



(19) 대한민국특허청(KR)
(12) 등록특허공보(B1)

(45) 공고일자 2012년10월02일
(11) 등록번호 10-1187554
(24) 등록일자 2012년09월25일

(51) 국제특허분류(Int. Cl.)
G06F 21/22 (2006.01) G06F 21/20 (2006.01)
(21) 출원번호 10-2011-7021880(분할)
(22) 출원일자(국제) 2005년06월10일
심사청구일자 2011년09월19일
(85) 번역문제출일자 2011년09월19일
(65) 공개번호 10-2011-0119808
(43) 공개일자 2011년11월02일
(62) 원출원 특허 10-2007-7000788
원출원일자(국제) 2005년06월10일
심사청구일자 2010년06월07일
(86) 국제출원번호 PCT/US2005/020638
(87) 국제공개번호 WO 2005/125081
국제공개일자 2005년12월29일
(30) 우선권주장
11/061,280 2005년02월17일 미국(US)
(뒷면에 계속)
(56) 선행기술조사문헌
KR1020020019562 A*
*는 심사관에 의하여 인용된 문헌

(73) 특허권자
마이크로소프트 코포레이션
미국 워싱턴주 (우편번호 : 98052) 레드몬드 원
마이크로소프트 웨이
(72) 발명자
베넷, 줄리, 디.
미국 98039 워싱턴주 메디나 에버그린 포인트 로
드 2203
우, 웨이
미국 98052 워싱턴주 레드몬드 173번 플레이스
노스이스트 3015
(뒷면에 계속)
(74) 대리인
제일특허법인

전체 청구항 수 : 총 10 항

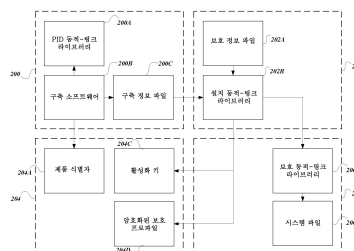
심사관 : 엄인권

(54) 발명의 명칭 소프트웨어 보안

(57) 요약

소프트웨어 일부분의 변조가 방지된다. 소프트웨어 일부분을 역설계하는 해커의 이해를 혼란스럽게 하기 위한 소프트웨어 이미지의 식별곤란화가 제공된다. 소프트웨어 일부분들이 함께 변조로부터의 보호를 요하는 소프트웨어 패키지를 구성하는지 검사하는 처리 또한 제공된다. 컴퓨팅 기기 상의 CPU 또는 캐쉬 메모리 등의 하드웨어 자원이 소프트웨어 변조를 방지하는 것이 가능한 종류에 속하는지를 판정하는 다른 처리 또한 제공된다. 스레드 보호는 보호적인 소프트웨어 일부분을 실행하는 다양한 스레드를 가지고 변조를 방지한다. 서비스 보호는 다양한 권한이 없는 서비스가 실행될 수 있게 하는 변조를 방지한다. 설치 구축 보호는 다양한 소프트웨어 일부분의 실행 중에 변조를 방지한다. 소프트웨어 식별곤란화는 함수 이름을 조작하고 다양한 소프트웨어 일부분에 점프 명령어를 추가하는 해커에 의한 역설계를 방지한다. 하드웨어 보호는 해커가 다양한 소프트웨어 일부분이 권한이 없는 하드웨어 일부분에서 실행되도록 강요하는 것을 방지한다. 프로파일은 중앙 계층 데이터베이스에 저장되고 이러한 프로파일은 변조로부터 보호된다. 소프트웨어 이미지를 포함하는 소프트웨어 일부분을 역설계 하는 해커의 이해를 혼란스럽게 하기 위한 소프트웨어 이미지의 식별곤란화가 제공된다. 이미지 검사는 설치될 소프트웨어 일부분의 진품 여부를 보장한다.

대표도 - 도2



(72) 발명자

추, 프랑크, 니안-트주

미국 98075 워싱턴주 삼마미쉬 3124-235번 플레이스 사우스이스트

포춘, 에릭

미국 피엠비 471 스위트 엠. 레드몬드 웨이 16625

엘-감말, 모하메드

미국 98072 워싱턴주 우던빌 128번 코트 노스이스트 15509

언샤우, 시몬, 디.

미국 98112 워싱턴주 시애틀 23번 애비뉴 이스트 1128

스와나수크, 레르크분

미국 98004 워싱턴주 벨레뷰 112-111번 애비뉴 사우스이스트

(30) 우선권주장

11/061,399 2005년02월17일 미국(US)

11/061,411 2005년02월17일 미국(US)

11/061,484 2005년02월17일 미국(US)

11/061,790 2005년02월17일 미국(US)

11/061,791 2005년02월17일 미국(US)

11/061,792 2005년02월17일 미국(US)

11/062,237 2005년02월17일 미국(US)

11/062,238 2005년02월17일 미국(US)

60/578,937 2004년06월12일 미국(US)

특허청구의 범위

청구항 1

컴퓨팅 기기(computing machine) 상에서 실행되는 소프트웨어의 변조(tampering)를 방지하기 위한 방법으로서,

권한이 없는 서비스의 실행을 방지하기 위해 식별 곤란화 처리된(obfuscated) 보호 동적-링크 라이브러리의 보호 함수를 호출하는 단계; 및

중앙 계층 데이터베이스의 키들 및 값들을 보호하기 위해 상기 식별 곤란화 처리된 보호 동적-링크 라이브러리의 또 다른 보호 함수를 호출하는 단계

를 포함하는 방법.

청구항 2

제1항에 있어서,

로그-온 모듈을 실행하는 단계를 더 포함하며,

상기 로그-온 모듈은, 상기 소프트웨어가 분류된 제품군(product class)에 대한 보호가 행해질 수 있는 경우에 상기 식별 곤란화 처리된 보호 동적-링크 라이브러리를 로딩하는 방법.

청구항 3

제2항에 있어서,

상기 식별 곤란화 처리된 보호 동적-링크 라이브러리를 로딩하는 동작이 실패하는 경우, 상기 컴퓨팅 기기를 종료하는 단계를 더 포함하는 방법.

청구항 4

제1항에 있어서,

상기 권한이 없는 서비스의 실행을 방지하기 위해 식별 곤란화 처리된 보호 동적-링크 라이브러리의 보호 함수를 호출하는 단계는, 허가된 서비스의 리스트를 가지고 검사를 행하여 상기 권한이 없는 서비스의 실행이 제거되게 하는 스레드를 생성하는 방법.

청구항 5

제1항에 있어서,

상기 중앙 계층 데이터베이스의 키들 및 값들을 보호하기 위해 상기 식별 곤란화 처리된 보호 동적-링크 라이브러리의 또 다른 보호 함수를 호출하는 단계는, 시스템 환경(system context)에서 키들 및 값들을 보호하기 위한 스레드를 생성하는 단계를 포함하는 방법.

청구항 6

제1항에 있어서,

상기 중앙 계층 데이터베이스의 키들 및 값들을 보호하기 위해 상기 식별 곤란화 처리된 보호 동적-링크 라이브러리의 또 다른 보호 함수를 호출하는 단계는, 사용자 환경(user context)에서 키들 및 값들을 보호하기 위한 스레드를 생성하는 단계를 포함하는 방법.

청구항 7

컴퓨팅 시스템의 하나 이상의 컴포넌트에 의해 실행되는 경우, 컴퓨팅 시스템 상에서 실행되는 소프트웨어의 변조를 방지하기 위한 동작들을 수행하는 컴퓨터-실행가능 명령어들을 기억하는 하나 이상의 컴퓨터-판독가능 기억 매체로서,

상기 동작들은,

권한이 없는 서비스의 실행을 방지하기 위해 식별 곤란화 처리된 보호 동적-링크 라이브러리의 보호 함수를 호출하는 동작; 및

중앙 계층 데이터베이스의 키들 및 값들을 보호하기 위해 상기 식별 곤란화 처리된 보호 동적-링크 라이브러리의 또 다른 보호 함수를 호출하는 동작

을 포함하는 하나 이상의 컴퓨터-판독가능 기억 매체.

청구항 8

제7항에 있어서,

상기 동작들은, 상기 소프트웨어가 분류된 제품군에 대한 보호가 행해질 수 있는 경우에 상기 식별 곤란화 처리된 보호 동적-링크 라이브러리를 로딩하도록 로그-온 모듈을 구현하는 동작을 더 포함하는 하나 이상의 컴퓨터-판독가능 기억 매체.

청구항 9

제8항에 있어서,

상기 동작들은, 상기 식별 곤란화 처리된 보호 동적-링크 라이브러리를 로딩하는 동작이 실패하는 경우에 상기 컴퓨팅 시스템을 종료하는 동작을 더 포함하는 하나 이상의 컴퓨터-판독가능 기억 매체.

청구항 10

제7항에 있어서,

상기 권한이 없는 서비스의 실행을 방지하기 위해 식별 곤란화 처리된 보호 동적-링크 라이브러리의 보호 함수를 호출하는 동작은, 허가된 서비스의 리스트를 가지고 검사를 행하여 상기 권한이 없는 서비스의 실행이 제거되게 하는 스레드를 생성하는 하나 이상의 컴퓨터-판독가능 기억 매체.

명세서

기술 분야

[0001] 본 발명은 일반적으로 보안에 관한 것으로, 보다 상세히는, 권한이 없는 개인 또는 권한이 없는 소프트웨어 일부분(piece)에 의한 변경을 차단함으로써 소프트웨어 변조(tampering; 變造)를 방지하는 것에 관한 것이다.

배경 기술

[0002] 소프트웨어는 컴퓨팅 기기를 강력하게 만든다. 이러한 기기는 사람의 결함이 있는 심장의 부정맥을 바로 잡고 사람이 천체의 위성군을 이해하게 할 수 있다. 그러나, 소프트웨어는 우발적인 장난 또는 의도적인 해악과 같은 간단한 것들에 취약하다. 우발적인 장난은 아이가 어떠한 방법으로 그의 부모의 퍼스널 컴퓨터에 접근하게 됨으로써 모르는 사이에 발행할 수 있으며, 이는 다른 예들 중에서도 특히, 데이터의 물리적 손실을 일으키거나 설정을 변경시켜 컴퓨팅 기기의 사용에 해로운 영향을 미친다. 의도적인 해악은 통상적으로 불법적인 목적으로 컴퓨터에 관한 전문 기술을 이용하는 사람들에게 대한 위악 어법의 표현인 "해커"에 의해, 예를 들면, 컴퓨팅 기기 상에 악성 소프트웨어가 실행되게 하거나 허가 없이 직접적으로 컴퓨팅 기기에 접근하여 프로그램과 데이터를 변조함으로써 발생한다.

[0003] 운영 체제는 메모리, CPU 시간, 디스크 공간, 및 주변 기기와 같은 컴퓨팅 기기 자원의 할당 및 이용을 제어하는 소프트웨어이다. 운영 체제는 애플리케이션이 의존하는 기반 소프트웨어이다. 인기있는 운영 체제는 윈도우즈 98, 윈도우즈 NT, 윈도우즈 XP, 맥 OS, 유닉스, 및 리눅스를 포함한다. 운영 체제는 때때로 특정 시장에 적절한 방식으로 패키징된다. 예를 들면, 작은 틈새 서버 시장에 이용되는 강력한 운영 체제는 소프트웨어 제조업체가 일반적인 대형 소비자 시장에서의 미숙한 사용자에게 적절한 다양한 방식으로 보안을 할 수 있다. 한가지 문제점은 미숙한 사용자들이 이렇게 보안을 한 운영 체제를 부주의로 인해 변경하여 심하게 손상을 입힐 수 있다는 것이다. 그러나 무엇보다 가장 치명적인 문제점은 해커들이 이 보안을 한 운영 체제를 다른 불법 목적에 이용하도록 변형시키기 위해 역설계(reverse engineer)함으로써 소프트웨어 제조업체가 손

해를 볼 수 있다는 점이다. 도 1은 이러한 문제점과 다른 문제점을 매우 상세히 도시한다.

[0004] 소프트웨어 이미지(108)는 컴퓨터 하드웨어를 동작하게 하는 명령어를 포함하는 운영 체제의 복제본 또는 사본을 나타낸다. 통상적으로 소프트웨어 이미지(108)는 컴퓨터 하드웨어의 어딘가에 저장된 파일이기 때문에 해커 또는 그의 악성 소프트웨어(102)가 소프트웨어 이미지(108)를 변경하거나 쉽게 교체될 수 있게 할 수 있다. 사용자(110)가 운영 체제 등의 시스템 소프트웨어를 실행하기 위해 소프트웨어 이미지를 호출할 때에, 변경되거나 교체된 소프트웨어 이미지가 소프트웨어 제조업체가 제공한 원본 대신에 실행된다.

[0005] 소프트웨어 이미지(108)의 변조는 통상적으로 해커 또는 악성 소프트웨어(102) 일부분들에 의해 수행되고, 사용자(110)에 의해서는 거의 수행되지 않는다. 그러나, 레지스트리(106)는 해커 또는 악성 소프트웨어(102) 일부분들 뿐 아니라 사용자(110)에 의해 의도적이지 않게 변조될 수 있다. 레지스트리(106)는 하나 이상의 사용자, 애플리케이션, 및 하드웨어 장치에 대하여 시스템을 구성하는 데에 채용될 수 있는 정보를 저장하는데 이용되는 시스템 소프트웨어 일부분이다. 예를 들면, 레지스트리는 사용자 컴퓨팅 기기에 대한 하드웨어 속도 향상 지원 및 3차원적인 렌더링은 활성화하면서, 서버 컴퓨팅 기기에 대해서는 이들 동일한 특징들을 비활성화하는 데에 이용될 수 있다.

발명의 내용

해결하려는 과제

[0006] 이들 정보의 일부분은 사용자(110)가 관리자 권한을 가지고 작업할 때 또는 부적절하게 레지스트리를 변경하는 권한을 얻은 해커 또는 악성 소프트웨어(102) 일부분들에 의해 변경될 수 있다. 해커 또는 악성 소프트웨어(102) 일부분들은 레지스트리(106) 조작을 시도하여 특정 청중 또는 마케팅 채널은 이용할 수 없는 레지스트리, 레지스트리 설정, 및 잠금 해지(unlock) 추가적인 특징의 정보를 변경하도록 하기 위한 라이선싱 계약을 극복할 수 있다. 한 가지 문제는 레지스트리를 변경함으로써 컴퓨팅 기기가 동작을 멈추거나 예측할 수 없는 행위를 할 수 있게 된다는 것이다.

[0007] 다른 문제는 실행되는 소프트웨어(104)를 변조하는 것을 포함한다. 해커, 또는 악성 소프트웨어(102) 일부분들은 적절하게 실행되는 소프트웨어(104)를 부적절하게 폐기하고 권한이 없거나 금지된 소프트웨어 서비스로 대체할 수 있다. 또한, 해커 또는 악성 소프트웨어(102) 일부분들은 소프트웨어 응답 또는 소프트웨어 호출을 모방하고 실행되는 소프트웨어(104)의 실행을 변조할 수 있다.

[0008] 우발적인 장난이나 의도적인 해악 모두에 의한, 소프트웨어 변조의 문제점이 발생하면, 사악한 해커 또는 이들의 악성 소프트웨어가 소프트웨어를 취약하게 할 수 있다는 것은 놀라운 일이 아니게 될 것이다. 소프트웨어 변조의 문제점을 해결하지 않는다면, 결국 사용자는 권한이 없는 개인에 의한 접근을 방지하면서 보안 컴퓨팅 체험을 제공하는 컴퓨터 제조업체를 더 이상 신뢰하지 않을 수 있다. 그러므로, 기존의 시스템에 관련된 상술한 문제점을 피하거나 줄이면서 소프트웨어 보안을 위한 시스템, 방법, 및 컴퓨터-판독가능 매체가 필요하다.

과제의 해결 수단

[0009] 본 발명에 따르면, 소프트웨어 변조를 방지하기 위한 시스템, 방법, 및 컴퓨터-판독가능 매체가 제공된다. 본 발명의 시스템 형태는 컴퓨팅 기기 상에 실행되는 소프트웨어 일부분의 변조(tampering)를 방지하기 위한 시스템을 포함한다. 본 발명의 시스템은 권한이 없는 서비스의 실행을 방지하기 위해 식별이 어려운(obfuscated) 보호 동적-링크 라이브러리의 보호 함수를 호출하기 위한 수단을 포함한다. 본 발명의 시스템은 중앙 계층 데이터베이스의 키 및 값을 보호하기 위해 식별이 어려운 보호 동적-링크 라이브러리의 다른 보호 함수를 호출하기 위한 수단을 더 포함한다.

[0010] 본 발명의 다른 양태에 따르면, 본 발명의 방법은 컴퓨팅 기기 상에 실행되는 소프트웨어 일부분의 변조를 방지하기 위한 방법을 포함한다. 본 발명의 방법은 권한이 없는 서비스의 실행을 방지하기 위해 식별이 어려운 보호 동적-링크 라이브러리의 보호 함수를 호출하는 단계를 포함한다. 본 발명의 방법은 중앙 계층 데이터베이스의 키 및 값을 보호하기 위해 식별이 어려운 보호 동적-링크 라이브러리의 다른 보호 함수를 호출하는 단계를 더 포함한다.

[0011] 본 발명의 다른 양태에 따르면, 본 발명의 컴퓨터-판독가능 형태는 컴퓨팅 기기 상에 실행되는 소프트웨어 일부분의 변조를 방지하는 방법을 구현하기 위한 컴퓨터-실행가능 명령어를 가지는 컴퓨터-판독가능 매체를 포

함한다. 이 방법은 권한이 없는 서비스의 실행을 방지하기 위해 식별이 어려운 보호 동적-링크 라이브러리의 보호 함수를 호출하는 단계를 포함한다. 또한 이 방법은 중앙 계층 데이터베이스의 키 및 값을 보호하기 위해 식별이 어려운 보호 동적-링크 라이브러리의 다른 보호 함수를 호출하는 단계를 더 포함한다

[0012] 본 발명에 따르면, 소프트웨어 변조를 방지하기 위한 시스템, 방법, 및 컴퓨터-판독가능 매체가 제공된다. 본 발명의 컴퓨터-판독가능 매체 형태는 컴퓨팅 시스템이 정보의 일부분들의 진품 여부를 검사하는 데에 이용하는 컴퓨터-실행가능 명령어 및 하나 이상의 데이터 구조를 가지는 컴퓨터-판독가능 매체를 포함한다. 컴퓨터 판독가능 매체는 보호 동적-링크 라이브러리의 개인 라이브러리 키로 암호화된 프로파일을 포함하는데, 이 프로파일은 프로파일 유형에 의존하는 포맷 및 사이즈를 가지는 프로파일 데이터를 포함한다. 프로파일은 또한 보호 동적-링크 라이브러리의 공개 라이브러리 키로 디지털적으로 서명된 시그니처(signature)를 더 포함하며, 프로파일은 프로파일을 식별하는 식별자와 프로파일 데이터의 체크섬(checksum)을 포함하는 검증 블랍(blob)을 더 포함한다.

[0013] 본 발명의 다른 양태에 따르면, 본 발명의 방법은 소프트웨어 일부분 간의 상호동작을 보호하기 위한 컴퓨터-구현 방법을 포함한다. 컴퓨터-구현 방법은 공개 호출 키 및 개인 호출 키를 가지는 제1 소프트웨어 일부분에 의해 중앙 계층 데이터베이스에 저장되는 프로파일로의 액세스를 개시하는 단계를 포함한다. 상기 제1 소프트웨어 일부분은 임의의 솔트 값(random salt value)을 송신한다. 이 방법은 임의의 솔트 값을 수신한 이후에, 개인 라이브러리 키를 가지는 제2 소프트웨어 일부분에 의해 상기 제2 소프트웨어 일부분이 상기 중앙 계층 데이터베이스에서 상기 프로파일을 검색하고 상기 개인 라이브러리 키를 이용하여 상기 프로파일을 복호화하는 단계를 더 포함한다.

[0014] 본 발명의 다른 양태에 따르면, 본 발명의 컴퓨터-판독가능 형태는 소프트웨어 일부분 간의 상호동작을 보호하기 위한 컴퓨터-구현 방법을 포함한다. 컴퓨터-구현 방법은 공개 호출 키 및 개인 호출 키를 가지는 제1 소프트웨어 일부분에 의해 중앙 계층 데이터베이스에 저장되는 프로파일로의 액세스를 개시하는 단계를 포함한다. 상기 제1 소프트웨어 일부분은 임의의 솔트 값을 송신한다. 컴퓨터-구현 방법은 임의의 솔트 값을 수신한 이후에, 개인 라이브러리 키를 가지는 제2 소프트웨어 일부분에 의해 제2 소프트웨어 일부분이 상기 중앙 계층 데이터베이스에서 상기 프로파일을 검색하고 상기 개인 라이브러리 키를 이용하여 상기 프로파일을 복호화하는 단계를 포함한다.

