보안토큰 기반의 공인인증서 이용기술 규격

Accredited Certificate Usage Specification for Hardware Security Module

v2.40

2016년 1월

KISA 한국인터넷진흥원
Korea Internet & Security Agency

목 차

보안토큰 기반 공인인인증서 이용기술 규격

Accredited Certificate usage specification for Hardware Security Module

1. 개 요

가입자의 개인키를 안전하게 관리 및 휴대할 수 있는 저장매체로써 USB, 스마트카드 등 보안토큰에 대한 요구가 증대됨에 따라, 응용프로그램에서 이러한 보안토큰의 호환성을 제공할 수 있도록 규격이 필요하다. 이에, 본 규격은 RSA사의 PKCS#11 인터페이스에 기반을 두고 보안토큰의 기능 중 반드시 필요하거나 구현이 권고되는 부분을 기술하여 개별적인 구현간의 상호연동이 보장될 수 있도록 한다.

2. 규격의 구성 및 범위

본 규격은 공인인증서 가입자 소프트웨어에서 전자서명생성키 등 비밀정보를 안전하게 저장·보관할 수 있도록 키 생성·전자서명 생성 등이 기기 내부에서 처리되는 보안토큰의 기본 요구사항과 보안토큰을 이용함에 있어 필요한 보안토큰 API의 기능적 요구사항을 명시한다.

첫 번째로, 보안토큰 기반 공인인인증서 이용을 위한 보안토큰과 보안토큰 API 및 공인인증서 가입자 소프트웨어의 인터페이스 모델 및 보안토큰, 보안토큰 구동프로그램, 공인인증기관에 대한 요구사항을 정의한다.

두 번째로, 보안토큰의 객체, 함수, 메커니즘 등 보안토큰API(PKCS#11) 프로파일을 정의한다.

3. 관련 표준 및 규격

3.1 국외 표준 및 규격

[PKCS1] RSA Laboratories PKCS#1, *RSA Cryptography Standard v2.1, 2001*

[PKCS5] RSA Laboratories PKCS#5, *Password-Based Cryptography Standard v2.0, 1999*

[PKCS8] RSA Laboratories PKCS#8, *Private-Key Information*

Syntax Standard, 1993

[PKCS11] RSA Laboratories PKCS#11, *Cryptographic Token Interface Standard v2.11, 2001*

[P11PROF] RSA Laboratories PKCS#11 Profile, *Conformance Profile Specification, 2001*

[RFC2511] IETF, RFC 2511, *Internet X.509 Certificate Request Message Format, March 1999*

[PKCS12] RSA Laboratories PKCS#12, *Personal Information Exchange Syntax Standard v1.0, 1999*

3.2 국내 표준 및 규격

[TTA-120012] TTA, TTAS.KO-12.0012, *전자서명 인증서 프로파일 표준*, 2000

[TTA-120004] TTA, TTAS.KO-12.0004, *128비트 블록암호알고리즘 표준*, 2000

[KCAC.TS.SIVID] KISA, KCAC.TS.SIVID, v1.30, *식별번호를 이용한 본인확인 기술규격*, 2009

[Bio보안토큰] 조달청, Bio보안토큰 이용기술규격, v1.0, *나라장터 Bio보안토큰 이용기술규격*, 2009

3.3 기타

해당사항 없음

4. 정의

4.1 전자서명법 용어 정의

본 규격에서 사용된 다음의 용어들은 전자서명법 및 동법 시행령, 공인인증기관의 시설 및 장비 등에 관한 규정(미래창조과학부 고시)에 정의되어 있다.

가) 인증서
나) 공인인증서
다) 공인인증기관

라) 전자서명인증체계
마) 가입자
바) 가입자 설비

4.2 용어의 정의

가) 보안토큰(HSM) : 전자서명생성키 등 비밀정보를 안전하게 저장·보관하기 위하여 키 생성, 전자서명 생성 등이 기기 내부에서 처리되도록 구현된 하드웨어 기기
나) 보안토큰 API : 보안토큰에 대한 응용프로그래밍 인터페이스
다) 메커니즘 : 암호기능을 구현하는 절차
라) 슬롯(Slot) : 보안토큰과의 입출력을 수행할 수 있는 논리적 리더기
마) 템플릿(Template) : 객체 속성 집합으로서 객체의 생성, 수정, 검색 등을 위하여 사용됨
바) 보안토큰 구동프로그램 : 보안토큰 API를 구현한 프로그램을 말하며, 보안토큰과 가입자 소프트웨어간의 인터페이스를 담당
사) Mutexes(Mutual Exclusion) : 다중 프로세스 내에서 동기화를 위해 사용되는 상호배제 객체
아) 보안토큰 제품정보(ID) : 보안토큰 구동프로그램을 식별하기 위해 사용하는 정보. USB형 보안토큰의 보안토큰 제품정보는 Vendor ID와 Product ID를 연접하여 사용하고, 스마트카드형 보안토큰은 ATR 정보를 사용함
자) 바이오 보안토큰 : 보안토큰의 기능을 만족하면서 바이오 정보를 이용하여 인증을 수행할 수 있도록 구현된 기기
차) 스마트인증 : 스마트폰과 같은 모바일 기기의 USIM, eSE 등 보안모듈을 모바일 통신을 통해 PC와 같은 타 기기에 연결하여 전자서명생성키 등 비밀정보를 안전하게 저장·보관하고, 키생성 및 전자서명 생성 등을 처리하는 하드웨어와 소프트웨어의 총칭

4.3 용어의 효력

본 규격에서 사용된 다음의 용어들은 전자서명인증체계의 공인인증서 가입자 소프트웨어가 보안토큰의 기능을 이용함에 있어서 보안토큰 구동프로그램(이하 구동프로그램) 및 가입자 소프트웨어 구현 정도를 의미하는 것으로 [RFC2119]를

준용하며 다음과 같은 의미를 지닌다.

가) 해야 한다, 필수이다, 강제한다. (기호 : M)
반드시 준수해야 한다.

나) 권고한다. (기호 : R)
보안성 및 상호연동을 고려하여 준수할 것을 권장한다.

다) 할 수 있다, 쓸 수 있다. (기호 : O)
주어진 상황을 고려하여 필요한 경우에 한해 선택적으로 사용할 수 있다.

라) 권고하지 않는다. (기호 : NR)
보안성 및 상호연동을 고려하여 사용하지 말 것을 권장한다.

마) 금지한다, 허용하지 않는다. (기호 : X)
반드시 사용하지 않아야 한다.

바) 언급하지 않는다, 정의하지 않는다. (기호 : -)
준수 여부에 대해 기술하지 않는다.

5. 약어

본 규격에서는 다음의 약어들에 대해 추가적으로 정의한다.

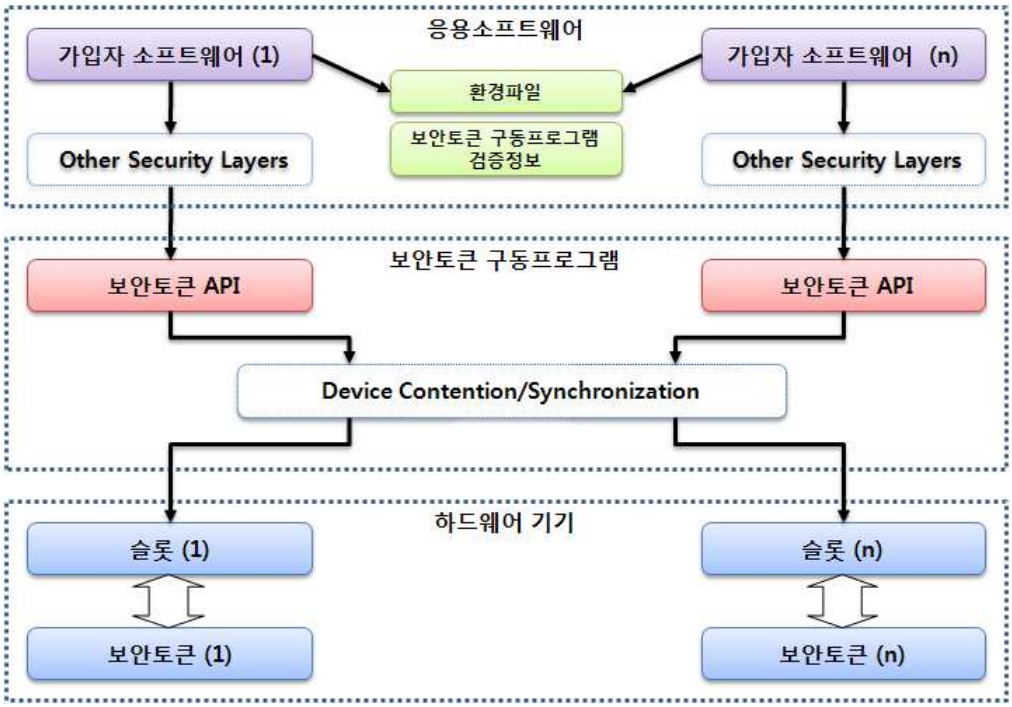
가) API : Application Programming Interface
나) ATR : Answer to Reset, 보안토큰 초기화 정보
다) eSE : Embedded Secure Element
라) HSM : Hardware Security Modules
마) microSD : micro Secure Digital
바) OID : Object Identifier
사) PKCS : Public Key Cryptography Standard
아) PIN : Personal Identification Number, 개인식별번호
자) USIM : Universal Subscriber Identity Module, 범용사용자식별모듈

6. 보안토큰 개요

6.1 가입자 소프트웨어와 보안토큰 간 인터페이스 모델

보안토큰은 구현된 하드웨어 기기에 따라 USB 토큰, 스마트카드, USIM,

eSE, microSD 등 다양한 형태를 가진다. 보안토큰 하드웨어 기기와 보안토큰을 이용하는 플랫폼 및 가입자 소프트웨어 간의 보안토큰 API(PKCS#11)의 일반적인 적용모델은 [그림 1]과 같이 도식화 된다.



[그림 2] 공인인증서 가입자 소프트웨어와 보안토큰 인터페이스 모델

일반적인 보안토큰 API 적용모델에서 보안토큰 API는 보안토큰 하드웨어 기기와의 인터페이스, 보안토큰 구동프로그램(PKCS#11 라이브러리 등)으로 구성된다. 하나의 보안토큰 API는 [그림 1]과 같이 다중 슬롯을 통하여 하나 이상의 보안토큰에 대한 인터페이스를 제공해야 한다.

가입자 소프트웨어는 보안토큰 API를 통해 보안토큰 하드웨어 기기를 이용하게 된다. 가입자 소프트웨어와 보안토큰 API 간의 인터페이스 구현은 [PKCS11] 및 [부록 7]의 PKCS#11 자바 인터페이스 사용 예를 참조하고, 무선통신단말 지원 보안토큰과의 인터페이스 구현은 [부록 9]의 무선통신단말 지원 보안토큰 표준 API를 사용 예를 참조할 것을 권고한다(무선 단말기에서의 구현 포함).

6.2 보안토큰 요구사항

보안토큰은 전자서명키 및 전자서명이 하드웨어 내부에서 생성되는 기능을 제공하여야 하며, 이 경우 암호프로세서(Crypto-processor)를 이용하여야 한

다.

보안토큰은 인가되지 않는 사용자 또는 가입자 소프트웨어의 접근·이용에 적절히 대처하기 위해 PIN 인증 및 PIN 입력오류 회수 제한 기능을 갖춰야 하며, 입력오류 허용 회수를 초과하는 경우 해당 보안토큰 잠금 기능을 제공하여야 한다.

또한, 보안토큰의 전자서명생성키 등 비밀정보는 기기 외부로 노출·유출되지 않도록 안전성을 보장해야 한다. 또한 차분전력분석기법 등 공격에 대한 대비책을 마련해야 한다. 다만, 보안토큰 외부에서 생성된 전자서명생성키 등을 보안토큰에 저장하는 것은 허용한다.

6.3 구동프로그램 요구사항

구동프로그램은 한 개의 읽기/쓰기 세션과 다중의 읽기 세션을 반드시 지원해야한다. 또한, 구동프로그램은 공개키 알고리즘으로 RSA 1024비트 및 2048비트를 지원해야 하며 안전한 쓰레드 사용을 위한 Mutexes를 지원해야 한다.

보안토큰 PIN은 가입자 소프트웨어, 구동프로그램 및 보안토큰에서 입력 가능하며, 입력된 PIN을 보호하기 위해 보안토큰과 암호채널을 생성할 수 있다.

가입자 소프트웨어는 시스템에 존재하는 구동프로그램의 위치정보를 파악할 수 있도록 [부록 4]에서 명시한 환경파일을 이용한 위치정보 관리를 지원해야 한다. 가입자 소프트웨어는 [부록 5]의 구동프로그램 검증정보를 확인하여 구동프로그램의 무결성 및 구현적합성을 확인할 수 있다. 가입자 소프트웨어에서 구동프로그램 검증정보 확인 처리 기능의 구현은 선택사항으로 한다.

6.4 공인인증기관 요구사항

공인인증기관은 보안토큰 기반으로 발급되는 공인인증서에는 '확장키 사용목적 확장필드'에 보안토큰 식별자를 추가하여 공인인증서를 발급하여야 한다. 이 경우, 공인인증서에 추가되는 식별자는 id-kisa-HSM으로 한다.

```
id-kisa-HSM OBJECT IDENTIFIER ::= {
  iso(1) member-body(2) korea(410) kisa(200004) npki(10) attributes(1) kisa-HSM(2)
}
```

공인인증기관은 보안토큰 기반으로 공인인증서를 발급하기 위해서는 [부록 5]의 구동프로그램 검증정보를 확인하여 공인인증서 발급 신청자의 보안토큰 구동프로그램의 무결성 및 구현적합성을 확인하여야 한다.

이러한 확인 결과는 [RFC2511]에서 정의하는 메시지 구조를 가지는 ir 메시지의 regInfo 부분에 부가 정보로 입력하여 전자서명키가 보안토큰 기반으로 생성되었음을 표시하여야 한다.

regInfo 부분에 전자서명키가 보안토큰 내에서 생성되었음을 표시하는 부가정보로 Attribute 형태는 id-kisa-HSM 명칭 형식을 이용하고, Value 값은 공인인증기관의 필요에 따라 선택적으로 사용할 수 있다. Value 값을 사용하지 않고자 하는 경우, 그 값은 NULL 인코딩한다.

7. 보안토큰 API(PKCS#11) 프로파일

본 장에서는 전자서명인증체계 가입자 소프트웨어에게 구동프로그램 기능의 호환성을 제공하기 위해 필요한 PKCS#11 저장객체, 함수, 메커니즘 등을 정의한다.

