(19) 대한민국특허청(KR)
(12) 공개특허공보(A)

(11) 공개번호 10-2012-0082027
(43) 공개일자 2012년07월20일

(51) 국제특허분류(Int. Cl.)
G06K 19/073 (2006.01)
(21) 출원번호 10-2012-7014514
(22) 출원일자(국제) 2010년11월03일
심사청구일자 2012년06월04일
(85) 번역문제출일자 2012년06월04일
(86) 국제출원번호 PCT/EP2010/006693
(87) 국제공개번호 WO 2011/054498
국제공개일자 2011년05월12일
(30) 우선권주장
0905312 2009년11월05일 프랑스(FR)

(71) 출원인
트러스티드 로직
프랑스 에프-92190 뤠동 뤼 드 라 베레리 6
(72) 발명자
레그노 니콜라스
프랑스 에프-92190 뤠동 뤼 드 라 베레리 6 트러스티드 로직
베틸라르 에리크
프랑스 에프-92190 뤠동 뤼 드 라 베레리 6 트러스티드 로직
(74) 대리인
리앤목특허법인

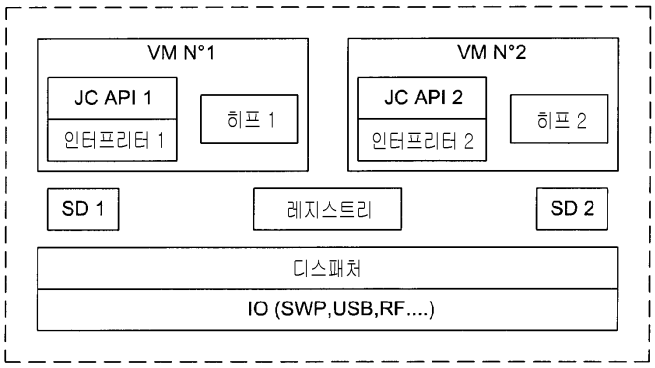
전체 청구항 수 : 총 18 항

(54) 발명의 명칭 **안전한 휴대용 객체**

(57) 요 약

본 발명은 (a) 객체 몸체 및 (b) 제1 실행 공간에서 제1 실행 엔진에 의해 실행되는 제1 애플리케이션이 저장되어 있는 적어도 하나의 메모리 및 프로세서를 지니는 마이크로-모듈을 포함하는, 스마트 카드 타입의 안전한 휴대용 객체에 관한 것이다. 본 발명은 제2 애플리케이션이 상기 적어도 하나의 메모리에 부가적으로 저장되는 것을 특징으로 하며, 이 경우에 상기 제2 애플리케이션은 상기 제1 실행 공간과는 다른 제2 실행 공간에서 상기 제1 실행 엔진과 다른 제2 실행 엔진에 의해 실행된다. 본 발명은 특히 스마트 카드들에 적용된다.

대 표 도 - 도1



특허청구의 범위

청구항 1

(a) 객체 몸체 및 (b) 제1 애플리케이션 및 제2 애플리케이션이 저장되어 있는 하나 이상의 메모리들 및 프로세서를 지니는 마이크로-모듈을 포함하는, 스마트 카드 타입의 안전한 휴대용 객체로서,

상기 안전한 휴대용 객체는, 상기 제1 및 제2 애플리케이션들을 실행하는 것이 가능한 제1 및 제2 실행 엔진을 부가적으로 포함하고, 상기 제1 애플리케이션은 제1 실행 공간에서 상기 제1 실행 엔진에 의해 실행되며 상기 제2 애플리케이션은 상기 제1 실행 공간과는 다른 제2 실행 공간에서 상기 제2 실행 엔진에 의해 실행되는 것을 특징으로 하는, 안전한 휴대용 객체.

청구항 2

제1항에 있어서, 상기 제2 실행 엔진은 상기 제1 실행 엔진과는 다른 것을 특징으로 하는, 안전한 휴대용 객체.

청구항 3

제1항 또는 제2항에 있어서, 상기 제1 실행 엔진은 제1 가상 머신(VM 1)의 실행 엔진(인터프리터 1)이며 상기 제2 실행 엔진은 제2 가상 머신(VM 2)의 실행 엔진(인터프리터 2)인 것을 특징으로 하는, 안전한 휴대용 객체.

청구항 4

제3항에 있어서, 상기 가상 머신들(VM 1,VM 2)은 Java Card™ 가상 머신들이며 상기 제1 및 제2 애플리케이션들은 Java Card™ 애플리케이션들인 것을 특징으로 하는, 안전한 휴대용 객체.

청구항 5

제3항 또는 제4항에 있어서, 상기 가상 머신들의 실행 공간에서 상기 가상 머신들에 의해 관리되는 애플리케이션들의 실행을 위해 상기 가상 머신들에 의해 요구되는 안전 조치들은 한 가상 머신과 다른 한 가상 머신 간에 서로 다르며, 상기 애플리케이션들은 상기 가상 머신들의 실행을 위해 필요한 보안 등급에 기초하여 상기 가상 머신들에 배치되는 것을 특징으로 하는, 안전한 휴대용 객체.

청구항 6

제3항 또는 제4항에 있어서, 상기 가상 머신들의 실행 공간에서 상기 가상 머신들에 의해 관리되는 애플리케이션들의 실행을 위해 상기 가상 머신들에 의해 요구되는 안전 조치들이 한 가상 머신과 다른 한 가상 머신 간에 서로 다르며, 상기 애플리케이션들은 상기 애플리케이션들의 인증 또는 상기 애플리케이션들의 검증 결과에 기초하여 상기 가상 머신들에 배치되는 것을 특징으로 하는, 안전한 휴대용 객체.

청구항 7

제1항 내지 제6항 중 어느 한 항에 있어서, 한 실행 공간은 특정 자원에 확장되며 다른 한 실행 공간은 상기 특정 자원에 확장되지 않는 것을 특징으로 하는, 안전한 휴대용 객체.

청구항 8

제7항에 있어서, 상기 특정 자원은 메모리 영역, 상기 메모리 영역에 저장된 데이터, 통신 채널, 통신 또는 데이터 저장 장치 또는 코드 라이브러리인 것을 특징으로 하는, 안전한 휴대용 객체.

청구항 9

제3항 내지 제8항 중 어느 한 항에 있어서, 상기 가상 머신들(VM 1,VM 2)의 자원들은 공유되는 것을 특징으로 하는, 안전한 휴대용 객체.

청구항 10

제9항에 있어서, 상기 공유된 자원들은 상기 애플리케이션들에 전용되는 메모리(히프(Heap)) 또는 상기 가상 머신들에 공유되는 일부 코드 라이브러리들을 포함하는 것을 특징으로 하는, 안전한 휴대용 객체.

청구항 11

제1항 내지 제10항 중 어느 한 항에 있어서, 상기 안전한 휴대용 객체는 근접장 무선 주파수 통신 모듈을 포함하는 휴대용 전화에 삽입되도록 설계된 가입자 식별 모듈이며, 상기 가입자 식별 모듈은 상기 근접장 무선 주파수 통신 모듈에 라우팅되는 제1 통신 인터페이스 및 제2 통신 인터페이스를 포함하며, 상기 제1 통신 인터페이스에 대한 액세스는 단지 상기 제1 실행 공간에서만 허가되는 것을 특징으로 하는, 안전한 휴대용 객체.

청구항 12

제1항 내지 제11항 중 어느 한 항에 있어서, 상기 제1 애플리케이션은 오직 상기 제2 실행 공간에서 상기 제2 실행 엔진에 의한 실행을 배제하도록 상기 제1 실행 공간에서 상기 제1 실행 엔진에 의해서만 실행되고, 상기 제2 애플리케이션은 오직 상기 제2 실행 공간에서 상기 제1 실행 엔진에 의한 어떠한 실행도 배제하도록 상기 제2 실행 공간에서 상기 제2 실행 엔진에 의해서만 실행되는 것을 특징으로 하는, 안전한 휴대용 객체.

청구항 13

(a) 객체 몸체 및 (b) 제1 애플리케이션 및 제2 애플리케이션이 저장되어 있는 하나 이상의 메모리들 및 프로세서를 지니는 마이크로-모듈을 포함하는, 스마트 카드 타입의 안전한 휴대용 객체를 획득하는 방법으로서, 상기 방법은,

상기 제1 및 제2 애플리케이션들을 실행하는 것이 가능한 제1 및 제2 실행 엔진이 제공되는 단계;

상기 제1 실행 엔진이 제1 실행 공간에서 상기 제1 애플리케이션을 실행하는 단계; 및

상기 제2 실행 엔진이 상기 제1 실행 공간과는 다른 제2 실행 공간에서 상기 제2 애플리케이션을 실행하는 단계;

를 포함하는 것을 특징으로 하는, 방법.

청구항 14

제13항에 있어서, 상기 방법은,

설치할 각각의 애플리케이션에 대해 할당되는 실행 공간이 식별되는 단계; 및

상기 애플리케이션이 상기 실행 공간에서 실행되도록 상기 애플리케이션이 설치되는 단계;

를 더 포함하는 것을 특징으로 하는, 방법.

청구항 15

제14항에 있어서, 상기 실행 공간은 적어도 상기 애플리케이션의 제공자의 식별자에 의해 식별되는 것을 특징으로 하는, 방법.

청구항 16

제14항에 있어서, 상기 실행 공간은 적어도 상기 애플리케이션의 인증 또는 검증 등급에 의해 결정되는 것을 특징으로 하는, 방법.

