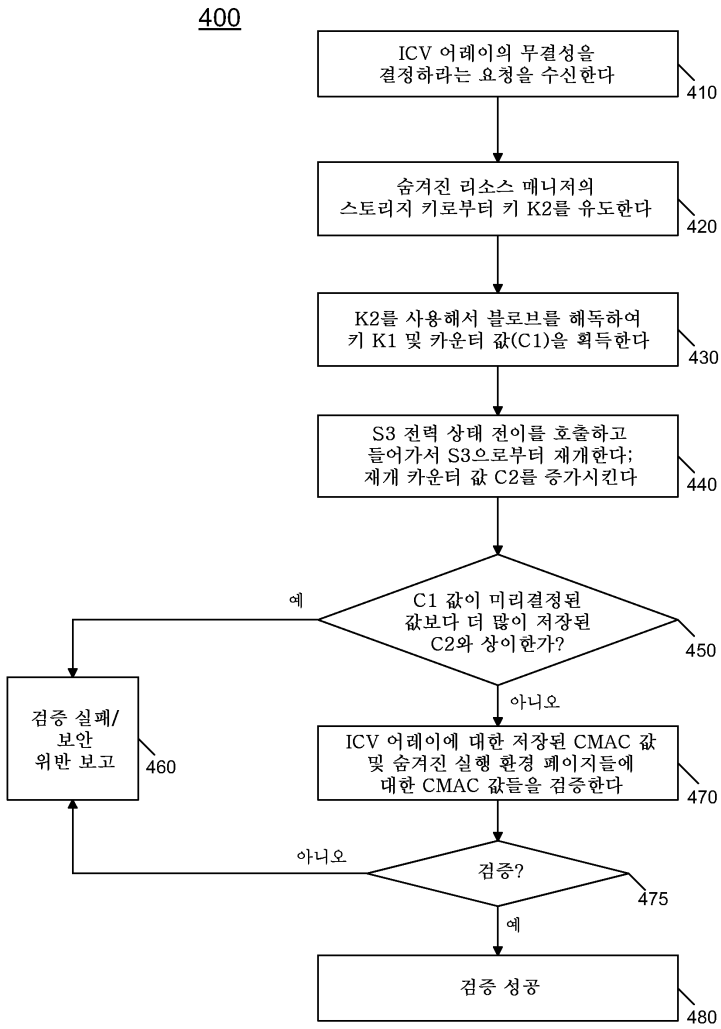
도면4


200
S0에서 S3으로의 전이를 개시한다
210
키 K1을 사용해서 각각의
숨겨진 메모리 페이지의
AES-CMAC 값들을 계산한다
220
CMAC 값들을 ICV(Integrity Check
Value) 어레이에 저장한다
230
K1을 사용해서 ICV 어레이의
AES-CMAC 값을 계산한다
240
CMAC 값을 제2 스토리지에 저장한다
250
카운터 C1 및 K1을 메모리에 기록한다
260
키 K2를 사용해서 K1 및 C1을 포함하는
메모리를 암호화하여 블로브를 생성한다
270
블로브를 S3 영구 메모리에 저장한다
280