7.1 저장 객체(Storage Object)

본 규격을 준용하는 구동프로그램 및 가입자 소프트웨어는 저장객체로써, 데이터 객체, 인증서 객체 및 키 객체를 생성하거나 처리 시 부록 1. PKCS#11 객체속성 프로파일에서 명시한 객체 속성 템플릿을 준용하여야 한다.

7.1.1 데이터 객체(Data Object)

구동프로그램이 CKA_OBJECT_ID 속성을 지원하는 경우 다수 데이터 객체를 구별하기 위해 해당 객체의 OID를 속성 값으로 사용해야 한다.

[KCAC.TS.SIVID]에서 정의한 난수(R)를 저장하는 데이터 객체가 하나 이상 존재할 경우 이를 구분하기 위하여, 구동프로그램은 레이블속성(CKA_LABEL)을 지원해야 하며, 이 레이블 속성에 키 및 인증서객체의 식별자로 이용되는 CKA_ID 값을 'R4VID=CKA_ID(16진수 표기)'와 같이 표현하여 하나 이상의 데이터 객체 난수(R)를 구분할 수 있어야 한다. 여기서 'R4VID'은 [KCAC.TS.SIVID]에서 정의한 난수(R)을 위한 식별자를 나타내며,

CKA_ID값은 16진수로 표기하고 이들의 값은 '='로 연결한다. 난수(R)는 비밀객체 속성을 가져야 한다.

바이오 보안토큰 구동프로그램이 바이오 보안토큰 등록정보 및 사용자 등록정보를 지원하는 경우 데이터 객체를 사용하여야 한다. 이 경우, 구동프로그램은 바이오 보안토큰 등록정보 및 사용자 등록정보를 식별하기 위해 CKA_LABEL을 지원하여야 한다.

바이오 보안토큰 등록정보의 레이블 속성(CKA_LABEL)은 하나 이상의 보안토큰 등록정보를 구분할 수 있도록 'DevAuth=일련번호'로 표기한다(이 경우 '일련번호'는 1바이트 16진수로 표기하고 'DevAuth'와 '='로 연결한다). 또한, 사용자 등록정보의 레이블 속성(CKA_LABEL)은 법인의 경우 'Corporation', 개인의 경우 'Person'으로 표기한다.

CKA_VALUE에 표기되는 바이오 보안토큰 등록정보 및 사용자 등록정보는 비밀객체 속성을 가져야 한다.

7.1.2 인증서 객체(Certificate Object)

인증서는 공개객체이며 하나 이상의 인증서를 구분하기 위하여 키 및 인증서 객체의 식별자로 이용되는 CKA_ID 속성을 지원해야 한다.

7.1.3 키 객체(Key Object)

키 객체는 개인키, 공개키, 비밀키를 저장하기 위하여 이용되며, 개인키 및 비밀키 객체는 비밀객체 속성을 가져야 한다.

하나 이상의 개인키 구분을 위해 키 및 인증서객체의 식별자로 이용되는 CKA_ID 속성을 지원해야 한다. CKA_ID 속성값은 인증서 확장필드의 소유자 키 식별자 값을 사용할 것을 권고한다.

공개키는 공개객체이며 하나 이상의 공개키를 구분하기 위하여 키 및 인증서 객체의 식별자로 이용되는 CKA_ID 속성을 지원해야 한다.

개인키 및 공개키 객체는 인증서에 명시된 키 사용목적에 부합하는 속성을 가져야 하며 부록 1. PKCS#11 객체 속성 프로파일을 준용해야 한다.

7.2 함수

본 절에서는 전자서명인증체계 가입자 소프트웨어가 보안토큰의 기능을 이용하기 위해 구동프로그램이 지원해야 하는 함수 및 반환값을 정의하며, 구동프로그램은 부록 2. 보안토큰API(PKCS#11) 함수 프로파일 및 부록 6. 보안토큰 API(PKCS#11) 반환값 프로파일을 각각 준용하여야 한다.

보안토큰의 초기화, 자원해제, 일반적인 정보획득의 기능함수로, 구동프로그램은 C_initialize, C_Finalize, C_GetInfo, C_GetFunctionList를 지원해야 한다.

보안토큰의 슬롯 및 토큰에 대한 정보 획득, 토큰이 지원할 수 있는 메커니즘 집합 획득 등 토큰과 슬롯간의 연결관리 기능함수로, 구동프로그램은 C_GetSlotList, C_GetSlotInfo, C_GetTokenInfo, C_GetMechanismList, C_GetMechanismInfo를 지원해야 한다.

보안토큰의 세션관리 기능함수로, 구동프로그램은 C_OpenSession, C_CloseSession, C_CloseAllSessions, C_GetSessionInfo, C_Login, C_Logout을 지원해야 한다.

보안토큰의 객체생성, 객체검색 등 객체관리 기능함수로, 구동프로그램은 C_CreateObject, C_DestroyObject, C_FindObjectsInit, C_FindObjects, C_FindObjectsFinal, C_GetAttributeValue, C_SetAttributeValue를 지원해야 한다.

보안토큰의 전자서명 생성 기능함수로, 구동프로그램은 C_SignInit, C_Sign, C_SignUpdate, C_SignFinal를 지원해야 한다.

보안토큰의 암호키 복호화 기능함수로, 구동프로그램은 C_DecryptInit, C_Decrypt, C_Unwrap을 지원해야 한다.

보안토큰의 전자서명키쌍 생성 기능함수로, 구동프로그램은 C_GenerateKeyPair, C_SeedRandom, C_GenerateRandom을 지원해야 한다.

7.3 메커니즘(Mechanism)

본 절에서는 전자서명인증체계 가입자 소프트웨어가 보안토큰 기능을 이용하기 위해 구동프로그램이 지원해야 하는 메커니즘을 정의하며, 구동프로그램은 부록 3. 보안토큰 API(PKCS#11) 메커니즘 프로파일을 준용하여야 한다.

RSA 알고리즘을 이용한 전자서명 생성 및 암호키 복호화 등의 기능을 지원

하기 위해, 구동프로그램은 PKCS#11에 기반을 둔 CKM_RSA_PKCS 메커니즘을 지원해야 한다. 보안토큰의 처리 용량을 고려하여 전자문서에 대한 해쉬값은 가입자 소프트웨어에서 생성 또는 보안토큰 내부에서 생성할 수 있다.

8. 무선통신 지원 보안토큰 표준 API

본 장에서는 전자서명인증체계 가입자 소프트웨어가 무선통신단말 지원 보안토큰 기능을 이용하기 위해 부록 9. 무선통신 지원 보안토큰 표준 API를 준용하여야 한다.

부록 1. 보안토큰 API(PKCS#11) 객체 속성 프로파일

1. 일반 객체(Common Object Attribute)

속성	데이터 타입	내용	처리
CKA_CLASS	CK_OBJECT_CLASS	Object class	M

o 본 프로파일은 CKA_CLASS로 다음 아래와 같은 객체를 지원해야 한다.
- CKO_DATA, CKO_CERTIFICATE, CKO_PUBLIC_KEY, CKO_PRIVATE_KEY

o 본 프로파일은 CKA_CLASS로 다음 아래와 같은 객체 지원을 권고한다.
- CKO_SECRET_KEY, CKO_DOMAIN_PARAMETERS

2. 저장 객체(Storage Object)

2.1 저장 공통 객체(Common Storage Object Attribute)

속성	데이터 타입	내용	처리
CKA_TOKEN	CK_BBOOL	IF True : 토큰객체, IF False : 세션 객체	M
CKA_PRIVATE	CK_BBOOL	IF True : Private 객체	M
CKA_MODIFIABLE	CK_BBOOL	IF True : 객체 수정	M
CKA_LABEL	RFC2279 string	객체 설명	M

2.2 데이터 객체(Data Object)

속성	데이터 타입	내용	처리
CKA_APPLICATION	RFC2279 String	어플리케이션 설명	M
CKA_OBJECT_ID	Byte Array	DER 인코딩된 OID(Object identifier)	O
CKA_VALUE	Byte Array	객체 값	M

※ 데이터 객체 속성 템플릿 :

```
CK_OBJECT_CLASS class = CKO_DATA;
CK_UTF8CHAR label[] = "R4VID=FF3082A0....";
CK_UTF8CHAR application[] = “Accredited PKI Application”;
CK_BYTE data[] = “Sample data”;
CK_BBOOL true = TRUE;
CK_ATTRIBUTE template[] = {
  {CKA_CLASS, &class, sizeof(class)},
  {CKA_TOKEN, &true, sizeof(true)},
  {CKA_PRIVATE, &true, sizeof(true)},
  {CKA_LABEL, label, sizeof(label)-1},
  {CKA_APPLICATION, application, sizeof(application)-1},
  {CKA_VALUE, data, sizeof(data)}
```

```
};
```

※ 바이오 보안토큰 등록정보 데이터 객체 속성 템플릿(바이오 보안토큰 사용시) :

```
CK_OBJECT_CLASS class = CKO_DATA;
CK_UTF8CHAR label[] = "DevAuth=FF";
CK_UTF8CHAR application[] = “Accredited PKI Application” ;
CK_BYTE data[] = "FEDCBA98....” ;
CK_BBOOL true = TRUE;
CK_ATTRIBUTE template[] = {
  {CKA_CLASS, &class, sizeof(class)},
  {CKA_TOKEN, &true, sizeof(true)},
  {CKA_PRIVATE, &true, sizeof(true)},
  {CKA_LABEL, label, sizeof(label)-1},
  {CKA_APPLICATION, application, sizeof(application)-1},
  {CKA_VALUE, data, sizeof(data)}
};
```

※ 법인사용자 등록정보 데이터 객체 속성 템플릿(바이오 보안토큰 사용시) :

```
CK_OBJECT_CLASS class = CKO_DATA;
CK_UTF8CHAR label[] = "Corporation";
CK_UTF8CHAR application[] = “Accredited PKI Application” ;
CK_BYTE data[] = "7C25....” ;
CK_BBOOL true = TRUE;
CK_ATTRIBUTE template[] = {
  {CKA_CLASS, &class, sizeof(class)},
  {CKA_TOKEN, &true, sizeof(true)},
  {CKA_PRIVATE, &true, sizeof(true)},
  {CKA_LABEL, label, sizeof(label)-1},
  {CKA_APPLICATION, application, sizeof(application)-1},
  {CKA_VALUE, data, sizeof(data)}
};
```

※ 개인사용자 등록정보 데이터 객체 속성 템플릿(바이오 보안토큰 사용시) :

```
CK_OBJECT_CLASS class = CKO_DATA;
CK_UTF8CHAR label[] = "Person";
CK_UTF8CHAR application[] = “Accredited PKI Application” ;
CK_BYTE data[] = "AFFBC....” ;
CK_BBOOL true = TRUE;
CK_ATTRIBUTE template[] = {
  {CKA_CLASS, &class, sizeof(class)},
  {CKA_TOKEN, &true, sizeof(true)},
  {CKA_PRIVATE, &true, sizeof(true)},
  {CKA_LABEL, label, sizeof(label)-1},
  {CKA_APPLICATION, application, sizeof(application)-1},
  {CKA_VALUE, data, sizeof(data)}
};
```

2.3 인증서 객체(Certificate Object)

2.3.1 X.509 인증서 공통 객체(Common X.509 Certificate Object Attribute)

속성	데이터 타입	내용	처리
CKA_CERTIFICATE_TYPE	CK_CERTIFICATE_TYPE	인증서 타입	M
CKA_TRUSTED	CK_BBOOL	어플리케이션은 신뢰된 인증서로 간주	O

※ CKA_TRUSTED 속성은 특정 어플리케이션에 의해 설정되어져서는 안되며, 토큰을 초기화 할 때 설정되어야 함.

2.3.2 X.509 공개키 인증서 객체(X.509 Public Key Certificate Object Attribute)

속성	데이터 타입	내용	처리
CKA_SUBJECT	Byte Array	DER 인코딩된 인증서 가입자 이름	M
CKA_ID	Byte Array	Key identifier(Public/private)	M
CKA_ISSUER	Byte Array	DER 인코딩된 인증서 발급자 이름	O
CKA_SERIAL_NUMBER	Byte Array	DER 인코딩된 인증서 일련번호	O
CKA_VALUE	Byte Array	BER 인코딩된 인증서	M

※ X.509 공개키 인증서 객체 속성 템플릿 :

```
CK_OBJECT_CLASS class = CKO_CERTIFICATE;
CK_CERTIFICATE_TYPE certType = CKC_X_509;
CK_UTF8CHAR label[] = “LABEL 명”;
CK_BYTE subject[] = {...};
CK_BYTE id[] = {123};
CK_BYTE certificate[] = {...};
CK_BBOOL true = TRUE;
CK_BBOOL false = FALSE;
CK_ATTRIBUTE template[] = {
  {CKA_CLASS, &class, sizeof(class)},
  {CKA_CERTIFICATE_TYPE, &certType, sizeof(certType)};
  {CKA_TOKEN, &true, sizeof(true)},
  {CKA_LABEL, label, sizeof(label)-1},
  {CKA_SUBJECT, subject, sizeof(subject)},
  {CKA_ID, id, sizeof(id)},
  {CKA_PRIVATE, false, sizeof(false)},
  {CKA_VALUE, certificate, sizeof(certificate)}
};
```

2.3.3 X.509 속성 인증서 객체(X.509 Attribute Certificate Object Attribute)

속성	데이터 타입	내용	처리
CKA_OWNER	Byte Array	DER 인코딩된 속성인증서 소유자 이름	M
CKA_AC_ISSUER	Byte Array	DER 인코딩된 속성인증서 발급자 이름	O
KA_SERIAL_NUMBER	Byte Array	DER 인코딩된 인증서 일련번호	O
CKA_ATTR_TYPES	Byte Array	BER 인코딩된 속성 OID	O
CKA_VALUE	Byte Array	BER 인코딩된 속성 인증서	M

※ 속성인증서를 사용할 경우 위 속성 인증서의 객체 속성을 준용할 것을 권고함.