청구항 17

제14항에 있어서, 상기 제1 실행 공간은 자원에 대한 액세스를 허용하며, 상기 제2 실행 공간은 상기 자원에 대한 액세스를 허용하지 않고, 할당하기 위한 상기 실행 공간은 적어도 상기 자원에 액세스하기 위한 상기 애플리케이션의 요구에 의해 식별되는 것을 특징으로 하는, 방법.

청구항 18

제17항에 있어서, 상기 자원은 상기 안전한 휴대용 객체의 통신 인터페이스인 것을 특징으로 하는, 방법.

명 세 서

기 술 분 야

[0001] 본 발명은 스마트 카드들과 같은 일반적으로 표준화된 안전한 휴대용 객체들의 분야에 관한 것이다. 그러한 객체들은 (a) 카드 몸체 및 (b) 마이크로-모듈을 포함한다. 상기 마이크로-모듈은 프로세서, 및 제1 애플리케이션 및 제2 애플리케이션이 저장된 하나 이상의 메모리들을 포함한다.

배 경 기 술

[0002] 스마트 카드들은 특히 한정된 하드웨어 및 소프트웨어 자원들을 지니는 일반적으로 표준화된 안전한 휴대용 객체들이다.

[0003] 현재 이용가능한 카드들 중 몇몇 카드들, 특히 소위 Java Card™ 카드들은 가상 머신을 포함한다. 그러한 가상 머신은 실행 엔진(인터프리터(interpreter))을 포함한다. 상기 실행 엔진은 스마트 카드 애플리케이션으로부터의 명령어들을 해석하고 상기 가상 머신에 의해 실제로 정의된 실행 공간에서 상기 명령어들을 실행하는 것이 가능하다.

[0004] 몇몇 카드들은 복수 개의 Java Card™ 애플리케이션들을 구현한다. 그러한 경우에, 그러한 Java Card™ 애플리케이션들 모두는 동일한 실행 공간 내에서 실행되며 종종 동일한 보안 규칙들에 적용을 받는다. 따라서, 비록 그러한 것이 반드시 필요하지는 않더라도, 실행을 위한 높은 수준의 보안을 요구하지 않는 몇몇 애플리케이션들은, 이러한 애플리케이션들이 상기 가상 머신에 의해 정의된 실행 공간 내에서 액세스하는 것이 가능하지 않은 기밀 자원(confidential resource)들에 액세스할 수 있다. 그러한 것은 보안 문제의 원인이 된다. 더욱이, 그리고 설령 그러한 것이 몇몇 애플리케이션들에만 필요하다 하더라도, 기대되는 신뢰성의 관점에서 볼 때 광범위한 안전 조치들의 적용이 적절한 것이라고 볼 수 없는 애플리케이션들을 포함한, 모든 애플리케이션들에 대해 광범위한 안전 조치들이 구현된다. 일 예로, 중복 계산 실행 또는 중복 데이터 저장은, 모든 애플리케이션들이 어떤 애플리케이션들인지에 관계없이, 모든 애플리케이션들에 대해 사용된다. 그 결과로, 카드 성능에 대한 영향 때문에, 카드 자원들이 이러한 카드 자원들을 필요로 하지 않는 애플리케이션들의 실행에 사용된다.

발명의 내용

해결하려는 과제

[0005] 위의 배경기술의 관점에서 볼 때, 본 발명이 해결하고자 하는 한가지 기술적 과제는 복수 개의 애플리케이션들의 실행에 대한 보안이 강화될 경우에, 특히 이러한 애플리케이션들이 특정 보안 자원들에 대한 액세스를 필요로 하지 않을 때, 그리고 카드 성능에 대한 보안 조치들의 구현의 영향이 제한될 경우에, 상기 조치들이 몇몇 애플리케이션들의 실행에 적합한 것이라고 볼 수 없을 때, 앞서 언급한 선행기술의 단점들을 완화하는, 복수 개의 애플리케이션들을 실행할 수 있는 스마트 카드 타입의 안전한 휴대용 객체들을 획득하는 것이다.

과제의 해결 수단

[0006] 위의 배경기술의 관점에서 본 발명이 해결하고자 하는 한가지 기술적 과제는 복수 개의 애플리케이션들의 실행에 대한 보안이 강화될 경우에, 특히 이러한 애플리케이션들이 특정 보안 자원들에 대한 액세스를 필요로 하지 않을 때, 그리고 카드 성능에 대한 보안 조치들의 구현의 영향이 제한될 경우에, 상기 조치들이 몇몇 애플리케이션들의 실행에 적합한 것이라고 볼 수 없을 때, 앞서 언급한 선행기술의 단점들을 완화하는, 복수 개의 애플리케이션들을 실행할 수 있는 스마트 카드 타입의 안전한 모바일 객체들을 획득하는 것이다.

[0007] 이러한 기술적 과제에 대하여 본 발명에 의해 제공되는 해결방안은 먼저 (a) 객체 몸체 및 (b) 제1 애플리케이션 및 제2 애플리케이션이 저장되어 있는 하나 이상의 메모리들 및 프로세서를 지니는 마이크로-모듈을 지니는, 스마트 카드 타입의 안전한 휴대용 객체를 포함하며, 상기 안전한 휴대용 객체는 상기 제1 및 제2 애플리케이션들을 실행하는 것이 가능한 제1 및 제2 실행 엔진을 부가적으로 포함하고, 상기 제1 애플리케이션은 제1 실행 공간에서 상기 제1 실행 엔진에 의해 실행되며 상기 제2 애플리케이션은 상기 제1 실행 공간과는 다

른 제2 실행 공간에서 상기 제2 실행 엔진에 의해 실행되는 것을 특징으로 한다.

[0008] 둘째로, 상기 해결방안은 (a) 객체 몸체 및 (b) 제1 애플리케이션 및 제2 애플리케이션이 저장되어 있는 하나 이상의 메모리들 및 프로세서를 지니는 마이크로-모듈을 포함하는, 스마트 카드 타입의 안전한 휴대용 객체를 획득하는 방법을 포함하며, 상기 방법은, 상기 제1 및 제2 애플리케이션들을 실행하는 것이 가능한 제1 및 제2 실행 엔진이 제공되는 단계; 상기 제1 실행 엔진이 제1 실행 공간에서 상기 제1 애플리케이션을 실행하는 단계; 및 상기 제2 실행 엔진이 상기 제1 실행 공간과는 다른 제2 실행 공간에서 상기 제2 애플리케이션을 실행하는 단계;를 포함한다.

[0009] 따라서, 상기 제1 애플리케이션이 예를 들면 중요한 보안 자원들에 확장되는 실행 공간에서 실행되어야 하는 경우에, 상기 제2 애플리케이션이 상기 객체를 더 안전하게 하도록 상기 자원들에 접근할 수는 없다. 더욱이, 상기 실행 엔진들에 의존하여, 한 애플리케이션과 다른 한 애플리케이션 간에 서로 다른 보안 조치들을 구현하는 것이 가능하게 됨으로써, 카드 성능이 개선된다.

[0010] 바람직하게는, (a) 상기 제2 실행 엔진이 상기 제1 실행 엔진과 다르며; (b) 상기 제1 실행 엔진이 제1 가상 머신의 실행 엔진이고 상기 제2 실행 엔진이 제2 가상 머신의 실행 엔진이며; (c) 상기 가상 머신들이 Java Card™ 가상 머신들이고 상기 제1 및 제2 애플리케이션들은 Java Card™ 애플리케이션들이며; (d) 상기 가상 머신들이 자신들의 실행 공간에서 관리하는 애플리케이션들의 실행을 위해 상기 가상 머신들에 의해 요구되는 보안 조치들이 한 가상 머신과 다른 한 가상 머신 간에 서로 다르고 상기 애플리케이션들이 상기 애플리케이션들의 실행에 필요한 보안 등급에 따라 상기 가상 머신들에 배치되며; (e) 상기 가상 머신들의 실행 공간에서 상기 가상 머신들에 의해 관리되는 애플리케이션들을 실행하기 위해 가상 머신들에 의해 요구되는 보안 조치들이 한 가상 머신과 다른 한 가상 머신 간에 서로 다르고, 상기 애플리케이션들이 상기 애플리케이션들의 검증의 결과 또는 상기 애플리케이션들의 인증에 의존하여 상기 가상 머신들에 배치되며; (f) 한 실행 공간이 특정 자원에 확장되며 다른 한 실행 공간이 상기 특정 자원에 확장되지 않고; (g) 상기 특정 자원이 메모리 영역, 상기 메모리 영역에 저장된 데이터, 통신 채널, 통신 또는 데이터 저장 장치 또는 코드 라이브러리이며; (h)상기 가상 머신들의 자원들이 공유되고; (i) 상기 공유된 자원이 상기 애플리케이션에 전용되는 메모리 또는 상기 가상 머신들에 공유된 몇몇 코드 라이브러리들을 포함하며; (j) 상기 객체가 근접장 무선 주파수 통신 모듈을 포함하는 모바일 전화에 삽입되도록 설계된 가입자 식별 모듈이고, 상기 가입자 식별 모듈이 근접장 무선 주파수 통신 모듈에 라우팅(routing)되는 제1 통신 인터페이스 및 제2 통신 인터페이스를 지니고, 상기 제1 통신 인터페이스에 대한 액세스가 단지 상기 제1 실행 공간에만 허용되며; (k) 상기 제1 애플리케이션이 상기 제2 실행 공간에서 상기 제2 실행 엔진에 의한 어떠한 실행도 배제하도록 오직 상기 제1 실행 공간에서 상기 제1 실행 엔진에 의해서만 실행되고, 상기 제2 애플리케이션은 상기 제2 실행 공간에서 상기 제1 실행 엔진에 의한 어떠한 실행도 배제하도록 오직 상기 제2 실행 공간에서 상기 제2 실행 엔진에 의해서만 실행되며, (l) 본 발명에 따른 상기 방법은 (a) 설치할 각각의 애플리케이션에 할당하기 위한 실행 공간이 식별되고 상기 애플리케이션이 상기 실행 공간에서 실행되도록 설치되는 단계; (b) 상기 실행 공간이 적어도 상기 애플리케이션의 제공자의 아이덴티티에 의해 식별되는 단계; (c) 상기 실행 공간이 적어도 상기 애플리케이션의 검증 또는 인증 등급에 의해 식별되는 단계; (d) 상기 제1 실행 공간이 자원에 대한 액세스를 제공하고, 상기 제2 실행 공간이 상기 자원에 대한 액세스를 허용하지 않고, 할당하기 위한 상기 실행 공간이 적어도 상기 자원에 액세스하도록 하는 상기 애플리케이션의 요구에 의해 적어도 식별되는 단계; 및 (e) 상기 자원이 상기 안전한 휴대용 객체의 통신 인터페이스인 단계;를 더 포함한다.