[0015] 본 발명의 다른 양태에 따르면, 본 발명의 시스템 형태는 소프트웨어 일부분 간의 상호동작을 보호하기 위한 시스템을 포함한다. 시스템은 공개 호출 키 및 개인 호출 키를 가지는 제1 소프트웨어 일부분에 의해 중앙 계층 데이터베이스에 저장되는 프로파일로의 액세스를 개시하기 위한 수단을 포함하며, 상기 제1 소프트웨어 일부분은 임의의 솔트 값을 송신한다. 시스템은 임의의 솔트 값을 수신한 이후에, 개인 라이브러리 키를 가지는 제2 소프트웨어 일부분에 의해 상기 제2 소프트웨어 일부분이 중앙 계층 데이터베이스에서 프로파일을 검색하고 개인 라이브러리 키를 이용하여 프로파일을 복호화하기 위한 수단을 더 포함한다.

[0016] 본 발명에 따르면, 소프트웨어 변조를 방지하기 위한 시스템, 방법, 및 컴퓨터-판독가능 매체가 제공된다. 본 발명의 방법은 컴퓨팅 기기 상에 실행되는 소프트웨어 일부분의 변조를 방지하기 위한 방법을 포함한다. 이 방법은 소프트웨어 일부분을 특징짓는 제품 종류에 대한 보호가 유용한 경우 컴퓨팅 기기 상에 식별이 어려운 보호 동적-링크 라이브러리를 로딩하는 단계를 포함한다. 이 방법은 컴퓨팅 기기 상에 실행되는 소프트웨어 일부분이, 변조로부터의 보호가 유용한 제품 종류에 여전히 존재하는지를 모니터링하기 위한 시스템 타이머를 생성하는 단계를 더 포함한다.

[0017] 본 발명의 다른 양태에 따르면, 본 발명의 컴퓨터-판독가능 매체 형태는 컴퓨팅 기기 상에 실행되는 소프트웨어 일부분의 변조를 방지하기 위한 방법을 포함한다. 이 방법은 소프트웨어 일부분을 특징짓는 제품 종류에 대한 보호가 유용한 경우 컴퓨팅 기기 상에 식별이 어려운 보호 동적-링크 라이브러리를 로딩하는 단계를 포함한다. 이 방법은 컴퓨팅 기기 상에 실행되는 소프트웨어 일부분이, 변조로부터의 보호가 유용한 제품 종류에 여전히 존재하는지를 모니터링하기 위한 시스템 타이머를 생성하는 단계를 더 포함한다.

[0018] 본 발명의 다른 양태에 따르면, 본 발명의 방법은 컴퓨팅 기기 상에 실행되는 소프트웨어 일부분의 변조를 방지하기 위한 방법을 포함한다. 이 방법은 컴퓨팅 기기 상에 식별이 어려운 보호 동적-링크 라이브러리의 보호 함수를 호출함으로써 멀티 실행 스레드를 생성하는 단계를 포함한다. 멀티 스레드 중 하나는 권한이 없는 서비스의 실행을 방지하고, 다른 멀티 스레드들은 중앙 계층 데이터베이스의 키 및 값을 보호한다. 이 방법은 멀티 스레드가 실행되고 있는지를 모니터링하기 위한 타이머를 생성하는 단계를 더 포함한다.

[0019] 본 발명의 다른 양태에 따르면, 본 발명의 컴퓨터-판독가능 매체 형태는 컴퓨팅 기기 상에 실행되는 소프트웨어

어 일부분의 변조를 방지하는 방법을 구현하기 위한 컴퓨터-실행가능 명령어가 저장된 컴퓨터-판독가능 매체를 포함한다. 이 방법은 컴퓨팅 기기 상에 식별이 어려운 보호 동적-링크 라이브러리의 보호 함수를 호출함으로써 멀티 실행 스레드를 생성하는 단계를 포함한다. 멀티 스레드 중 하나는 권한이 없는 서비스의 실행을 방지하고, 다른 멀티 스레드들은 중앙 계층 데이터베이스의 키 및 값을 보호한다. 이 방법은 멀티 스레드가 실행되고 있는지를 모니터링하기 위한 타이머를 생성하는 단계를 더 포함한다.

[0020] 본 발명에 따르면, 소프트웨어 변조를 방지하기 위한 시스템, 방법, 및 컴퓨터-판독가능 매체가 제공된다. 본 발명의 방법은 서비스 변조를 방지하기 위한 변경을 포함한다. 이 방법은 컴퓨팅 기기를 시동할 때, 로그-온 모듈에 의해 권한이 없는 서비스의 실행을 방지하는 스레드를 생성하기 위해 식별이 어려운 보호 동적-링크 라이브러리의 보호 함수를 호출하는 단계를 포함한다. 이 방법은 식별이 어려운 보호 동적-링크 라이브러리와 통신하여 권한이 없는 서비스들의 리스트에 관련된 암호화된 보호 프로파일을 액세스하는 단계를 더 포함한다.

[0021] 본 발명의 다른 양태에 따르면, 본 발명의 컴퓨터-판독가능 형태는 서비스 변조를 방지하는 방법을 구현하기 위한 컴퓨터-실행가능 명령어가 저장된 컴퓨터-판독가능 매체를 포함한다. 상기 방법은 컴퓨팅 기기를 시동할 때, 로그-온 모듈에 의해 권한이 없는 서비스의 실행을 방지하는 스레드를 생성하기 위해 식별이 어려운 보호 동적-링크 라이브러리의 보호 함수를 호출하는 단계를 포함한다. 이 방법은 식별이 어려운 보호 동적-링크 라이브러리와 통신하여 권한이 없는 서비스들의 리스트에 관련된 암호화된 보호 프로파일을 액세스하는 단계를 더 포함한다.

[0022] 본 발명의 다른 양태에 따르면, 본 발명의 컴퓨터-판독가능 형태는 컴퓨팅 기기 상에 실행되는 소프트웨어 일부분의 변조를 방지하는 방법을 구현하기 위한 컴퓨터-실행가능 명령어를 가지는 컴퓨터-판독가능 매체를 포함한다. 이 방법은 권한이 없는 서비스의 실행을 방지하기 위해 식별이 어려운 보호 동적-링크 라이브러리의 보호 함수를 호출하는 단계를 포함한다. 이 방법은 중앙 계층 데이터베이스의 키 및 값을 보호하기 위해 식별이 어려운 보호 동적-링크 라이브러리의 다른 보호 함수를 호출하는 단계를 더 포함한다.

[0023] 본 발명에 따르면, 소프트웨어 변조를 방지하기 위한 시스템, 방법, 및 컴퓨터-판독가능 매체가 제공된다. 본 발명의 컴퓨터-판독가능 매체 형태는 컴퓨팅 시스템이 소프트웨어 일부분을 설치하는 데에 이용하는 컴퓨터-판독가능 명령어가 저장된 컴퓨터-판독가능 매체를 포함한다. 컴퓨터-판독가능 매체는 중앙 계층 데이터베이스가 변조되었는지 모니터링하고, 컴퓨팅 시스템에 저장된 파일 세트의 진품 여부를 검사하고, 권한이 없는 서비스를 실행하는 것을 중지시키는 컴퓨터-판독가능 명령어를 포함하는 보호 동적-링크 라이브러리를 포함한다.

[0024] 본 발명의 다른 양태에 따르면, 본 발명의 방법은 소프트웨어 구축을 보호하기 위한 컴퓨터-구현 방법을 포함한다. 이 방법은 소프트웨어 일부분을 시판된 소프트웨어 패키지의 종류에 속할 것으로 식별하는 제품 식별자를 생성하는 단계를 포함한다. 이 방법은 제품 식별자에 기초하여 소프트웨어 일부분이 소프트웨어 변조 보호가 유용한 시판된 소프트웨어 패키지의 종류에 속한다면 소프트웨어 변조를 방지하기 위한 보호 동적-링크 라이브러리의 설치를 용이하게 하는 단계를 더 포함한다.

[0025] 본 발명의 또 다른 양태에 따르면, 본 발명의 컴퓨터-판독가능 형태는 소프트웨어 구축을 보호하기 위한 컴퓨터-구현 방법을 실행하기 위한 컴퓨터-실행가능 명령어가 저장된 컴퓨터-판독가능 매체를 포함한다. 상기 방법은 소프트웨어 일부분을 시판된 소프트웨어 패키지의 종류에 속할 것으로 식별하는 제품 식별자를 생성하는 단계를 포함한다. 이 방법은 제품 식별자에 기초하여 소프트웨어 일부분이 소프트웨어 변조 보호가 유용한 시판된 소프트웨어 패키지의 종류에 속한다면 소프트웨어 변조를 방지하기 위한 보호 동적-링크 라이브러리의 설치를 용이하게 하는 단계를 더 포함한다.

[0026] 본 발명의 다른 양태에 따르면, 본 발명의 시스템 형태는 소프트웨어 구축을 보호하기 위한 시스템을 포함한다. 시스템은 소프트웨어 일부분을 시판된 소프트웨어 패키지의 종류에 속할 것으로 식별하는 제품 식별자를 생성하기 위한 수단을 포함한다. 시스템은 제품 식별자에 기초하여 상기 소프트웨어 일부분이 소프트웨어 변조 보호가 유용한 시판된 소프트웨어 패키지의 종류에 속한다면 소프트웨어 변조를 방지하기 위한 보호 동적-링크 라이브러리의 설치를 촉진하기 위한 수단을 더 포함한다.

[0027] 본 발명에 따르면, 소프트웨어 변조를 방지하기 위한 시스템, 방법, 및 컴퓨터-판독가능 매체가 제공된다. 본 발명의 컴퓨터-판독가능 매체 형태는 컴퓨팅 시스템이 소프트웨어 변조를 보호하는 데에 이용하는 컴퓨터-실행가능 명령어를 가지는 컴퓨터-판독가능 매체를 포함한다. 컴퓨터-판독가능 매체는 데이터베이스가 변조되었는지 모니터링하고, 파일 세트의 진품 여부를 검사하고, 권한이 없는 서비스를 실행하는 것을 중지시키는

식별이 어려운 컴퓨터-실행가능 명령어를 포함하는 보호 동적-링크 라이브러리를 포함한다.

[0028] 본 발명의 다른 양태에 따르면, 본 발명의 방법은 컴퓨팅 시스템이 소프트웨어 변조를 방지하는 데에 이용하는 컴퓨터-실행가능 명령어가 저장된 컴퓨터-판독가능 매체를 생성하기 위한 컴퓨터-구현 방법을 포함한다. 이 방법은 데이터베이스가 변조되었는지 모니터링하고, 파일 세트의 진품 여부를 검사하고, 권한이 없는 서비스를 실행하는 것을 중지시키는 식별이 어려운 보호 동적-링크 라이브러리 형성하는 단계를 포함하며 상기 보호 동적-링크 라이브러리에서는 제어 흐름, 필드, 메소드, 및 클래스의 이름의 식별이 곤란하다. 이 방법은 식별이 어려운 보호 동적-링크 라이브러리의 체크섬을 포함하는 디지털적으로 서명된 카달로그 파일을 생성하는 단계를 더 포함한다.

[0029] 본 발명의 다른 양태에 따르면, 본 발명의 컴퓨터-판독가능 형태는 소프트웨어 변조를 방지하는 방법을 구현하기 위한 컴퓨터-실행가능 명령어가 저장된 컴퓨터-판독가능 매체를 포함한다. 이 방법은 데이터베이스가 변조되었는지 모니터링하고, 파일 세트의 진품 여부를 검사하고, 권한이 없는 서비스를 실행하는 것을 중지시키는 식별이 어려운 보호 동적-링크 라이브러리 형성하는 단계를 포함하며, 상기 보호 동적-링크 라이브러리에서는 제어 흐름, 필드, 메소드, 및 클래스의 이름의 식별이 곤란하다. 이 방법은 상기 식별이 어려운 보호 동적-링크 라이브러리의 체크섬을 포함하는 디지털적으로 서명된 카달로그 파일을 생성하는 단계를 더 포함한다.

[0030] 본 발명의 다른 양태에 따르면, 본 발명의 컴퓨터-판독가능 형태는 컴퓨팅 시스템이 소프트웨어 변조를 방지하는 데에 이용하는 컴퓨터-실행가능 명령어가 저장된 컴퓨터-판독가능 매체를 생성하기 위한 시스템을 포함한다. 이 시스템은 데이터베이스가 변조되었는지 모니터링하고, 파일 세트의 진품 여부를 검사하고, 권한이 없는 서비스를 실행하는 것을 중지시키는 식별이 어려운 보호 동적-링크 라이브러리 형성하기 위한 수단을 포함하며, 상기 보호 동적-링크 라이브러리에서는 제어 흐름, 필드, 메소드, 및 클래스의 이름의 식별이 곤란하다. 이 시스템은 상기 식별이 어려운 보호 동적-링크 라이브러리의 체크섬을 포함하는 디지털적으로 서명된 카달로그 파일을 생성하기 위한 수단을 더 포함한다.

[0031] 본 발명에 따르면, 소프트웨어 변조를 방지하기 위한 시스템, 방법, 및 컴퓨터-판독가능 매체가 제공된다. 본 발명의 방법은 소프트웨어 변조를 방지하기 위한 방법을 포함한다. 이 방법은 컴퓨팅 기기 상에 식별이 어려운 보호 동적-링크 라이브러리를 로딩하기 전에 식별이 어려운 보호 동적-링크 라이브러리의 제1 체크섬을 계산하는 단계를 포함한다. 이 방법은 소프트웨어 변조를 방지하기 위한 보안 오류 조치가 취해져야 하는지를 판정하기 위하여 상기 식별이 어려운 보호 동적-링크 라이브러리의 소프트웨어 이미지의 진품 여부를 검사하는 단계를 더 포함한다.

[0032] 본 발명의 다른 양태에 따르면, 본 발명의 컴퓨터-판독가능 매체 형태는 소프트웨어 변조를 방지하는 방법을 구현하기 위한 컴퓨터-실행가능 명령어를 가지는 컴퓨터-판독가능 매체를 포함한다. 이 방법은 컴퓨팅 기기 상에 식별이 어려운 보호 동적-링크 라이브러리를 로딩하기 전에 식별이 어려운 보호 동적-링크 라이브러리의 제1 체크섬을 계산하는 단계를 포함한다. 이 방법은 소프트웨어 변조를 방지하기 위한 보안 오류 조치가 취해져야 하는지를 판정하기 위하여 상기 식별이 어려운 보호 동적-링크 라이브러리의 소프트웨어 이미지의 진품 여부를 검사하는 단계를 더 포함한다.

[0033] 본 발명의 다른 양태에 따르면, 본 발명의 시스템 형태는 소프트웨어 변조를 방지하기 위한 시스템을 포함한다. 이 시스템은 컴퓨팅 기기 상에 식별이 어려운 보호 동적-링크 라이브러리를 로딩하기 전에 식별이 어려운 보호 동적-링크 라이브러리의 제1 체크섬을 계산하기 위한 수단을 포함한다. 이 시스템은 소프트웨어 변조를 방지하기 위한 보안 오류 조치가 취해져야 하는지를 판정하기 위하여 식별이 어려운 보호 동적-링크 라이브러리의 소프트웨어 이미지의 진품 여부를 검사하기 위한 수단을 더 포함한다.

[0034] 본 발명에 따르면, 소프트웨어 변조를 방지하기 위한 시스템, 방법, 및 컴퓨터-판독가능 매체가 제공된다. 본 발명의 방법은 컴퓨팅 기기가 소프트웨어 일부분에 대하여 지원가능한 기기의 종류에 속하는지를 판정하기 위한 컴퓨터-구현 방법을 포함한다. 이 방법은 컴퓨팅 기기의 CPU 상에, CPU의 식별을 나타내는 명령어 세트가 존재하는지 검사하는 단계를 포함한다. 이 방법은 CPU가 자가 식별을 지원하는 아키텍처를 가지고 있다면 컴퓨팅 기기 상에 식별이 어려운 보호 동적-링크 라이브러리를 로딩하는 단계를 더 포함한다.

[0035] 본 발명의 다른 양태에 따르면, 본 발명의 컴퓨터-판독가능 매체 형태는 컴퓨팅 기기가 소프트웨어 일부분에 대하여 지원가능한 기기의 종류에 속하는지를 판정하기 위한 방법을 구현하기 위한 컴퓨터-실행가능 명령어가 저장된 컴퓨터-판독가능 매체를 포함한다. 상기 방법은 상기 컴퓨팅 기기의 CPU 상에, 상기 CPU의 식별을 나타내는 명령어 세트가 존재하는지 검사하는 단계를 포함한다. 이 방법은 CPU가 자가 식별을 지원하는 아키텍

치를 가지고 있다면 컴퓨팅 기기 상에 식별이 어려운 보호 동적-링크 라이브러리를 로딩하는 단계를 더 포함한다.

[0036] 본 발명에 따르면, 소프트웨어 변조를 방지하기 위한 시스템, 방법, 및 컴퓨터-판독가능 매체가 제공된다. 본 발명의 방법은 소프트웨어 변조를 방지하기 위한 방법을 포함한다. 이 방법은 컴퓨팅 기기의 시동 중에, 로그-온 모듈에 의해 식별이 어려운 동적-링크 라이브러리의 보호 함수를 호출하여 시스템과 관련된 중앙 계층 데이터베이스의 키 및 값을 보호하는 제1 스레드를 생성하는 단계를 포함한다. 이 방법은 사용자에게 의한 컴퓨팅 기기로의 로그-온 중에, 로그-온 모듈에 의한 식별이 어려운 동적-링크 라이브러리의 보호 함수를 호출하여 사용자와 관련된 중앙 계층 데이터베이스의 키 및 값을 보호하는 제2 스레드를 생성하는 단계를 더 포함한다.

[0037] 본 발명의 다른 양태에 따르면, 본 발명의 컴퓨터-판독가능 매체 형태는 소프트웨어 변조를 방지하기 위한 방법을 구현하기 위한 컴퓨터-실행가능 명령어가 저장된 컴퓨터-판독가능 매체를 포함한다. 상기 방법은 컴퓨팅 기기의 시동 중에, 로그-온 모듈에 의해 식별이 어려운 동적-링크 라이브러리의 보호 함수를 호출하여 시스템과 관련된 중앙 계층 데이터베이스의 키 및 값을 보호하는 제1 스레드를 생성하는 단계를 포함한다. 이 방법은 사용자에게 의한 상기 컴퓨팅 기기로의 로그-온 중에, 상기 로그-온 모듈에 의한 상기 식별이 어려운 동적-링크 라이브러리의 보호 함수를 호출하여 상기 사용자와 관련된 중앙 계층 데이터베이스의 키 및 값을 보호하는 제2 스레드를 생성하는 단계를 더 포함한다.

도면의 간단한 설명

[0038] 도 1은 허가되지 않은 개인들에 의한 컴퓨팅 기기에서의 소프트웨어 변조의 문제점을 보여주는 통상적인 시스템을 도시하는 블록도.

도 2는 변조를 방지하도록 소프트웨어 일부분을 구축하기 위한 예시적인 소프트웨어 컴포넌트를 도시하는 블록도.

도 3a는 소프트웨어 변조를 방지하도록 상호동작하는 예시적인 소프트웨어 컴포넌트를 도시하는 블록도.

도 3b는 본 발명의 일 실시예에 따른, 레지스트리 또는 동적-링크 라이브러리 중 하나에 저장된 예시적인 프로파일 및 소프트웨어 일부분들의 안전한 상호동작에 의한 프로파일의 추출을 도시하는 블록도.

도 4a 내지 4z는 본 발명의 일 실시예에 따른, 소프트웨어 변조를 방지하기 위한 방법을 예시하는 처리의 도면.

발명을 실시하기 위한 구체적인 내용

[0039] 본 발명의 기술한 양태 및 다수의 부수적인 이점은 첨부된 도면에 관련하여 이루어질 이하 상세한 설명을 참조하면 보다 잘 이해되어질 것이므로 보다 쉽게 인식될 것이다.