※ X.509 속성인증서 객체 템플릿 :

```
CK_OBJECT_CLASS class = CKO_CERTIFICATE;
CK_CERTIFICATE_TYPE certType = CKC_X_509_ATTR_CERT;
CK_UTF8CHAR label[] = "An attribute certificate object";
CK_BYTE owner[] = {...};
CK_BYTE certificate[] = {...};
CK_BYTE subject[] = {...};
CK_BYTE id[] = {123};
CK_BBOOL true = TRUE;
CK_BBOOL false = FALSE;
CK_ATTRIBUTE template[] = {
  {CKA_CLASS, &class, sizeof(class)},
  {CKA_CERTIFICATE_TYPE, &certType, sizeof(certType)};
  {CKA_TOKEN, &true, sizeof(true)},
  {CKA_LABEL, label, sizeof(label)-1},
  {CKA_OWNER, owner, sizeof(owner)},
  {CKA_SUBJECT, subject, sizeof(subject)},
  {CKA_ID, id, sizeof(id)},
  {CKA_PRIVATE, false, sizeof(false)},
  {CKA_VALUE, certificate, sizeof(certificate)}
 };
```

2.4 키 객체(Key Object)

2.4.1 키 공통 객체(Common Key Attribute)

속성	데이터 타입	내용	처리
CKA_KEY_TYPE	CKA_KEY_TYPE	키 타입	M
CKA_ID	Byte Array	KEY Identifier	M
CKA_START_DATE	CK_DATE	키의 유효 시작 날짜	-
CKA_END_DATE	CK_DATE	키의 유효 만료 날짜	-
CKA_DERIVE	CK_BBOOL	IF TRUE : 키 유도 지원	O
CKA_LOCAL	CK_BBOOL	IF TRUE : 토큰 내에서 키생성	O
CKA_KEY_GEN_MECHANISM	CK_MECHANISM_TYPE	키 생성 메카니즘 Identifier	O

※ 보안토큰 API 키 속성과 X.509V3 인증서와의 Mapping Table

공개키/개인키를 위한 X.509 v3 인증서 Key Usage	공개키/개인키를 위한 보안토큰 API 속성
dataEncipherment	CKA_ENCRYPT
digitalSignature, keyCertSign, cRLSign	CKA_VERIFY
digitalSignature, keyCertSign, cRLSign	CKA_VERIFY_RECOVER
keyAgreement	CKA_DERIVE
keyEncipherment	CKA_WRAP
nonRepudiation	CKA_VERIFY
nonRepudiation	CKA_VERIFY_RECOVER

2.4.2 공개키 객체(Public Key Object)

2.4.2.1 공개키 공통 객체(Common Public Key Object Attribute)

속성	데이터 타입	내용	처리
CKA_SUBJECT	Byte array	DER 인코딩된 소유자 이름	M
CKA_ENCRYPT	CK_BBOOL	암호화	O
CKA_VERIFY	CK_BBOOL	부가형 전자서명 검증	M
CKA_VERIFY_RECOVER	CK_BBOOL	복원형 전자서명 검증	O
CKA_WRAP	CK_BBOOL	키 암호화	M
CKA_TRUSTED	CK_BBOOL	신뢰되는 키	O

※ CKA_VERIFY 속성은 전자서명 검증 공개키에 설정되어야 하며, CKA_WRAP 속성은 암호키분배 공개키에 설정 되어야 함.

2.4.2.2 RSA 공개키 객체(RSA Public Key Object Attribute)

속성	데이터 타입	내용	처리
CKA_MODULUS	Big integer	Modulus n	M
CKA_MODULUS_BITS	CK_ULONG	n의 길이(비트)	M
CKA_PUBLIC_EXPONENT	Big integer	Public exponent e	M

※ RSA공개키 객체 템플릿(C_GenerateKeyPair 함수 이용시) :

```
CK_OBJECT_CLASS class = CKO_PUBLIC_KEY;
CK_KEY_TYPE keyType = CKK_RSA;
CK_UTF8CHAR label[] =  An RSA public key object ;
CK_BYTE exponent[] = {...};
CK_ULONG bits = 1024;
CK_BYTE id[] = {123};
CK_BBOOL true = TRUE;
CK_BBOOL false = FALSE;
CK_ATTRIBUTE template[] = {
   {CKA_CLASS, &class, sizeof(class)},
   {CKA_KEY_TYPE, &keyType, sizeof(keyType)},
```

```
    {CKA_TOKEN, &true, sizeof(true)},
    {CKA_LABEL, label, sizeof(label)-1},
    {CKA_ID, id, sizeof(id)},
    {CKA_WRAP, &false, sizeof(false)},
    {CKA_ENCRYPT, &false, sizeof(false)},
    {CKA_MODULUS_BITS, &bits, sizeof(bits)},
    {CKA_PRIVATE, false, sizeof(false)},
    {CKA_VERIFY, true, sizeof(true)},
    {CKA_PUBLIC_EXPONENT, exponent, sizeof(exponent)}
};
```

※ RSA공개키 객체 템플릿(C_CreateObject 함수 이용시) :

```
CK_OBJECT_CLASS class = CKO_PUBLIC_KEY;
CK_KEY_TYPE keyType = CKK_RSA;
CK_UTF8CHAR label[] =  An RSA public key object ;
CK_BYTE exponent[] = {...};
CK_BYTE modulus[] = {...};
CK_BYTE subject[] = {...};
CK_BYTE id[] = {123};
CK_BBOOL true = TRUE;
CK_BBOOL false = FALSE;
CK_ATTRIBUTE template[] = {
   {CKA_CLASS, &class, sizeof(class)},
   {CKA_KEY_TYPE, &keyType, sizeof(keyType)},
   {CKA_TOKEN, &true, sizeof(true)},
   {CKA_LABEL, label, sizeof(label)-1},
   {CKA_ID, id, sizeof(id)},
   {CKA_WRAP, &false, sizeof(false)},
   {CKA_ENCRYPT, &false, sizeof(false)},
   {CKA_MODULUS, modulus, sizeof(modulus)},
   {CKA_SUBJECT, subject, sizeof(subject)},
   {CKA_PRIVATE, false, sizeof(false)},
   {CKA_VERIFY, true, sizeof(true)},
   {CKA_PUBLIC_EXPONENT, exponent, sizeof(exponent)}
};
```

※ 전자서명 검증 공개키에는 CKA_VERIFY를 TRUE로 설정하고, CKA_WRAP 및 CKA_ENCRYPT을 FALSE로 설정하여야 함

※ 암호키 분배 공개키에는 CKA_VERIFY를 FALSE로 설정하고, CKA_WRAP 및 CKA_ENCRYPT을 TRUE로 설정하여야 함

2.4.2.3 EC 공개키 객체(Elliptic Curve Public Key Object Attribute)

속성	데이터 타입	내용	처리
CKA_EC_PARAMS (CKA_ECDSA_PARAMS)	Byte Array	DER인코딩된 ASNI X9.62 파라메터 값	M
CKA_EC_POINT	Byte Array	DER인코딩된 ASNI X9.62 ECPoint 값 Q	M

※ EC 공개키를 사용할 경우 위 객체 속성을 준용할 것을 권고함.

※ EC 공개키 객체 템플릿(C_CreateObject 함수 이용시) :

```
CK_OBJECT_CLASS class = CKO_PUBLIC_KEY;
CK_KEY_TYPE keyType = CKK_EC;
CK_UTF8CHAR label[] =  An EC public key object ;
CK_BYTE ecParams[] = {...};
CK_BYTE ecPoint[] = {...};
CK_BBOOL true = TRUE;
CK_ATTRIBUTE template[] = {
  {CKA_CLASS, &class, sizeof(class)},
  {CKA_KEY_TYPE, &keyType, sizeof(keyType)},
  {CKA_TOKEN, &true, sizeof(true)},
  {CKA_LABEL, label, sizeof(label)-1},
  {CKA_EC_PARAMS, ecParams, sizeof(ecParams)},
  {CKA_EC_POINT, ecPoint, sizeof(ecPoint)}
};
```

2.4.3 개인키 객체(Private Key Object)

2.4.3.1 개인키 공통 객체 속성(Common Private Key Object Attribute)

속성	데이터 타입	내용	처리
CKA_SUBJECT	Byte Array	DER인코딩된 인증서 소유자 이름	M
CKA_SENSITIVE	CK_BBOL	TRUE : Plain-Text 형태로 Export 불가	M
CKA_SECONDARY_AUTH	CK_BBOL	IF TRUE : 두 번의 인증 요구	O
CKA_AUTH_PIN_FLAGS	CK_FLAGS	IF TRUE : 두 번의 PIN 인증, IF FALSE : 속성값 0	O
CKA_DECRYPT	CK_BBOL	TRUE : 복호화	O
CKA_SIGN	CK_BBOL	TRUE : 부가형 전자서명	M
CKA_SIGN_RECOVER	CK_BBOL	IF TRUE : 복원형 전자서명	O
CKA_UNWRAP	CK_BBOL	TRUE : 키 복호화	M
CKA_EXTRACTABLE	CK_BBOL	FALSE : 키의 Export 불가	M
CKA_ALWAYS_SENSITIVE	CK_BBOL	IF TRUE : 항상 Plain-Text 형태로 Export 불가	O
CKA_NEVER_EXTRACTABLE	CK_BBOL	IF TRUE : 키가 Plain-Text 형태로 나오지 않음	M

※ CKA_SIGN 속성은 전자서명 생성 개인키에 설정되어야 하며, CKA_UNWRAP 속성은 암호 키분배 개인키에 설정 되어야 함.

2.4.3.2 RSA 개인키 객체(RSA Private Key Object Attribute)

속성	데이터 타입	내용	처리
CKA_MODULUS	Big integer	Modulus n	M
CKA_PUBLIC_EXPONENT	Big integer	Public exponent e	M
CKA_PRIVIATE_EXPONENT	Big integer	Private exponent d	M
CKA_PRIME_1	Big integer	Prime p	M
CKA_PRIME_2	Big integer	Prime q	M

속성	데이터 타입	내용	처리
CKA_EXPONENT_1	Big integer	Private exponent d modulo p-1	M
CKA_EXPONENT_2	Big integer	Private exponent d modulo p-2	M
CKA_COEFFICIENT	Big integer	CRT coefficient q-1mod p	M

※ RSA 개인키 객체 템플릿(C_GenerateKeyPair 함수 이용 시) :

```
CK_OBJECT_CLASS class = CKO_PRIVATE_KEY;
CK_KEY_TYPE keyType = CKK_RSA;
CK_UTF8CHAR label[] =  "LABEL 명" ;
CK_BYTE id[] = {123};
CK_BBOOL true = TRUE;
CK_BBOOL false = FALSE;
CK_ATTRIBUTE template[] = {
   {CKA_CLASS, &class, sizeof(class)},
   {CKA_KEY_TYPE, &keyType, sizeof(keyType)},
   {CKA_TOKEN, &true, sizeof(true)},
   {CKA_LABEL, label, sizeof(label)-1},
   {CKA_ID, id, sizeof(id)},
   {CKA_SENSITIVE, &true, sizeof(true)},
   {CKA_DECRYPT, &false, sizeof(false)},
   {CKA_SIGN, &true, sizeof(true)},
   {CKA_UNWRAP, &false, sizeof(false)},
   {CKA_PRIVATE, &true, sizeof(true)},
};
```

※ RSA 개인키 객체 템플릿(C_CreateObject 함수 이용시) :

```
CK_OBJECT_CLASS class = CKO_PRIVATE_KEY;
CK_KEY_TYPE keyType = CKK_RSA;
CK_UTF8CHAR label[] =  "LABEL 명" ;
CK_BYTE subject[] = {...};
CK_BYTE id[] = {123};
CK_BYTE modulus[] = {...};
CK_BYTE publicExponent[] = {...};
CK_BYTE privateExponent[] = {...};
CK_BYTE prime1[] = {...};
CK_BYTE prime2[] = {...};
CK_BBOOL true = TRUE;
CK_BBOOL false = FALSE;
CK_BYTE exponent1[] = {...};
CK_BYTE exponent2[] = {...};
CK_BYTE coefficient[] = {...};
CK_ATTRIBUTE template[] = {
   {CKA_CLASS, &class, sizeof(class)},
   {CKA_KEY_TYPE, &keyType, sizeof(keyType)},
   {CKA_TOKEN, &true, sizeof(true)},
   {CKA_LABEL, label, sizeof(label)-1},
   {CKA_SUBJECT, subject, sizeof(subject)},
   {CKA_ID, id, sizeof(id)},
```

```
    {CKA_SENSITIVE, &true, sizeof(true)},
    {CKA_SIGN, &true, sizeof(true)},
    {CKA_UNWRAP, &false, sizeof(false)},
    {CKA_DECRYPT, &false, sizeof(false)},
    {CKA_PRIVATE, &true, sizeof(true)},
    {CKA_MODULUS, modulus, sizeof(modulus)},
    {CKA_PUBLIC_EXPONENT, publicExponent, sizeof(publicExponent)},
    {CKA_PRIVATE_EXPONENT, privateExponent, sizeof(privateExponent)},
    {CKA_PRIME_1, prime1, sizeof(prime1)},
    {CKA_PRIME_2, prime2, sizeof(prime2)},
    {CKA_EXPONENT_1, exponent1, sizeof(exponent1)},
    {CKA_EXPONENT_2, exponent2, sizeof(exponent2)},
    {CKA_COEFFICIENT, coefficient, sizeof(coefficient)}
};
```

※ 전자서명 생성 개인키에는 CKA_SIGN을 TRUE로 설정하고, CKA_UNWRAP 및 CKA_DECRYPT을 FALSE로 설정하여야 함

※ 암호키 분배 개인키에는 CKA_SIGN을 FALSE로 설정하고, CKA_UNWRAP 및 CKA_DECRYPT을 TRUE로 설정하여야 함

2.4.3.3 EC 개인키 객체 속성(Elliptic Curve Private Key Object Attribute)

속성	데이터 타입	내용	처리
CKA_EC_PARAMS (CKA_ECDSA_PARAMS)	Byte Array	DER인코딩된 ASNI X9.62 파라메터 값	M
CKA_VALUE	Big integer	ASNI X9.62 개인키 값 d	M

※ EC 개인키를 사용할 경우 위 객체 속성을 준용할 것을 권고함.