발명의 효과

[0011] 본 발명은 복수 개의 애플리케이션들의 실행에 대한 보안이 강화될 경우에, 특히 이러한 애플리케이션들이 특정 보안 자원들에 대한 액세스를 필요로 하지 않을 때, 그리고 카드 성능에 대한 보안 조치들의 구현의 영향이 제한될 경우에, 상기 조치들이 몇몇 애플리케이션들의 실행에 적합한 것이라고 볼 수 없을 때, 앞서 언급한 선행기술의 단점들을 완화하는, 복수 개의 애플리케이션들을 실행할 수 있는 스마트 카드 타입의 안전한 휴대용 객체들을 제공한다.

[0012] 본 발명은 첨부된 도면들을 참조하여 이하의 비-제한적인 설명을 숙지하면 양호하게 이해할 수 있을 것이다.

도면의 간단한 설명

[0013] 도 1은 객체가 2개의 개별 Java Card™ 가상 머신들을 포함하는 경우에 본 발명에 따른 객체의 실시예를 개략

적으로 보여주는 도면이다.

도 2는 객체가 2개의 가상 머신들을 포함하고 메모리가 공유된 애플리케이션들에 전용된 경우에 본 발명에 따른 객체의 실시예를 개략적으로 보여주는 도면이다.

도 3은 객체가 2개의 가상 머신들을 포함하고 제1 가상 머신이 제1 통신 수단에 액세스하고 제2 가상 머신이 제2 통신 수단에 액세스하며 상기 제1 및 제2 통신 수단이 분리되어 있는 경우에 본 발명에 따른 객체의 실시예를 개략적으로 보여주는 도면이다.

발명을 실시하기 위한 구체적인 내용

[0014] 본 발명은 스마트 카드 타입의 안전한 휴대용 객체에 관한 것이다. 상기 객체는 첫째로 객체 몸체, 예를 들면 표준 치수들을 갖는 플라스틱 카드 몸체, 및 둘째로 마이크로-모듈을 포함한다.

[0015] 본 발명에 따른 객체의 마이크로-모듈은 적어도 2 개의 애플리케이션들 및 상기 2 개의 애플리케이션들의 데이터가 저장되어 있는 적어도 하나의 메모리 및 적어도 하나의 프로세서를 포함한다. 상기 2 개의 애플리케이션들은 상기 프로세서에 의해 간헐적으로 실행된다.

[0016] 본 발명에 따른 객체들의 자원들, 특히 하드웨어 자원들, 좀더 구체적으로는 메모리 크기가 한정되어 있다.

[0017] 본 발명에 의하면, 제1 애플리케이션은 제1 실행 공간에서 제1 실행 엔진에 의해 실행되고 제2 애플리케이션은 제2 실행 공간에서 제2 실행 엔진에 의해 실행된다. 바람직하게는, 상기 제1 실행 엔진은 제1 가상 머신의 일부이며 상기 제2 실행 엔진은 제2 가상 머신의 일부이다. 결과적으로, 본 발명에 따른 객체는 적어도 2 개의 가상 머신들, 예컨대 Java Card™ 가상 머신들을 포함한다. 상기 2 개의 가상 머신들은 아마도 일부 자원들을 공유할 수 있다. 상기 가상 머신들은 서로 다른 보안 등급들을 제공할 수 있다.

[0018] 실행 공간들

[0019] 각각의 애플리케이션은 소정의 특성들을 갖는 실행 공간에서 실행되도록 설계된다. 이러한 특성들은 예를 들면 다음과 같은 것들을 포함할 수 있다.

[0020] - 상기 실행 공간의 특성, 좀더 구체적으로 설명하면 소정의 명령어 집합에 의해 정의된 프로세서 또는 가상 머신의 타입;

[0021] - 특히, 일부 데이터의 기밀 등급, 또는 일부 데이터의 중요성(일부 데이터의 무결성 보호의 중요성)에 따라, 또는 주변 장치들 또는 다른 시스템들과의 통신을 위한 채널들, 또는 코드 라이브러리들에 따라, 메모리에서 일부 데이터를 판독 또는 편집할 수 있는 능력과 같은 자원들에 액세스하는 수단의 제공;

[0022] - 애플리케이션들의 코드 및 상기 애플리케이션들에 의해 처리되는 다른 애플리케이션들 또는 외부의 파티(party)들로부터의 데이터를 보호하는데 목적을 둔 보안 조치들.

[0023] 본 발명은 공통된 특성으로서 적어도 실행 공간의 특성을 갖는 실행 공간들에서 실행되도록 설계된 적어도 2 개의 애플리케이션들을 포함하는 객체들에 관한 것이다. 예를 들면, 상기 애플리케이션들은, 양자 모두가 Java Card™ 가상 머신에서 실행되도록 설계된 2 개의 Java Card™ 애플리케이션들일 수 있다. 상기 실행 공간들의 다른 특성들과 관련하여, 일부 실행 공간들은 공유된 것들이며 다른 실행공간들은 서로 다른 것들이다.

[0024] 본 발명에 의하면, 상기 객체는 개별 실행 공간을 가지고 2 개의 애플리케이션들을 제공한다. 그러나, 이러한 공간들은 상기 2 개의 애플리케이션들에 공통된 요건들에 따라, 일부 컴포넌트들을 공유한다.

[0025] 좀더 일반적으로 말하면, 본 발명에 따른 객체는 몇몇 실행 공간들을 제공할 수 있으며 각각의 애플리케이션이 자체의 실행 공간을 가지거나 또는 애플리케이션들이 요구되는 특성들의 집합들 및 요구되는 보안 등급과 같은 기준에 따라 적은 개수의 실행 공간들로 그룹화되는 몇몇 애플리케이션들의 실행을 허용할 수 있다.

[0026] 자원들의 공유

[0027] 상기 안전한 휴대용 객체는, 상기 애플리케이션들 간에 배포되어 있는, 한정 분량의 서로 다른 자원들을 포함한다. 상기 객체의 한 자원은 메모리이다. 상기 객체들 대부분은 사용자들과의 상호작용, 다른 컴퓨터 시스템

들과의 데이터 교환 또는 외부의 물리적 시스템들의 관측 또는 외부의 물리적 시스템들에 대한 작용을 허용하는 추가적인 기능 컴포넌트들을 부가적으로 포함한다. 이러한 자원들 모두는, 소정의 자원이 한 애플리케이션에 대하여 별도로 마련되어 있었던 이유 및 본 발명에 따른 객체가 이러한 배포를 관리하는 경우에 소정의 자원이 여러 애플리케이션들 간에 공유되는 이유 중 어느 한 이유로, 상기 애플리케이션들 간에 배포되어야 한다.

[0028] 메모리

[0029] 본 발명에 따른 객체는 하나 이상의 타입들일 수 있는 하나 이상의 메모리들을 포함한다. 상기 메모리는, 예를 들면 RAM, EEPROM 타입의 것과 같은 영구적이고 재기록가능한 메모리 또는 ROM일 수 있다. 본 발명에 따른 객체들에 담겨져 있는 메모리의 분량이 특히 한정되어 있다. 특히, ROM 메모리의 크기는 약 수백 Kb이며 상기 EEPROM 메모리의 크기는 약 수십 Kb이다.

[0030] 각각의 애플리케이션은 컴퓨터 프로그램이며, 상기 컴퓨터 프로그램의 코드는 본 발명에 따른 안전한 휴대용 객체의 메모리에 저장된다. 더욱이, 각각의 애플리케이션은 적어도 하나의 메모리 내에 데이터를 저장한다.

[0031] 본 발명에 의하면, 각각의 실행 공간은 특히 메모리에 할당된 메모리의 하나 이상의 부분들을 지닌다. 메모리의 다른 부분들은 여러 실행 공간들 간에 공유되어 있는 소프트웨어 컴포넌트들에 부가적으로 할당될 수도 있고, 여러 실행 공간들에 동시 접근가능한 공유된 메모리 영역들을 형성할 수 있다.

[0032] 예약 자원들

[0033] 한 애플리케이션 또는 애플리케이션 카테고리에 대해 일부 자원들이 예약된다. 이때, 상기 자원들은 한 실행 공간에 의해 직접 관리되고, 상기 자원을 사용할 필요가 있는 어느 한 애플리케이션은 그러한 실행 공간에 의해 실행된다.