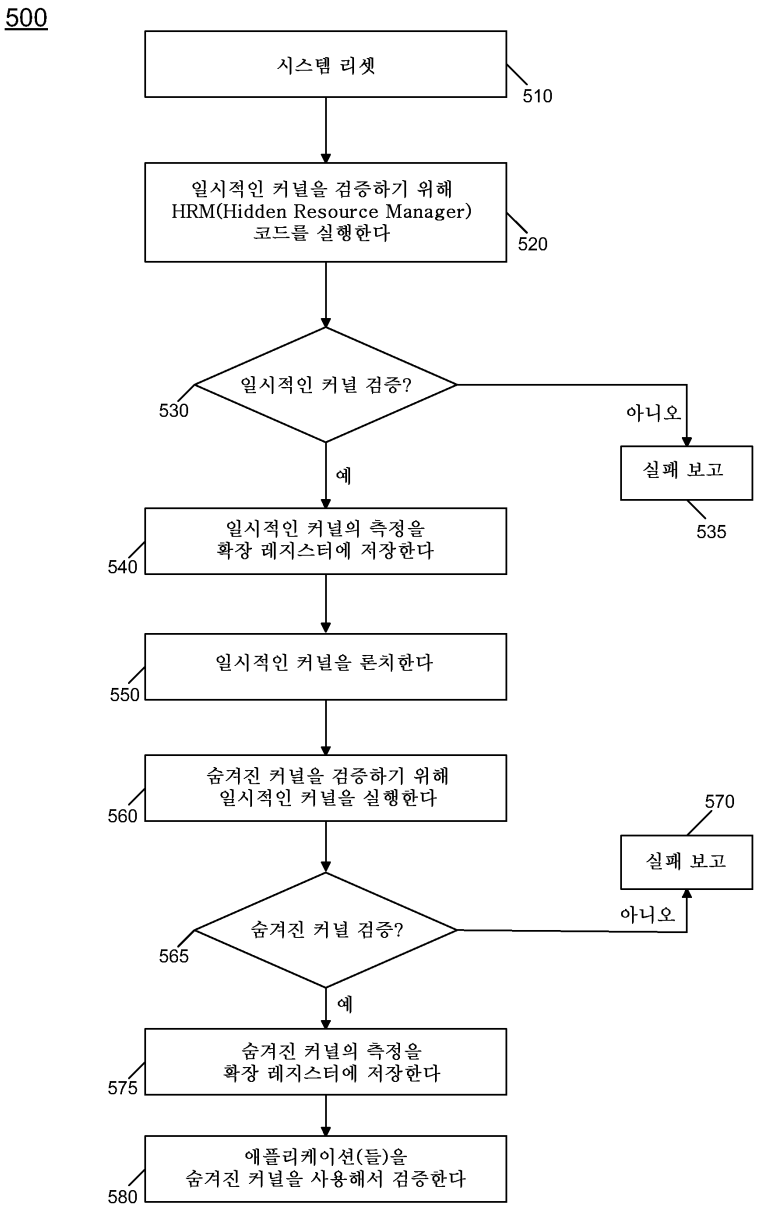
도면5


300
S3에서 S0으로의 전이를 개시한다 - HRM은 S3 재개 통지를 수신한다
310
메모리에서 ICV 어레이의 무결성이 검증되었는가?
320
아니오
예
숨겨진 파티션을 초기화 코드로 로딩한다
325
숨겨진 실행 환경 페이지들을 숨겨진 페이지 테이블에 존재하지 않는 것으로 표시한다
330
숨겨진 실행 부팅 로더로 제어를 이동한다
335
페이지 액세스를 요청한다
340
페이지 폴트?
345
아니오
예
360
페이지를 초기화 코드로 로딩한다
페이지에 대한 유효 ICV가 있는가?
350
아니오
예
페이지 액세스
380
폴트 페이지에 대한 ICV를 계산하고 ICV 어레이의 ICV와 비교한다
355
390
보안 위반을 보고한다
검증?
370
아니오
예
메모리 페이지 레인지 레지스터들을 사용해서 메모리의 페이지를 보호한다
375