※ EC 개인키 객체 템플릿(C_CreateObject 함수 이용시) :

```
CK_OBJECT_CLASS class = CKO_PRIVATE_KEY;
CK_KEY_TYPE keyType = CKK_EC;
CK_UTF8CHAR label[] =  An EC private key object ;
CK_BYTE subject[] = {...};
CK_BYTE id[] = {123};
CK_BYTE ecParams[] = {...};
CK_BYTE value[] = {...};
CK_BBOOL true = TRUE;
CK_BBOOL false = FALSE;
CK_ATTRIBUTE template[] = {
  {CKA_CLASS, &class, sizeof(class)},
  {CKA_KEY_TYPE, &keyType, sizeof(keyType)},
  {CKA_TOKEN, &true, sizeof(true)},
  {CKA_LABEL, label, sizeof(label)-1},
  {CKA_SUBJECT, subject, sizeof(subject)},
  {CKA_ID, id, sizeof(id)},
  {CKA_SENSITIVE, &true, sizeof(true)},
  {CKA_DERIVE, &true, sizeof(true)},
```

```
  {CKA_PRIVATE, & true, sizeof(true)},
  {CKA_EC_PARAMS, ecParams, sizeof(ecParams)},
  {CKA_VALUE, value, sizeof(value)}
};
```

2.4.4 비밀키 객체(Secret Key Object)

속성	데이터 타입	내용	처리
CKA_SENSITIVE	CK_BBOL	IF TRUE : 민감한 정보	M
CKA_ENCRYPT	CK_BBOL	IF TRUE : 암호화	M
CKA_DECRYPT	CK_BBOL	IF TRUE : 복호화	M
CKA_SIGN	CK_BBOL	IF TRUE : 부가형 서명	-
CKA_VERIFY	CK_FLAGS	IF TRUE : 부가형 서명 검증	-
CKA_WRAP	CK_BBOL	IF TRUE : 키 암호화	M
CKA_UNWRAP	CK_BBOL	IF TRUE : 키 복호화	M
CKA_EXTRACTABLE	CK_BBOL	IF TRUE : 키의 Export 지원	M
CKA_ALWAYS_SENSITIVE	CK_BBOL	IF TRUE : 항상 Plain-Text 형태로 Export 불가	O
CKA_NEVER_EXTRACTABLE	CK_BBOL	IF TRUE : 키의 Export 지원 절대 금지	O

2.4.4.1 비밀키 공통 객체(Common Secret Key Object)

※ 비밀키 객체를 사용할 경우 위 객체 속성을 준용할 것을 권고함.

2.4.4.2 3DES 비밀키 객체(3DES secret key objects)

속성	데이터 타입	내용	처리
CKA_VALUE	Byte Array	키 값	M

※ 3DES 비밀키 객체 템플릿 :

```
CK_OBJECT_CLASS class = CKO_SECRET_KEY;
CK_KEY_TYPE keyType = CKK_DES3;
CK_UTF8CHAR label[] =  A DES3 secret key object ;
CK_BYTE value[24] = {...};
CK_BBOOL true = TRUE;
CK_ATTRIBUTE template[] = {
  {CKA_CLASS, &class, sizeof(class)},
  {CKA_KEY_TYPE, &keyType, sizeof(keyType)},
  {CKA_TOKEN, &true, sizeof(true)},
  {CKA_LABEL, label, sizeof(label)-1},
  {CKA_PRIVATE, & true, sizeof(true)},
  {CKA_ENCRYPT, &true, sizeof(true)},
  {CKA_DECRYPT, &true, sizeof(true)},
  {CKA_VALUE, value, sizeof(value)}
};
```

2.4.4.3 SEED 비밀키 객체(SEED secret key objects)

속성	데이터 타입	내용	처리
CKA_VALUE	Byte Array	키 값	M

※ SEED 비밀키 객체 템플릿 :

```
CK_OBJECT_CLASS class = CKO_SECRET_KEY;
CK_KEY_TYPE keyType = CKK_SEED;
CK_UTF8CHAR label[] =  A SEED secret key object ;
CK_BYTE value[24] = {...};
CK_BBOOL true = TRUE;
CK_ATTRIBUTE template[] = {
  {CKA_CLASS, &class, sizeof(class)},
  {CKA_KEY_TYPE, &keyType, sizeof(keyType)},
  {CKA_TOKEN, &true, sizeof(true)},
  {CKA_LABEL, label, sizeof(label)-1},
  {CKA_PRIVATE, & true, sizeof(true)},
  {CKA_ENCRYPT, &true, sizeof(true)},
  {CKA_DECRYPT, &true, sizeof(true)},
  {CKA_VALUE, value, sizeof(value)}
};
```

부록 2. 보안토큰 API(PKCS#11) 함수 프로파일

API 함수 정의	내용	처리
기본목적(General Purpose)		
C_Initialize	보안토큰 API를 초기화	M
C_Finalize	보안토큰 API와 연관된 다양한 자원을 해제	M
C_GetInfo	보안토큰 API에 대한 일반적인 정보를 획득	M
C_GetFunctionList	지원하는 보안토큰 API 함수 집합을 획득	M
슬롯 & 토큰 관리(Slot and Token Management)		
C_GetSlotList	시스템에 있는 슬롯 집합을 획득	M
C_GetSlotInfo	특정 슬롯에 관한 정보를 획득	M
C_GetTokenInfo	특정 토큰에 관한 정보를 획득	M
C_WaitForSlotEvent	슬롯 이벤트(토큰 추가, 제거)발생에 따른 대기상태	O
C_GetMechanismList	토큰이 지원하는 메커니즘 집합을 획득	M
C_GetMechanismInfo	특정 메커니즘에 관한 정보를 획득	M
C_InitToken	토큰 초기화	O
C_InitPIN	사용자 PIN 초기화	O
C_SetPIN	사용자 PIN 수정	O
세션 관리(Session Management)		
C_OpenSession	응용시스템과 토큰간의 연결을 생성	M
C_CloseSession	세션을 종료	M
C_CloseAllSessions	특정 토큰과 연관된 모든 세션을 종료	M
C_GetSessionInfo	세션에 대한 정보 획득	M
C_GetOperationState	세션의 암호 오퍼레이션 상태 획득	O
C_SetOperationState	세션 암호 오퍼레이션 상태 수정	O
C_Login	토큰에 로그인	M
C_Logout	토큰으로부터 로그아웃	M
객체 관리(Object management)		
C_CreateObject	객체 생성	M
C_CopyObject	복사본 객체 생성	O
C_DestroyObject	객체 파기	M
C_GetObjectSize	객체 사이즈 획득	O
C_GetAttributeValue	객체 속성 획득	M
C_SetAttributeValue	객체 속성 수정	M
C_FindObjectInit	객체 검색 초기화	M
C_FindObjects	객체 검색	M
C_FindObjectsFinal	객체 검색 종료	M
암호화(Encryption)		
C_EncryptInit	암호화 기능을 초기화	O
C_Encrypt	단일 부분에 대한 암호화 수행	O
C_EncryptUpdate	다중 부분에 대한 암호화 수행	O

API 함수 정의	내용	처리
C_EncryptFinal	다중 부분에 대한 암호화 수행 종료	O
복호화(Encryption)		
C_DecryptInit	복호화 기능 초기화	M
C_Decrypt	단일 암호화 부분에 대한 복호화 수행	M
C_DecryptUpdate	다중 암호화 부분에 대한 복호화 수행	O
C_DecryptFinal	다중 암호화 부분에 대한 복호화 수행 종료	O
메시지 압축(Message Digesting)		
C_DigestInit	메시지 압축 수행 초기화	O
C_Digest	단일 부분 데이터 압축 수행	O
C_DigestUpdate	다중 부분에 대한 압축 수행	O
C_DigestKey	키 압축 수행	O
C_DigestFinal	다중 부분에 대한 압축 수행 종료	O
전자서명 및 MAC (Signing & MACing)		
C_SignInit	서명 기능을 초기화	M
C_Sign	단일 부분 서명을 수행	M
C_SignUpdate	다중 부분에 대한 서명 수행	O
C_SignFinal	다중 부분에 대한 서명 수행 종료	O
C_SignRecoverInit	복원형 서명 기능 초기화	O
C_SignRecover	복원형 서명 수행	O
전자서명 및 MAC 검증		
C_VerifyInit	검증 기능을 초기화	O
C_Verify	단일 부분 서명에 대한 검증을 수행	O
C_VerifyUpdate	다중 부분에 대한 검증 수행	O
C_VerifyFinal	다중 부분에 대한 검증 종료	O
C_VerifyRecoverInit	복원형 검증 기능 초기화	–
C_VerifyRecover	복원형 검증 기능 수행	–
Dual-purpose cryptographic		
C_DigestEncryptUpdate	압축 및 암호화 수행	O
C_DecryptDigestUpdate	복호화 및 압축수행	O
C_SignEncryptUpdate	서명 및 암호화 수행	O
C_DecryptVerifyUpdate	복호화 및 검증 수행	O
키 관리(Key Management)		
C_GenerateKey	비밀키 혹은 도메인 파라미터 생성	O
C_GenerateKeyPair	비대칭키쌍 생성	M
C_WrapKey	키 암호화	O
C_UnWrapKey	키 복호화	M
C_DeriveKey	기본 키로부터 키 호출(Vendor-defined)	–
렌덤 넘버 생성(Random Number Generation)		
C_SeedRandom	seed 값 추가	M
C_GenerateRandom	랜덤 혹은 의사랜덤을 생성	M

API 함수 정의	내용	처리
Parallel 함수 관리(Parallel function 관리)		
C_GetFunctionStatus	Legacy 함수는 항상 CKR_FUNCTION_NOT_PARALLEL 리턴함	-
C_CancelFunction	Legacy 함수는 항상 CKR_FUNCTION_NOT_PARALLEL 리턴함	-

부록 3. 보안토큰 API(PKCS#11) 메커니즘 프로파일

메카니즘 정의	내용	처리
RSA 메카니즘		
CKM_RSA_PKCS	PKCS#1 Version1.5에 정의된 RSA 암호시스템과 블록형식에 기반한 메커니즘	M
CKM_RSA_PKCS_KEY_PAIR_GEN	PKCS#1에 정의된 RSA키를 생성하는 메커니즘	M
CKM_RSA_PKCS_OAEP	PKCS#1 Version2.0에 정의된 RSA 암호시스템과 블록형식에 기반한 메커니즘	O
CKM_SHA1_RSA_PKCS	SHA-1 해쉬알고리즘에 기반한 전자서명 메커니즘	O
CKM_SHA256_RSA_PKCS	SHA256 해쉬알고리즘에 기반한 전자서명 메커니즘	O
ECDSA 메카니즘		
CKM_ECDSA_PKCS_KEY_PAIR_GEN	키를 생성하는 메커니즘	O
CKM_SHA1_ECDSA_PKCS	SHA-1 해쉬알고리즘에 기반한 전자서명 메카니즘	O
ECDH 메카니즘		
CKM_ECDH1_DERIVE	ANSI X9.63에 기반한 키 유도 메카니즘	O
CKM_ECDH1_COFACTOR_DERIVE	ANSI X9.63에 기반한 ECDH 키 유도 메카니즘	O
CKM_ECDH1_COFACTOR_DERIVE	ANSI X9.63에 기반한 Cofactor ECDH 키 유도 메카니즘	O
Block cipher 메커니즘		
CKM_DES3_GEN	블록 Cipher 키 생성 메커니즘	O
CKM_DES3_ECB	ECB(Electronic codebook) 메커니즘(암/복호화)	O
CKM_DES3_CBC	CBC(Cipher-block chaining) 메커니즘(암/복호화)	O
CKM_DES3_CBC_PAD	CBC_PAD (PKCS Padding이 추가된 Cipher-block chaining) 메커니즘(암/복호화)	O
CKM_DES3_MAC_GENERAL	General-legnth MACing 메커니즘(서명/검증)	O
CKM_DES3_MAC	General MACing 메커니즘(서명/검증)	O
CKM_SEED_GEN	블록 Cipher 키 생성 메커니즘	O
CKM_SEED_ECB	ECB(Electronic codebook) 메커니즘(암/복호화)	O
CKM_SEED_CBC	CBC(Cipher-block chaining) 메커니즘(암/복호화)	O
CKM_SEED_CBC_PAD	CBC_PAD (PKCS Padding이 추가된 Cipher-block chaining) 메커니즘(암/복호화)	O
CKM_ARIA_KEY_GEN	블록 Cipher 키 생성 메커니즘	O
CKM_ARIA_ECB	ECB(Electronic codebook) 메커니즘(암/복호화)	O
CKM_ARIA_CBC	CBC(Cipher-block chaining) 메커니즘(암/복호화)	O
CKM_ARIA_CBC_PAD	CBC_PAD (PKCS Padding이 추가된 Cipher-block chaining) 메커니즘(암/복호화)	O
CKM_ARIA_MAC_GENERAL	General-legnth MACing 메커니즘(서명/검증)	O
CKM_ARIA_MAC	General MACing 메커니즘(서명/검증)	O
CKM_AES_KEY_GEN	블록 Cipher 키 생성 메커니즘	O
CKM_AES_ECB	ECB(Electronic codebook) 메커니즘(암/복호화)	O
CKM_AES_CBC	CBC(Cipher-block chaining) 메커니즘(암/복호화)	O
CKM_AES_CBC_PAD	CBC_PAD (PKCS Padding이 추가된 Cipher-block chaining) 메	O

메카니즘 정의	내용	처리
	커니즘(암/복호화)	
CKM_AES_MAC_GENERAL	General-legnth MACing 메커니즘(서명/검증)	O
CKM_AES_MAC		
CKM_SHA_1	FIPS PUB 180-1에 정의된 해쉬 메커니즘	O
CKM_SHA_1_HMAC_GENERAL	General-length SHA-1-HMAC 메커니즘	O
CKM_SHA_1_HMAC	SHA-1-HMAC 메커니즘	O
CKM_SHA_1_KEY_DERIVATION	SHA-1 Key 유도 메커니즘	O
SHA-256 메커니즘		
CKM_SHA256	FIPS PUB 180-2에 정의된 해쉬 메커니즘	O
CKM_SHA256_HMAC_GENERAL	General-length SHA-256-HMAC 메커니즘	O
CKM_SHA256_HMAC	SHA-256-HMAC 메커니즘	O
CKM_SHA256_KEY_DERIVATION	SHA-256 Key 유도 메커니즘	O
PKCS#5 and PKCS#5-style password-based encryption 메커니즘		
CKM_PKCS5_PBKD2	PKCS#5에 정의된 패스워드 기반 암호화 키 생성 메커니즘	O
PKCS#12 password-based encryption/authentication 메커니즘		
CKM_PBE_SHA1_DES3_EDE_CBC	3DES를 이용한 패스워드 기반 암호화 메커니즘	O
CKM_PBE_SHA1_SEED_EDE_CBC	SEED를 이용한 패스워드 기반 암호화 메커니즘	O
CKM_PBA_SHA1_WITH_SHA1_HMAC	Salt와 패스워드로부터 160bit secret 키를 생성하기 위한 메커니즘	O

부록 4. 환경파일을 이용한 보안토큰 구동프로그램 위치정보 관리

1. 위치정보 관리

보안토큰 API와 관련한 구동프로그램이 설치되어 있는 위치정보 및 구동프로그램에 대한 정보를 관리하기 위해 환경파일을 이용한다.