[0034] 일부 객체들은, 단지 일부 애플리케이션들에 의해서만 합법적으로 사용될 수 있는 하나 이상의 통신 또는 데이터 저장 장치들을 지닌다. 그러한 경우에는, 한 실행 공간 내에서 직접 이러한 장치들을 관리하는 것이 바람직하다. 상기 장치들에 액세스할 필요가 있는 애플리케이션들은 상기 실행 공간 내에서 실행되지만, 다른 애플리케이션들은 그러한 액세스를 제공하지 않는 실행 공간들에서 실행된다.

[0035] 본 발명에 따른 객체는 단지 승인된 제공자들에 의해서만 사용될 수 있는 자원들을 포함할 수 있다. 이러한 자원들은 주변 장치들일 수 있고, 또한 예를 들면 코드 라이브러리들일 수 있으며, 그러한 자원들에 대한 이용에는 소정의 수수료가 지급될 필요가 있다. 그러한 경우에, 관련 자원들을 가지고 한 실행 공간에서 상기 승인된 애플리케이션들을 실행하고, 상기 자원들에 대한 액세스를 허용하지 않는 실행 공간에서 다른 애플리케이션들을 실행하는 것이 바람직하다.

[0036] 공유 자원들

[0037] 일부 자원들은 여러 애플리케이션, 또는 심지어는 모든 애플리케이션들 간에 공유될 필요가 있다. 이때, 서로 다른 실행 공간들을 허가하고 상기 서로 다른 실행 공간들이 그러한 자원들을 이용할 수 있게 하는 것이 바람직하다. 이를 달성하는 한 가지 수단은 상기 실행 공간들 외부에 필연적으로 위치해 있는 특정의 소프트웨어 컴포넌트의 도움으로 관련된 각각의 자원을 관리하는 것이다. 상기 실행 공간들은 상기 컴포넌트를 이용하여 해당 자원에 액세스한다.

[0038] 통신 채널들

[0039] 잠재적인 공유 자원이 특히 중요시되는 경우는 통신 수단이다. 단지 하나의 애플리케이션만이 그러한 통신 수단을 이용하는 경우에, 상기 통신 수단이 한 실행 공간에 의해 관리되게 하고 상기 애플리케이션이 그러한 실행 공간에 의해 실행되게 하는 것이 가능하다. 그 반면에, 여러 애플리케이션이 상기 통신 수단을 이용할 것 같은 경우에, 한 소프트웨어 컴포넌트는 상기 통신 수단의 공유를 허용해야 한다. 상기 컴포넌트는, 단지 상기 실행 공간에서 실행되는 애플리케이션들만이 상기 통신 수단에 접근할 수 있는 경우에 한 실행 공간의 일부일 수 있거나, 상기 애플리케이션들 모두가 상기 통신 수단에 접근할 수 있도록 모든 실행 공간들에 의해

이용가능할 수 있다.

[0040] 본 발명의 한 실시 형태에서는, 본 발명에 따른 객체에서 실행될 수 있는 애플리케이션들은 식별자, 예를 들면 문자들의 개수 또는 문자열로 표시되며, 상기 통신 수단은 수신자가 메시지를 처리해야 하는 애플리케이션의 식별자의 형태로 표시되어 있는 메시지들을 수신하는 것을 가능하게 한다. 그러한 경우에, 수신된 메시지들의 최우선 처리 기능을 수행하는 소프트웨어 컴포넌트는, 수신된 메시지의 수신자인 애플리케이션의 식별자를 식별하기 위해 수신된 모든 메시지를 적어도 부분적으로 복호화한다. 상기 소프트웨어 컴포넌트는 메시지들을 수신할 것 같은 각각의 애플리케이션에 대하여, 상기 애플리케이션이 실행되는 실행 공간의 식별자와 상기 애플리케이션의 식별자를 연관시키는 대응 테이블에 부가적으로 액세스할 수 있다. 그리고나서, 상기 소프트웨어 컴포넌트는 상기 메시지를, 상기 수신자 애플리케이션이 실행되는 실행 공간에 포워드(forward)한다. 필요한 경우에, 상기 소프트웨어 애플리케이션은, 예를 들면 자신의 실행 공간을 유효화하게 하고 상기 애플리케이션으로부터 수신된 메시지들을 처리하기 위한 절차를 촉발(triggering)시킴으로써, 상기 수신자 애플리케이션에 의한 상기 메시지의 처리를 촉발시킨다.

[0041] 소프트웨어 컴포넌트들의 배포

[0042] 본 발명에 따른 안전한 휴대용 객체는 동일한 타입의 적어도 2 개의 안전한 실행 공간들을 포함한다. 이러한 공간들은 가상 머신들에 의해 정의된다. 상기 가상 머신들은 예를 들면 Java Card™ 가상 머신들일 수 있다. 한 실행 공간은 여러 소프트웨어 컴포넌트들에 의해 정의되는 것이 일반적이며, 상기 여러 소프트웨어 컴포넌트들 중에서 상기 가상 머신에 의해 형성된 실행 엔진 *stricto sensu*가 식별되는 것이 전형적이고, 이 경우에 다른 컴포넌트들은 장치 드라이버들, 메모리 관리자들, 보안 관리자들 또는 코드 라이브러리들(API)과 같은 추가 엘리먼트들인 것처럼 보인다. 본 발명에서는, 이러한 컴포넌트들 중 일부가 상기 실행 공간들 간에 공유된다. 다시 말하면, 상기 컴포넌트들의 코드들은 상기 메모리에 단지 한 번만 저장된다. 상기 컴포넌트 부분에 대해, 다른 컴포넌트들은 한 실행 공간에 특정한 것들이다. 그러한 방식으로, 다른 컴포넌트들은 단지 한 실행 공간과는 다른 실행 공간을 배제한 한 실행 공간에 접근하기 쉬운 컴포넌트들일 수도 있고, 이러한 다른 컴포넌트들의 여러 버전이 존재할 수도 있지만, 서로 다른 속성들을 지닌다.

[0043] 실행 엔진

[0044] 본 발명에서는, 상기 실행 공간들이 동일한 특성을 지니는데, 다시 말하면 상기 실행 공간들에서 실행될 수 있는 애플리케이션 포맷이 동일한 것이다. 특히, 가능할 때마다 서로 다른 애플리케이션들에 대해 동일한 실행 엔진을 이용하고 그에 따라 상기 실행 엔진을 공유하는 건 결과적으로 당연한 것이다. 상기 애플리케이션들이 고유 포맷으로 이루어진 경우에, 다시 말하면 상기 애플리케이션들이 프로세서(물리적 컴포넌트)에 의해 직접 실행되는 것에 목적을 둔 경우에, 상기 실행 엔진은, 본질적으로 공유되는, 상기 프로세서이다. 상기 애플리케이션들이 다른 포맷으로 이루어진 경우에, 다시 말하면 어떤 적합한 실행 엔진이 가상 머신인 경우에, 상기 가상 머신을 구현하는 코드의 공유는 가능하지만 부여되지 않는다.

[0045] 본 발명의 한 실시 형태에서는, 설령 상기 실행 엔진들이 동일한 애플리케이션들을 실행할 수 있다는 점에서 상기 실행 엔진들이 동일한 기본적인 기능적 속성들을 지닌다 하더라도, 상기 실행 엔진들은 서로 다른 보조 특성들을 지님으로써, 서로 다른 코드들을 지닌다. 예를 들면, 한 실행 엔진이 지닐 수 있지만 다른 실행 엔진이 지닐 수 없다는 보조 특성들의 예들은 지금까지 제시된 것들이었다.

[0046] 한 실시예에서는, 한 실행 엔진이 상기 애플리케이션에 의해 처리되는 데이터를 보호하도록 설계된 추가적인 보안 조치들을 포함한다. 예를 들면, 상기 객체에 저장되거나 외부 시스템과의 통신이 이루어지게 하는 데이터는 암호화될 수 있다. 이러한 안전 조치들에는 계산 시간 및 아마도 메모리 크기 면에서 비용이 들기 때문에, 다른 실행 엔진은 기밀 데이터를 처리하지 않는 애플리케이션들에 대한 양호한 성능을 제공하도록 그러한 보안 조치들을 포함하지 않는다.

[0047] 한 실시예에서, 한 실행 엔진은, 상기 실행이 중단된 경우, 예를 들면 상기 객체에 대한 전력 공급이 갑자기 차단된 경우 또는 상기 객체가 과열되는 경우나 또는 상기 객체의 물리적 컴포넌트가 손상된 경우를 포함해서, 처리되는 데이터의 무결성을 보호하도록 설계된 추가적인 보안 조치들을 포함한다. 또한, 직면하게 되는 장애(disturbance)들에는 제3 파티가 이용하지 못하게 하거나 상기 객체에 의한 불법적 행위를 감행하지 못하게 하는 데이터를 판독하려고 하는 제3 파티, 예를 들면 허가되어선 안 되는 은행 거래를 허가하려고 하

는 제3 파티로부터의 상기 객체에 대한 공격들도 있다. 그러한 보안 조치들은 종종 중복 계산들의 실행 또는 중복 데이터의 저장을 포함한다. 그러므로, 상기 보안 조치들에는 기대되는 신뢰성의 관점에서 볼 때 반드시 적절한 것이라 볼 수 없는 비용이 든다. 다른 실행 엔진은 그러한 중복성을 제공하지 않는데, 이는 또한 데이터 무결성의 유해물에 대한 양호한 성능을 제공한다.