[0040] 소프트웨어는 강력하지만 취약하다는 양면성이 있다. 본 발명의 다양한 실시예는 소프트웨어 일부분의 변조를 방지한다. 일 실시예는 소프트웨어 이미지를 포함하는 소프트웨어 일부분들을 역설계하는 해커의 이해를 혼란스럽게 하기 위한 소프트웨어 이미지의 식별곤란화(obfuscation)이다. 다른 실시예는 소프트웨어 일부분들 모두가 변조로부터 보호를 요하는 소프트웨어 패키지를 구성하는지를 검사하는 것이다. 또 다른 실시예는 컴퓨팅 기기 상의 CPU 또는 캐쉬 메모리 등의 하드웨어 자원이 소프트웨어 변조를 방지하는 것이 가능한 종류에 속하는지 판정하는 것이다. 다른 실시예는 컴퓨팅 기기 상의 특정한 중대한 파일들이 변조되었는지를 체크하는 것을 포함한다. 추가적인 실시예는 레지스트리가 변조되었는지를 판정하는 것을 포함한다. 몇몇의 다른 실시예는 실행되고 있는 소프트웨어 서비스들이 변조되었는지를 판정하는 것을 포함한다.

[0041] 도 2는 제품 식별자(PID) 동적-링크 라이브러리(200A), 구축 소프트웨어(200B), 및 구축 정보 파일(200C)을 포함하는 소프트웨어 컴포넌트(200)를 도시한다. 구축 소프트웨어 컴포넌트(200)는 변조를 방지하기 위한 소프트웨어 일부분이 컴퓨터 시스템 상에 설치될 수 있는지를 판정하는 역할을 담당한다. 소프트웨어 일부분은 시스템 소프트웨어(운영 체제), 애플리케이션 소프트웨어, 및 네트워크 소프트웨어 등의 임의의 적절한 소프트웨어 일부분들을 포함한다. 제품 식별자 동적-링크 라이브러리(200A)는 파일들과는 별도로 저장된 실행가능한 루틴의 종류에 속한다. 이들 루틴은 "DLL" 등의 특정 확장자를 가지며, 프로그램이 필요로 할 때만 로딩된다. 이들 동적-링크 라이브러리는 마이크로소프트 윈도우즈 계열의 운영 체제 및 OS/2의 특징으로서 적절히 존재할 수 있다. 동적-링크 라이브러리는 몇 가지 특징을 가진다: 첫째, 통상적으로 사용되기 전까지는

어떠한 메모리도 소비하지 않는다; 둘째, 동적-링크 라이브러리는 별개의 파일이기 때문에 호출 프로그램, 또는 다른 동적-링크 라이브러리의 운영에 영향을 미치지 않으면서 프로그래머가 바로 그 모듈에만 정정이나 개선을 할 수 있다; 마지막으로, 프로그래머는 다른 프로그램과 동일한 동적-링크 라이브러리를 이용할 수 있다.

[0042] 구축 정보 파일(200C)은 텍스트 기반 파일 종류에 속하며 설치 중에, 구축 소프트웨어(200B) 등의 구축 애플리케이션에 의해 이용되는 정보를 포함한다. 통상적으로, (보호 정보 파일(202A) 등의) 다른 구축 정보 파일들 중에서도 특히, 구축 정보 파일(200C)은 구축 소프트웨어(200B) 등의 구축 애플리케이션의 작성 이전에 생성된다. 정보 파일에 저장될 수 있는 정보의 예로는 레지스트리 변경, 파일 이름, 및 소스 매체 상의 소스 파일의 위치를 포함한다. 보호 정보 파일(202A) 뿐만 아니라 구축 정보 파일(200C)은 개인 섹션 또한 포함할 수 있다. 이들 개인 섹션은 구축 소프트웨어(200B)에 의존하며 구축 소프트웨어(200B) 등의 특정 구축 애플리케이션에 의해 이용되는 특수화된 정보를 저장하는 데 이용되어 소프트웨어 변조를 방지하는 소프트웨어 일부분들을 설치할 수 있다.

[0043] 구축 소프트웨어(200B)는 저장 매체 또는 메모리 중 하나에서의 시스템 소프트웨어, 애플리케이션 소프트웨어, 또는 네트워크 소프트웨어 등의 다른 프로그램을 설치하는 기능을 가지는 프로그램이다. 구축 소프트웨어(200B)는 기기, 프린터, 모니터, 및 네트워크의 특정 조합에 대한 애플리케이션을 구축하는 빈번하고 복잡한 처리 전반에 걸쳐 사용자를 안내하는 데에 이용될 수 있다. 제품 식별자 동적-링크 라이브러리(200A)는 고유한 제품 식별자를 생성하고 레지스트리(204)에 이 생성된 제품 식별자(204A)를 저장하는데, 이는 이하 설명할 것이다. 구축 정보 파일(200C)은 소프트웨어 변조를 방지하는 소프트웨어 일부분의 설치를 위하여 설치 동적-링크 라이브러리(202B)를 준비한다. 설치 컴포넌트(202)는 보호 정보 파일(202A) 및 설치 동적-링크 라이브러리(202B)를 포함한다. 보호 정보 파일(202A)은 소프트웨어 변조를 방지하는 소프트웨어 일부분의 설치를 위한, 레지스트리 변경, 파일 이름, 및 소스 파일의 위치 등의 각종 정보 일부분을 포함한다. 또한, 보호 정보 파일(202A)은 시스템 파일(206B)의 설치를 위한 정보 및 시판된 소프트웨어 패키지에 특정된 다양한 소프트웨어 컴포넌트를 포함한다.

[0044] 소프트웨어 컴포넌트(202) 내에 포함된, 설치 동적-링크 라이브러리(202B)는 특정 시장 부분에 특정되고 이 특정 시장 부분에 적절한 언어 및 내용을 제공할 수 있는, 보호 동적-링크 라이브러리(206A) 등의, 소프트웨어 변조를 방지하는 실질적인 소프트웨어 일부분을 설치한다. 설치 동적-링크 라이브러리(202B)는 또한 설치가 보안된 설치임을 빠르게 식별하기 위해 보호 동적-링크 라이브러리(206A)에 구현된 변조를 방지하는 소프트웨어 일부분을 허용하도록 활성화 키(204C)를 설정한다. 또한, 설치 동적-링크 라이브러리(202B)는 다른 것들 중에서도 특히, 하드웨어 및/또는 언어에 속하는 정보를 포함하는 복수의 암호화된 보호 프로파일, 및 각종 시스템 파일(206B)도 설치하여 무언가가 변조되었는지를 판정할 수 있게 한다.

[0045] 런타임 컴포넌트(206)는 도움말 파일, 및 언어와 같은 시장-특정 내용 파일을 포함하는 시스템 파일(206B)을 가지고 소프트웨어 변경을 방지하는 소프트웨어 일부분을 구현하는 보호 동적-링크 라이브러리(206A)를 포함한다.

[0046] 중앙 계층 데이터베이스(204)는 제품 식별자(204A), 활성화 키(204C), 및 암호화된 보호 프로파일(204D)을 포함한다. 중앙 계층 데이터베이스(204)는 하나 이상의 사용자, 애플리케이션, 및 하드웨어 장치에 대하여 시스템을 구성하는 데 필요한 정보를 저장하는 데 이용된다. 중앙 계층 데이터베이스(204)는 각 사용자에게 대한 프로파일, 컴퓨터 상에 설치된 애플리케이션, 및 각각이 생성할 수 있는 문서 유형; 폴더 및 애플리케이션 아이콘에 대한 속성 시트 설정; 시스템 상에 존재하는 하드웨어; 및 사용되고 있는 포트와 같은, 동작 중에 운영 체제가 계속적으로 참조하는 정보를 포함한다. 중앙 계층 데이터베이스(204)는, 또한, 상술한 바와 같이, 소프트웨어 변조의 방지를 돕는 정보 일부분들을 저장하는 데에 이용될 수 있다. 정보의 적합한 일부분은 해커가 파일을 변경할 때 등에서, 데이터가 변조되었는지 검사하는 데에 이용되는 계산된 값들인 체크섬(checksum)을 포함한다. 체크섬은 데이터의 모든 바이트를 일련의 산술 또는 논리적 동작과 순차적으로 결합함으로써 주어진 데이터 청크(chunk)에 대해 계산되는 것이 바람직하다. 소프트웨어 변조가 확인되었거나 유효한 기간 동안, 저장된 데이터를 이용하여 동일한 방식으로 새로운 체크섬이 계산될 수 있다. 2개의 체크섬이 일치하지 않는다면, 소프트웨어 변조가 일어난 것일 수 있고, 변조를 고치거나 컴퓨팅 기기를 종료하기 위한 적절한 단계들을 취할 수 있다.

[0047] 도 3a는 보호 동적-링크 라이브러리(306A)에 의해 구현된 소프트웨어 변조를 방지하기 위한 소프트웨어 일부분의 실행을 도시한다. 도 3a의 각종 구성요소는 도 2a의 각종 구성요소와 유사하며, 간결히 하기 위해 이들을 더 설명하지는 않을 것이다. 로그-온 컴포넌트(308)는 로그-온 모듈(308A)을 포함하며, 컴퓨터 시스템을

시동시킬 때, 로그-온 모듈(308A)은 제품 식별자(304A)에 의존하는 보호 동적-링크 라이브러리(306A)를 로딩하기를 시도한다. 중앙 계층 데이터베이스(308)에 적절한 제품 식별자가 존재하면, 보호 동적-링크 라이브러리(306A)는 로그-온 모듈(308A)에 의해 로딩될 것이다. 로그-온 모듈(308A)은 바람직하게는 사용자 이름 및 패스워드를 수신하는 소프트웨어 일부분이며 사용자가 컴퓨팅 기기 상의 시스템 소프트웨어를 액세스할 수 있게 되기 전에 정보를 확인한다.

[0048] 컴퓨터 시스템의 소프트웨어의 실행 중에, 보호 동적-링크 라이브러리(306A)는 각종 암호화된 보호 프로파일(304D)을 읽음으로써 각종 시스템 파일(306B)이 변조되지 않았음을 확인한다. 또한, 보호 동적-링크 라이브러리(306A)는 각종 키 설정 및 그 관련 값들이 변조되었는지 판정하기 위하여 이들을 검사한다. 보호 동적-링크 라이브러리(306A)는 먼저 레지스트리 및 컴퓨터 시스템 상에서 실행되도록 허가된 서비스들을 검사하기 위한 타이머를 생성한다. 시스템이 변조되었다면, 보호 동적-링크 라이브러리(306A)는, 변조를 처리하는 다른 적절한 방법들 중에서도 특히, 이 변조를 교정하는 것을 시도하거나 컴퓨팅 기기를 종료하기 시작한다.

[0049] 예를 들면, 보호 동적-링크 라이브러리(306A)는 중앙 계층 데이터베이스(304)를 모니터링하고 변조가 있었다면 키들 및 그들과 관련된 값들을 리셋한다. 보호 동적-링크 라이브러리(306A)는 또한, 암호화된 보호 프로파일(304D) 및 보호 동적-링크 라이브러리(306A)에 포함된 정보에 대하여, 각종 시스템 파일(306B) 등의 중대한 파일을 체크하여 변조가 있었는지 여부를 판정한다. 또한, 보호 동적-링크 라이브러리(306A)는 컴퓨터 시스템 상에서 실행되는 소프트웨어 서비스들이 권한이 있는지를 확인하도록 체크하고 권한이 없는 소프트웨어 서비스를 제거한다.

[0050] 도 3b는 로그-온 모듈(308A) 등의 소프트웨어 일부분에 의해 액세스될 수 있는 암호화된 보호 프로파일(304D)을 도시한다. 로그-온 모듈(308A)은 보호 동적-링크 라이브러리(306B)와 통신하여 암호화된 보호 프로파일(304D)을 액세스한다. 로그-온 모듈(308A)과 보호 동적-링크 라이브러리(306B) 간의 상호 운용은 소프트웨어 일부분들 간에 정보의 전송 또는 통신이 안전하게 이루어질 수 있다는 것을 알리는 적절한 프로토콜에 의해 이루어지는 것이 바람직하다.

[0051] 한 적절한 프로토콜은 로그-온 모듈(308A)에 의해 보호 동적-링크 라이브러리(306B)로 전송되는 숫자, 알파벳, 또는 알파벳 숫자의 문자열 정보를 포함할 수 있는 임의의 솔트 값(salt value)의 전송을 포함한다. 보호 동적-링크 라이브러리(306B)는 2개의 키, 즉, 공개 라이브러리 키 및 개인 라이브러리 키를 가지는 것이 바람직하다. 암호화 및 디지털 서명에서, 이들 키는 송신되거나 수신될 암호화 및 복호화 정보에 이용되는 비트 스트링을 포함한다. 암호화는 2개의 서로 다른 유형의 키, 한 명 이상에 알려지는 공개 키, 및 한 명에게만 알려지는 개인 키에 공통적으로 의존한다.

[0052] 보호 동적-링크 라이브러리(306B)는 개인 라이브러리 키를 이용하여 프로파일(312 내지 316) 등의 프로파일을 복호화한다. 각 프로파일은 각종 정보 일부분이 저장될 수 있는 무정형의 데이터 구조인 블랍으로 알려진 데이터 구조로서 적절하게 조직화된다. 바람직하게는, 프로파일(312 내지 316)의 블랍은 보호 동적-링크 라이브러리(306B)의 개인 라이브러리 키를 이용하여 암호화된다. 프로파일(312)은 보호 동적-링크 라이브러리(306B)의 개인 라이브러리 키에 의해 암호화되었던 블랍 1을 포함한다. 블랍 1은 그 사이즈와 포맷이 하드웨어 프로파일 또는 언어 프로파일 등의 프로파일 유형에 의존하는 프로파일 데이터 1(312A)을 포함한다. 블랍 1은 또한 보호 동적-링크 라이브러리(306B)의 공개 라이브러리 키로 디지털적으로 서명된, 프로파일 데이터 1(312A)의 체크섬(checksum)을 포함하는 시그니처 1(312B)을 포함한다. 블랍 1은 또한 로그-온 모듈(308A)의 공개 호출 키로 디지털적으로 서명되고 로그-온 모듈(308A)의 개인 호출 키로 암호화도 된, 프로파일(312)의 식별자와 프로파일 데이터 1(312A)의 체크섬인 검증 블랍 1(312C)을 포함한다.

[0053] 프로파일(314)은 로그-온 모듈(308A)의 공개 호출 키로 디지털적으로 서명되고 로그-온 모듈(308A)의 개인 호출 키로 암호화도 된, 프로파일 데이터 2(314B)에 의해 기술된 프로파일(314)의 식별자와 그 데이터의 체크섬인 검증 블랍 2(314A)을 포함하는 블랍 2로서 조직화된다. 블랍 2는 그 사이즈와 포맷이 하드웨어 프로파일 또는 언어 프로파일 등의 프로파일 유형에 의존하는 프로파일 데이터 2(314B)를 포함한다. 블랍 2는 또한 보호 동적-링크 라이브러리(306B)의 공개 라이브러리 키로 디지털적으로 서명된, 프로파일 데이터 2(312B)의 체크섬(checksum)인 시그니처 2(314C)를 포함한다.

[0054] 프로파일(316)은 블랍 3으로서 조직화된다. 블랍 3은 보호 동적-링크 라이브러리(306B)의 공개 라이브러리 키로 디지털적으로 서명된 프로파일 데이터 3(316C)의 체크섬인 시그니처 3(316A)을 포함한다. 블랍 3은 로그-온 모듈(308A)의 공개 호출 키로 디지털적으로 서명되고 로그-온 모듈(308A)의 개인 호출 키로 암호화도 된, 프로파일(316)의 식별자와 그 프로파일 데이터 3(316C)의 체크섬인 검증 블랍 3(316B)을 포함한다. 블랍

3은 그 사이즈와 포맷이 하드웨어 프로파일 또는 언어 프로파일 등의 프로파일 유형에 의존하는 프로파일 데이터 3(313C)을 포함한다. 프로파일(312 내지 316)의 블랍들(1 내지 3) 각각에 대하여, 프로파일 데이터(1 내지 3), 시그니처(1 내지 3) 및 검증 블랍(1 내지 3) 등의 다양하게 명명된 조직은 블랍(1 내지 3) 또는 프로파일(312 내지 316) 각각에서 서로 다르게 위치된다. 이들 서로 다른 배치는 레지스트리(304) 내에 포함된 프로파일 데이터의 변조를 방지하는 것을 돕는다. 위치 정보를 포함하는 헤더가 블랍(1 내지 3) 내에 존재하여 각각의 명명된 조직이 블랍(1 내지 3) 내에 상주하지를 판정할 수 있게 한다.

[0055] 보호 동적-링크 라이브러리(306B)가 그 개인 라이브러리 키를 이용하여 프로파일 또는 블랍을 복호화함으로써 원하는 프로파일을 찾았다면, 로그-온 모듈(308A)에 프로파일을 반환할 것이다. 로그-온 모듈(308A)에 데이터를 반환하기 전에, 보호 동적-링크 라이브러리(306B)는 찾아낸 프로파일의 식별자 및 프로파일 데이터로부터 체크섬을 계산한다. 체크섬은 최초에 로그-온 모듈(308A)에 의해 전송되었던 임의의 솔트 값과 프로파일을 비교한 결과를 포함한다. 본질적으로 솔트 값은 해커가 보호 동적-링크 라이브러리(306A)를 간단히 애플리케이션하여 로그-온 모듈(308A)에 잘못된 프로파일을 반환하지 못하게 한다.

[0056] 체크섬이 계산되었다면, 보호 동적-링크 라이브러리(306B)는 본질적으로 각종 정보 일부분(예를 들면, 성공, 실패, 및 성공 또는 실패를 상세히 기술하는 플래그), 찾아낸 프로파일 내의 검증 블랍, 및 프로파일의 식별자, 그 데이터, 결과, 및 임의의 솔트 값으로부터 계산된 체크섬을 가지는 결과 데이터 구조인 결과를 로그-온 모듈(308A)에 반환한다. 로그-온 모듈(308A)은 보호 동적-링크 라이브러리(306B)로부터 반환된 결과를 검사한다. 로그-온 모듈(308A)은 자신의 개인 호출 키를 이용하여 검증 블랍을 복호화하여 프로파일의 식별자와 프로파일의 데이터의 체크섬을 획득한다. 또한, 로그-온 모듈(308A)은 자신의 공개 호출 키를 이용함으로써 검증 블랍의 시그니처를 검사한다. 게다가, 로그-온 모듈(308A)은 복호화된 검증 블랍 체크섬, DLL의 결과, 및 최초로 보호 동적-링크 라이브러리(306A)로 전달된 임의의 솔트 값으로부터 체크섬을 계산한다. 로그-온 모듈(308A)은 계산된 체크섬과 보호 동적-링크 라이브러리(306B)에 의해 반환된 체크섬이 일치하는지를 판정하는 검사를 수행한다. 체크섬이 일치하지 않는다면, 로그-온 모듈(308A)은 시스템이 변조되었던 것이라고 결론짓는다.

[0057] 도 4a 내지 4z는 소프트웨어 변조를 방지하기 위한 방법(400)을 도시한다. 명료함을 위하여, 이하 방법(400)의 상세한 설명은 도 2a의 소프트웨어 컴포넌트(200, 202, 204, 및 206), 도 3a 내지 3b의 로그-온 모듈(308A), 및 보호 동적-링크 라이브러리(306B)에 관련하여 도시된 각종 구성요소들을 참조한다. 시작 블록으로부터, 방법(400)은 연속 단말("단말 A")과 종료 단말("단말 B") 사이에 정의된 방법 단계 세트(402)로 진행된다. 방법 단계 세트(402)는 소프트웨어 변조를 방지하기 위한 소프트웨어 일부분의 식별곤란화를 기술한다.

[0058] (도 4c의) 단말 A로부터, 방법(400)은 소프트웨어 변조를 방지하는 소프트웨어 일부분을 실행하는, 보호 동적-링크 라이브러리에 대하여 소스 코드가 생성되는 블록(414)으로 진행된다. 블록(416)을 참조하면, 소프트웨어 코드가 컴파일되고 링킹되어 보호 동적-링크 라이브러리(DLL)를 생성한다. 그 다음, 블록(418)에서, 필드, 메소드, 및 클래스의 이름들을 포함하는 식별곤란화 제어 파일이 생성된다. 그 다음 방법(400)은 보호 DLL 및 식별곤란화 제어 파일을 입력으로서 이용하여 식별곤란화 처리가 실행되는 블록(420)으로 진행된다. 블록(422)을 참조하면, 제어 흐름, 및 필드, 메소드, 클래스의 이름의 식별이 어려운 곳인, 식별곤란화된 DLL이 생성된다. 식별곤란화 처리는 엔지니어의 보호 동적-링크 라이브러리를 역설계하기를 시도하는 해커가 보호 동적-링크 라이브러리를 디버깅하는 것을 어렵게 만든다. 식별곤란화의 일례는 보호 동적-링크 라이브러리에 점프 명령어 삽입 또는 프로그래밍 명령어의 재정렬이다.