환경파일의 파일명 및 위치는 다음의 [표 1]과 같이 정의한다. 스마트폰 등 모바일 플랫폼과 같이 구동프로그램이 별도로 설치 및 제공되지 않고 모바일 앱의 구성모듈로 존재하는 경우 구동프로그램 위치정보 관리가 필요치 않으며, 별도의 환경파일을 구성하지 않는다.

구분		환경파일
데스크톱 플랫폼	Windows 계열	%SYSTEM%\npki_pkcs11.cnf
	Unix, Linux 계열	(사용자계정)/.npki_pkcs11.cnf
	Mac	(사용자계정)/.npki_pkcs11.cnf

[표 1] 보안토큰 구동프로그램 환경파일

2. 환경파일 구성

환경파일은 [PKCS#11.Driver] 섹션으로 시작하며, 하나의 키(Driver)로 구성된다. Driver에는 하위 섹션명이 나열되며 하나 이상의 하위 섹션명은 공백문자로 구분한다. 하위 섹션명은 보안토큰 제품정보(ID)를 사용한다.

하위 섹션명은 네 개의 키(Info, Name, Programs, SignatureToken)로 구성된다. Info에는 구동프로그램 정보와 구동프로그램 버전정보를 나열하며 두 정보는 ':'로 구분한다. 단, 바이오 보안토큰의 경우 구동프로그램 정보 앞에 BIO_를 연접하여 사용하고 스마트인증의 경우 Mobile_을 연접하여 사용한다.

Name은 보안토큰 API 관련 구동프로그램 위치정보 및 구동프로그램 명이며, Programs에 기타 보안토큰 관련 구동프로그램 위치정보 및 구동프로그램 명을 나열하며 하나 이상의 구동프로그램은 ','로 구분한다. SignatureToken은 구동프로그램 검증정보의 위치정보 및 파일명을 기입하며, 보안토큰 구동프로그램 검증정보는 공인인증체계 기술규격에 적합한 보안토큰에 대해서만 생성된다.

3. 환경파일(npki_pkcs11.cnf) 예제

```
[PKCS#11.Driver] //섹션
Driver=Vid_0000&Pid_0001 Vid_0001&Pid_0001 Vid_0002&Pid_0001 USIM_001//하위 섹션명

[Vid_0000&Pid_0001] //하위 섹션명
Info=KISA Token:0.0.0.1 //Info 키
Name=C:\Program Files\pkcs11.dll //Name 키
Programs=C:\Program Files\cert.dll,C:\Program Files\device.sys

[Vid_0001&Pid_0001]
Info=KISA Token2:0.0.0.2
Name=C:\Windows\npki_pkcs11.dll
Programs=C:\Program Files\cert1.dll,C:\Program Files\device1.sys
SignatureToken=C:\Program Files\Vid_0001&Pid_0001.der

[Vid_0002&Pid_0001]
Info=BIO_KISA Token3:0.0.0.3
Name=C:\Program Files\npki2_pkcs11.dll
Programs=C:\Program Files\cert2.dll,C:\Program Files\device2.sys
SignatureToken=C:\Program Files\Vid_0002&Pid_0001.der

[USIM_0001]
Info=Mobile_KISA Token4:0.0.0.4
Name=C:\Program Files\usim\npki3_pkcs11.dll
Programs=C:\Program Files\usim\cert3.dll,C:\Program Files\usim\device3.sys
SignatureToken=C:\Program Files\usim\USIM_0001.der
```

부록 5. 보안토큰 구동프로그램 검증정보 처리

1. 보안토큰 구동프로그램 검증정보

보안토큰 구동프로그램 검증정보(이하 검증정보)는 해당 플랫폼에 설치된 구동프로그램의 무결성 및 구현적합성을 확인할 수 있는 정보로써, SignatureValue의 DER 인코딩 값으로 구성된다.

2. 검증정보 확인

가입자 소프트웨어는 [부록 4.] 환경파일 위치정보를 통해 검증정보 파일의 위치를 확인하고, 해당 검증정보의 해쉬값(hashValue) 및 전자서명값(signature) 검증을 통해 구동프로그램의 무결성 및 구현적합성을 확인할 수 있다.

특히, 전자서명값 검증은 해당 플랫폼에 설치된 최상위인증기관 인증서 또는 정보통신망을 통해 안전하게 획득한 최상위인증기관 인증서를 통해 검증하여야 한다.

모바일 플랫폼의 경우 [부록 4]의 구동프로그램 위치정보 환경파일을 사용하지 않으며, 구동프로그램이 모바일 앱의 구성모듈로 사용되는 경우 별도의 검증정보를 사용하지 않는다.

3. 검증정보 ASN.1

```
ClientHSM DEFINITIONS ::= BEGIN

IMPORTS

 -- Imports from RFC 3280 [PROFILE], Appendix A.1
AlgorithmIdentifier, Certificate, CertificateList, CertificateSerialNumber, Name
FROM PKIX1Explicit88 { iso(1) identified-organization(3) dod(6) internet(1)
security(5) mechanisms(5) pkix(7) mod(0) pkix1-explicit(18) };

SignatureToken ::= SEQUENCE {
        driverName        PrintableString,
          -- 보안토큰 구동프로그램 DLL 이름
        hashID            AlgorithmIdentifier,
          -- 해쉬 알고리즘
```

```
        hashValue       OCTET STRING
          -- 보안토큰 구동프로그램 DLL에 대한 해쉬값으로, 가입자소프트웨어는
          -- 시스템에 설치된 driverName의 DLL을 입력값으로 해쉬하여 hashValue와
          -- 비교함으로서 사용자PC에 설치된 무결성을 검증
}

SignerAndSerialNumber ::= SEQUENCE {
     issuer Name,
     serialNumber CertificateSerialNumber }

SignatureValue ::= SEQUENCE {
        toBeSigned              SEQUENCE OF SignautreToken,
          -- 보안토큰 구동프로그램 DLL이 여러 DLL로 구성되어 있을 경우
          -- 각 DLL에 대한 무결성을 보증하기 위해 SEQUENCE OF로 구성
        signatureAlgorithm      AlgorithmIdentifier,
          -- 전자서명 알고리즘의 OID
        signerAndSerialNumber   SignerAndSerialNumber,
          -- 전자서명 생성자의 (KISA 3280 루트인증서)
        signature               OCTET STRING
          -- toBeSigned에 대한 전자서명값
}
END
```

부록 6. 보안토큰 API(PKCS#11) 반환값 프로파일

종류	반환값 및 내용	처리
일반	CKR_OK 보안토큰이 성공적으로 기능을 수행	M
	CKR_GENERAL_ERROR 보안토큰에 어떠한 복구 불가능한 에러가 발생하였음을 의미함. CKR_HOST_MEMORY, CKR_FUNCTION_FAILED와 함께 발생하는 경우, CKR_GENERAL_ERROR가 우선 반환됨	M
	CKR_HOST_MEMORY 보안토큰 구동프로그램을 구동하는 컴퓨터에 메모리가 부족할 경우 발생	M
	CKR_FUNCTION_FAILED 보안토큰 구동프로그램 특정 함수의 기능이 수행될 수 없을 경우 발생. 해당 함수가 세션을 사용하였다면 CK_SESSION_INFO 구조체의 ulDeviceError 항목에서 오류 발생 원인을 찾을 수 있음	M
세션	CKR_SESSION_HANDLE_INVALID 함수가 호출될 당시 해당 세션 핸들이 유효하지 않을 경우 발생하는 오류임. CKR_DEVICE_REMOVED, CKR_SESSION_CLOSED와 함께 발생하는 경우, CKR_SESSION_HANDLE_INVALID가 우선 반환됨	M
	CKR_DEVICE_REMOVED 보안토큰이 함수실행 도중 제거될 경우 발생	M
	CKR_SESSION_CLOSED 세션이 함수도중 종료된 경우 발생	M
보안토큰	CKR_DEVICE_MEMORY 보안토큰에 메모리가 부족할 경우 발생. CKR_DEVICE_ERROR, CKR_DEVICE_REMOVED와 함께 발생하는 경우, CKR_DEVICE_MEMORY가 우선 반환됨	M
	CKR_DEVICE_ERROR 보안토큰 또는 슬롯에 어떠한 문제가 발생할 경우 발생	M
	CKR_DEVICE_REMOVED 보안토큰이 함수실행 도중 제거될 경우 발생	M
	CKR_TOKEN_NOT_PRESENT 보안토큰이 함수 실행시점에 존재하지 않을 경우 발생	M
순차수행	CKR_CANCEL 보안토큰 구동프로그램이 가입자 S/W와 함께 기능을 상호 순차적으로 수행할 때, 가입자 S/W가 CKR_CANCEL를 반환하면 보안토큰 구동프로그램은 CKR_FUNCTION_CANCEL을 반환한다.	R
Mutex	CKR_MUTEX_BAD Mutex를 처리하는 함수가 잘못된 mutex 객체를 전달 받을 경우 발생	R

종류	반환값 및 내용	처리
	CKR_MUTEX_NOT_LOCKED	R
	Mutex 잠김을 해제하고자 하는 함수가 해당 mutex가 잠겨있지 않음을 발견할 경우 발생	
입출력	CKR_ARGUMENTS_BAD	R
	보안토큰 구동프로그램으로 전달되는 함수 인자가 잘못돼 있을 경우 발생	
	CKR_ATTRIBUTE_READ_ONLY	R
	가입자 S/W가 변경할 수 없는 속성의 값을 변경하고자 하는 경우 발생	
	CKR_ATTRIBUTE_TYPE_INVALID	R
	가입자 S/W가 템플릿에 유효하지 않는 속성타입을 명시하여 함수를 호출하는 발생	
	CKR_BUFFER_TOO_SMALL	R
	가입자 S/W의 버퍼 메모리가 작아서 보안토큰 구동프로그램 결과를 전달할 수 없는 경우 발생	
	CKR_CANT_LOCK	R
	C_Initialize 함수에만 해당하는 오류로써, 가입자 S/W가 안전한 기능 수행을 위해 thread-safety Locking을 요청하였는데, 보안토큰 구동프로그램이 해당 기능을 제공하지 않을 경우 발생	
	CKR_CRYPTOKI_ALREADY_INITIALIZED	R
	C_Initialize 함수에만 해당하는 오류로써, 보안토큰 구동프로그램이 이미 초기화되어 있음을 의미	
	CKR_CRYPTOKI_NOT_INITIALIZED	M
	C_Initialize 및 C_GetFunctionList 이외의 함수에 해당하는 오류로써, 보안토큰 구동프로그램이 초기화되지 않는 상황임을 의미	
	CKR_DATA_INVALID	R
	보안토큰 구동프로그램에 입력되는 입력값이 유효하지 않을 경우 발생. CKR_DATA_LEN_RANGE보다 낮은 우선순위를 가짐. 단, 본 반환값은 CKM_RSA_X_509 메커니즘에만 해당	
	CKR_DATA_LEN_RANGE	R
	보안토큰 구동프로그램에 입력되는 입력값의 길이가 유효하지 않을 경우 발생	
	CKR_DOMAIN_PARAMS_INVLALID	R
	유효하지 않거나 지원되지 않는 도메인 파라미터가 보안토큰 구동프로그램에 입력되는 경우 발생	
	CKR_ENCRYPTED_DATA_INVALID	R
	복호화 함수에 입력되는 암호문이 유효하지 않을 경우 발생. CKR_ENCRYPTED_DATA_LEN_RANGE보다 낮은 우선순위를 가짐	
	CKR_ENCRYPTED_DATA_LEN_RANGE	R
	복호화 함수에 입력되는 암호문의 길이가 유효하지 않을 경우 발생.	
	CKR_FUNCTION_CANCELED	R

종류	반환값 및 내용	처리
	보안토큰 구동프로그램이 가입자 S/W와 함께 기능을 상호 순차적으로 수행할 때, 가입자 S/W가 CKR_CANCEL를 반환하면 보안토큰 구동프로그램은 CKR_FUNCTION_CANCEL을 반환한다. 보호된 경로를 통한 PIN 입력 수행도중 발생할 수 있음	
	CKR_FUNCTION_NOT_PARALLEL	R
	지정된 세션에 parallel로 시행되는 함수가 없음을 알림, C_GetFunctionStatus 와 C_CancelFunction.에 의해 반환됨	
	CKR_FUNCTION_NOT_SUPPORTED	R
	해당 함수가 구동프로그램에서 지원되지 않을 경우 발생	
	CKR_INFORMATION_SENSITIVE	R
	정보가 민감하여 제공되어질 수 없음을 나타냄	
	CKR_KEY_CHANGED	R
	기존에 저장된 키가 변경되었음을 알림	
	CKR_KEY_FUNCTION_NOT_PERMITTED	R
	허가되지 않은 키의 속성을 사용하고자 했을 때 나타나는 메시지	
	CKR_KEY_HANDLE_INVALID	M
	키를 위한 핸들이 유효하지 않을 때 발생	
	CKR_KEY_INDIGESTIBLE	R
	C_DigestKey 함수에만 해당하는 메시지로써, 지정된 키의 값이 특정 이유로 인해 받아질 수 없을 때 발생	
	CKR_KEY_NEEDED	R
	C_SetOperationState 함수에만 해당하는 메시지로써, 하나 이상의 키가 있어야 하는데 없기 때문에 발생	
	CKR_KEY_NOT_NEEDED	R
	해당 사항이 없는 키가 추가적으로 제공 되었을 경우 발생	
	CKR_KEY_NOT_WRAPPABLE	R
	CKA_UNEXTRACTABLE 속성이 TRUE로 설정되어 있지 않더라도, 보안토큰 구동프로그램은 해당 키를 Wrap할 수 없음	
	CKR_KEY_SIZE_RANGE	R
	요청된 cryptographic 처리과정이 원칙상 문제가 없어도 키 사이즈가 범위 밖일 경우 발생	
	CKR_KEY_TYPE_INCONSISTENT	R
	지정된 키가 해당 메커니즘의 키가 아닐 경우 발생	
	CKR_KEY_UNEXTRACTABLE	R
	CKA_UNEXTRACTABLE 속성이 TRUE로 설정되어 있기 때문에, 해당 개인키 또는 비밀키는 Wrap 될 수 없음	
	CKR_MECHANISM_INVALID	R
	지정되지 않거나 지원되지 않는 메커니즘 사용 시 발생	
	CKR_MECHANISM_PARAM_INVALID	R