[0048] 한 실시예에서, 한 실행 엔진은 애플리케이션에 할당된 메모리의 부분들을 제외한 액세스들을 방지하기 위해 상기 실행 엔진에 의해 실행된 애플리케이션들에 의한 메모리 액세스들의 제어를 포함하는 반면에, 다른 실행 엔진은 그러한 제어들의 기능을 제공하지 않는다. 따라서, 신속하게 그리고 적은 메모리 요건들을 가지고 애플리케이션들을 실행하는 제2 실행 엔진은, 메모리 액세스가 먼저 자동으로나 수동으로 검증된 애플리케이션 또는 애플리케이션들과 함께 동반하는 인증서에 의해 승인될 수 있는, 제공자가 충분히 신뢰성 있는 것으로 생각되는 애플리케이션들에 대해 별도로 마련된다.

[0049] 한 실시예에서, 한 실행 엔진은 다른 실행 엔진보다 신속하지만, 전력 소비와 같은 일부 물리적 자원들의 관점에 볼 때 비용이 많이 든다. 이러한 경우에, 상기 실행 엔진의 선택은 상기 실행 성능 및 물리적 자원들의 소비 간 타협의 결과로 이루어진다.

[0050] 한 실시예에서, 한 실행 엔진은 다른 실행 엔진보다 효율적이지만, 한 실행 엔진의 이용은 상기 한 실행 엔진에 의해 실행되는 애플리케이션들의 개수에 의존하는 가격결정(pricing) 또는 상기 애플리케이션들의 크기 또는 복잡도와 같은 특성들에 의존한다.

[0051] 방금 설명되었던 본 발명의 형태에 대한 한 변형예에서, 상기 실행 공간들이 동일한 타입을 지니지만 일부 세부사항들에서는 다르다. 상기 실행 공간들은 예를 들면 동일한 표준을 충족시키는 가상 머신들에 의해 정의될 수 있지만, 서로에 대해 반드시 양립할 필요가 없는 일부 확장들을 추가로 제공할 수 있다. 이리하여, 각각의 호환가능하지 않은 확장 패밀리에 대한 실행 공간을 상기 실행 엔진들에 제공하는 것이 바람직하다.

[0052] 본 발명의 다른 한 변형예에서, 상기 실행 엔진들은 상기 코드의 일부를 공유한다. 이는 상기 실행 엔진들이 충분히 똑같이 제작되고 예를 들면 단지 어떤 특정 동작들의 완료시에만 다른 경우에 가능하고 심지어는 바람직하다. 이러한 변형예의 여러 실시 형태가 가능하다. 2 가지의 실시 형태들에서, 상기 실행 엔진들은 각각의 명령어에 대해 실행되는, 복호화라 불리는, 코드의 한 부분이 상기 명령어에 의해 수행되는 동작을 복호화하는 것에 그리고 각각의 동작에 특정한 것인, 코드의 다른 한 부분에 제어 기능을 전달하는 것에 목적을 둔 가상 기계들이다.

[0053] 하나의 제1 실시 형태에서, 특히 명령어 복호화 코드를 포함하는 상기 실행 엔진의 코어는 2 개의 실행 공간들에 대해 공유된다. 한 메모리 셀은 유효 실행 공간의 식별자를 담고 있다. 상기 복호화 코드가 상기 실행 공간에 따라 서로 다르게 실행되어야 하는 명령어를 검출하는 경우에, 상기 복호화 코드는 한 테이블로부터의 유효 실행 공간에서 수행될 동작에 상응하는 코드의 주소를 판독하고, 이러한 방식으로 위치해 있는 코드에 제어 기능을 전달한다.

[0054] 제2 실시 형태에서, 각각의 실행 공간은 각각의 실행 공간에 특정한 실행 엔진을 지닌다. 상기 복호화 코드가 고려 중에 있는 실행 공간들 모두에서 동일한 방식으로 실행되어야 하는 명령어를 검출하는 경우에, 상기 복호화 코드는 상기 2 개의 실행 공간들에 대해 공유된 코드의 부분에 실행 기능을 전달한다.

[0055] 여기서 유념해야 할 점은 본 발명에 따른 객체의 동일한 메모리 또는 서로 다른 메모리들에 담기게 되는 것을 허용한다는 점이다. 첫 번째의 예에서, 2 개의 Java Card™ 가상 머신들이 상기 객체의 단일 ROM 메모리에 담기게 된다. 두 번째의 예에서, 제1 가상 머신이 상기 객체의 ROM 메모리에 담기게 되며, 제2 가상 머신은 EEPROM 메모리에서 상기 객체의 다른 메모리에 담기게 된다.

[0056] 추가 컴포넌트들

[0057] 실행 공간들이 실행 엔진을 공유하거나 또는 이와는 다르다는 점과는 별개로, 상기 실행 공간들은 자체의 서로 다른 추가 컴포넌트들을 공유하거나 자체의 서로 다른 추가 컴포넌트들을 지닐 수 있다. 예를 들면, 전형적인 Java Card™ 가상 머신은 상기 애플리케이션들의 코드를 만드는 명령어들을 해석하는 기능을 수행하는 실행 엔진에 부가해서, Java Card API(애플리케이션 프로그래밍 인터페이스(application programming interface))와 같은 차등 라이브러리들, 방화벽, 애플리케이션 레지스트리, 메모리 관리자, 디스패처(dispatcher) 및 상기 애플리케이션들의 요구들에 따라 추가로 개발된 다수의 라이브러리를 포함한다. 경우에

따라, 각각의 애플리케이션이 자체의 컴포넌트를 지니도록 각각의 컴포넌트가 어느 컴포넌트이든 서로 다른 애플리케이션들에 의해 공유되는 것이 바람직할 수 있다. 전형적으로, 공유하는 것은 메모리를 절약하는 것을 가능하게 하고 애플리케이션들 간의 통신을 용이하게 해 주는 반면에, 분리는 고립을 증가시키고 해당 컴포넌트의 서로 다른 구현들을 허용한다. 여기서 유념해야 할 점은 각각의 경우에 일부 컴포넌트들이 공유될 수 있는 반면에 다른 컴포넌트들이 분리된다는 점이다.

[0058] 실행 공간의 선택

[0059] 본 발명에서, 각각의 애플리케이션에는 상기 객체에서 이용가능한 실행 공간들로부터 하나의 실행 공간이 할당된다. 특히 상기 할당이 식별될 때와 상기 할당이 효과를 발휘할 때가 서로 다른, 이러한 할당의 여러 실시형태들이 가능하다. 또한, 상기 할당은 여러 기준에 기초하여 이루어질 수 있다. 할당 방법들 및 할당 기준에 대한 몇 가지 예가 본 명세서에서 언급될 것이지만, 여기서 이해할 점은 다른 형태들이 가능하다는 점이다.

[0060] 할당 방법들

[0061] 본 발명에 따른 안전한 휴대용 객체 상에의 애플리케이션 설치는 상기 객체 내에 상기 애플리케이션의 코드를 로딩하여, 상기 코드가 상기 객체의 메모리에 배치되게 하는 것을 적어도 포함한다. 로딩은 상기 메모리 내에 상기 코드를 저장하도록 상기 객체를 적절하게 프로그래밍한 후에 상기 코드를 상기 객체에 전송하는 것으로 이루어질 수 있지만, 다른 방법들이 가능한데, 특히 상기 애플리케이션의 코드는 상기 객체가 제작될 때 상기 객체의 ROM 내에 배치될 수 있다. 본 발명에서, 상기 애플리케이션을 설치하는 방법은, 상기 애플리케이션에 할당된 실행 공간을 식별하는 것을 포함한다. 상기 할당은 하나 이상의 단계들에서, 로딩 전, 로딩 후 또는 로딩과 함께 식별될 수 있다.

[0062] 첫 번째 할당 형태에서, 한 애플리케이션의 코드가 상기 객체의 메모리 내에 로딩될 경우에, 상기 코드는 할당 기준에 따라 식별된, 상기 실행 공간들 중 한 실행 공간에 할당되는 상기 메모리의 영역 내에 배치된다. 상기 메모리는 예를 들면 상기 객체가 제작되는 동안에 상기 코드가 로딩되는 경우에 ROM일 수 있고, 상기 객체가 제작된 후에 상기 코드가 로딩되는 경우에 EEPROM 타입의 재기록가능한 메모리일 수 있다. 이러하여, 상기 애플리케이션은 상기 애플리케이션을 설치하고 이를 실행하는 기능을 수행하는, 그러한 실행 공간에 할당된다.

[0063] 두 번째 할당 형태에서, 상기 애플리케이션의 코드가 상기 객체의 메모리 내에 로딩되는 경우에, 상기 애플리케이션에 공급되는 코드, 상기 객체 내에 이미 존재해 있는 코드 또는 앞서 언급한 2 가지 코드의 조합을 실행하는 것을 포함하는 설치 프로세스가 수행된다. 이러한 방법 동안, 상기 실행 공간은 할당 기준에 기초하여 식별되며, 상기 애플리케이션을 상기 식별된 실행 공간에 연관시켜 주는 정보가 상기 메모리 내에 저장된다. 선택적으로는, 상기 실행 공간이 상기 애플리케이션을 설치하기 위한 추가적인 절차를 실행한다. 상기 애플리케이션이 실행될 필요가 있는 경우에, 상기 실행의 기능을 수행하는 소프트웨어 컴포넌트는 상기 애플리케이션을 상기 실행 공간에 연관시켜 주는 상기 정보를 판독하고 그러한 방식으로 식별된 실행 공간에서의 상기 애플리케이션의 실행을 촉발시킨다.