[0059] 상술한 식별곤란화 처리는 불법 목적으로 소프트웨어를 리버스 엔지니어링하는 것을 시도하는 디컴파일러를 좌절시키도록 소프트웨어를 어셈블링할 때 심볼의 재명명을 제공하는 다수의 적절한 기법들 중 하나이다. 식별곤란화 처리는 불법 디컴파일러로부터의 보호를 향상시키면서도 애플리케이션이 손상되지 않은 상태를 유지시킨다. 식별곤란화의 목적은 소프트웨어 일부분의 다수의 양상의 지식적인 개념을 이해하는 해커의 마음에 부담을 주어 혼란스럽게 하는 것이다. 식별곤란화 처리는 해커인 인간 해석자(interpreter)를 혼란시키는 것 뿐만 아니라, 로직의 확실성에 의존하는 디컴파일러를 교란시킬 수도 있을 것이다. 식별곤란화 처리는 이 중 몇몇이 잘못된 로직에 도달하는 무수한 디컴파일 가능성을 생성하여, 변환을 불확실하게 한다.

[0060] (도 4d의) 단말 A1으로부터, 방법(400)은 식별곤란화된 보호 DLL의 체크섬을 계산하는 블록(426)으로 진행된다. 그 다음에 블록(428)에서 카탈로그 파일이 생성된다. 카탈로그 파일의 한 적절한 구현은 데이터베이스이다. 카탈로그는 각 소프트웨어 컴포넌트의 이름, 그 버전, 및 그 시그니처(예를 들면, 체크섬)를 포함한다. 그 다음, 블록(430)에서, 식별곤란화된 보호 DLL의 체크섬이 카탈로그 파일에 위치한다. 그 다음

방법(400)은 이 방법이 식별곤란화된 보호 DLL의 체크섬을 포함하는 카달로그 파일의 체크섬을 계산하는 블록(432)으로 진행된다. 블록(434)을 참조하면, 카달로그 파일의 체크섬 또한 개인 키를 이용하여 암호화된다. 다시 말하면, 카달로그 파일은 개인 키에 의해 디지털적으로 서명된다. 이 시점에서, 방법(400)은 검사될 수 있는 카달로그 파일 및 식별곤란화된 보호 DLL을 생성한다. 검사는 식별곤란화된 보호 DLL이 특정 소프트웨어 제조업체 등의 알려진 소스로부터 유래하였고 식별곤란화된 보호 DLL이 변조되지 않았음을 보장하는 것을 돕는다. 그 다음 방법(400)은 종료 단말 B로 계속된다.

[0061] (도 4A의) 단말 B로부터, 방법(400)은 연속 단말("단말 C")과 종료 단말("단말 D") 사이에 정의된 방법 세트(404)로 진행된다. 방법 단계 집합(404)은 컴퓨팅 기기 상에, 소프트웨어 변조를 방지하기 위한 소프트웨어 일부분들을 구현하는 보호 동적-링크 라이브러리의 구축을 기술한다.

[0062] (도 4e의) 단말 C로부터, 방법(400)은 이 방법이 소프트웨어 일부분에 대한 제품 식별자를 획득하고 이 식별자를 레지스트리 등의 중앙 계층 데이터에 저장하는 블록(438)으로 진행된다. 사용자가 소프트웨어 일부분의 패키징에 통상적으로 첨부되는 키로부터 획득할 수 있는 제품 식별자가 이 사용자에게 의해 제공된다. 블록(440)을 참조하면, 이 방법은 제품 식별자를 디코딩하여 소프트웨어 일부분을 특징짓는 종류를 식별한다. 소프트웨어 일부분은 시스템 소프트웨어, 애플리케이션 소프트웨어, 또는 네트워크 소프트웨어 등의 임의의 적절한 소프트웨어 일부분을 포함할 수 있다. 그 다음, 판정 블록(442)에서, 식별된 종류에 대한 보호가 유용한지를 판정하기 위한 검사가 이루어진다. 결과가 "아니오"라면, 방법(400)은 다른 연속 단말("단말 L")로 진행하여 실행을 종료한다. 한편, 판정 블록(442)에서의 검사 결과가 "예"라면, 방법(400)은 설치 동적-링크 라이브러리가 호출되는 블록(444)으로 진행된다. 블록(446)을 참조하면, 설치 동적-링크 라이브러리가 제품 식별자에 기초하여, 레지스트리 등의 중앙 계층 데이터베이스에 활성화 키를 설치한다. 활성화 키는 컴퓨팅 기기로부터 하역 시간 소모적일 수 있는 제품 식별자 디코딩을 하지 않고 소프트웨어 변조를 방지시키기 위한 소프트웨어 일부분이 컴퓨팅 기기 상에 실행되는지를 신속하게 체크하고 판정할 수 있게 한다.

[0063] (도 4f의) 단말 C1로부터 설치 동적-링크 라이브러리는 레지스트리 등의 중앙 계층 데이터베이스로 암호화된 보호 프로파일을 설치한다. 블록(450)을 참조하면, 컴퓨팅 기기를 시동시킬 때, 시스템 소프트웨어 등의 소프트웨어 일부분의 로그-온 모듈이 실행된다. 그 다음, 블록(452)에서, 로그-온 모듈은 레지스트리 등의 중앙 계층 데이터베이스에 저장된 제품 식별자를 디코딩하여, 소프트웨어 일부분을 특징짓는 제품 종류를 획득한다. 방법(400)은 식별된 제품 종류에 대한 보호가 유용한지를 판정하기 위한 검사가 수행되는 판정 블록(454)으로 진행된다. 판정 블록(454)에서의 검사 결과가 "아니오"라면, 방법(400)은 실행을 종료하는 단말 L로 진행된다. 반면에, 판정 블록(454)에서의 검사 결과가 "예"라면, 블록(456)에 나타난 바와 같이 로그-온 모듈(308A)은 식별이 어려운 보호 동적-링크 라이브러리를 컴퓨팅 기기에 로딩하는 것을 시도한다. 이 방법은 다른 연속 단말("단말 C2")로 계속된다.

[0064] (도 4g의) 단말 C2로부터, 방법(400)은 이 방법이 설치될 보호 동적-링크 라이브러리의 체크섬이 계산하는 블록(458)으로 진행된다. 이하 명료함을 위하여 식별이 어려운 보호 동적-링크 라이브러리는 보호 동적-링크 라이브러리로 알려질 것임을 유의한다. 블록(460)을 참조하면, 이 방법은 시스템 파일에 포함된 카달로그 파일의 리스트를 획득한다. 그 다음, 블록(462)에서, 방법은 리스트로부터 카달로그 파일을 취득하고 카달로그 파일에 포함된 체크섬을 찾기 위한 검색을 한다. 시그니처의 검사 및 복호화가 일어난다. 방법(400)은 2개의 체크섬들 간에 일치가 있는지를 판정하기 위한 검사가 수행되는 판정 블록(464)으로 진행된다. 판정 블록(464)에서의 검사 결과가 "아니오"라면, 방법(400)은 다른 연속 단말("단말 C3")로 계속된다. 반면에, 판정 블록(464)에서의 검사 결과가 "예"라면, 방법(400)은 다른 연속 단말("단말 C4")로 계속된다. 도 4g에 관련하여 기술된 처리 단계들은 소프트웨어 이미지의 진품 여부(즉, 소프트웨어 이미지가 처음부터 원하는 소프트웨어 제조업체에 의해 출하되고 배포되었는지)를 판정하기 위하여 식별이 어려운 보호 동적-링크 라이브러리의 소프트웨어 이미지를 검사하는 처리를 기술한다. 변조가 이루어졌다면, 검사 처리는 로딩될 식별이 어려운 보호 동적-링크 라이브러리의 소프트웨어 이미지가 변조되었음을 인지하고, 컴퓨팅 기기 종료 등의 보안 오류 조치가 취해져야 할 수 있다.

[0065] (도 4h의) 단말 C3로부터, 방법(400)은 검색할 카달로그 파일이 더 있는지를 판정하기 위한 검사가 수행되는 판정 블록(466)으로 진행된다. 판정 블록(466)에서의 검사 결과가 "예"라면, 방법은 다른 연속 단말("단말 C5")로 진행된다. (도 4g의) 단말 C5로부터, 방법(400)은 상술한 처리 단계들이 반복되는 블록(462)으로 다시 루핑한다. 반면에, 판정 블록(466)에서의 검사 결과가 "아니오"라면, 방법(400)은 설치될 보호 동적-링크 라이브러리가 카달로그 파일에 부수적으로 딸려온 진품이 아니므로 변조가 되었을 수 있음을 나타내는 블록(468)으로 계속된다. 그 다음 방법(400)은 다른 연속 단말("단말 C7")로 계속된다. (도 4h의) 단말 C4로부터

터, 방법(400)은 신뢰되는 카탈로그 파일 및 보호 동적-링크 라이브러리가 존재하는지 여부를 판정하기 위한 검사가 수행되는 판정 블록(470)으로 진행된다. 판정 블록(470)에서의 검사 결과가 "아니오"라면, 방법(400)은 블록(462)으로 다시 루핑하여 상술한 처리 단계들이 반복되는 단말 C5로 진행한다. 판정 블록(470)에서 기술된 검사의 한 적절한 구현은 마이크로소프트 윈도우즈에 의해 제공된 신뢰-체크 애플리케이션 프로그래밍 인터페이스 세트이다. 그렇지 않고, 판정 블록(470)에서의 검사 결과가 "예"라면, 방법은 다른 연속 단말("단말 C6")로 진행된다.

[0066] (도 4i의) 단말 C6로부터, 방법(400)은 이 방법이 신뢰-체크 애플리케이션 프로그래밍 인터페이스 세트를 호출함으로써 찾아낸 카탈로그 파일을 검사하고 반환 값으로서 상태 데이터를 획득하는 블록(472)으로 진행한다. 블록(474)에서 방법은 신뢰-체크 애플리케이션 프로그래밍 인터페이스 세트를 통하여 획득된 상태 데이터를 이용하여 신뢰 제공자 정보를 획득한다. 그 다음, 블록(476)에서, 방법은 획득된 신뢰 제공자 정보를 이용하여 인증 체인으로부터 기본 서명자(base signer)를 획득한다. 방법(400)은 이 방법이 기본 서명자의 환경(context)을 이용하여 인증 체인의 마지막 요소를 검사하는 블록(478)으로 진행한다. 판정 블록(480)을 참조하면, 마지막 요소가 적절한 공개 키를 포함하였는지를 판정하기 위한 검사가 수행된다. 반면에, 판정 블록(480)에서의 검사 결과가 "아니오"라면, 방법은 블록(462)으로 다시 루핑하여 상술한 처리 단계들이 반복되는 단말 C5로 계속된다.

[0067] (도 4j의) 단말 C7로부터, 방법(400)은 보호 동적-링크 라이브러리가 성공적으로 로딩되었는지를 판정하기 위한 검사가 수행되는 판정 블록(482)으로 진행된다. 판정 블록(482)에서의 검사 결과가 "아니오"라면, 블록(484)에 나타난 바와 같이, 방법은 시스템을 종료하기 시작한다. 방법(400)은 단말 L로 진행하고 실행을 종료한다. 반면에, 판정 블록(482)에서의 검사 결과가 "예"라면, 방법(400)은 이 방법이 종료될 수 없는 스레드에서 실행되는 제1 타이머를 생성하는 블록(486)으로 진행한다. 그 다음 방법은 연속 단말("단말 C8")과 종료 단말 D로 나타난 2개의 실행 경로로 계속된다. 이렇게 도 4j의 실행 경로가 분리되는 것은 컴퓨팅 기기 상에 실행되는 몇몇의 소프트웨어 일부분이 동시에 발생하는 것을 예시하는 것이다.

[0068] (도 4k의) 단말 C8로부터, 방법(400)은 미리-구성된 기간이 만료되었는지를 판정하는 검사가 수행되는 판정 블록(488)으로 진행한다. 판정 결과가 "아니오"라면, 방법(400)은 상술한 처리 단계들이 반복되는 판정 블록(488)으로 다시 루핑한다. 판정 블록(488)에서의 검사 결과가 "예"라면, 방법(400)은 이 방법이 레지스트리 등의 중앙 계층 데이터베이스에 저장되는 활성화 키를 체크하여 제품 클래스를 디코딩하는 블록(490)으로 진행한다. 판정 블록(492)을 참조하면 디코딩된 제품 종류에 대한 보호가 유용한지를 판정하기 위한 다른 검사가 수행된다. 판정 블록(492)에서의 검사 결과가 "예"라면, 방법(400)은 상술한 처리 단계들이 반복되는 단말 C8로 다시 루핑한다. 판정 블록(480)에서의 검사 결과가 "예"라면, 방법(400)은 단말 C7로 계속된다. 반면에, 판정 블록(492)에서의 검사 결과가 "아니오"라면, 코어 애플리케이션 프로그래밍 인터페이스가 변조된 것어서 이 방법은 컴퓨팅 기기를 종료하는 등의 보안 조치를 취한다. 블록(494) 참조한다. 그 다음 방법(400)은 단말 L로 진행하고 실행을 종료한다.

[0069] (도 4a의) 단말 D로부터, 방법(400)은 연속 단말("단말 E")과 종료 단말("단말 F") 사이에 정의된 방법 단계 세트(406)로 진행한다. 방법 단계 세트(406)는 컴퓨팅 기기가 지원될 수 있는 기기의 종류에 속하는지를 판정한다.

[0070] (도 4l의) 단말 E로부터, 방법(400)은 이 방법이 컴퓨팅 기기의 CPU 상에 명령어 세트가 존재하는지 검사하여 그 식별자를 획득하는 블록(496)으로 진행한다. 블록(498)에서 명령어 세트가 유용한지를 판정하는 검사가 수행된다. 판정 블록(498)에서의 검사 결과가 "아니오"라면, 블록(499)에 나타난 바와 같이 컴퓨팅 기기의 CPU는 자가 식별을 지원하지 않는 아키텍처를 가진다. 한 적절한 결론은 CPU가 소프트웨어 변조를 방지하기 위한 소프트웨어 일부분들을 지원할 수 있는 것이다. 그 다음 방법(400)은 종료 터미널 F로 진행한다. 반면에, 판정 블록(498)에서의 검사 결과가 "예"라면, 블록(497)에서 로그-온 모듈은 보호 동적-링크 라이브러리와 통신하여 배타적인 CPU 종류에 관련된 암호화된 보호 프로파일을 액세스한다. 다시 말하면, 컴퓨팅 기기를 동작시키는 각종 파라미터를 기술하는 컴퓨팅 기기 상에 설치된 파일이 존재한다. 예를 들면, 언어 파일은 운영 체제가 사용자에게 정보를 표시하는 데에 이용되는 언어를 기술한다. 다른 예로서, 하드웨어 파일은 운영 체제가 지원하는 컴퓨팅 기기 일부분을 기술한다. 문제는 해커 또는 악성 소프트웨어에 의해 파일들이 쉽게 변조될 수 있는 것이다. 컴퓨팅 기기 상에 설치된 원본 파일을 나타내는 체크섬이 암호화된 보호 프로파일 내에 포함된다. 원본 파일이 변조되었다면, 암호화된 보호 프로파일에 저장된 체크섬과 변조된 파일은 일치하지 않아, 원본 파일이 변조되었음을 알 수 있다. 예를 들면, 해커가 지원불가능한 컴퓨팅 기기를 지원하는 것으로 운영 체제를 속이도록 하드웨어 파일을 변경하였다면, 변경된 하드웨어 파일의 체크섬은 암호화

된 보호 파일에 저장된 체크섬과 달라질 수 있어, 변조를 판정할 수 있게 된다. 도 4a를 다시 참조해 보면, 블록(495)에서 로그-온 모듈은 보안을 향상시키는 데에 이용되는 임의 솔트 값의 보호 동적-링크 라이브러리로 전달된다. 임의 솔트 값은 함수 호출을 못하게 하거나 차단시키는 것을 어렵게 한다. 이하 기술될 바와 같이, 임의 솔트 값은 보호 동적-링크 라이브러리에 의해 송신된 데이터 및 임의 솔트 값에 기초하는 체크섬의 계산에 이용된다. 그 체크섬은 변조가 일어나는지를 판정하는 보호 동적-링크 라이브러리에 의해 계산된 체크섬과 비교된다. 그 다음 방법(400)은 다른 연속 단말("단말 E1")로 계속된다.

[0071] (도 4m의) 단말 E1로부터, 방법(400)은 보호 동적-링크 라이브러리가 개인 라이브러리 키를 이용하여 암호화된 보호 프로파일("프로파일")을 복호화하는 블록(493)으로 진행한다. 블록(491)에서 보호 동적-링크 라이브러리는 프로파일을 시스템과 비교한다. 그 다음, 판정 블록(489)에서, 프로파일이 발견되었는지를 판정하기 위한 검사가 수행된다. 각 프로파일은 64 비트와 같은 복수의 비트인 것이 바람직한 이름을 가진다. 판정 블록(489)에서의 검사 결과가 "아니오"라면, 방법(400)은 상술한 처리 단계들을 반복하는 블록(493)으로 다시 루핑한다. 반면에, 판정 블록(489)에서의 검사 결과가 "예"라면, 방법(400)은 보호 동적-링크 라이브러리가 프로파일의 식별자, 그 데이터, 비교 결과 및 솔트 값에 대한 체크섬을 계산하는 블록(487)으로 진행한다. 블록(485)에서 보호 동적-링크 라이브러리는 결과 데이터 구조, 프로파일의 검증 블랍, 및 계산된 체크섬을 반환한다. 그 다음 방법은 다른 연속 단말("단말 E2")로 계속된다.

[0072] (도 4n의) 단말 E2로부터, 방법(400)은 로그-온 모듈이 개인 호출 키를 이용하여 검증 블랍을 복호화하고 검사 체크섬을 획득하는 블록(483)으로 진행한다. 블록(481)에서 로그-온 모듈은 공개 호출 키를 이용하여 체크섬의 서명을 검사한다. 그 다음, 블록(479)에서, 로그-온 모듈은 복호화된 검사 체크섬, 수신된 결과 데이터 구조, 및 솔트 값으로부터 체크섬을 계산한다. 방법(400)은 이 방법이 로그-온 모듈에 의해 계산된 체크섬과 보호 동적-링크 라이브러리를 비교하는 블록(477)으로 진행한다. 판정 블록(475)에서 체크섬들이 일치하는지를 판정하기 위한 검사가 수행된다. 판정 블록(475)에서의 검사 결과가 "아니오"라면, 방법(400)은 단말 L로 계속되고 실행은 종료된다. 그렇지 않고, 판정 블록(475)에서의 검사 결과가 "예"라면, 방법(400)은 다른 연속 단말("단말 E3")로 계속된다.

[0073] (도 4o의) 단말 E3로부터, 방법(400)은 이 방법이 프로파일로부터 배타적인 컴퓨팅 기기 종류들의 리스트를 추출하는 블록(473)으로 진행한다. 그 다음 블록(471)에서 방법은 컴퓨팅 기기로 하여금 CPU 정보(예를 들면, 벤더 신원, 유형, 계열, 모델 번호, 브랜드 신원, 및 특징 세트)를 검토하도록 한다. 그 다음, 판정 블록(469)에서, 벤더 신원이 배타적인 리스트 상에 있는지를 판정하기 위한 검사가 수행된다. 판정 블록(469)에서의 검사 결과가 "아니오"라면, 방법(400)은 단말 F로 계속된다. 그렇지 않고, 검사 결과가 "예"라면, 방법은 다른 연속 단말("단말 E4")로 계속된다.

[0074] (도 4p의) 단말 E4로부터, 방법(400)은 유형, 계열, 및 모델 번호가 배타적인 리스트 상에 있는지 판정하기 위한 검사가 수행되는 판정 블록(467)으로 계속된다. 판정 블록(467)에서의 검사 결과가 "예"라면, 방법(400)은 다른 연속 단말("단말 E7")로 계속된다. 그렇지 않고, 판정 블록(467)에서의 검사 결과가 "아니오"라면, 방법(400)은 브랜드 신원 및 특징 세트가 배타적인 리스트 상에 있는지를 판정하기 위한 검사가 수행되는 다른 판정 블록(465)으로 계속된다. 판정 블록(465)에서의 검사 결과가 "예"라면, 방법(400)은 단말 E7로 계속된다. 그렇지 않고, 판정 블록(465)에서의 검사 결과가 "아니오"라면, 방법(400)은 다른 연속 단말("단말 E5")로 계속된다.