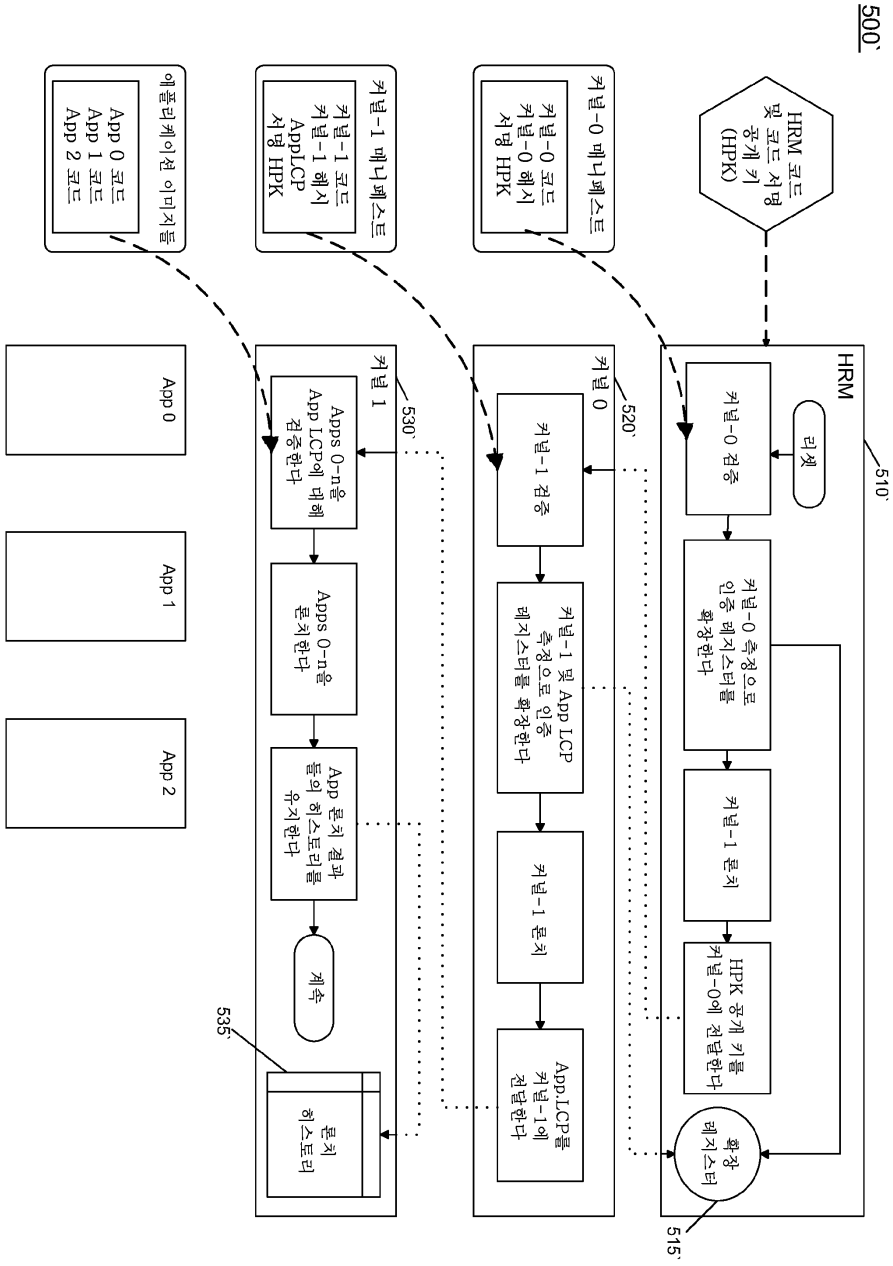
도면6


400
ICV 어레이의 무결성을 결정하라는 요청을 수신한다
410
숨겨진 리소스 매니저의 스토리지 키로부터 키 K2를 유도한다
420
K2를 사용해서 블로브를 해독하여 키 K1 및 카운터 값(C1)을 획득한다
430
S3 전력 상태 전이를 호출하고 들어가서 S3으로부터 재개한다; 재개 카운터 값 C2를 증가시킨다
440
C1 값이 미리결정된 값보다 더 많이 저장된 C2와 상이한가?
450
예
아니오
검증 실패/ 보안 위반 보고
460
ICV 어레이에 대한 저장된 CMAC 값 및 숨겨진 실행 환경 페이지들에 대한 CMAC 값들을 검증한다
470
검증?
475
아니오
예
검증 성공
480