[0064] 세 번째 할당 형태에서, 상기 애플리케이션이 실행될 경우에, 상기 객체의 사용자로부터의 명시적인 요구에 따라 또는 시간 이벤트(예컨대, 타이머 이벤트 또는 타임아웃 이벤트)와 같은 내부 또는 외부 자극 다음에 또는 통신 장치 또는 수단을 통한 요구의 수신시, 상기 실행의 기능을 수행하는 소프트웨어 컴포넌트는 상기 할당 기준에 기초하여 상기 실행 공간을 식별하고 그러한 방식으로 식별된 실행 공간에 의한 상기 애플리케이션의 실행을 촉발시킨다.

[0065] 할당 기준

[0066] 상기 할당은 상기 애플리케이션의 제공자 또는 개발자에 의해 결정될 수 있다. 그러한 경우에, 애플리케이션이 애플리케이션의 개발자 또는 제공자에 의해 배포되어 상기 객체 내에 로딩되는 애플리케이션은 상기 애플리케이션의 실행가능한 코드에 부가해서, 이용할 실행 공간을 식별하는 정보의 적어도 일부분을 포함한다.

[0067] 애플리케이션들의 각각의 개발자 또는 제공자 또는 개발자들 또는 제공자들의 그룹들에 다른 실행 공간을 할당하는 것이 가능하다. 그러한 경우에, 상기 애플리케이션은 카드의 개발자 또는 제공자를 식별하는 정보의

일부분을 가지고 카드 내에 로딩된다.

[0068] 상기 실행 공간은 상기 객체 상에 상기 애플리케이션을 설치하는 사람에 의해서나 또는 상기 애플리케이션에 공급되는 표시들에 따라 그러한 설치 중이나 그러한 설치 후에 상기 객체를 통해 자동으로 선택될 수 있다. 상기 정보는 상기 애플리케이션의 요구들 또는 상기 애플리케이션의 실행에 관한 권고들에 관련된 것이다. 상기 정보는 예를 들면 다음과 같은 것에 관련된 것일 수 있다.

[0069] - 상기 애플리케이션에 의해 실행되는 보안 등급;

[0070] - 전형적으로 처리되는 데이터의 기밀 등급;

[0071] - 상기 애플리케이션에 의해 기대되는 신뢰 등급;

[0072] - 상기 애플리케이션이 이용할 가능성이 있는 장치들;

[0073] - 상기 애플리케이션에 의해 이용되는 통신 수단;

[0074] - 상기 실행 엔진에 의해 지원되는 확장들;

[0075] - 상기 애플리케이션이 기능을 수행하는데 필요한 코드 라이브러리들;

[0076] - 좀더 일반적으로 말하면, 상기 애플리케이션이 기능을 수행하는데 필요한 임의의 자원.

[0077] 예를 들면, 제1 실행 공간이 소정 장치에 대한 액세스를 허용하는 반면에 제2 실행 공간이 동일한 장치에 대한 액세스를 정의하는 경우에, 상기 장치에 대한 액세스를 요구하는 애플리케이션들은 상기 제1 실행 공간에 할당되는 반면에 다른 애플리케이션들은 상기 제2 실행 공간에 할당된다.

[0078] 상기 실행 공간은 상기 애플리케이션의 정적 분석에 의해, 다시 말하면 몇몇 특성들을 결정하기 위한 상기 애플리케이션의 코드의 수동 또는 자동 검사에 의해 결정될 수 있다. 예를 들면,

[0079] - 애플리케이션들이 단지 허가된 메모리 액세스들만을 수행하는 것이 상기 정적 분석을 통해 나타나게 되는 경우에 일부 메모리 액세스들이 상기 애플리케이션들에 허가된 메모리 영역들의 가능한 오버런(overrun)으로부터 보호받지 못하는 실행 공간이 할당되고 애플리케이션들이 허가된 영역들을 제외한 메모리 액세스들을 수행할 수 있음이 상기 정적 분석을 통해 나타나게 되는 경우에 상기 메모리 액세스들 모두가 상기 애플리케이션들에 대해 검증되는 실행 공간이 할당되며;

[0080] - 애플리케이션들이 상기 애플리케이션들에 대한 일부 공격들을 허용하는 것이 상기 정적 분석을 통해 나타나게 되는 경우에 상기 애플리케이션들에 대한 일부 공격들로부터의 보호를 위한 조치들을 포함하는 한 실행 공간이 할당되며, 상기 애플리케이션들의 동작이 그러한 공격들에 의해 중단되지 않음이 상기 정적 분석을 통해 나타나게 되는 경우에 상기 애플리케이션들에 대한 보호를 위한 그러한 조치들이 없는 실행 공간이 할당된다.

[0081] 여기서 유념해야 할 점은 이러한 서로 다른 기준이 서로 다른 방식들로 조합될 수 있다는 점이다. 예를 들면, 실행 공간들은 애플리케이션들의 일부 제공자들에게 할당될 수 있으며, 다른 규칙들은 알려져 있지 않은 제공자들의 애플리케이션들에 할당된 실행 공간을 식별하는데 사용될 수 있다. 상기 애플리케이션에 선택적으로 공급되는 정보가 사용될 수 있으며, 상기 정보가 누락된 경우에 정적 분석으로 상기 실행 공간이 식별될 수 있다. 이와는 반대로, 항상 결정적이지 않은 정적 분석은 모든 경우에 수행될 수 있으며, 상기 애플리케이션에 공급되는 표시들은 단지 상기 정적 분석이 결정적이지 않은 경우에만 사용될 수 있다. 본 명세서에 기재된 선택 방법들을 조합하는 다른 식별 방법들이 가능하다.

[0082] 예 1: 개별 가상 머신들

[0083] 도 1에 예시된 본 발명의 제1 실시예에서, 상기 안전한 휴대용 객체는 적어도 2 개의 Java Card™ 가상 머신들(VM 1,VM 2)이 실행되는 스마트 카드이다.

[0084] 각각의 가상 머신은 자체의 실행 엔진(인터프리터 1,인터프리터 2), 자체의 라이브러리들(JC API 1,JC API 2), 및 애플리케이션들에 전용되는, 자체의 메모리, 또는 히프(heap)(히프 1,히프 2)를 지닌다. 그러한 편성은 제1 실행 공간에서 그리고 제2 실행 공간에서 실행되는 애플리케이션들 간의 높은 수준의 고립을 제공하는 이점을 제공하는데, 그 이유는 상기 가상 머신들(VM 1,VM 2)이 상기 애플리케이션들로 하여금 상기 가상 머신

들의 히프(히프 1,히프 2)를 제외한 메모리에 액세스할 수 없게 하기 때문이다. 각각의 애플리케이션은 특정의 가상 머신의 히프에 저장되고, 이들 데이터를 상기 히프에 저장하며 상기 가상 머신 내에서 실행된다.

[0085] 일부 컴포넌트들은 상기 가상 머신들 간에 공유된다. 단일의 Java Card™ 가상 머신을 지니는 종래의 Java Card™ 스마트 카드에서, 애플리케이션 레지스트리는 각각의 애플리케이션의 이용가능성(설치하에서, 실행가능하고, 차단되는 등등)과 같은 각각의 애플리케이션에 대한 표시들 및 각각의 애플리케이션에 제공되었던 승인들을 가지고, 설치된 애플리케이션들의 리스트를 특히 관리하는 소프트웨어 컴포넌트이다. 이러한 실시예에서, 애플리케이션 레지스트리는 실행 공간들 간에 공유되고 종래의 정보 외에도, 각각의 애플리케이션에 대해 상기 애플리케이션에 할당된 실행 공간의 식별자를 보유한다.

[0086] 스마트 카드의 서로 다른 ISO 7816, SWP, USB, RF 및 다른 인터페이스들에 대한 IO 입력/출력 드라이버들은 또한, 상기 스마트 카드만이 각각의 인터페이스에 대해 하나의 입력/출력 드라이버를 지니도록 상기 가상 머신들 간에 공유된다.

[0087] 상기 스마트 카드에 의해 수신된 메시지들은 수신자 애플리케이션을 결정하도록 수신된 각각의 메시지를 분석함으로써 상기 메시지들을 디스패치하고 상기 메시지들 상에의 상기 애플리케이션의 실행을 촉발시키는 소프트웨어 컴포넌트에 의해 처리된다. 그러한 수신된 메시지들의 디스패처는 또한 상기 가상 머신들 간에 공유된다. 한 메시지가 수신될 때, 일단 상기 수신자 애플리케이션이 식별될 경우에, 상기 수신된 메시지의 디스패처는 상기 수신자 애플리케이션이 실행되는 실행 공간을 식별하기 위해 상기 애플리케이션 레지스트리에 질의(querry)을 하고, 상기 실행 공간 내에서의 상기 애플리케이션에 의한 상기 메시지의 처리를 촉발시킨다.

[0088] 예 2: SD 보안 도메인들

[0089] 상기 가상 머신들 간에 상기 애플리케이션들을 배포하기 위해, 기존의 분류 기반구조를 이용하는 것이 바람직하며, 상기 분류 기반구조의 전개가 지금부터 기재될 것이다.

[0090] 도 1에 예시된 바와 같이, 보안 도메인(SD1,SD2)이라 불리는 특정 애플리케이션은 상기 가상 머신들(VM1,VM2) 각각에서 실행된다. 상기 보안 도메인은 한 세트의 애플리케이션들의 보안 관련 데이터를 관리한다. 일반적으로, 애플리케이션들의 각각의 제공자는 또한 그 제공자의 모든 애플리케이션들에 대해 사용되는 보안 도메인을 제공한다. 특히, 상기 보안 도메인은 상기 제공자에 대해 특정한 비밀 키들을 보유하며 다른 애플리케이션들 대신에 몇 가지 동작들, 특히 해당 제공자(또는 필요한 키들이 제공되는 대표자)만이 자체의 애플리케이션들에 영향을 미치도록 상기 애플리케이션들의 코드 및 로딩된 데이터가 암호화되고 인증된 경우에, 애플리케이션들의 설치 및 데이터의 로딩을 수행한다.