[0075] (도 4q의) 단말 E5로부터, 방법(400)은 컴퓨팅 기기 신원의 CPU, 즉 중앙 처리 장치를 결정하기 위한 확장된 명령어가 존재하는지를 판정하는 검사가 수행되는 판정 블록(463)으로 계속된다. 판정 블록(463)에서의 검사 결과가 "아니오"라면, 방법은 단말 F로 계속된다. 그렇지 않고, 판정 블록(463)에서의 검사 결과가 "예"라면, 블록(461)에서 방법은 컴퓨팅 기기의 CPU, 즉 중앙 처리 장치의 이름을 획득하기 위한 확장된 명령어를 호출한다. 그 다음, 블록(459)에서, 방법은 중앙 처리 장치의 캐쉬 사이즈를 획득하기 위한 확장된 명령어를 호출한다. 방법(400)은 다른 연속 단말("단말 E6")로 계속된다.

[0076] (도 4r의) 단말 E6로부터, 방법(400)은 중앙 처리 장치의 이름이 배타적인 리스트 상에 있는지를 판정하는 검사가 수행되는 판정 블록(457)으로 진행한다. 판정 블록(457)에서의 검사 결과가 "예"라면, 방법은 단말 E7로 계속된다. 판정 블록(457)에서의 검사 결과가 "아니오"라면, 판정 블록(455)에서 CPU의 캐쉬 사이즈가 배타적인 리스트 상에 있는지를 판정하기 위한 다른 검사가 수행된다. 판정 블록(455)에서의 검사 결과가 "예"라면, 방법(400)은 단말 E7로 계속된다. 그렇지 않고, 결과가 "아니오"라면, 방법(400)은 단말 F로 계속된다.

- [0077] (도 4s의) 단말로부터, 블록(455)에서 방법은 컴퓨팅 기기 상에 어떠한 보호도 유용하지 않다는 오류 메시지를 발행하고 시스템을 종료하기 시작한다. 그 다음 방법(400)은 단말 L로 계속되고 실행을 종료한다.
- [0078] (도 4b의) 단말 F로부터, 방법(400)은 연속 단말("단말 G")과 종료 단말("단말 H") 사이에 정의된 방법 단계 세트(408)로 계속된다. 방법 단계 세트(408)는 설치 파일이 변조되었는지 검사한다.
- [0079] (도 4s의) 단말 G로부터, 블록(453)에서 로그-온 모듈은 보호 동적-링크 라이브러리와 통신하여 컴퓨팅 기기 상의 중대한 파일과 관련된 암호화된 보호 파일을 액세스한다. 블록(451)을 참조하면 (도시된 일련의) 도 41 내지 4n의 단계들(495 내지 475)은 컴퓨팅 기기 상의 중대한 파일에 관련된 암호화된 보호 프로파일을 발견하는 관점에서 실행된다. 그 다음, 블록(449)에서, 방법은 프로파일로부터 중대한 파일 리스트를 추출한다. 블록(447)에서 컴퓨팅 기기 상의 중대한 파일이 변경되었는지 판정하는 검사가 수행된다. 판정 블록(447)에서의 검사 결과가 "예"라면, 방법(400)은 단말 L로 계속되고 실행을 종료한다. 그렇지 않고, 판정 블록(447)에서의 검사 결과가 "아니오"라면, 방법은 단말 H로 계속된다. 단말 G와 단말 H 사이의 처리 단계들은 동시에 실행되어 컴퓨팅 기기 상의 중대한 파일들이 변경되거나 변조되었는지를 계속하여 체크할 수 있음을 유의한다.
- [0080] (도 4b의) 단말 H로부터, 방법(400)은 연속 단말("단말 I")과 종료 단말("단말 J") 사이에 정의된 방법 단계 세트(410)로 진행한다. 방법 단계 세트(410)는 레지스트리가 변조되었는지를 판정한다.
- [0081] (도 4t의) 단말 I로부터, 블록(445)을 참조하면, 시동 시에, 로그-온 모듈은 시스템 환경의 레지스트리의 값 및 키를 보호하기 위해 보호 동적-링크 라이브러리의 보호 함수를 호출한다. 블록(443)에서 보호 동적-링크 라이브러리는 시스템 환경의 레지스트리의 각종 키 및 값을 보호하기 위한 스레드를 생성한다. 그 다음, 블록(441)에서, 보호 동적-링크 라이브러리는 방금 생성된 스레드에 대한 핸들을 로그-온 모듈에 반환한다. 방법(400)은 이 방법이 시스템 환경에서 레지스트리의 값 및 키를 보호하는 스레드의 존속을 체크하는 타이머를 생성하는 블록(439)으로 진행한다. 그 다음, 방법(400)은 연속 단말("단말 I2"), 판정 블록(437), 및 다른 연속 단말("단말 I4")로 나타낸 3개의 독립적인 실행 경로로 진행한다. 이들 독립적인 실행 경로는 동시 발생을 나타낸다. 판정 블록(437)에서 미리-구성된 기간이 만료되었는지를 판정하기 위한 검사가 수행된다. 판정 블록(437)에서의 검사 결과가 "아니오"라면, 방법(400)은 다른 연속 단말("단말 I3")로 진행하고 상술한 처리 단계가 반복되는 판정 블록(437)으로 다시 루핑한다. 그렇지 않고, 판정 블록(437)에서의 검사 결과가 "예"라면, 방법(400)은 다른 연속 단말("단말 I1")로 계속된다.
- [0082] (도 4u의) 단말 I1으로부터, 방법(400)은 스레드가 종료되었는지 판정하기 위한 검사가 수행되는 판정 블록(435)으로 진행한다. 판정 블록(435)에서의 검사 결과가 "아니오"라면, 방법은 스레드가 보류되었는지를 판정하기 위한 다른 검사가 수행되는 판정 블록(433)으로 계속된다. 판정 블록(433)에서의 검사 결과가 "아니오"라면, 방법(400)은 단말 I3로 계속되어 상술한 처리 단계가 반복되는 판정 블록(437)으로 다시 루핑한다. 판정 블록(435, 433)에서의 검사 결과가 "예"라면, 방법(400)은 시스템이 변조되었다고 여기는 블록(431)으로 진행하여 방법은 시스템을 종료하기 시작한다. 그 다음 방법(400)은 단말 L로 진행하고 실행을 종료한다.
- [0083] (도 4v의) 단말 I2로부터, 방법(400)은 스레드가 보호 동적-링크 라이브러리와 통신하여 보호될 시스템 레지스트리 키 및 값의 리스트를 액세스하는 블록(429)으로 진행한다. 블록(427)에 나타난 바와 같이, 도 41 내지 4n의 (도시된 일련의) 단계들(495 내지 475)이 수행된다. 그 다음, 블록(425)에서, 스레드는 레지스트리 키를 보호하기 위하여 임의의 변경들의 통지에 가입한다. 판정 블록(423)에서 변경의 통지가 있는지를 판정하기 위한 검사가 수행된다. 판정 블록(423)에서의 검사 결과가 "아니오"라면, 방법은 상술한 처리 단계가 반복되는 판정 블록(423)으로 다시 루핑한다. 그렇지 않고, 판정 블록(423)에서의 검사 결과가 "예"라면, 블록(421)에서 방법은 변조된 키 및 그 값을 교정한다. 그 다음 방법(400)은 상술한 처리 단계들이 반복되는 판정 블록(423)으로 다시 루핑한다.
- [0084] (도 4w의) 단말 I4로부터, 방법(400)은 사용자에게 의한 로그-온 중에, 로그-온 모듈이 사용자 환경의 레지스트리의 값 및 키를 보호하기 위한 보호 동적-링크 라이브러리의 보호 함수를 호출하는 블록(419)으로 진행한다. 블록(417)에서 보호 동적-링크 라이브러리는 사용자 환경에서 레지스트리의 각종 키 및 값을 보호하기 위한 스레드를 생성한다. 그 다음, 블록(415)에서, 보호 동적-링크 라이브러리는 방금 생성된 스레드에 대한 핸들을 로그-온 모듈에 반환한다. 방법(400)은 이 방법이 사용자 환경의 레지스트리의 값 및 키를 보호하는 스레드의 존속을 체크하는 타이머를 생성하는 블록(413)으로 진행한다. (도시된 일련의) 도 4t 내지 4u의 단계들(437 내지 421)은 사용자 환경에 대해 실행된다. 그 다음 방법(400)은 단말 J로 계속된다.
- [0085] (도 4b의) 단말 J로부터, 방법(400)은 연속 단말("단말 K")과 종료 단말("단말 L") 사이에 정의된 방법 단계

세트(412)로 진행한다. 방법 세트(412)는 서비스들이 변조되었거나 권한이 없는 서비스가 수행되고 있는지를 판정한다. (도 4x의) 단말 k로부터, 시동 시에, 로그-온 모듈은 권한이 없는 서비스의 실행을 방지하기 위한 보호 동적-링크 라이브러리의 보호 함수를 호출한다. 블록(407)에서 보호 동적-링크 라이브러리는 권한이 없는 서비스의 실행을 방지하기 위한 스레드를 생성한다. 그 다음, 블록(405)에서, 보호 동적-링크 라이브러리는 방금 생성된 스레드에 대한 핸들을 로그-온 모듈에 반환한다. 그 다음 방법(400)은 이 방법이 권한이 없는 서비스들의 실행을 방지하는 스레드의 존속을 체크하는 타이머를 생성하는 블록(403)으로 계속된다. 블록(401)을 참조하면 도 4t 내지 4u의 단계(437 내지 431)가 실행된다. 그 다음, 블록(401A)에서, 스레드는 보호 동적-링크 라이브러리와 통신하여 권한이 없는 서비스들의 리스트에 관련된 암호화된 보호 프로파일을 액세스한다. 그 다음 방법(400)은 다른 연속 단말("단말 K1")로 계속된다.

[0086] (도 4y의) 단말 K1으로부터, 도 4l 내지 4n의 단계들(495 내지 475)이 수행된다. 판정 블록(401C)에서 권한이 없는 서비스가 실행되는지를 판정하기 위한 검사가 수행된다. 판정 블록(401C)의 검사 결과가 "아니오"라면, 방법(400)은 다른 연속 단말("단말 K2")로 계속된다. 그렇지 않고, 판정 블록(401C)에서의 검사 결과가 "예"라면, 방법(400)은 이 방법이 권한이 없는 서비스에 의존하는 모든 서비스들을 중지시키는 블록(401D)으로 진행한다. 그 다음 블록(401E)에서 권한이 없는 서비스가 실행 목록으로부터 삭제된다.

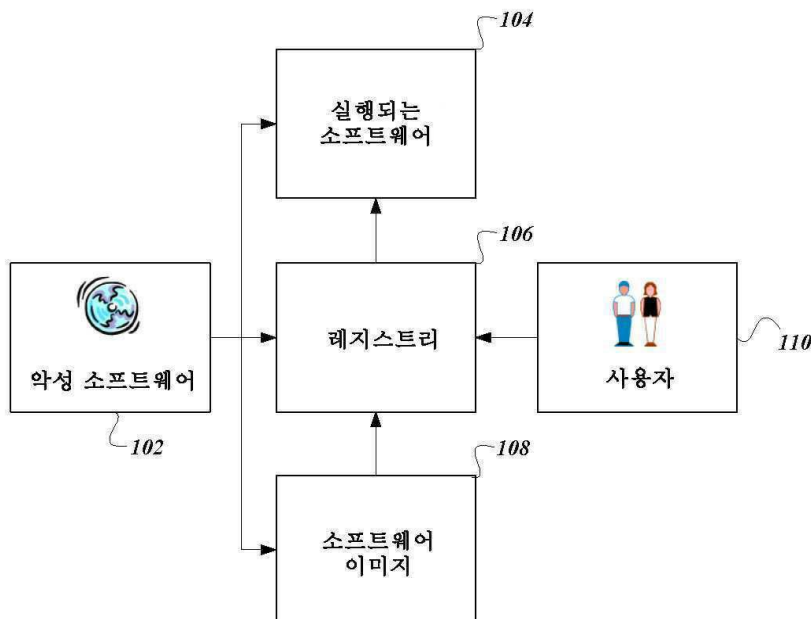
[0087] (도 4j의) 단말 K2로부터, 방법(400)은 권한이 없는 서비스가 더 있는지 판정하기 위한 검사가 수행되는 판정 블록(401F)으로 진행한다. 판정 블록(401F)에서의 검사 결과가 "예"라면, 방법은 다른 연속 단말("단말 K3")로 계속된다. (도 4y의) 단말 K3로부터, 방법(400)은 상술한 처리 단계가 반복되는 판정 블록(401C)으로 다시 루핑된다. 그렇지 않고, 판정 블록(401F)에서의 검사 결과가 "아니오"라면, 방법(400)은 스레드가 미리-구성된 기간 동안 슬립(sleep) 상태에 있는 블록(401G)으로 계속된다. 그 다음 방법(400)은 상술한 처리 단계들이 반복되는 단말 K3를 통하는 판정 블록(401C)으로 다시 루핑된다. 단말 K와 L 사이에 정의된 방법 단계 세트(412)와 관련하여 기술된 단계들은 다른 방법 단계 세트와 동시에 실행되는 것이 바람직하다.

[0088] 본 발명의 바람직한 실시예가 도시되고 기술되었지만, 본 발명의 사상 및 범위로부터 벗어나지 않는 다양한 변경이 이루어질 수 있다고 인식될 것이다.

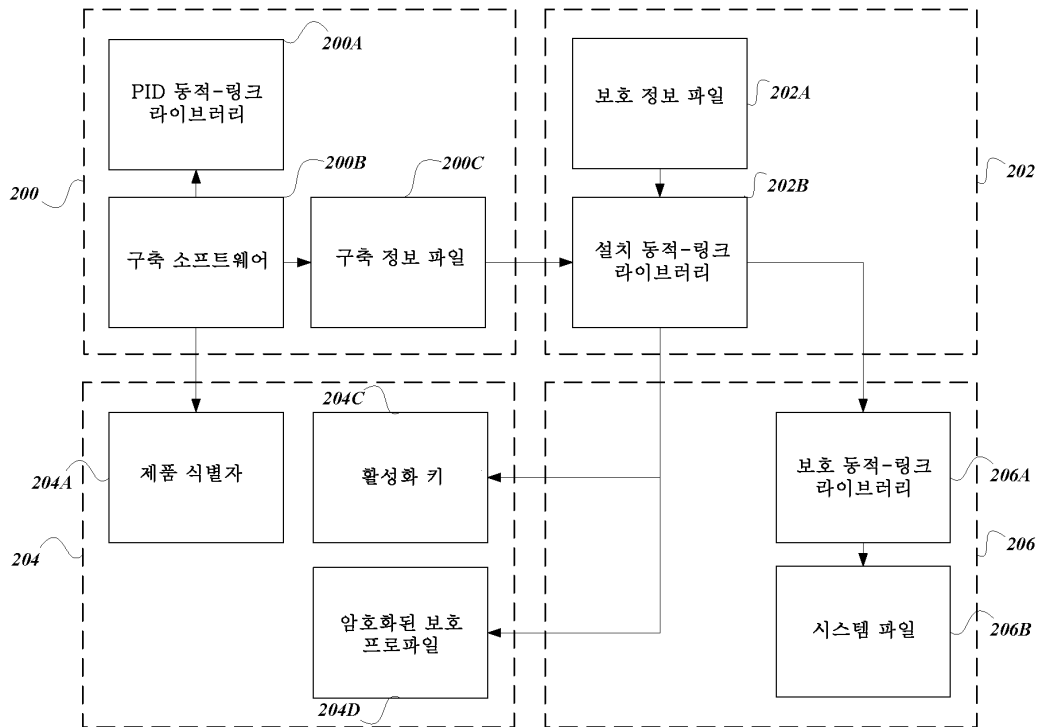
[0089] 독점적인 특징 또는 특권이 청구된 본 발명의 실시예가 이하 정의된다.