종류	반환값 및 내용	처리
	유효하지 않는 메커니즘 매개변수가 사용될 경우 발생	
	CKR_NEED_TO_CREATE_THREADS	R
	C_Initialize 가 새로운 thread를 만들 수 없는 상황, library가 제역할을 하지 못해 새로운 thread를 만들 수 없을 때 발생 (C_Initialize에만 해당하는 메시지)	
	CKR_NO_EVENT	R
	C_GetSlotEvent 함수에만 해당하는 메시지로써, non-blocking 모드에서 C_GetSlotEvent 호출되고 새로운 슬롯 이벤트가 없을 때 발생	
	CKR_OBJECT_HANDLE_INVALID	M
	지정된 object 핸들이 유효하지 않을 때 발생	
	CKR_OPERATION_ACTIVE	R
	이미 처리과정이 진행되고 있어 지정된 처리과정을 진행 시키지 못하게 될 때 발생	
	CKR_OPERATION_NOT_INITIALIZED	R
	해당 세션에 해당하는 적절한 타입의 액티브한 실행이 일어나지 않을 경우 발생	
	CKR_PIN_EXPIRED	R
	지정된 PIN이 만료되고 요청된 처리가 PIN 값이 바뀔 때까지 실행되지 않음	
	CKR_PIN_INCORRECT	M
	저장된 PIN과 일치 하지 않을 경우 발생	
	CKR_PIN_INVALID	R
	지정된 PIN에 유효하지 않는 문자가 존재할 경우 발생	
	CKR_PIN_LEN_RANGE	M
	지정된 PIN이 너무 길거나 너무 짧을 경우 발생	
	CKR_PIN_LOCKED	M
	지정된 PIN 이 lock 되어 있다. 보안토큰이 인증과정 중 더 이상의 과정을 허용치 않을 때 발생	
	CKR_RANDOM_NO_RNG	R
	지정된 토큰이 랜덤 변수 생성기가 없을 때 발생. CKR_RANDOM_SEED_NOT_SUPPORTED보다 높은 우선순위를 가짐	
	CKR_RANDOM_SEED_NOT_SUPPORTED	R
	보안토큰의 랜덤 변수 생성기가 응용프로그램으로부터 seeding을 받지 않고자 할때 발생. 본 메시지는 C_SeedRandom 함수에 의해서만 이용.	
	CKR_SAVED_STATE_INVALID	R
	제공된 저장된 암호화 처리 상태가 유효하지 않아, 지정된 세션에 저장될 수 없을 경우 발생	
	CKR_SESSION_COUNT	M

종류	반환값 및 내용	처리
	세션을 열려는 시도가 실패할 경우 발생. C_OpenSession 함수에서만 이용.	
	CKR_SESSION_EXISTS	R
	토큰이 있는 세션이 이미 열려있음	
	CKR_SESSION_PARALLEL_NOT_SUPPORTED	R
	parallel 세션을 지원하지 않음. C_OpenSession 함수에서만 이용.	
	CKR_SESSION_READ_ONLY	M
	읽기만 허용되는 세션임을 설명. CKR_TOKEN_WRITE_PROTECTED 보다 낮은 우선 순위를 가짐	
	CKR_SESSION_READ_ONLY_EXISTS	R
	읽기 세션이 이미 존재하여 SO가 logged in 되지 않음	
	CKR_SESSION_READ_WRITE_SO_EXISTS	R
	읽기/쓰기 세션이 이미 존재, 읽기 세션은 열리지 않음	
	CKR_SIGNATURE_LEN_RANGE	R
	전자서명 길이가 유효하지 않을 경우 발생. CKR_SIGNATURE_INVALID보다 우선순위를 가짐	
	CKR_SIGNATURE_INVALID	R
	전자서명값이 유효하지 않을 경우 발생	
	CKR_SLOT_ID_INVALID	M
	지정된 슬롯 아이디가 유효하지 않을 경우 발생	
	CKR_STATE_UNSAVEABLE	R
	단순히 토큰이 현 상태를 저장 하지 못할 경우 발생. CKR_OPERATION_NOT_INITIALIZED 보다 낮은 우선순위를 가짐	
	CKR_TEMPLATE_INCOMPLETE	M
	object를 만들기 위해 지정된 템플릿이 미완성되고 필요한 요소들이 부족한 상황일 경우 발생	
	CKR_TEMPLATE_INCONSISTENT	M
	object를 만들기 위해 지정된 템플릿에 충돌요소가 존재할 경우 발생	
	CKR_TOKEN_NOT_RECOGNIZED	M
	슬롯에 있는 토큰을 인식하지 못할 경우 발생	
	CKR_TOKEN_WRITE_PROTECTED	R
	보안토큰에 쓰기 보호가 되어있어 업무를 수행하지 못할 경우 발생. CKR_SESSION_READ_ONLY보다 높은 우선 순위를 가짐	
	CKR_UNWRAPPING_KEY_HANDLE_INVALID	M
	다른 키를 풀(Unwrap)려고 사용되는 키 핸들이 유효하지 않을 경우 발생	
	CKR_UNWRAPPING_KEY_SIZE_RANGE	R
	제공된 키의 키 사이즈가 허용범위를 넘어선 경우 발생	
	CKR_UNWRAPPING_KEY_TYPE_INCONSISTENT	R
	메커니즘이 맞지 않아 키를 풀 수 없을 경우 발생	

종류	반환값 및 내용	처리
	CKR_USER_ALREADY_LOGGED_IN 이미 로그인 한 상태를 알림	M
	CKR_USER_ANOTHER_ALREADY_LOGGED_IN 다른 사용자가 로그인 하여 로그인이 안됨	M
	CKR_USER_NOT_LOGGED_IN 로그인을 하지 않아 작업이 수행되지 않을 경우 발생	M
	CKR_USER_PIN_NOT_INITIALIZED 사용자의 핀이 초기화되지 않을 경우 발생	R
	CKR_USER_TOO_MANY_TYPES 보안토큰이 수용할 수 있는 이상의 동시 접속이 이루어질 때 발생	R
	CKR_USER_TYPE_INVALID CK_USER_TYPE에 유효하지 않는 값이 입력되었을 경우 발생	R
	CKR_WRAPPED_KEY_INVALID 제공된 wrapped key가 유효하지 않을 경우 발생	R
	CKR_WRAPPED_KEY_LEN_RANGE wrapped key의 길이가 유효하지 않을 경우 발생. CKR_WRAPPED_KEY_INVALID 보다 낮은 우선 순위를 가짐	R
	CKR_WRAPPING_KEY_HANDLE_INVALID 다른 키를 wrap 하기 위한 키 핸들이 유효하지 않을 경우 발생	R
	CKR_WRAPPING_KEY_SIZE_RANGE 키 길이로 인해 wrapping 에러난 경우 발생	R
	CKR_WRAPPING_KEY_TYPE_INCONSISTENT 지정되지 않은 메카니즘으로 wrap한 경우 발생	R
바이오 보안토큰 관리	CKR_BIO_FINGER_AUTH_NEEDED 지문인증없이 카드 사용	R
	CKR_BIO_SENSOR_ERROR 지문센서 H/W 에러	R
	CKR_BIO_NO_FINGER_INFO 지문템플릿 없음. 지문정보가 등록되어 있지 않아 오류발생	R
	CKR_BIO_SENSOR_TIMEOUT 지문센서 타임아웃	R
	CKR_BIO_FINGER_AUTH_FAILED 지문인증 실패	R
	CKR_BIO_NO_FINGER_AUTH_ID 지문인증 ID없음	R
	CKR_BIO_FINGER_ALGO_FAILED 지문 알고리즘 비정상 종료	R
	CKR_BIO_INVALID_USER	R

종류	반환값 및 내용	처리
	사용자 등록정보 불일치	

※ 바이오 보안토큰 API 반환값 헤더파일 :

```
#define CKR_BIO_FINGER_AUTH_NEEDED      0x89000001
#define CKR_BIO_SENSOR_ERROR            0x89000002
#define CKR_BIO_NO_FINGER_INFO          0x89000003
#define CKR_BIO_SENSOR_TIMEOUT          0x89000004
#define CKR_BIO_FINGER_AUTH_FAILED      0x89000005
#define CKR_BIO_NO_FINGER_AUTH_ID       0x89000006
#define CKR_BIO_FINGER_ALGO_FAILED      0x89000007
#define CKR_BIO_INVALID_USER            0x89000008
```

※ 바이오 보안토큰 관리와 관련된 반환값 처리 요구사항 'R'은 바이오 보안토큰 구동프로그램의 경우 'M'으로 처리한다.

부록 7. PKCS#11 사용 예

1. 개요

공인인증서 사용자가 공인인증서 저장매체로 보안토큰을 이용할 수 있도록 가입자 소프트웨어에서 PKCS#11 관련 기능을 구현할 때 참고할 수 있도록 구현 예를 제공한다.

2. PKCS#11 헤더 include

표준규격 API를 이용하여 구현하기 위해서는 cryptoki 헤더 파일을 include 시켜야 한다.

- PKCS#11 v2.20 이후 버전

```
#include “cryptoki.h”
```

- PKCS#11 v2.20 이전 버전

```
#include “pkcs11.h”
```

3. 토큰 세션 열기

```
// 출력값
CK_SESSION_HANDLE    hSession;     /* 토큰과의 세션 핸들 */
/ ------------------------------------------------------------------------
// 변수 선언
CK_RV nRv =    CKR_OK;              /* 에러 코드 확인 */
unsigned long    i = 0;
unsigned long    ulSlotCnt = 0;       /* 토큰이 꽂혀있는 슬롯의 개수 */
unsigned long    pSlotList[5];        /* 토큰이 꽂혀 있는 슬롯 목록 */
CK_TOKEN_INFO    tokenInfo;           /* 토큰 정보 */

// PKCS11 모듈 초기화
if ((nRv = C_Initialize(NULL_PTR)) != CKR_OK)
        goto FINISH;

// 토큰이 꽂혀있는 슬롯의 개수 확인
if ((nRv =    C_GetSlotList(TRUE, NULL_PTR, &ulSlotCnt)) != CKR_OK)
        goto FINISH;
if (ulSlotCnt    <= 0)
        goto FINISH;
```

```
// 토큰이 꽂혀있는 슬롯 목록 획득
if ((nRv =    C_GetSlotList(TRUE, pSlotList, &ulSlotCnt)) != CKR_OK)
        goto FINISH;

// 각 슬롯의 토큰 정보를 확인하여 사용할 토큰 선택
for (i=0; i<ulSlotCnt; i++)
{
        if ((nRv = C_GetTokenInfo(pSlotList[i],    &tokenInfo)) != CKR_OK)
        goto FINISH;

        // 획득한 토큰 정보 출력
}

// 사용하고자 하는 토큰 선택 (index)

// 세션 열기
nRv =    C_OpenSession(pSlotList[index], CKF_RW_SESSION | CKF_SERIAL_SESSION,
            &pApplication, NULL_PTR, &hSession);
        goto FINISH;

FINISH:
        C_Finalize(NULL_PTR);
        return nRv;
```

4. 토큰에 있는 인증서 목록 중 사용하고자 하는 인증서 선택

```
// 입력값
CK_SESSION_HANDLE    hSession;    /* 토큰과의 세션 핸들 (2. 단계에서 획득) */

// 출력값 (인증서,   keyID)
CK_BYTE    pOutCert[3072], pOutKeyId[32];
CK_ULONG    lOutCertLen = 0, lOutKeyIdLen = 0;
/------------------------------------------------------------------------
// 변수 선언
CK_RV nRv =    CKR_OK;              /* 에러 코드 확인 */
Unsigned long    i = 0;
CK_OBJECT_HANDLE  hCertAry[15]; /* 인증서 객체 핸들 목록 */
Unsigned long    ulCertCnt = 0;       /* 인증서 개수 */

/* 인증서 검색을 위한 템플릿 선언 */
CK_OBJECT_CLASS     certObject    = CKO_CERTIFICATE;
CK_CERTIFICATE_TYPE    x509Cert   = CKC_X_509;
CK_ATTRIBUTE     pX509CertTemplate[] = {
        {CKA_CLASS,             &certObject,    sizeof(certObject)},
                {CKA_CERTIFICATE_TYPE,    &x509Cert,   sizeof(x509Cert)}};

/* 인증서, keyID 획득을 위한 템플릿 선언 */
```

```
BYTE            pCertVal[2200] = {0x00, };
BYTE            pKeyId[20] = {0x00, };

CK_ATTRIBUTE    pValueTemplate[] = {
        {CKA_VALUE, pCertVal, 0},
        {CKA_ID, pKeyId, 0}};

// 인증서 객체를 찾기 위한 초기화
if ((nRv =    C_FindObjectsInit(hSession, pX509CertTemplate, 2)) != CKR_OK)
        goto FINISH;

if ((nRv =    C_FindObjects(hSession, hCertAry, 15, &ulCertCnt)) != CKR_OK)
        goto FINISH;

for (i=0;    i<ulCount; i++)
{
        // 템플릿 초기화
        pValueTemplate[0].ulValueLen =    sizeof(pCertVal);

        if ((nRv = C_GetAttributeValue(hSession,    hCertAry[i], pValueTemplate, 1)) !=
CKR_OK)
        goto FINISH;

        // 획득한 인증서 사용자 선택을 위해 보여줌
        // 인증서 : pValueTemplate[0].pValue,    pValueTemplate[0].ulValueLen
}

// 사용하고자 하는 인증서 선택 (index)

// 선택된 인증서와 keyId 리턴
// keyID는 인증서와 쌍인 개인키, 공개키, 랜덤값을 찾는데 사용됨.

pValueTemplate[0].ulValueLen    = sizeof(pCertVal);
pValueTemplate[1].ulValueLen    = sizeof(pKeyId);

if ((nRv = C_GetAttributeValue(hSession,    hCertAry[index],    pValueTemplate, 2)) !=
CKR_OK)
        goto FINISH;

memcpy(pOutCert,    pValueTemplate[0].pValue, pValueTemplate[0].ulValueLen);
lOutCertLen =    pValueTemplate[0].ulValueLen;

memcpy(pOutKeyId,    pValueTemplate[1].pValue, pValueTemplate[1].ulValueLen);
lOutKeyIdLen    = pValueTemplate[1].ulValueLen;

FINISH:

        C_FindObjectsFinal(hSession);
        return nRv;
```