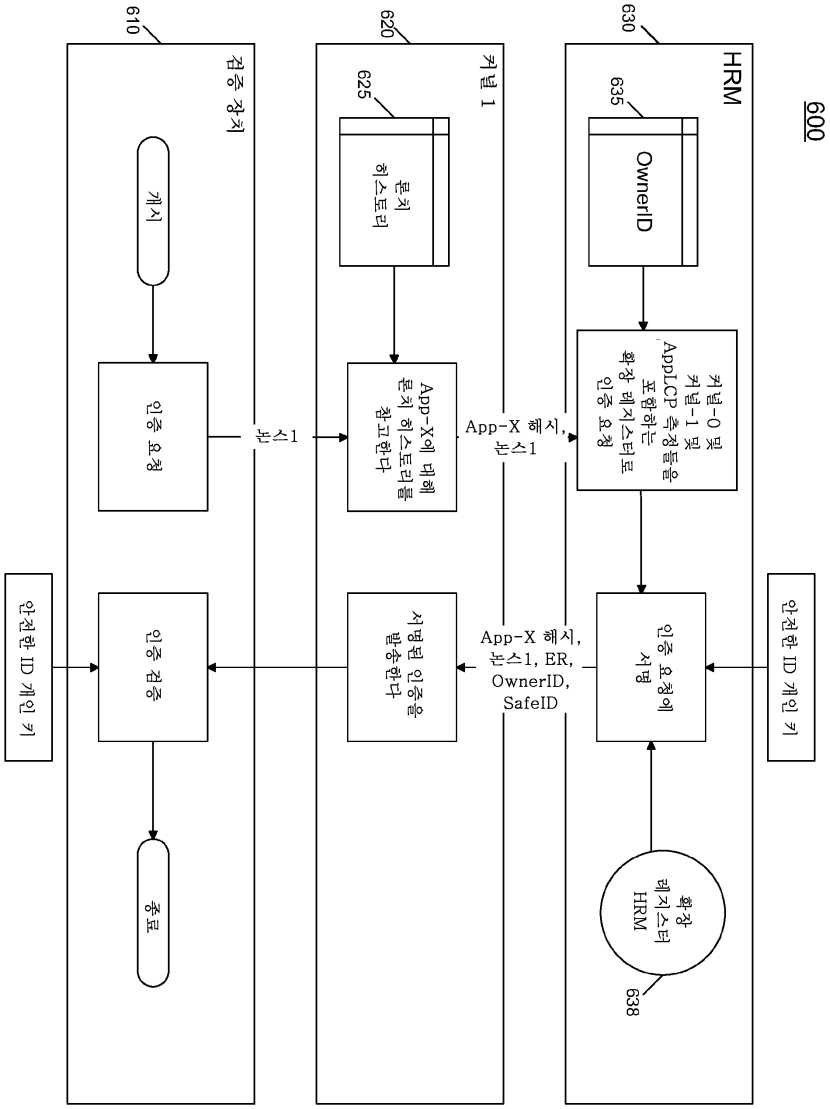
도면7


500
시스템 리셋
510
일시적인 커널을 검증하기 위해
HRM(Hidden Resource Manager)
코드를 실행한다
520
일시적인 커널 검증?
530
아니오
실패 보고
535
예
일시적인 커널의 측정을
확장 레지스터에 저장한다
540
일시적인 커널을 론치한다
550
숨겨진 커널을 검증하기 위해
일시적인 커널을 실행한다
560
570
실패 보고
숨겨진 커널 검증?
565
아니오
예
숨겨진 커널의 측정을
확장 레지스터에 저장한다
575
애플리케이션(들)을
숨겨진 커널을 사용해서 검증한다
580