도면

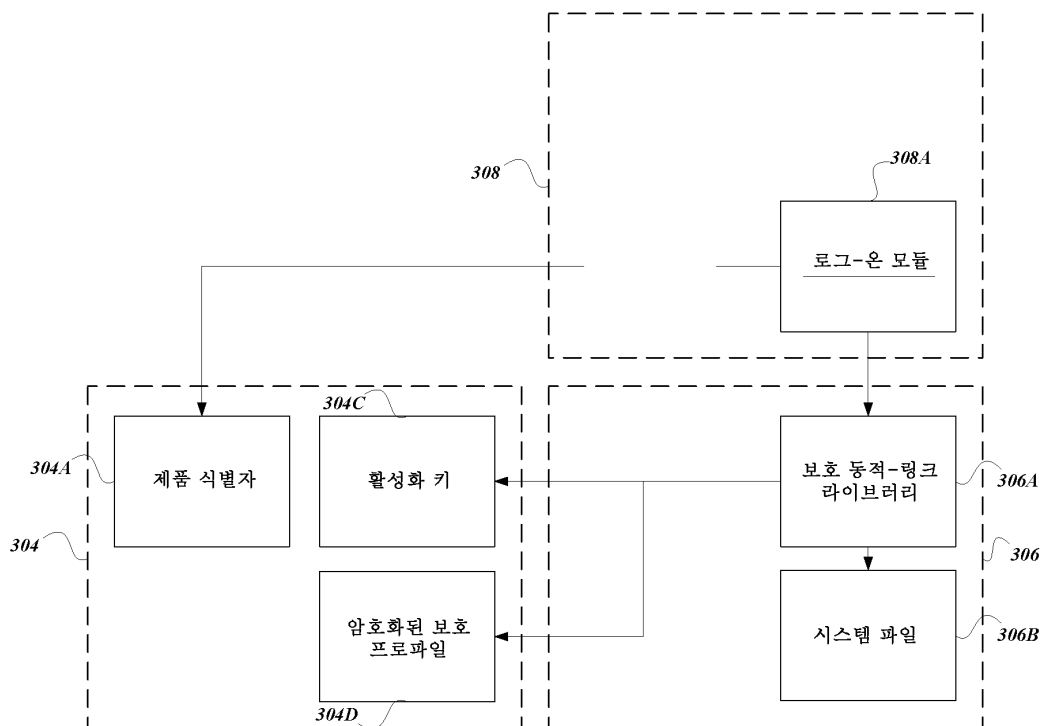
도면1



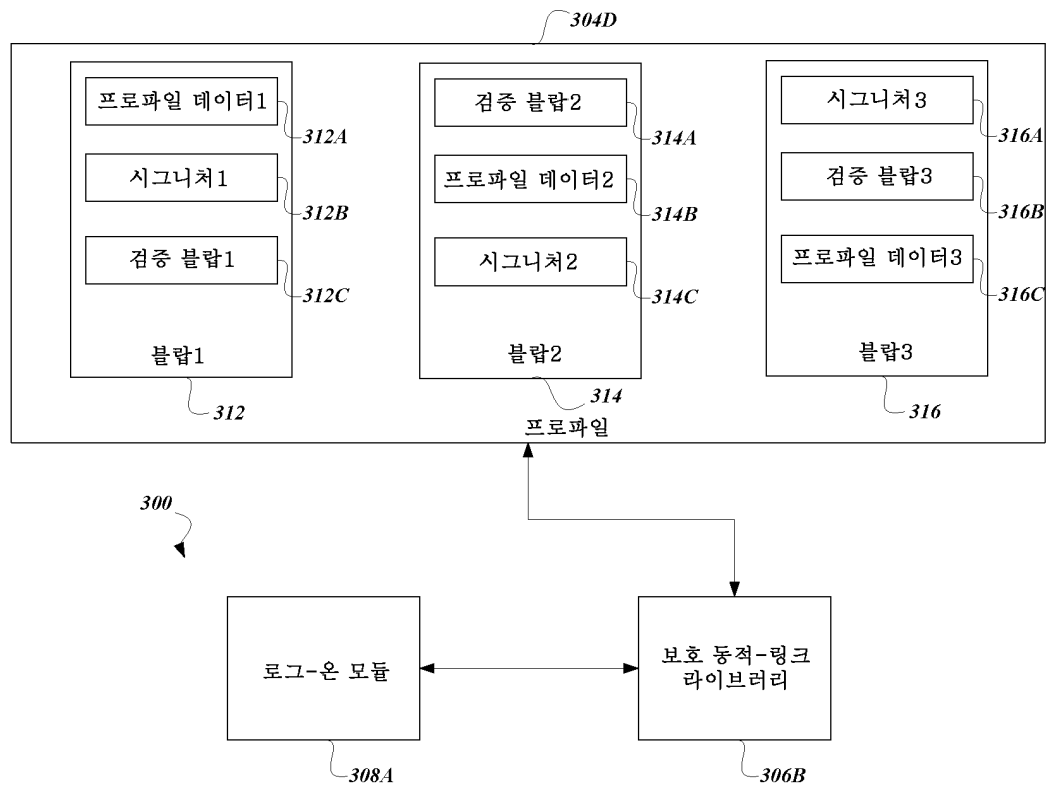
도면2



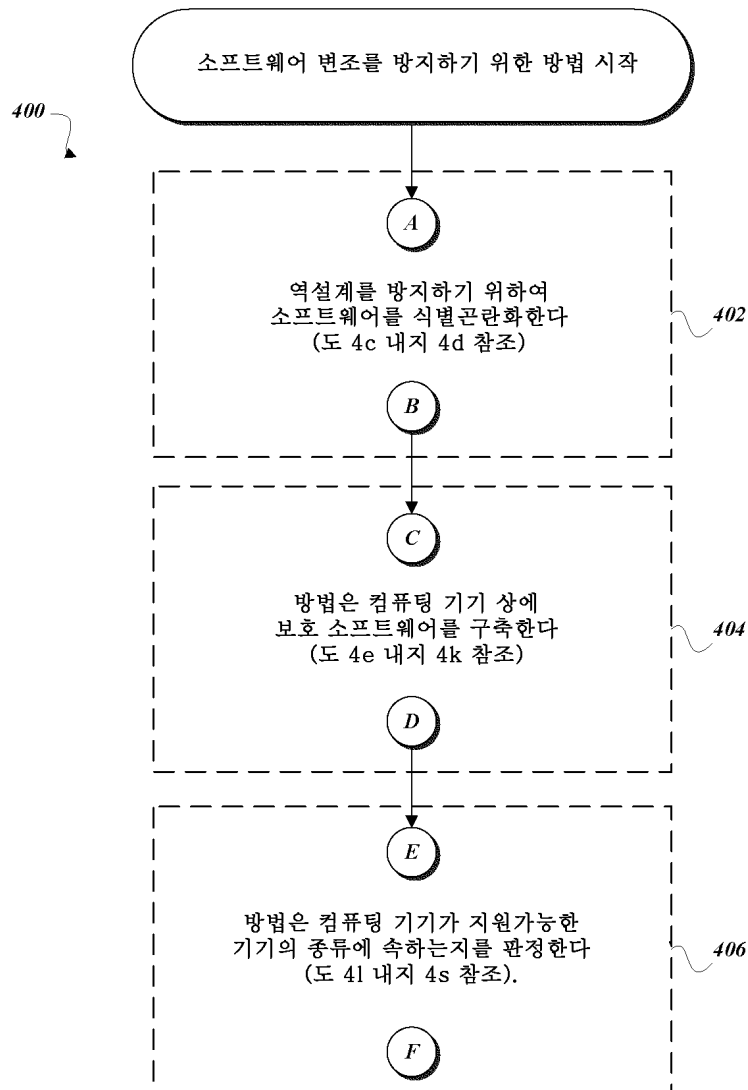
도면3a



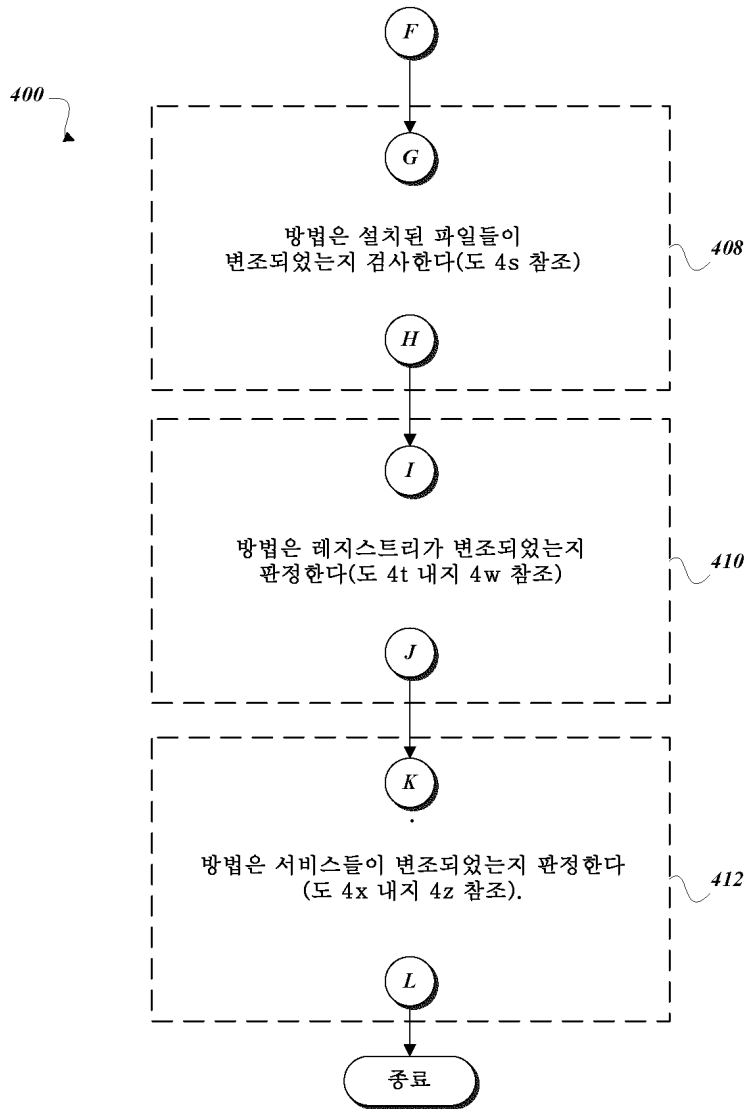
도면3b



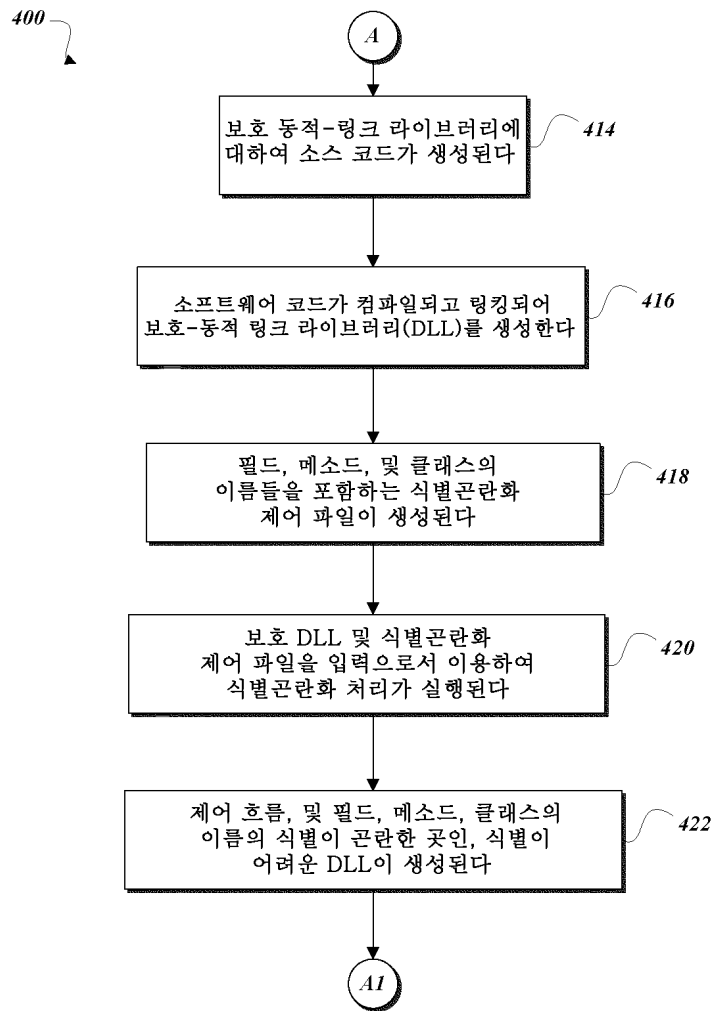
도면4a



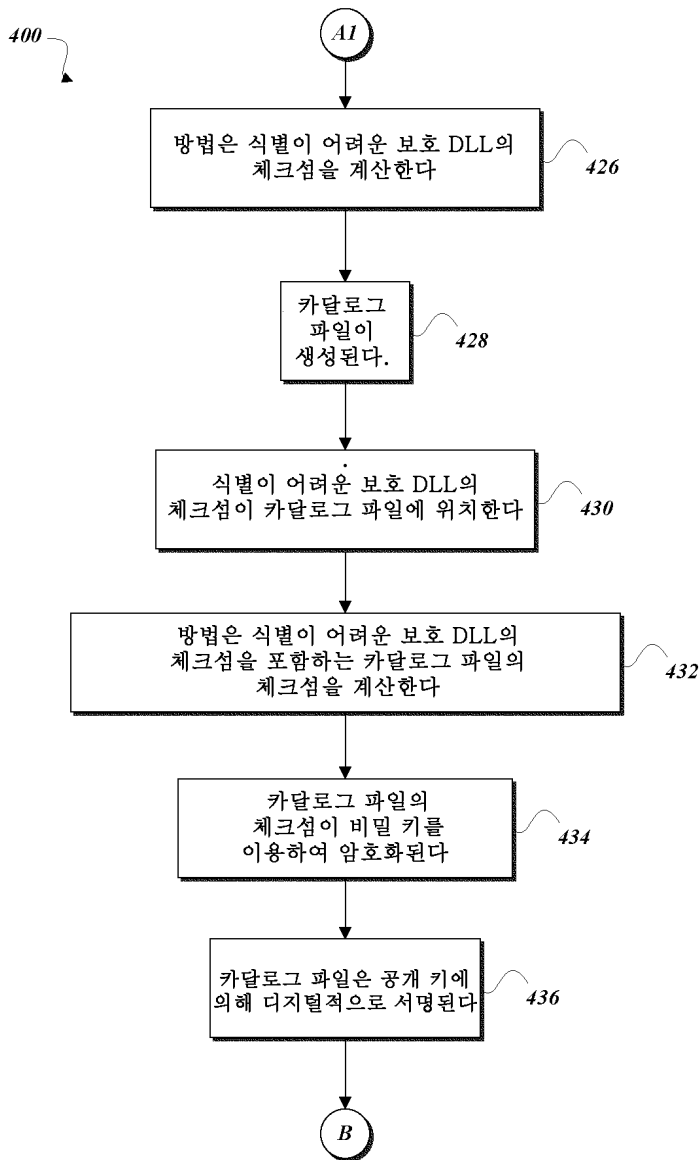
도면4b



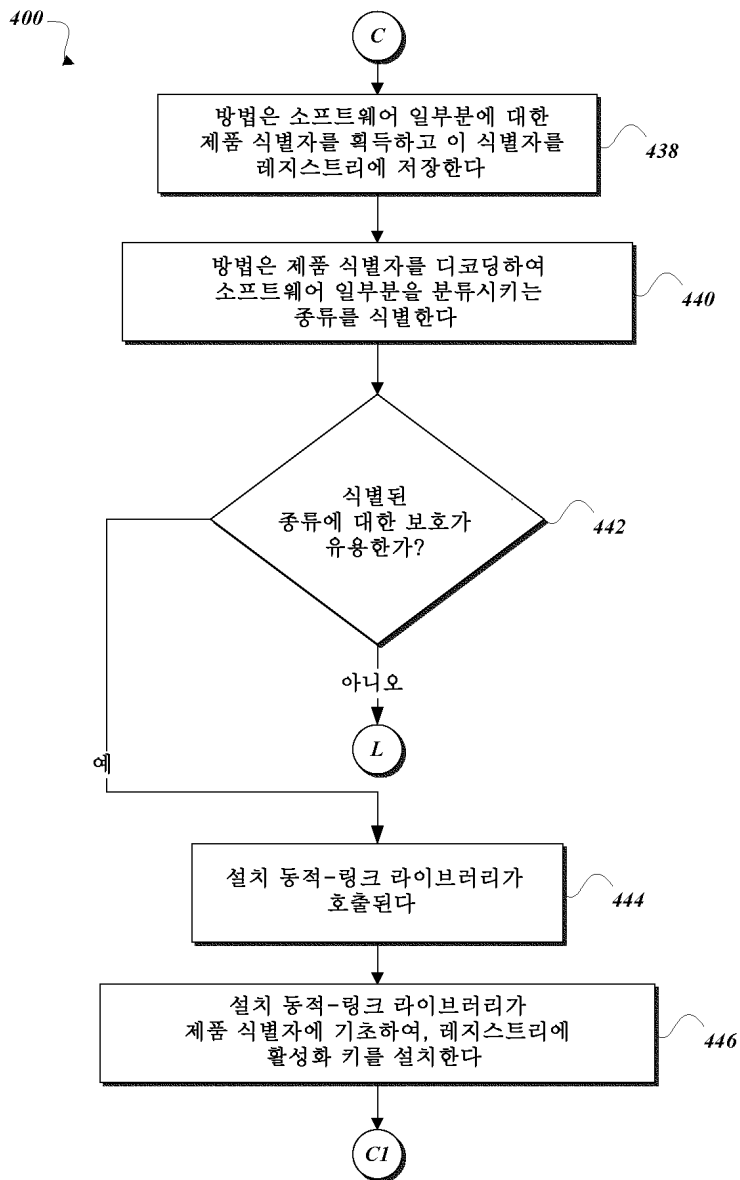
도면4c



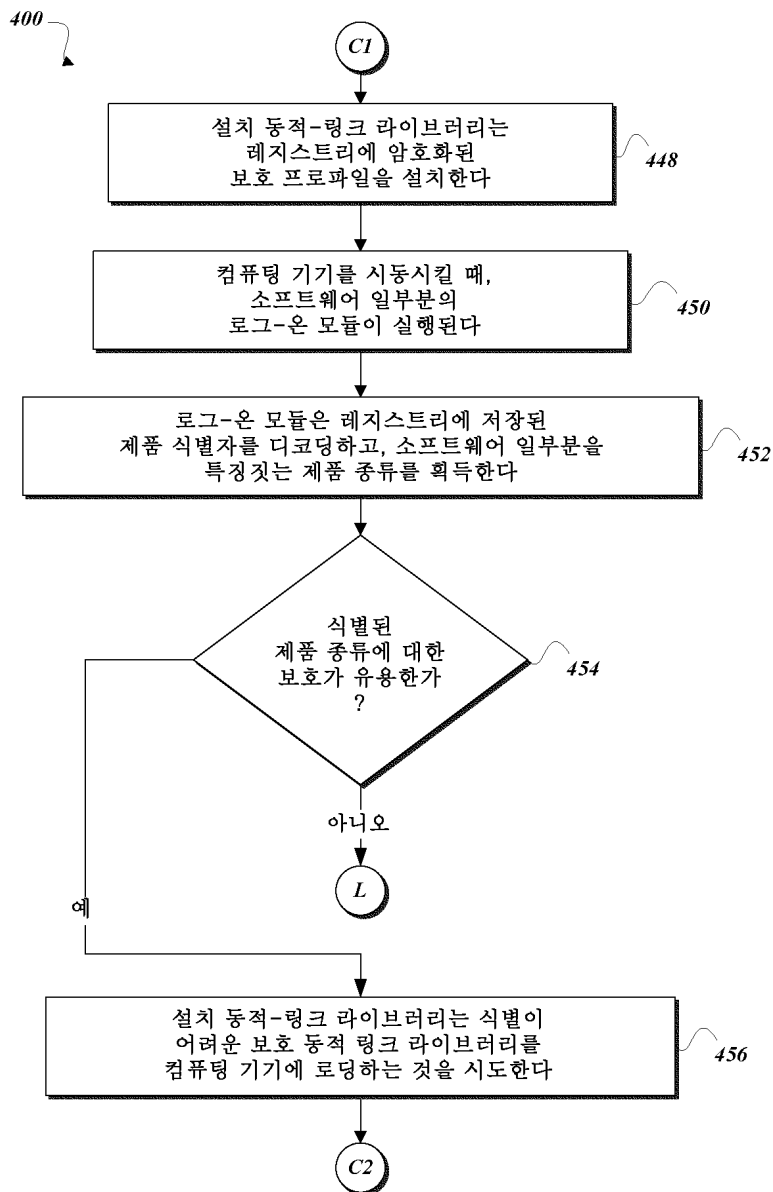
도면4d



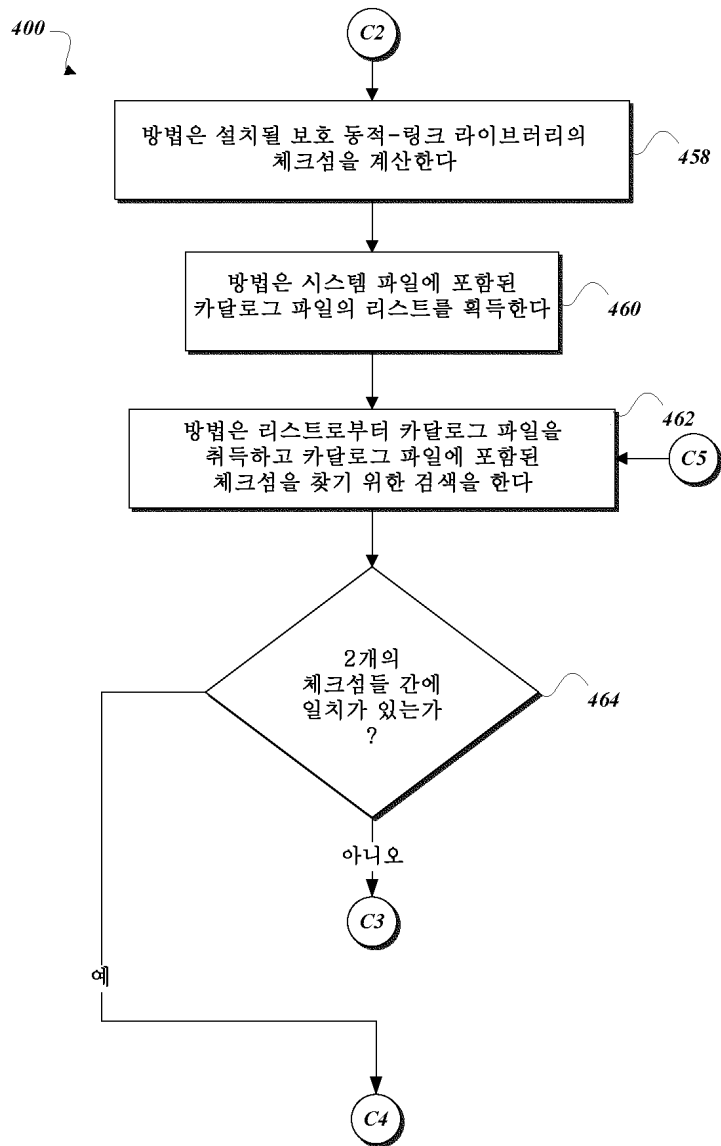
도면4e