5. 선택한 인증서를 이용한 서명

```
// 입력값
CK_SESSION_HANDLE   hSession;     /* 토큰과의 세션 핸들 (2. 단계에서 획득) */
unsigned char   keyid[20];          /* 인증서의 keyId 값 (3. 단계에서 획득) */
CK_MECHANISM_TYPE   hashAlg;     /* 서명 시 사용할 해쉬 알고리즘 */
unsigned char   *pData;             /* 서명 하고자 하는 원본 */
int   nDataLen;                     /* 서명 하고자 하는 원본 길이 */

// 출력 값 (서명 값)
CK_BYTE   out[1024];
CK_ULONG outLen   = 0;
/-------------------------------------------------------------------------
// 변수 선언
CK_RV nRv =   CKR_OK;               /* 에러   코드 확인 */
CK_OBJECT_HANDLE  hPriKeyAry[5];  /* 개인키   객체 핸들 목록 */
Unsigned long   ulPriKeyCnt = 0;      /* 개인키 개수 */

CK_KEY_TYPE   key_type = 0;          /* 개인키 종류 (알고리즘) */
CK_ATTRIBUTE   pKeyTypeTmpl[] = {{ CKA_KEY_TYPE, &key_type, sizeof(CK_KEY_TYPE) }};

/* keyid에 해당하는 개인키 핸들 획득 */
CK_OBJECT_CLASS   objPriKey = CKO_PRIVATE_KEY;
CK_ATTRIBUTE    pPriKeyTemplate[] = {
        {CKA_CLASS, &objPriKey,   sizeof(objPriKey)},
        {   CKA_ID,    &keyid,    sizeof(keyid)}};

// 개인키 객체를 찾기 위한 초기화
if ((nRv =   C_FindObjectsInit(hSession, pPriKeyTemplate, 2)) != CKR_OK)
        goto FINISH;

if ((nRv =   C_FindObjects(hSession, hPriKeyAry, 5, &ulPriKeyCnt)) != CKR_OK)
        goto FINISH;

// 해당 keyID와 일치하는 개인키 개수 확인
// ERR_NOT_EXIST,   ERR_KEY_ID_COINCIDE 에러 값 정의
if   (ulPriKeyCnt == 0) { nRv = ERR_NOT_EXIST; goto FINISH; }
else if   (ulPriiKeyCnt != 1) { nRv = ERR_KEY_ID_COINCIDE, goto FINISH; }

// 개인키 알고리즘 확인
if ((nRv =   C_GetAttributeValue(hSession, hPriKeyAry[0], pKeyTypeTmpl, 1)) != CKR_OK)
        goto FINISH;

if (key_type   == CKK_RSA)
{
        // RSA 서명인 경우 많은 보안 토큰들이 해쉬알고리즘을 이용한 서명이 아닌
        // RSA_PKCS 즉, RSA 개인키 암호화를 지원하므로 원본 메시지에   대한 DigestInfo
```

```
        // 구성 후 개인키 암호화 수행한다.
        CK_BYTE pDigestInfo[100];
        CK_ULONG lDigestInfoLen = 0;
        if ((nRv = MakeDigestInfo(data, pData, nDataLen, &pDigestInfo,
&lDigestInfoLen)) != CKR_OK)
        goto FINISH;

        CK_MECHANISM mechanism = { CKM_RSA_PKCS, NULL, 0 };

        if ((nRv = C_SignInit(hSession, &mechanism, hPriKeyAry[0])) != CKR_OK)
                goto FINISH;

        if ((nRv = C_Sign(hSession, pDigestInfo, lDigestInfoLen, out, &outLen)) !=
CKR_OK)
                goto FINISH;
}
else
{
        CK_MECHANISM signMech = { 0, 0, 0 };
        if (hashAlg == CKM_SHA_1)
                signMech.mechanism = CKM_KCDSA_SHA1;
        else if (nHash == ALGID_HS_SHA256)
                signMech.mechanism = CKM_KCDSA_SHA256;
        else
        {
                // ERR_INVALID_HASH_ALG 값 정의
                nRv = ERR_INVALID_HASH_ALG;
                goto FINISH;
        }

        if ((nRv = C_SignInit(m_hSession, &signMech, hPriKeyAry[0])) != CKR_OK)
                goto FINISH;

        if ((nRv = C_Sign(hSession, pData, nDataLen, pSign, &lSignLen))!= CKR_OK)
                goto FINISH;
}
FINISH:

        C_FindObjectsFinal(hSession);
        return nRv;
```

```
int MakeDigestInfo(CK_MECHANISM_TYPE hashAlg, unsigned char *pData, int nDataLen,
unsigned char **pDigestInfo, int *pDigestInfoLen)
{
        CK_RV nRv = CKR_OK;             /* 에러 코드 확인 */
        CK_BYTE pDigest[32];
        CK_ULONG lDigestLen = 32;
```

```
        CK_MECHANISM hashMech = { hashAlg,    0, 0 };
        C_DigestInit(m_hSession, &hashMech);
        if ((nRv = C_Digest(m_hSession, pData,    nDataLen, pDigest, &lDigestLen))    !=
CKR_OK)
        return nRv;

        // 지원 되는 각 해쉬알고리즘의 DigestInfo 구성을 위한 헤더 정보
        static const BYTE SHA1_PADDING[] =   {0x30, 0x21, 0x30, 0x09, 0x06, 0x05, 0x2b,
        0x0e, 0x03, 0x02, 0x1a,    0x05, 0x00, 0x04, 0x14 };
        static const BYTE SHA256_PADDING[]    = {0x30, 0x31, 0x30, 0x0d, 0x06, 0x09,
0x60,
        0x86, 0x48, 0x01, 0x65,    0x03, 0x04, 0x02, 0x01, 0x05, 0x00, 0x04, 0x20 };

        if (hashAlg == CKM_SHA_1)
        {
                memcpy(pDigestInfo, 0, SHA1_PADDING,    sizeof(SHA1_PADDING));
                memcpy(pDigestInfo,    sizeof(SHA1_PADDING), pDigest, lDigestLen);
                *pDigestInfoLen =    sizeof(SHA1_PADDING) + lDigestLen;
        }
        else if (nHash == ALGID_HS_SHA256)
        {
                memcpy(pDigestInfo, SHA256_PADDING,    sizeof(SHA256_PADDING));
                memcpy(pDigestInfo,    sizeof(SHA256_PADDING), pDigest, lDigestLen);
                *pDigestInfoLen = sizeof(SHA256_PADDING)    + lDigestLen;
        }
        else
        {
                // ERR_INVALID_HASH_ALG 값 정의
                return ERR_INVALID_HASH_ALG;
        }
        return 0;
}
```

6. 본인확인을 위한 랜덤값 획득

```
// 입력값
CK_SESSION_HANDLE    hSession;      /* 토큰과의 세션 핸들 (2. 단계에서 획득) */
unsigned char    keyid[20];            /* 인증서의 keyId 값 (3. 단계에서 획득) */

// 출력값 (랜덤값)
unsigned char    out[32];
int outLen =    0;
/-------------------------------------------------------------------------------
// 변수 선언
CK_OBJECT_CLASS    objData       = CKO_DATA;
CK_ATTRIBUTE       dataTemplate  = { CKA_CLASS, & objData,    sizeof(CK_OBJECT_CLASS)};
CK_OBJECT_HANDLE     hAry[10];
CK_ULONG               ulCount = 0;
```

```
CK_ULONG          i =   0;
CK_BYTE           pLabel[512] = {0x00, };
CK_BYTE           pRandom[64];

// 찾고자 하는 random 값의 라벨을 만든다.
sprintf(pLabel,   "R4VID=%s", toHexString(keyid));

// data 객체의 개수 확인
if ((nRv =   C_FindObjectsInit(hSession, &dataTemplate, 1)) != CKR_OK)
        goto FINISH;

if ((nRv =   C_FindObjects(hSession, hAry, 10, &ulCount)) != CKR_OK)
        goto FINISH;

if (ulCount   == 0) { nRv = DATA_OBJ_NOT_EXIST; goto FINISH; }

// data 객체 중 라벨이 동일한 객체를 찾는다.
for (i = 0;   i<ulCount; i++)
{
        CK_BYTE ckaLabel[32];
        CK_ATTRIBUTE randomTemplate =
        {{CKA_LABEL, (CK_VOID_PTR)   ckaLabel, sizeof(ckaLabel)},
        {CKA_VALUE, pRandom,   sizeof(pRandom)}};

        if ((nRv =   C_GetAttributeValue(hSession, hAry[i], &randomTemplate, 2)) !=
CKR_OK)
        goto FINISH;

        // 라벨 값 비교 (대소문자 구분하지 않음)
        if ((randomTemplate[0].ulValueLen ==   strlen(pLabel)) &&
                (_memicmp(randomTemplate[0].pValue,   (char*)pLabel,
                randomTemplate[0].ulValueLen) == 0))
        {
                // 동일한 라벨에 해당하는 객체의 값 즉, 랜덤값을 리턴한다.
                memcpy(out, randomTemplate[1].pValue,   randomTemplate[1].ulValueLen);
                outLen = randomTemplate[1].ulValueLen;
                return 0;
        }
}
FINISH:
        C_FindObjectsFinal(hSession);
        return nRv;
```

부록 8. PKCS#11 자바 인터페이스 사용 예

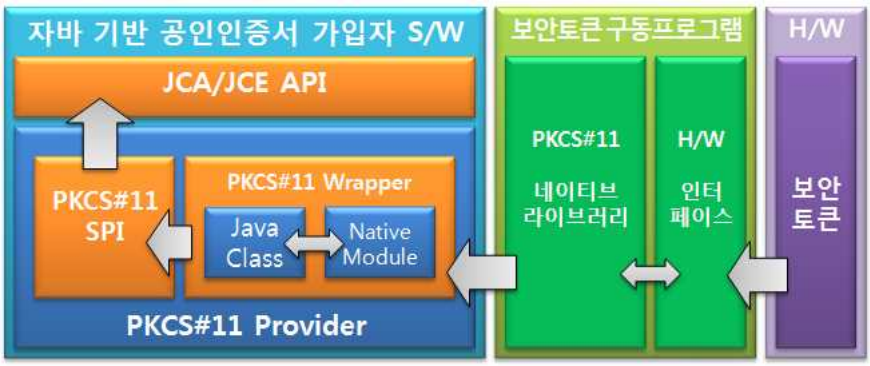
1. 개요

1.1. 인터페이스 모델

자바를 이용하여 가입자 소프트웨어를 구현하는 경우, 보안토큰 이용을 위해 자바에서 지원하는 Sun PKCS#11 Provider와 IBM, IAIK 등 써드파티의 솔루션을 이용할 수 있다. 본 규격에서는 성능 및 플랫폼 상의 제한을 받지 않는 한 Sun PKCS#11 Provider의 이용을 권장한다.

C/C++ 등으로 구현된 PKCS#11 네이티브 라이브러리를 이용하기 위해 아래와 같이 자바 Wrapper 클래스와 Wrapper 네이티브 모듈로 구성된 PKCS#11 Wrapper와 PKCS#11 SPI를 갖춘 JCA/JCE 프로바이더를 이용할 수 있다.

또한, 직접 자바를 이용하여 PKCS#11 라이브러리를 구현하여 이용되는 경우에도 PKCS#11 Wapper 등을 구성하여 JCA/JCE 프로바이더와 동일한 인터페이스를 유지할 수 있다.



1.2. Sun PKCS#11 Provider

자바 기반의 가입자 소프트웨어 구현 시 보안토큰과 관련한 부분은 “Sun PKCS#11 Provider”를 참고한다. 자바 5.0부터 자바플랫폼에서 네이티브 PKCS#11 토큰 이용을 보장하기 위해 Sun PKCS#11 Provider가 추가되었으며, 이 프로바이더는 자바 기반으로 구현된 가입자 소프트웨어가 네이티브 PKCS#11 토큰에 접근할 수 있도록 한다.

2. Sun PKCS#11 Provider 사용 예

2.1. 프로바이더 설정

자바 PKCS#11 프로바이더(SunPKCS11)를 사용하기 위해서는 설정파일을 이용하여 등록하여 사용하거나 프로그램에서 동적으로 등록하여 사용 가능하다.

```
String pkcs11Config =
   "name = SmartCard" +
   "library = C:\Windows\system32\pkcs11.dll";
byte[] pkcs11configByte = pkcs11Config.getByte();
ByteArrayInputStream configStream
   = new ByteArrayInputStream(pkcs11Config.getByte);
Provider pkcs11Provider = new sun.security.pkcs11.SunPKCS11(configStream);
Security.addProvider(pkcs11Provider);
```

2.2. 보안토큰 로그인

보안토큰을 사용하기 위해서는 자바의 키저장 객체(KeyStore)를 프로바이더와 보안토큰 비밀번호(PIN)를 이용하여 로딩 함으로서 보안토큰에 로그인 가능하다.

```
char[] pin = ...;
KeyStore ks = KeyStore.getInstance("PKCS11");
ks.load(null, pin);
```

2.3. 키쌍 생성

키쌍을 생성하기 위해서는 KeyPairGenerator 클래스를 프로바이더와 알고리즘을 이용하여 객체를 생성한 후 genKeyPair 메소드를 호출하여 키쌍을 생성한다.

```
KeyPairGenerator kpg = KeyPairGenerator.getInstance("RSA", pkcs11Provider);
KeyPair kp = kpg.genKeyPair();
```

2.4. 전자서명생성키 불러오기

보안토큰에 저장되어 있는 전자서명생성키를 얻기 위해서는 키저장 객체의 getKey() 메소드를 이용한다. 키저장 객체(Key store)에는 실제 전자서명생성키가 저장되는 것이 아니라 해당 키에 대한 참조 값이 저장된다.

```
PrivateKey mykey = (PrivateKey)keystore.getKey(키저장 객체 명칭, 비밀번호);
```

2.5. 인증서 저장 및 가져오기

getCertificate 및 setCertificate 메소드를 호출하여 인증서 또는 인증서 체인을 저장하거나 가져오기 가능하다.

```
인증서 불러오기
Certificate myCert = keystore.getCertificate(키저장 객체 명칭);

인증서 저장
keystore.setCertificateEntry(키저장 객체 명칭, 인증서 객체);
```

2.6. 전자서명 생성

Signature 클래스에 프로바이더와 알고리즘을 지정하여 인스턴스 객체를 생성한 후 sign() 메소드를 이용하여 문자열에 대한 전자서명을 생성한다.

```
Signature sig = Signature.getInstance("SHA256withRSA",pkcs11Provider);
sig.initSign(전자서명생성키 객체);
sig.update(전자서명 하고자 하는 문자열);
byte[] sigBytes = sig.sign();
```

부록 9. 무선통신 지원 보안토큰 표준 API

1. 무선통신 지원 보안토큰 표준 API 이용 공통 규격

■ NFC 보안토큰 사용을 위한 NFC Adapter 제어는 서비스 앱에서 처리한다.

■ 표준 API 라이브러리의 클래스명은 'TokenControl'로 정의하고,
이를 이용하기 위해 다음 함수를 필수로 지원한다.