[0091] 본 명세서의 예에서, 각각의 가상 머신이 단지 하나의 보안 도메인만을 포함하는 것이 바람직하다. 그러한 방식으로, 각각의 가상 머신(VM 1,VM 2)은 애플리케이션들의 제공자와 연관되어 있다. 실제로, 가상 머신들의 분리는 제1 가상 머신을 통해 실행되는 애플리케이션들 및 제2 가상 머신(VM 2)을 통해 실행되는 애플리케이션들 간의 고립 정도를 제공한다. 특히, 이는 제공자 1의 애플리케이션들이 제공자 2의 데이터에 액세스할 수 없는 능력 및 이와는 반대로 제공자 2의 애플리케이션들이 제공자 1의 데이터에 액세스할 수 없는 능력의 보장을 제공한다.

[0092] 상기 스마트 카드 내에 한 애플리케이션이 로딩되는 경우에, 상기 애플리케이션이 상기 스마트 카드 내에 이미 로딩된 것으로 생각되고 상기 애플리케이션에 의해 참조되기 때문이라는 이유 및 보안 도메인이 해당 애플리케이션과 동시에 로딩되기 때문이라는 이유 중 어느 하나의 이유로, 상기 애플리케이션이 상기 관련 보안 도메인을 지닌다. 상기 스마트 카드상의 애플리케이션 관리자는 상기 가상 머신 내의 상기 애플리케이션의 설치를 촉발시켜, 상기 제공된 보안 도메인이 동작하게 한다. 상기 애플리케이션이 상기 스마트 카드 내에 아직 존재하지 않은 경우에, 상기 애플리케이션 관리자는 상기 메모리의 일부를 별도로 마련하고 상기 메모리의 그러한 부분을 이용하는 새로운 가상 머신을 시동한다. 그리고나서, 상기 애플리케이션 관리자는 상기 새로운 가상 머신 내의 새로운 보안 도메인의 설치를 촉발시킨 다음에 상기 새로운 가상 머신 내의 새롭게 로딩된 애플리케이션의 설치를 촉발시킨다. 상기 보안 도메인이 공급되지도 않고 아직 존재하지도 않은 경우에, 또는 상기 보안 도메인이 새롭게 공급되지만 새로운 가상 머신이 시동될 수 없는 경우에(예컨대, 상기 스마트 카드가 그러한 기능을 지원하지 못하기 때문에, 또는 메모리의 충분한 분량이 이용가능하지 않기 때문에), 상기 애플리케이션의 로딩이 실패한다.

[0093] 따라서, 본 명세서에 기재된 구현예에 따르는 스마트 카드는 높은 보안 요건들을 지니는 애플리케이션들을 위한 매체로서의 기능을 안전하게 수행할 수 있다. 예를 들면, 상기 스마트 카드는 (은행 카드 소지자의 은행에 의해 제공된 애플리케이션에 전용되는 한 가상 머신(VM)을 지니는) 은행 카드로서의 기능과 동시에 (교통 운영자에 의해 제공된 애플리케이션들에 전용되는 한 가상 머신을 지니는) 교통 카드로서의 기능을 수행할 수 있다. 상기 스마트 카드는 추가로 모바일 전화에 삽입되는 SIM 카드일 수 있으며, 원격 통신 애플리케이션들에 전용되는 가상 머신의 실행을 허용할 수 있다.

[0094] 예 3: 서로 다른 가상 머신들

[0095] 이러한 예에서, 상기 스마트 카드는, 비록 여러 가상 머신들이 동일한 애플리케이션들을 실행할 수 있다 하더라도, 각각의 애플리케이션에 대해, 일부 가상 머신들이 다른 가상 머신들보다 더 적합하도록 서로 다른 속성들을 지니는 여러 가상 머신들의 실행을 허용한다. 그 차이점들은 서로 다른 애플리케이션들의 보안 및 상기 서로 다른 애플리케이션들에 의해 처리되는 데이터에 관련된 것들이다.

[0096] 상기 스마트 카드 내에 한 애플리케이션을 로딩하는 동안, 상기 애플리케이션의 코드는 검증기(verifier)라 불리는 소프트웨어 컴포넌트에 의해 분석된다. 이러한 컴포넌트는 상기 스마트 카드 내에 로딩되고 상기 스마트 카드에 의해 실행되거나 상기 스마트 카드의 보안 기능을 수행하는 엔티티(entity)에 의해 *stricto sensu* 를 로딩하기 전에 실행된다. 상기 검증기는 보안 관련 문제들을 가지고 상기 애플리케이션의 속성들을 결정하기 위해 상기 애플리케이션의 코드의 정적 분석을 실행한다. 상기 애플리케이션의 코드의 정적 분석은 상기 애플리케이션의 코드에 공급되는 선택적 주석(optional annotation)들에 의해 지원될 수 있다. 특히, 상기 검증기는, 상기 애플리케이션이 상기 메모리에 액세스할 수 있거나 그러한 액세스가 허가되지 않은 다른 자원들에 액세스할 수 있는 어떠한 위험도 전혀 없다는 결론을 내릴 수 있다. 따라서, 상기 검증기는 (어떠한 보증도 제공되지 않은) 안전하지 않은 애플리케이션들로부터 (상기 검증기가 금지된 액세스들이 없음을 보증하는) 안전한 애플리케이션들을 차별화함으로써, 또는 좀더 정확한 결과를 제공함으로써, 상기 애플리케이션의 보안 등급을 계산한다.

[0097] 상기 애플리케이션이 충분히 안전한 것으로 상기 검증기에 의해 식별되는 경우에, 상기 애플리케이션은 상기 메모리 또는 다른 자원들에 대한 일부 액세스들, 즉 상기 검증기가 불가능한 것으로 보증하는 액세스들을 검증하지 않는 가상 머신 내에 로딩된 다음에 상기 가상 머신에 의해 실행된다. 그 반면에, 상기 검증기가 그러한 보증을 제공하지 않는 경우에, 상기 애플리케이션은 잠재적으로 위험한 액세스를 신중하게 검증하고 실행시 허가되지 않은 액세스들을 거부하는 다른 가상 머신에 의해 실행된다. 그러한 방식으로, 동일한 스마트 카드는 어느 적합한 애플리케이션을 실행할 수 있지만, 신속하게 안전한 것으로 보증될 수 있는 그러한 애플리케이션들을 실행한다.

[0098] 한 변형 실시예에서, 상기 스마트 카드는 동일한 애플리케이션들을 실행할 수 있는 2 개의 가상 머신들을 제공하고, 제1 가상 머신은 상기 제1 가상 머신에서 실행되는 애플리케이션들에 의해 처리되는 데이터의 무결성 또는 신뢰성에 영향을 주는데 목적을 둔 소프트웨어 또는 하드웨어 공격들로부터의 보호를 위한 조치들을 지니지만, 제2 가상 머신은 그러한 보안 조치들을 지니지 않는다. 민감한 데이터를 처리하는 애플리케이션들은 상기 제1 가상 머신에 할당되는 반면에, 다른 애플리케이션들은 상기 제2 가상 머신에 할당되는데, 이는 보호 조치들이 없음으로써 더 효율적이게 된다. 상기 제2 가상 머신은 또한 개발중에 있는 애플리케이션들을 테스트 및 디버깅하기 위해 사용될 수 있다. 여기서 유념할 점은 이러한 마지막 사례에서, 가상 머신들 모두가 가상 머신 내에서 실행되는 애플리케이션들의 기능적인 동작의 관점에서 볼 때 동일할 필요가 있지만, 외부 관측들의 견지에서 볼 때 서로 다른 것일 필요가 있다.

[0099] 예 4: 공유 메모리

[0100] 도 2에 제시된 실시예에서, 상기 스마트 카드는 적어도 2 개의 Java Card™ 가상 머신들(VM 1,VM 2)을 포함하는데, 여기서 각각의 가상 머신은 자체의 실행 엔진(인터프리터 1, 인터프리터 2) 및 자체의 라이브러리들(JC API 1,JC API 2)을 지닌다. 그러나, 히프 애플리케이션들에 전용되는 메모리는 공유된다. 애플리케이션 레지스트리, 수신된 메시지 디스패처 및 ISO 7816, SWP, USB RF 등등을 위한 IO 드라이버들과 같은, 본질적으로 단 하나인 컴포넌트들은 서로 다른 가상 머신들에 의해 마찬가지로 공유된다.

[0101] 상기 애플리케이션들에 전용되는 메모리의 공유는 각각의 가상 머신이 상기 애플리케이션들을 위한 자체의 특정 메모리 공간을 지니는 경우와 관련해서 메모리를 절약하는 것을 가능하게 한다. 실제로, 메모리의 각각의 분할은 메모리의 손실 문제의 원인이 되는데, 그 이유는 이를 필요하지 않은 하나의 가상 머신에 위치해 있는 빈 메모리 공간의 일부가 다른 가상 머신에 대해 손실되기 때문이다. 그러한 공유는 또한 메모리 관리에 전용되는 메모리에 대해 규모의 경제(economies of scale)를 허용할 수 있다. 이러한 측면은 특히 RAM에 관한 자원들이 한정되어 있는 스마트 카드들에 대해 특히 중요하다.