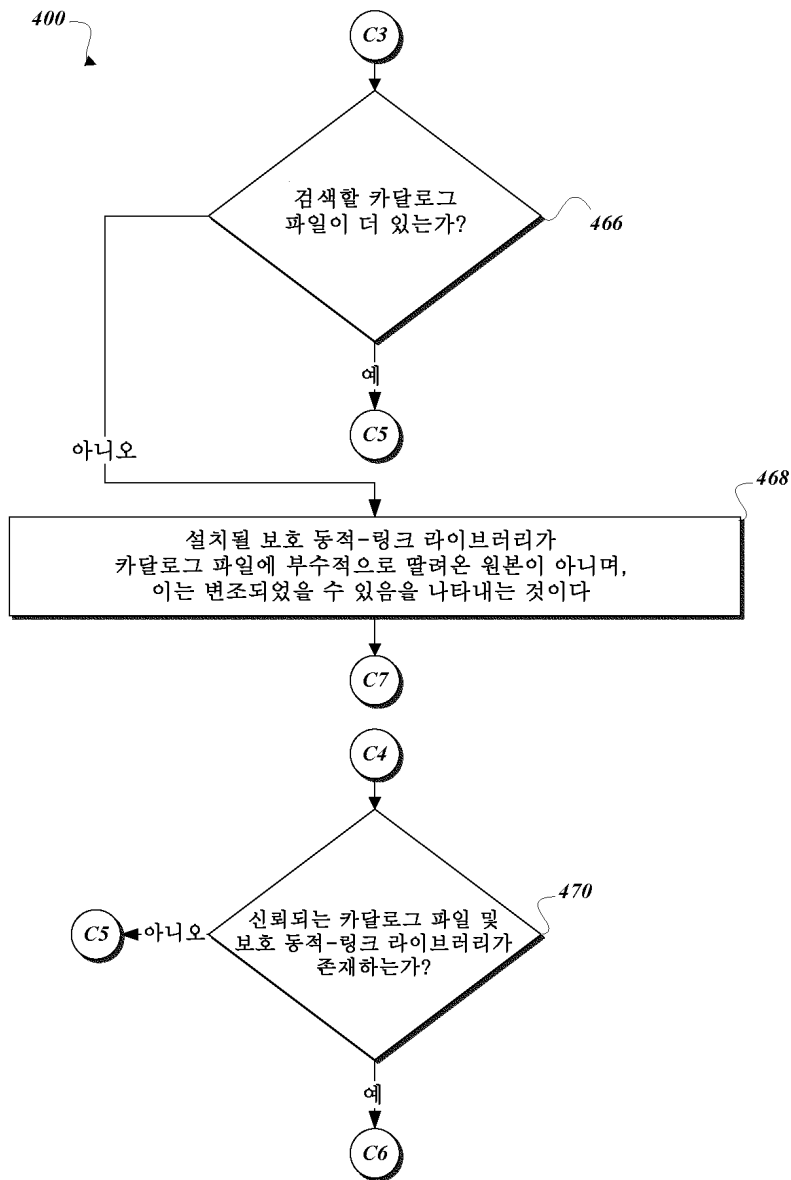
도면4f



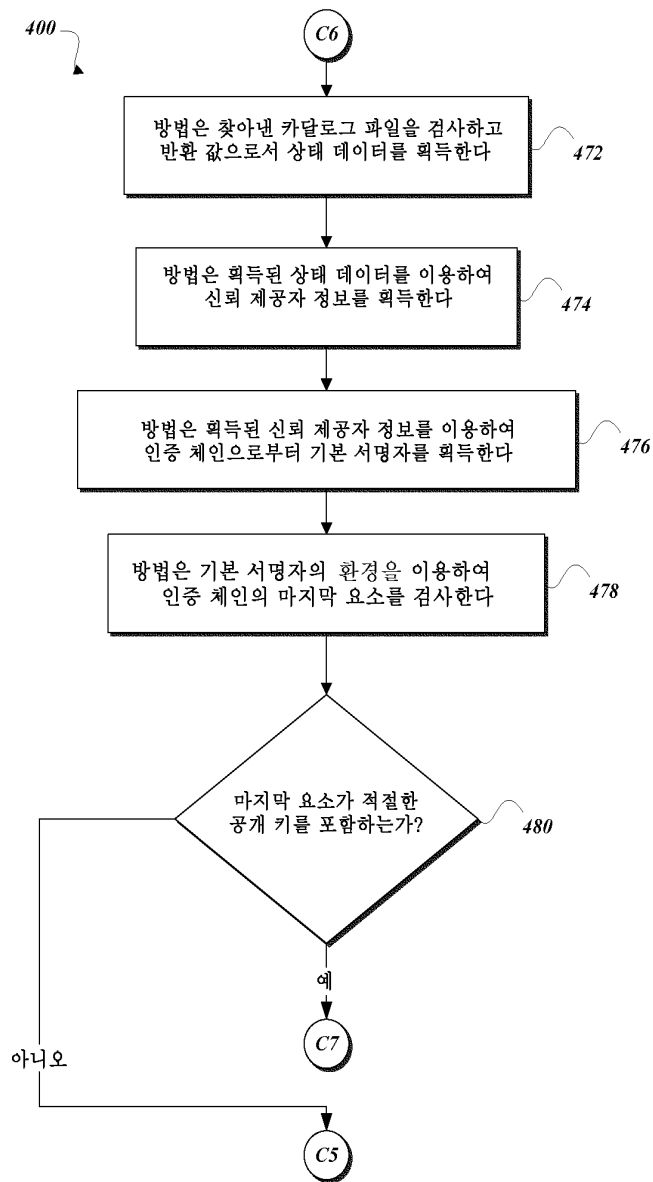
도면4g



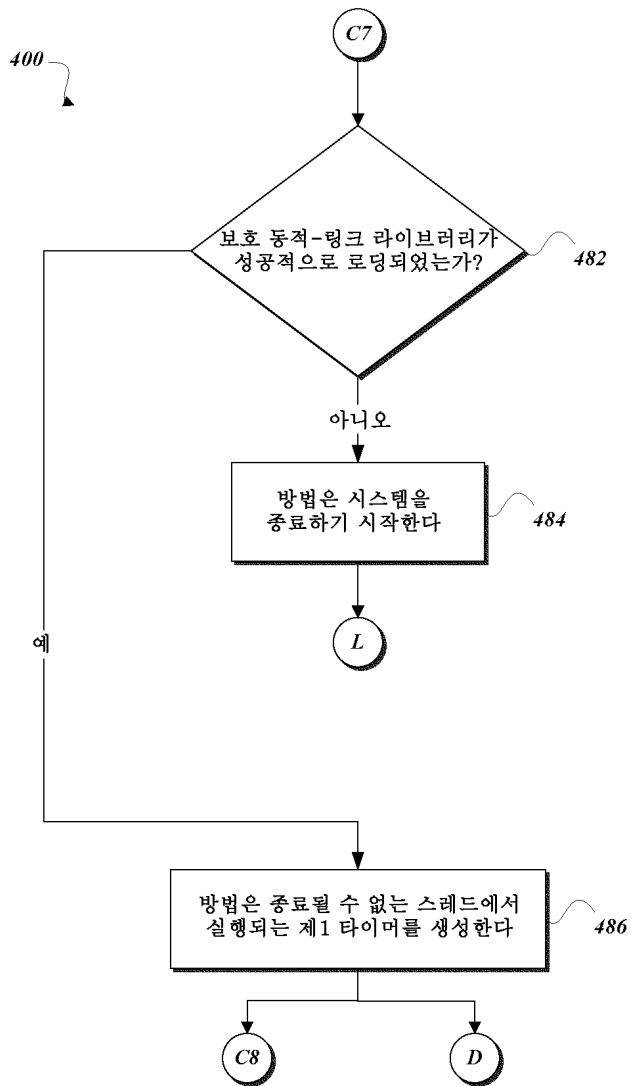
도면4h



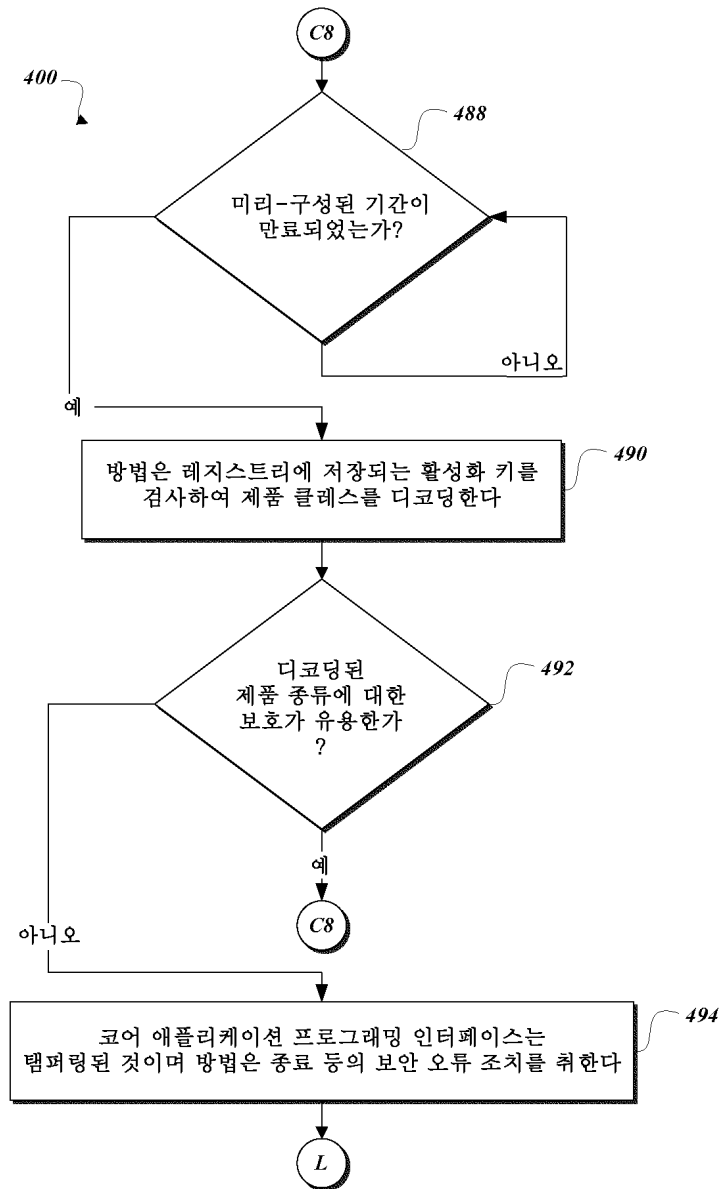
도면4i



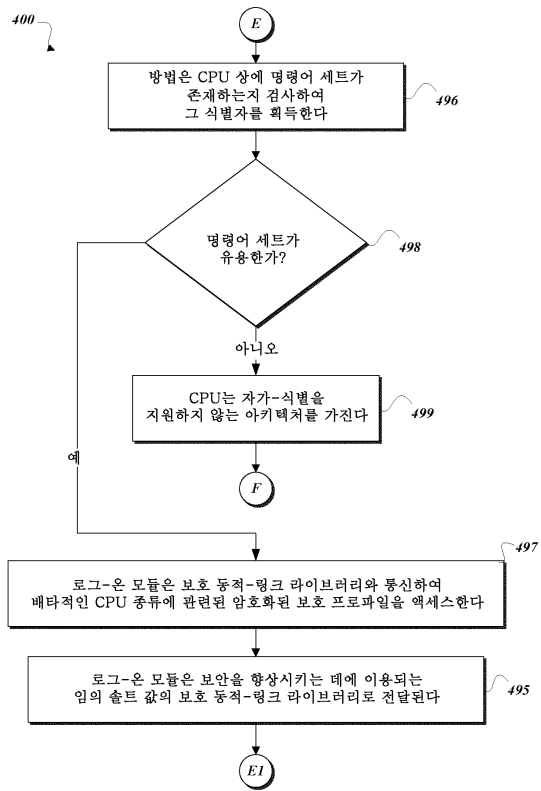
도면4j



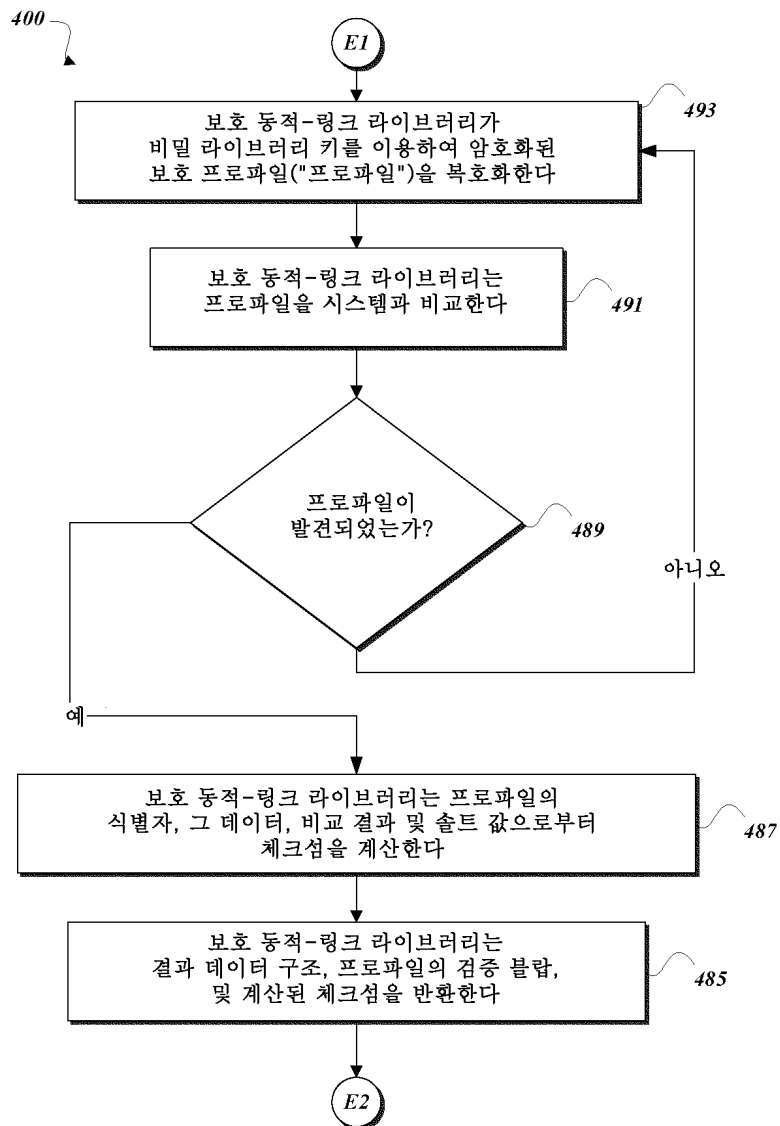
도면4k



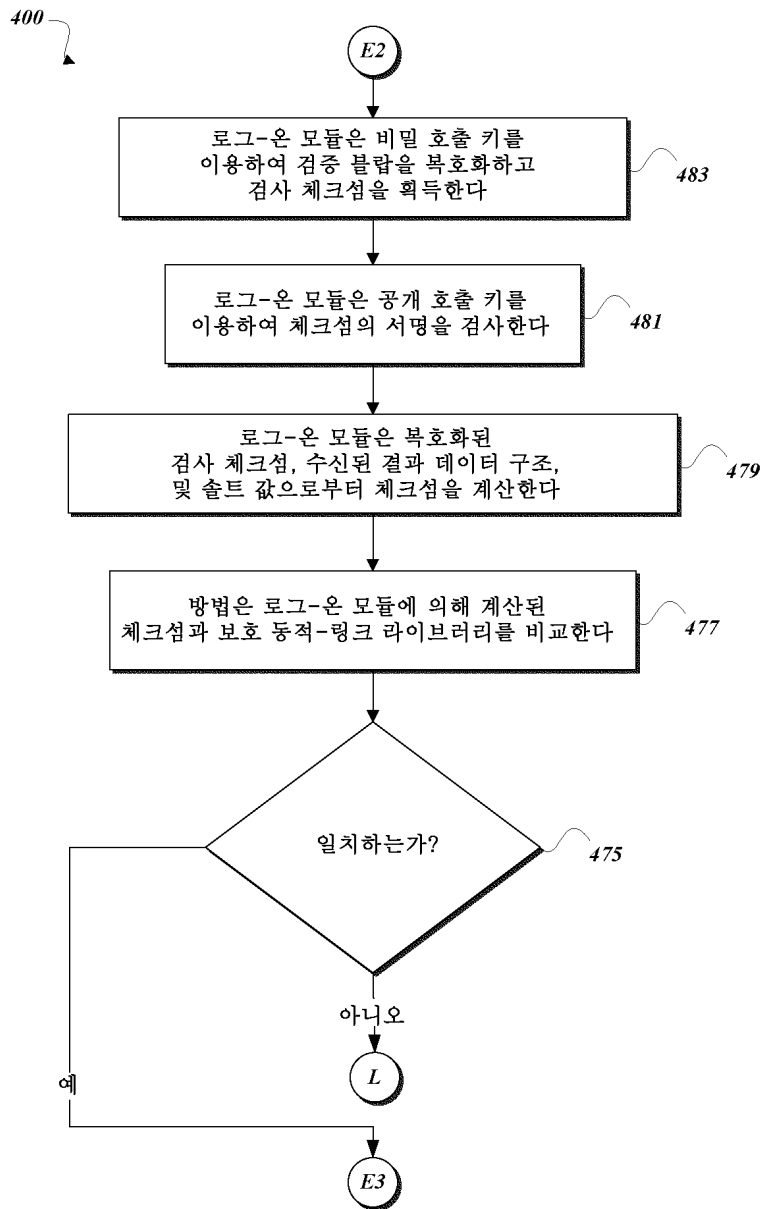
도면41



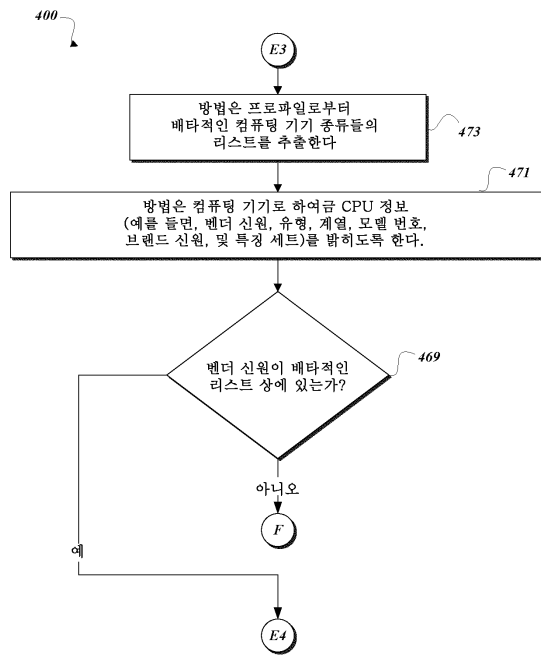
도면4m



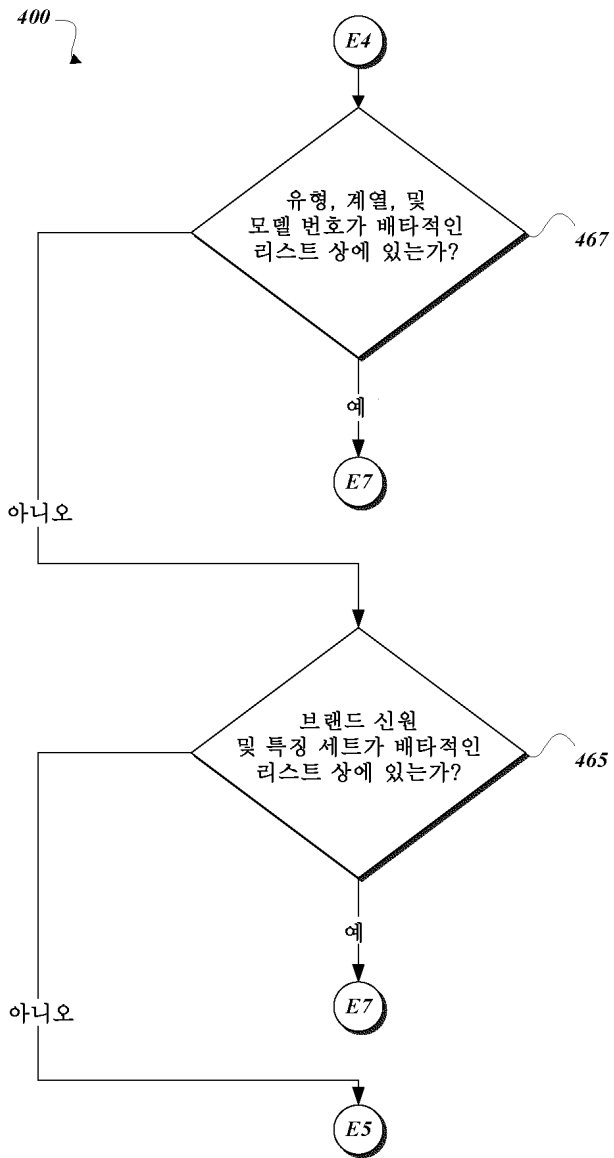
도면4n