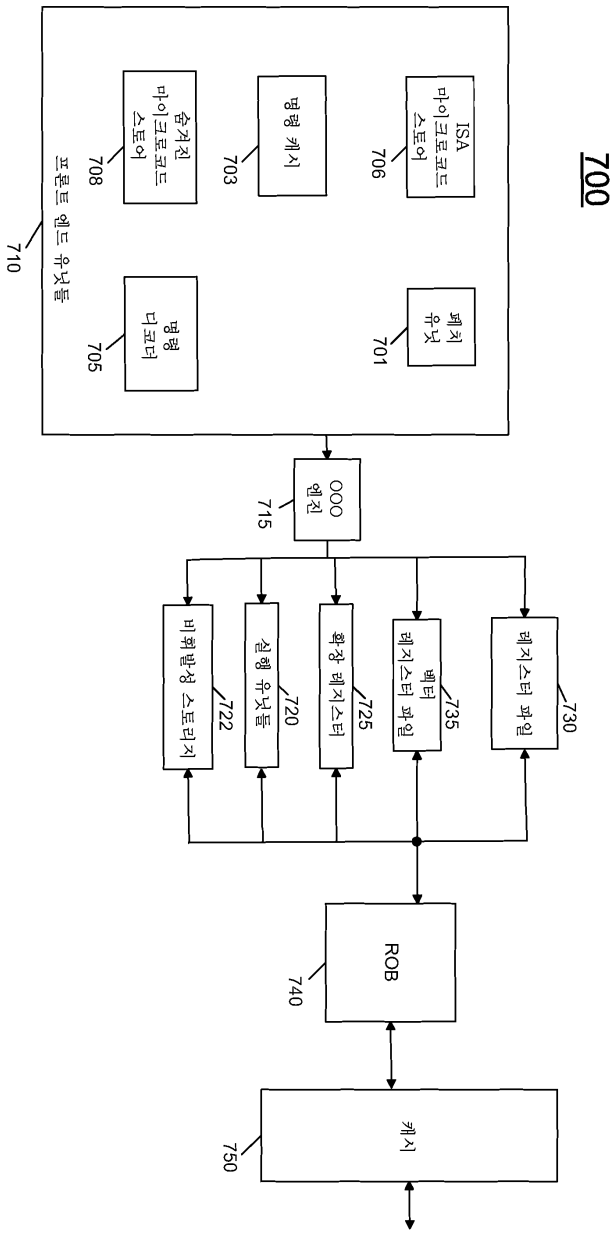
도면8


500`
HRM 코드 및 코드 서명 공개 키 (HPK)
HRM
510`
리셋
커널-0 검증
커널-0 측정으로 인증 레지스터를 확장한다
커널-1 론치
HPK 공개 키를 커널-0에 전달한다
확장 레지스터
515`
커널-0 매니페스트
커널-0 코드 커널-0 해시 서명 HPK
커널 0
520`
커널-1 검증
커널-1 및 App LCP 측정으로 인증 레지스터를 확장한다
커널-1 론치
App.LCP를 커널-1에 전달한다
커널-1 매니페스트
커널-1 코드 커널-1 해시 AppLCP 서명 HPK
커널 1
530`
Apps 0-n을 App LCP에 대해 검증한다
Apps 0-n을 론치한다
App 론치 결과들의 히스토리를 유지한다
계속
535`
론치 히스토리
애플리케이션 이미지들
App 0 코드 App 1 코드 App 2 코드
App 0
App 1
App 2

도면9


600
HRM
630
OwnerID
635
커널-0 및 커널-1 및 AppLCP 측정들을 포함하는 확장 레지스터로 인증 요청
App-X 헤시, 논스1
확장 레지스터 HRM
638
인증 요청에 서명
안전한 ID 개인 키
커널 1
620
론치 히스토리
625
App-X에 대해 론치 히스토리를 참고한다
서명된 인증을 발송한다
App-X 헤시, 논스1, ER, OwnerID, SafeID
검증 장치
610
개시
인증 요청
논스1
인증 검증
안전한 ID 개인 키
종료

도면10


700
프론트 엔드 유닛들
710
ISA 마이크로코드 스토어
706
명령 캐시
703
숨겨진 마이크로코드 스토어
708
명령 디코더
705
페치 유닛
701
OOO 엔진
715
비휘발성 스토리지
722
실행 유닛들
720
확장 레지스터
725
벡터 레지스터 파일
735
레지스터 파일
730
ROB
740
캐시
750