[0102] 상기 애플리케이션들이 저장되어 있는 메모리가 공유되어 있다는 점은 또한 애플리케이션들 간의 데이터 교환을 용이하게 한다. 따라서, 그러한 편성은 애플리케이션들 간에 통신할 수 있는 능력이 필요한 경우에 그리고 서로로부터 상기 애플리케이션들을 고립시킴으로써 이루어지는 것과는 다른 보안이 제공될 수 있는 경우에 바람직하다. 상기 메모리의 공유는 한 애플리케이션이 다른 가상 머신에서 실행되는 다른 애플리케이션에 의해 제공되는 인터페이스를 호출할 수 있게 함으로써 한 가상 머신으로부터 상기 다른 가상 머신에서 실행되는 코드를 촉발시킬 수 있게 해 준다.

[0103] 또한, 설령 상기 가상 머신들이 분리되어 있다 하더라도, 상기 가상 머신들은 상기 가상 머신과는 무관하게 구현되는 코드, 예컨대 데이터 포맷들, 또는 수치 계산 라이브러리들을 처리하는 코드의 일부 라이브러리들에 대하여 공유된 액세스를 지닐 수 있다.

[0104] 이러한 예의 일부로서, 상기 가상 머신들은 서로 다른 가능성들을 제공한다. 예를 들면, 한 가상 머신은 상기 스마트 카드의 전력 공급의 중단들과 같은 방법들에 의한 몇몇 보안 검증들의 실행을 회피하는데 목적을 둔 물리적 공격들로부터의 보호를 위한 조치들을 지닌다. 상기 가상 머신은 스마트 카드 소지자가 불법으로 수정하려고 시도할 수 있는, 민감한 데이터를 처리하는 애플리케이션들의 실행에 전용된다. 다른 가상 머신은 그러한 보안 조치들을 제공하지 않기 때문에 좀더 신속해진다. 이는 높은 수준의 보호를 필요로 하지 않는 애플리케이션들에 대해 사용된다.

[0105] 다른 한 예에서, Java Card™ 가상 머신들은 상이한 확장들을 추가하는데, 이는 Java Card™ 언어와 호환하지 않을 수 있다. 그러한 경우에, 상기 언어에 특정한 확장을 필요로 하는 애플리케이션들은 해당 확장을 제공하는 가상 머신에 의해 실행된다. 어떠한 특정 확장도 필요로 하지 않는 애플리케이션들은 어떠한 가상 머신에 의해서도 실행될 수 있으며 최선의 성능을 제공하는 가상 머신이 선택되는 것이 바람직하다.

[0106] 예 5: 개별 통신

[0107] 도 3에 예시되어 있는, 이러한 예에서, 상기 스마트 카드는 모바일 전화에 삽입되도록 설계된 가입자 식별 모듈(subscriber identification module; SIM) 카드이며, 이 경우에 상기 모바일 전화는 근접장 무선 주파수 통신(near field radiofrequency communication) 모듈(NFC 모듈)을 포함한다.

[0108] 이때, 상기 스마트 카드는 상기 모바일 전화, 좀더 구체적으로 기술하면 상기 모바일 전화의 NFC 모듈과의 통신을 위한 제1 논리 인터페이스를 지닌다. 그러한 제1 논리 인터페이스는 SWP(Single Wire Protocol) 프로토콜을 사용하여 상기 모바일 전화와의 카드 통신을 구현하는 인터페이스이다. 이 때문에, 상기 인터페이스는 상기 모듈에 라우팅되며 상기 SWP 프로토콜을 사용하여 통신이 수행된다.

[0109] 더욱이, 상기 스마트 카드는 상기 모바일 전화, 특히 상기 모바일 전화의 메인 프로세서와의 통신을 위한 제2 논리 인터페이스를 지닌다. 그러한 제2 논리 인터페이스는 표준 ISO 7816에서 정의된 통신 프로토콜 또는 프로토콜들에 기초하여 상기 모바일 전화와의 카드 통신을 구현하는 인터페이스이다.

[0110] 이러한 예에서, 상기 스마트 카드는 2 개의 개별 Java Card™ 가상 머신들(VM 1, VM 2)의 실행을 허용한다. 상기 2개의 가상 머신들(VM 1,VM 2) 각각은 자체의 실행 엔진(인터프리터 1,인터프리터 2) 및 자체의 라이브러리들(JC API 1,JC API 2)을 지닌다. 상기 제1 가상 머신(VM 1)은 무접촉식 애플리케이션들, 예를 들면 교통 애플리케이션들에 전용되며, 단지 상기 제1 가상 머신(VM 1)이 상기 모바일 전화의 NFC와의 통신을 위한 논리 인터페이스에 액세스할 뿐이다. 상기 제2 가상 머신(VM 2)은 전화 애플리케이션에 전용되며 단지 상기 제2 가상 머신(VM 2)이 상기 프로세서의 전화 기능을 구현하기 위한 상기 모바일 전화의 프로세서와의 통신용 인터페이스에 액세스할 뿐이다.

[0111] 따라서, 상기 제1 통신 인터페이스에 대한 액세스는 단지 상기 제1 실행 공간에서만 허가되지만 상기 제2 통

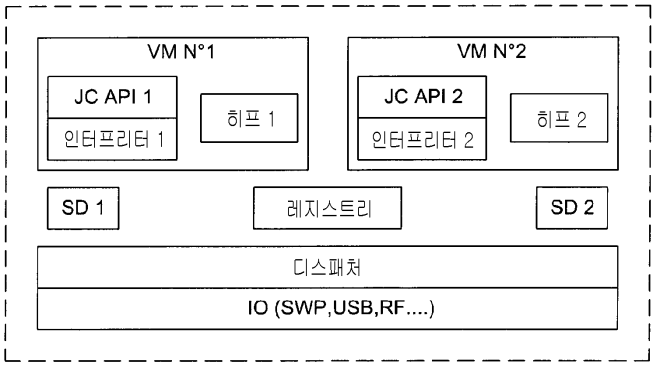
신 인터페이스에 대한 액세스는 단지 상기 제2 실행 공간에서만 허가된다.

[0112] 여기서 유념해야 할 점은 일부 컴포넌트들이 상기 가상 머신들 간에 공유된 상태로 있는 것이 바람직하다는 점이다. 따라서, 상기 스마트 카드는 상기 가상 머신들 모두에 설치된 애플리케이션들을 리스트화하는 단지 하나의 애플리케이션 레지스트리만을 지닌다. 예를 들면, 이는 다른 인터페이스와 통신하도록 설계된 애플리케이션을 로딩하기 위한 인터페이스를 사용하는 것을 가능하게 한다.

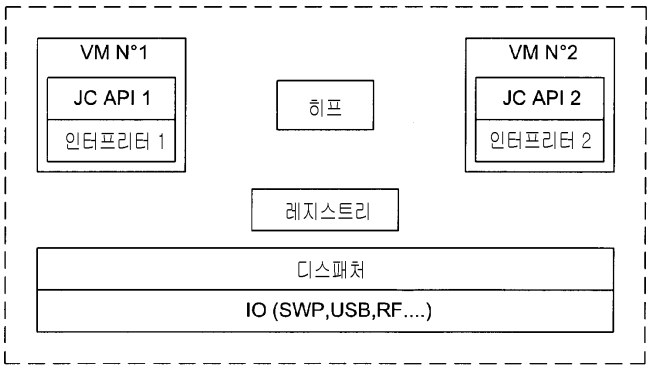
[0113] 또한, 여기서 유념해야 할 점은 히프 1, 히프 2 애플리케이션들에 전용되는 메모리가 상기 가상 머신들 간에 공유될 수도 있고 공유되지 않을 수도 있는데, 각각의 접근방안의 이점들은 위에서 개략적으로 설명된 구현예들과 관련지어 논의되어 있다. 메모리의 한 부분이 항상 공유될 수 있으며, 특히 상기 코드 라이브러리들이 상기 2 개의 실행 공간들에 대하여 공유되어 있으면서, 특히 민감 또는 기밀 데이터를 공유하지 않는 것이 바람직한 데이터를 위한 메모리는 각각의 가상 머신 내에 별도로 마련되어 있다.

도면

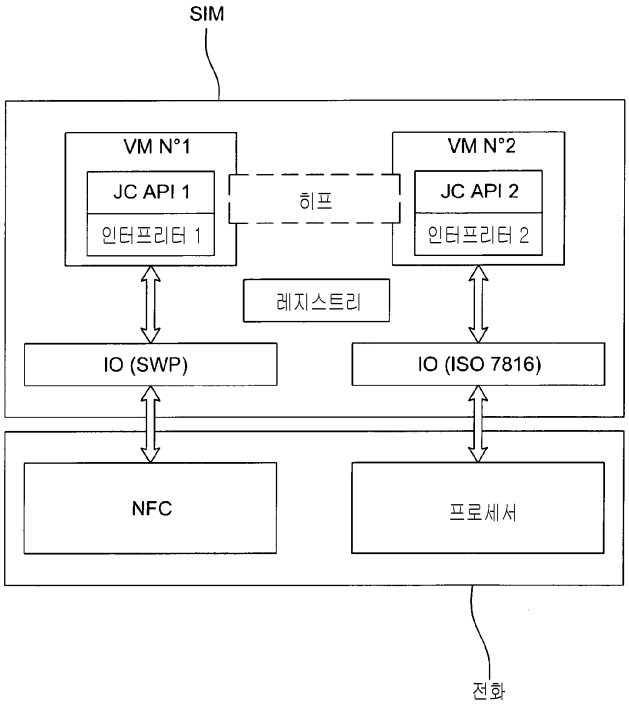
도면1



도면2



도면3


SIM
VM N°1
JC API 1
인터프리터 1
힙
VM N°2
JC API 2
인터프리터 2
레지스트리
IO (SWP)
IO (ISO 7816)
NFC
프로세서
전화