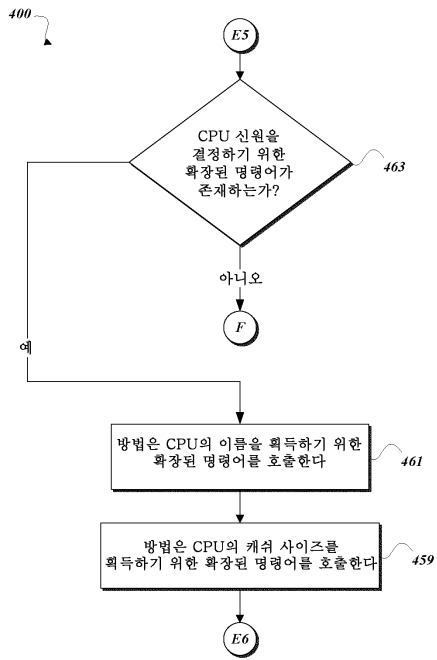
도면40



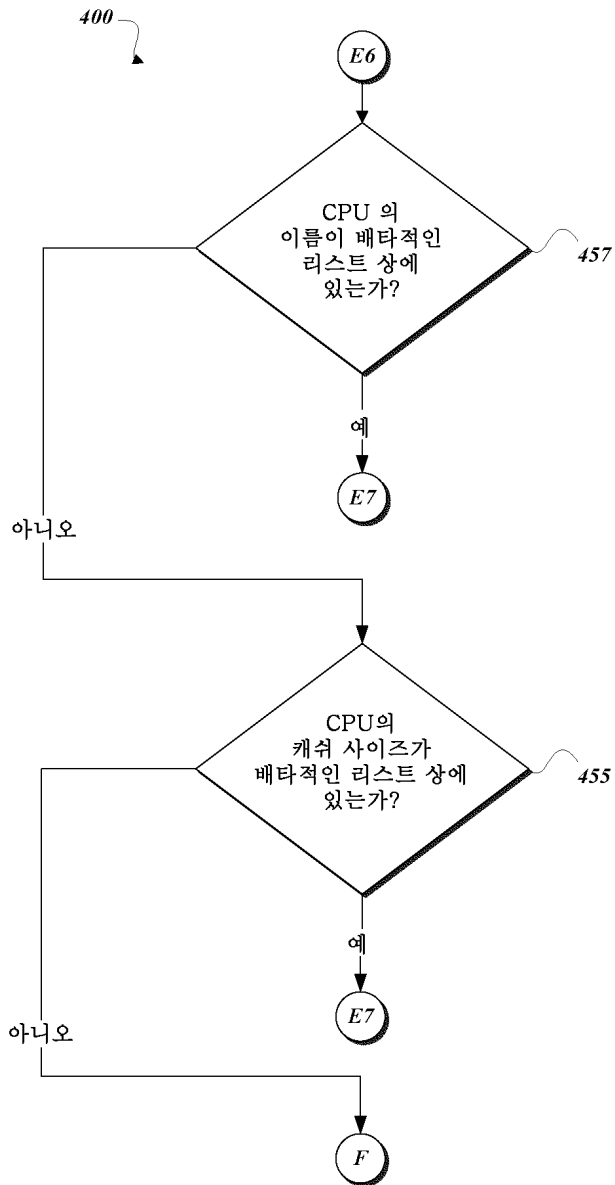
도면4p



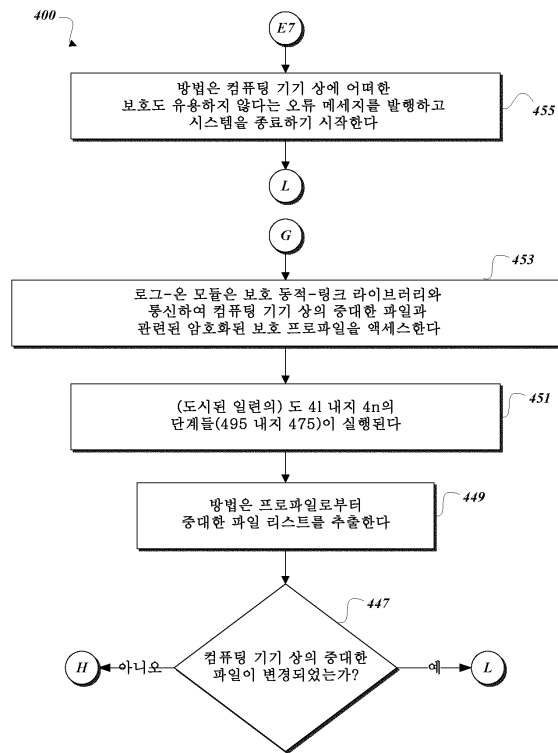
도면4q



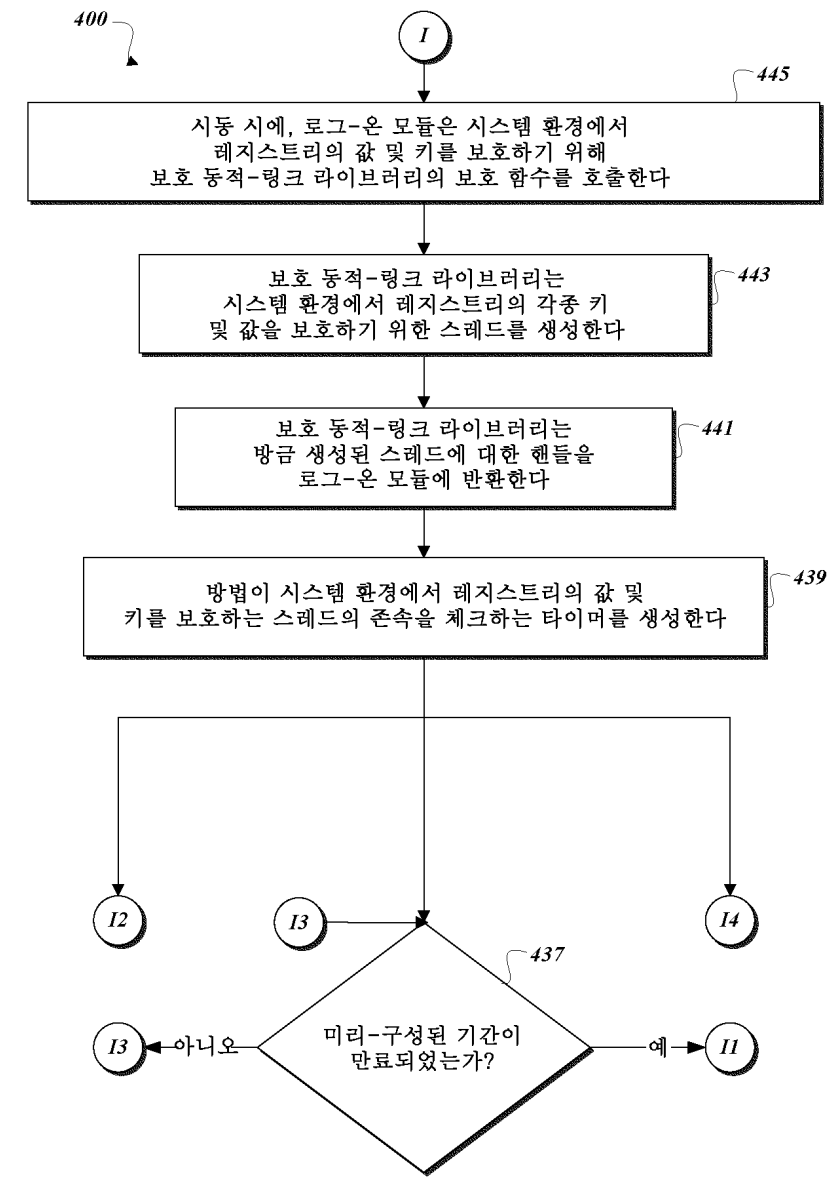
도면4r



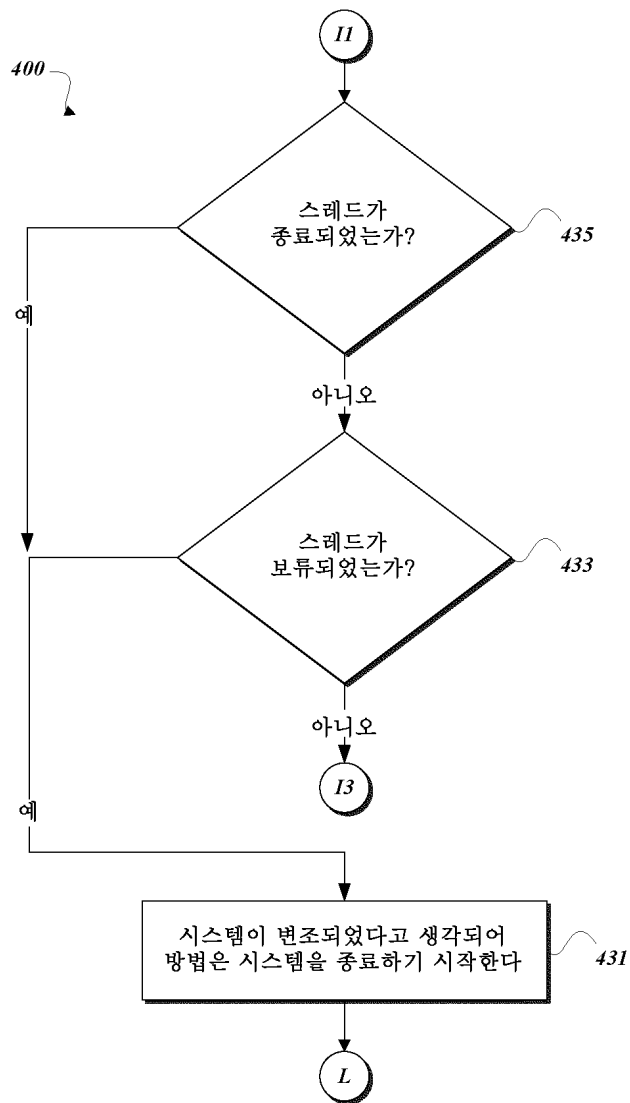
도면4s



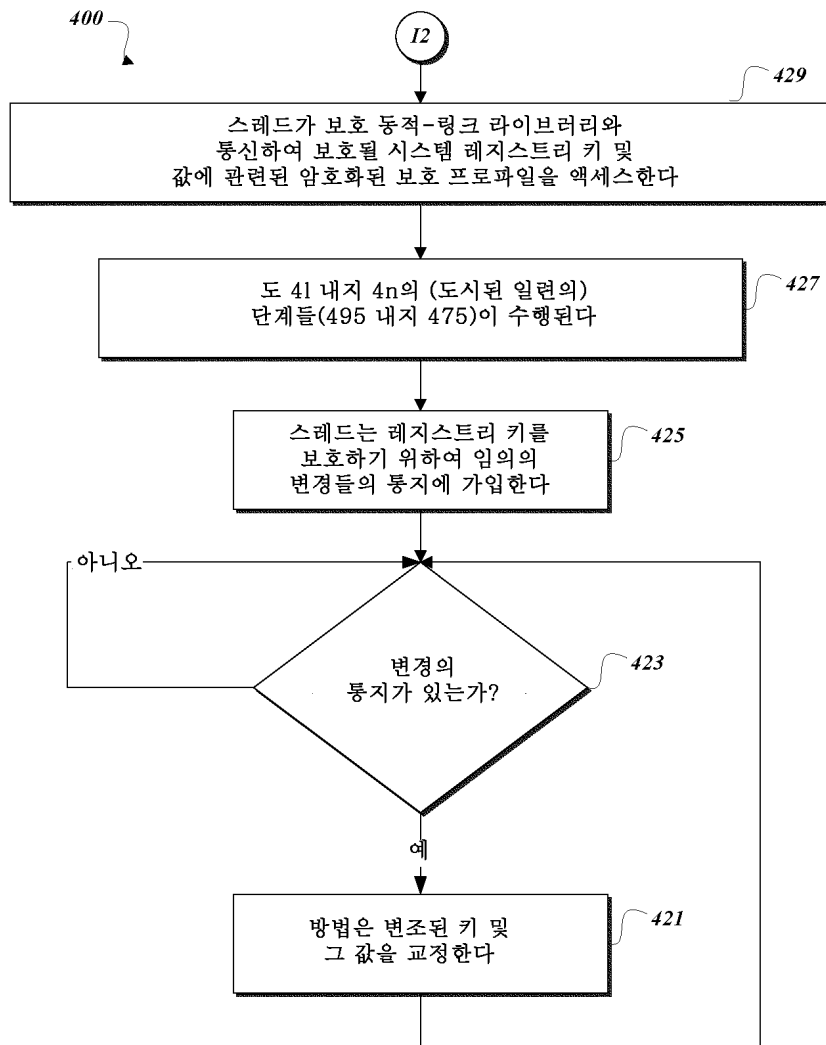
도면4t



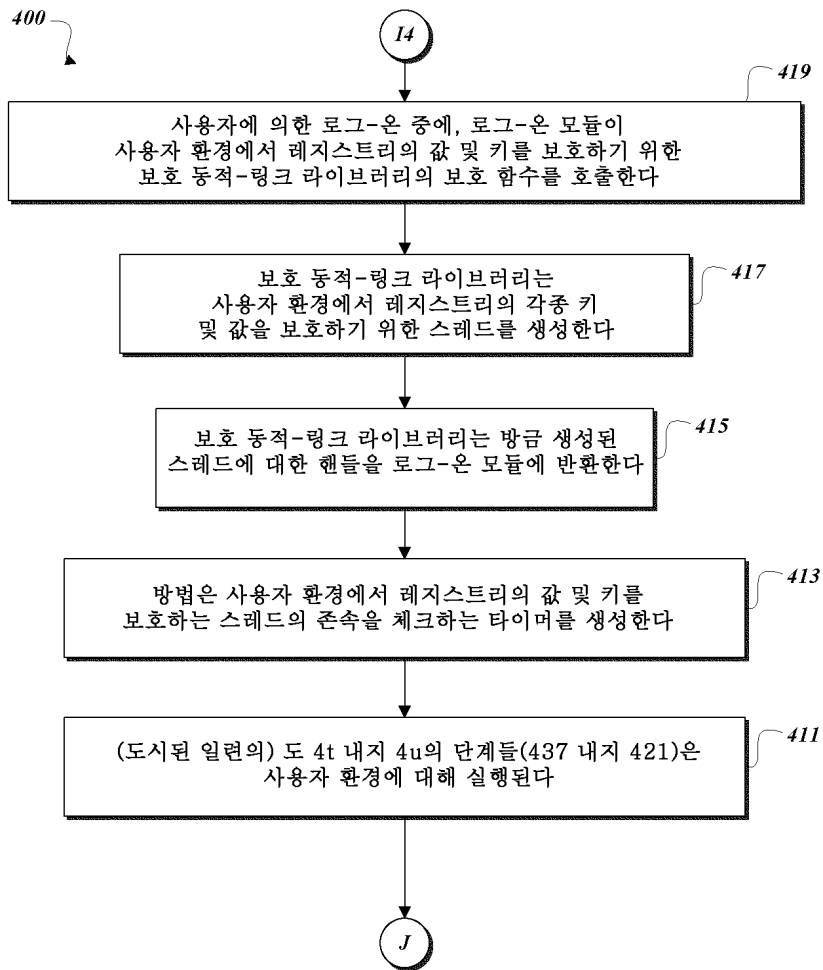
도면4u



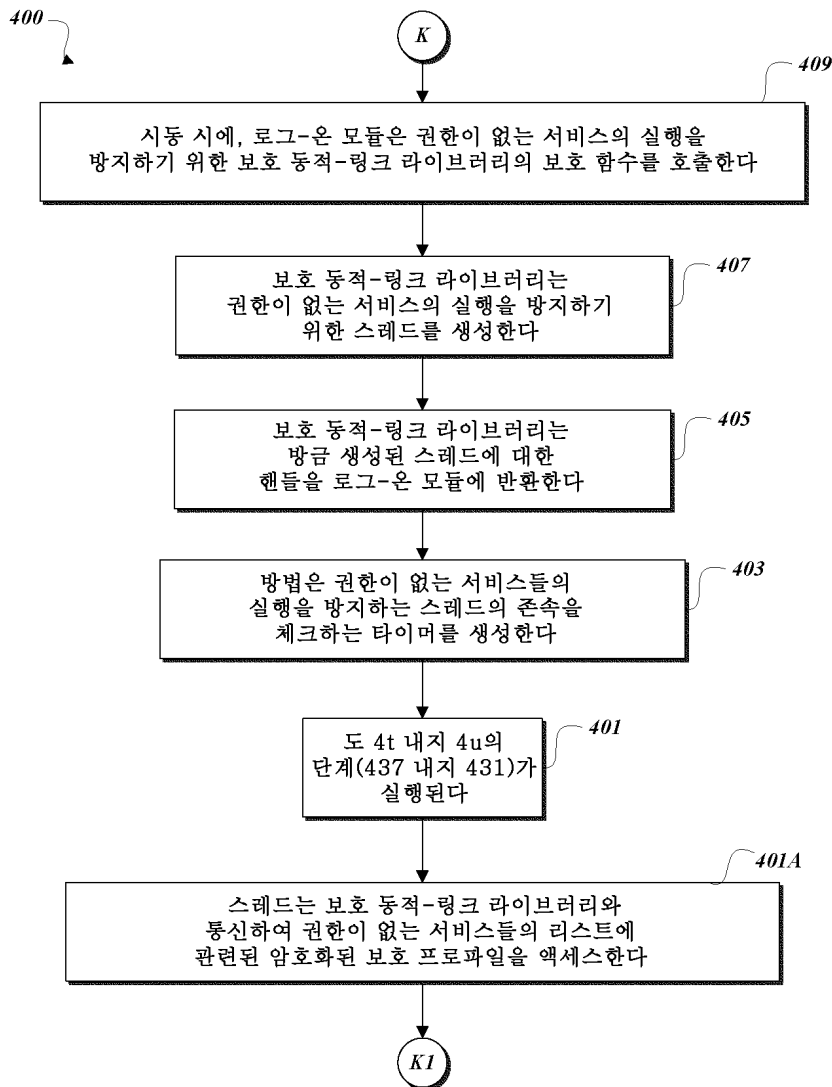
도면4v



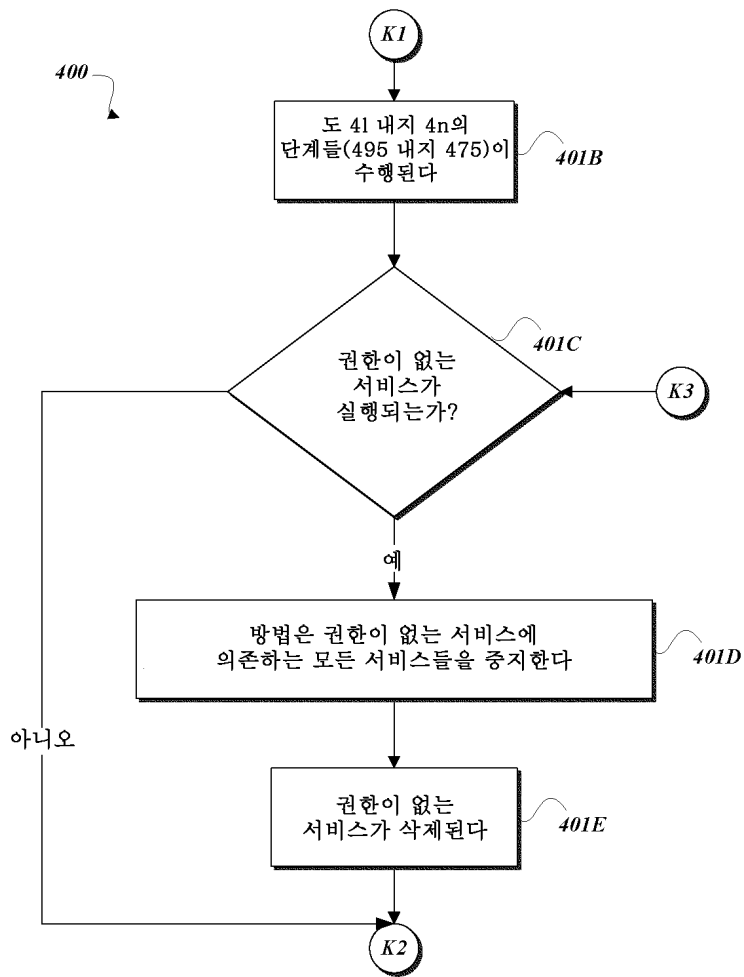
도면4w



도면4x



도면4y



도면4z