함수	기능	비고
TokenControl(IsoDep isoDep)	NFC 타입 보안토큰 객체 생성자	• isoDep : NFC 보안토큰 접근 객체
TokenControl(Context appContext, String deviceType)	보안토큰 객체 생성자	• appContext : 서비스 앱 Context 객체 • deviceType : 보안토큰 종류에 따라 입력 ex) microSD : "MSD" 블루투스 : "BLE"
boolean Connect()	보안토큰에 연결	• True : 연결 성공 • False : 연결 실패
void Transmit(Bundle requestBundle, Bundle responseBundle)	보안토큰 명령 수행	• requestBundle : 명령요청 Bundle • responseBundle : 처리결과 Bundle
boolean Disconnect()	보안토큰 연결해제	• True : 연결해제 성공 • False : 연결해제 실패

■ 보안토큰 종류 별 TokenControl 객체 생성의 예

보안토큰 종류	TokenControl 객체 생성의 예

NFC	Tag mTag =
	
	intent.getParcelableExtra(NfcAdapter.EXTRA_TAG);
	if (mTag != null) {
	IsoDep isoDep = IsoDep.get(mTag);
	TokenControl mToken= new
	TokenControl(isoDep);
	}

microSD	TokenControl mToken = new TokenControl (getApplicationContext(), “MSD”);
BLE	TokenControl mToken = new TokenControl (getApplicationContext(), “BLE”);

2. 무선통신 지원 보안토큰 표준 API 서비스 제공 기능 목록

함수	요청 명령 (REQUEST_ACTION)
보안토큰 정보 조회	GET_TOKEN_INFO
애플릿 제어정보 조회	GET_APPLET_CONTROL_INFO
발행기관 키 및 PIN 상태 조회	GET_KEY_PIN_STATUS
PIN 인증	VERIFY_PIN
PIN 변경	PUT_PIN
난수 생성	GET_CHALLENGE
발행기관과 보안토큰 상호인증 요청	MUTUAL_AUTHENTICATE
인증서 목록 조회	GET_CERTIFICATE_LIST
인증서 조회	GET_CERTIFICATE
인증서 R 값 조회	GET_CERTIFICATE_R
키 길이 조회	GET_KEY_LENGTH
전자서명 생성	GENERATE_SIGNATURE
복호화	DECRYPT
공개키/개인키 키 쌍 생성	GENERATE_KEY_PAIR
개인키 주입	STORE_PRIVATE_KEY
공개키 주입	STORE_PUBLIC_KEY
인증서 저장	PUSH_CERTIFICATE
인증서 R 값 저장	PUT_CERTIFICATE_R
인증서 삭제	DELETE_CERTIFICATE

3. 무선통신 지원 보안토큰 표준 API 기능별 세부 이용 규격

3.1 보안토큰 정보 조회

보안토큰 발급 시 설정된 보안토큰의 기본 정보를 조회한다.

■ 요청 Bundle 구성

파라메타		데이터 형식	필수 여부
키	값		
REQUEST_ACTION	“GET_TOKEN_INFO”	String	필수

■ 응답 Bundle 구성

파라메타		데이터 형식	필수 여부
키	값		
REQUEST_ACTION	“GET_TOKEN_INFO”	String	필수
RESULT_CODE	응답코드 표 참고	String	필수
LABEL	보안토큰 라벨	ByteArray	
MANUFACTURER_ID	제조사 ID	ByteArray	
MODEL	보안토큰 모델명	ByteArray	
CSN	보안토큰 고유일련번호	ByteArray	
MAX_PIN_LENGTH	PIN 최대 길이	Integer	
MIN_PIN_LENGTH	PIN 최소 길이	Integer	
HW_VER	보안토큰 하드웨어 버전	ByteArray	
FW_VER	보안토큰 펌웨어 버전	ByteArray	
ERROR_MESSAGE	에러 메시지	String	

3.2 애플릿 제어정보 조회

표준화 규격 버전, 인증서 최대 저장 개수 등의 보안토큰 애플릿의 정보를 조회한다.

■ 요청 Bundle 구성

파라메타		데이터 형식	필수 여부
키	값		
REQUEST_ACTION	"GET_APPLET_CONTROL_INFO"	String	필수

■ 응답 Bundle 구성

파라메타		데이터 형식	필수 여부
키	값		
REQUEST_ACTION	"GET_APPLET_CONTROL_INFO"	String	필수
RESULT_CODE	응답코드 표 참고	String	필수
ISSUER_CODE	발행기관 식별자	ByteArray	
VERSION	표준화 규격 버전	Byte	
CERT_MAX_COUNT	인증서 최대 저장 개수	Integer	
SUPPORT_FUNCTION	지원기능 코드 참고	Byte	
USER_AUTH_MODE	사용자 인증 방법 코드 참고	Byte	
CRYPTO_ALG	기본 암호화 알고리즘 코드 참고	Byte	
ERROR_MESSAGE	에러 메시지	String	

※ 지원기능 코드

코드	의미
0x41	RSA Private Key 1024bit 지원
0x42	RSA Private Key 2048bit 지원
0x81	RSA CRT Private Key 1024bit 지원
0x82	RSA CRT Private Key 2048bit 지원
0xC1	RSA Private Key, RSA CRT Private Key 1024bit 지원
0xC2	RSA Private Key, RSA CRT Private Key 2048bit 지원

* EC-DSA, EC-KCDSA 등 전자서명 알고리즘을 지원할 수 있다.

※ 사용자 인증 방법 코드

코드	의미
0x00	보안토큰 단말기로부터 사용자 인증
0x01	프로그램에서 입력 받은 PIN값을 이용한 사용자 인증

※ 기본 암호화 알고리즘 코드

코드	의미
0x10	3DES 사용
0x20	SEED 사용
0x40	AES-128 사용

3.3 발행기관 키 및 PIN 상태 조회

보안토큰의 발행기관 키 설정 여부, 초기 PIN 변경 여부, PIN 인증 여부 등을 조회한다.

■ 요청 Bundle 구성

파라메타		데이터 형식	필수 여부
키	값		
REQUEST_ACTION	“GET_KEY_PIN_STATUS”	String	필수

■ 응답 Bundle 구성

파라메타		데이터 형식	필수 여부
키	값		
REQUEST_ACTION	“GET_KEY_PIN_STATUS”	String	필수
RESULT_CODE	응답코드 표 참고	String	필수
ISSUER_KEY_STATUS	발행기관 키 설정 상태 0 = 미설정 1 = 설정	Integer	
PIN_INIT_STATUS	PIN 설정 상태 0 = 초기 PIN 상태 1 = 사용자에 의한 PIN 변경 상태	Integer	
PIN_VERIFY_STATUS	PIN 인증 상태 0 = 미 인증 상태 1 = 인증 완료 상태	Integer	
ERROR_MESSAGE	에러 메시지	String	

3.4 PIN 인증

보안토큰에 사용자 PIN 인증을 요청한다.

■ 요청 Bundle 구성

파라메타		데이터 형식	필수여부
키	값		
REQUEST_ACTION	“VERIFY_PIN”	String	필수
PIN	사용자가 입력한 PIN 값	ByteArray	필수

■ 응답 Bundle 구성

파라메타		데이터 형식	필수여부
키	값		
REQUEST_ACTION	“VERIFY_PIN”	String	필수
RESULT_CODE	응답코드 표 참고	String	필수
RETRY_COUNT	인증 실패 시 남은 재시도 횟수	Integer	
ERROR_MESSAGE	에러 메시지	String	

3.5 PIN 변경

보안토큰에 설정된 기존 사용자 PIN 을 새로운 PIN 으로 변경한다. PIN 검증이 완료된 상태에서 사용 가능하다.

■ 요청 Bundle 구성

파라메타		데이터 형식	필수 여부
키	값		
REQUEST_ACTION	“PUT_PIN”	String	필수
PIN	사용자가 입력한 새로운 PIN 값	ByteArray	필수

■ 응답 Bundle 구성

파라메타		데이터 형식	필수 여부
키	값		
REQUEST_ACTION	“PUT_PIN”	String	필수
RESULT_CODE	응답코드 표 참고	String	필수
ERROR_MESSAGE	에러 메시지	String	

3.6 난수 생성

보안토큰에 난수 생성을 요청한다. Mutual Authenticate 를 위한 난수 생성 시 난수 사용 용도(RANDOM_PURPOSE)는 “0x01”, 난수 값의 길이(RANDOM_LENGTH)는 16으로 설정하여 사용한다.

■ 요청 Bundle 구성

파라메타		데이터 형식	필수 여부
키	값		
REQUEST_ACTION	“GET_CHALLENGE”	String	필수
RANDOM_PURPOSE	난수 사용 용도 0x00 = 외부사용 목적 0x01 = 토큰내부의 Security 관련 Procedure 에서 사용 목적	Byte	필수
RANDOM_LENGTH	생성할 난수 값의 길이	Integer	필수

■ 응답 Bundle 구성

파라메타		데이터 형식	필수 여부
키	값		
REQUEST_ACTION	“GET_CHALLENGE”	String	필수
RESULT_CODE	응답코드 표 참고	String	필수
RANDOM	요청한 길이만큼 보안토큰에서 생성된 난수 값	ByteArray	
ERROR_MESSAGE	에러 메시지	String	

3.7 발행기관과 보안토큰 상호인증 요청

보안토큰에 발행기관 키가 설정되어 있을 경우 보안토큰은 발행기관 키를 이용하여 발행기관을 검증한 후 발행기관이 보안토큰을 검증할 수 있는 인증 데이터를 생성한다.

■ 요청 Bundle 구성

파라메타		데이터 형식	필수 여부
키	값		
REQUEST_ACTION	“MUTUAL_AUTHENTICATE”	String	필수
ISSUER_ENC_DATA	발행기관에 의해 암호화된 데이타	ByteArray	필수

■ 응답 Bundle 구성

파라메타		데이터 형식	필수 여부
키	값		
REQUEST_ACTION	“MUTUAL_AUTHENTICATE”	String	필수
RESULT_CODE	응답코드 표 참고	String	필수
TOKEN_ENC_DATA	보안토큰에 의해 암호화된 데이터	ByteArray	
ERROR_MESSAGE	에러 메시지	String	

3.8 인증서 목록 조회

보안토큰에 저장된 인증서의 개수, 각 인증서의 종류 및 인증서 크기 정보를 조회한다.

■ 요청 Bundle 구성

파라메타		데이터 형식	필수 여부
키	값		
REQUEST_ACTION	“GET_CERTIFICATE_LIST”	String	필수

■ 응답 Bundle 구성

파라메타		데이터 형식	필수 여부
키	값		
REQUEST_ACTION	“GET_CERTIFICATE_LIST”	String	필수
RESULT_CODE	응답코드 표 참고	String	필수
CERT_COUNT	저장된 인증서의 개수	Integer	
CERT1_TYPE	1 번 인증서의 유무 및 종류	Byte	
CERT2_TYPE	2 번 인증서의 유무 및 종류	Byte	
CERT3_TYPE	3 번 인증서의 유무 및 종류	Byte	
CERT4_TYPE	4 번 인증서의 유무 및 종류	Byte	
CERT5_TYPE	5 번 인증서의 유무 및 종류	Byte	
CERT6_TYPE	6 번 인증서의 유무 및 종류	Byte	
CERT7_TYPE	7 번 인증서의 유무 및 종류	Byte	
CERT1_LENGTH	1 번 인증서의 크기	Integer	
CERT2_LENGTH	2 번 인증서의 크기	Integer	
CERT3_LENGTH	3 번 인증서의 크기	Integer	
CERT4_LENGTH	4 번 인증서의 크기	Integer	
CERT5_LENGTH	5 번 인증서의 크기	Integer	

CERT6_LENGTH	6 번 인증서의 크기	Integer	
CERT7_LENGTH	7 번 인증서의 크기	Integer	
ERROR_MESSAGE	에러 메시지	Integer	

※ 인증서의 유무 및 종류 코드

코드	의미
0x00	인증서 없음
0x01	서명용 범용
0x02	서명용 용도 제한용(은행, 보험, 신용카드)
0x03	서명용 용도 제한용(조달청)
0x04	서명용 용도 제한용(증권, 보험)
0x05	키 분배용 범용
0xFF	종류는 표기하지 않으나 인증서는 존재

3.9 인증서 조회

지정된 위치의 인증서 데이터를 조회한다.

■ 요청 Bundle 구성

파라메타		데이터 형식	필수 여부
키	값		
REQUEST_ACTION	“GET_CERTIFICATE”	String	필수
CERTIFICATE_INDEX	인증서 저장위치번호 (1 ~ 7 까지의 번호 사용 가능)	Integer	필수

■ 응답 Bundle 구성

파라메타		데이터 형식	필수 여부
키	값		
REQUEST_ACTION	“GET_CERTIFICATE”	String	필수
RESULT_CODE	응답코드 표 참고	String	필수
CERTIFICATE	인증서 데이터	ByteArray	
ERROR_MESSAGE	에러 메시지	String	

3.10 인증서 R값 조회

지정된 인증서의 R 값을 조회한다.

■ 요청 Bundle 구성

파라메타		데이터 형식	필수 여부
키	값		
REQUEST_ACTION	“GET_CERTIFICATE_R”	String	필수
CERTIFICATE_INDEX	인증서 저장위치번호 (1 ~ 7 까지의 번호 사용 가능)	Integer	필수

■ 응답 Bundle 구성

파라메타		데이터 형식	필수 여부
키	값		
REQUEST_ACTION	“GET_CERTIFICATE_R”	String	필수
RESULT_CODE	응답코드 표 참고	String	필수
CERTIFICATE_R	인증서 R 값	ByteArray	
ERROR_MESSAGE	에러 메시지	String	

3.11 키 길이 조회

지정된 인증서의 키 길이 값을 조회한다.

■ 요청 Bundle 구성

파라메타		데이터 형식	필수 여부
키	값		
REQUEST_ACTION	“GET_KEY_LENGTH”	String	필수
CERTIFICATE_INDEX	인증서 저장위치번호 (1 ~ 7 까지의 번호 사용 가능)	Integer	필수

■ 응답 Bundle 구성

파라메타		데이터 형식	필수 여부
키	값		
REQUEST_ACTION	“GET_KEY_LENGTH”	String	필수
RESULT_CODE	응답코드 표 참고	String	필수
KEY_LENGTH	키 길이 0x20 = 1024bits Key 0x40 = 2048bits Key	Byte	
ERROR_MESSAGE	에러 메시지	String	

3.12 전자서명 생성

보안토큰에 입력된 데이터에 대한 전자서명 값을 요청한다.

■ 요청 Bundle 구성

파라메타		데이터 형식	필수 여부
키	값		
REQUEST_ACTION	"GENERATE_SIGNATURE"	String	필수
CERTIFICATE_INDEX	인증서 저장위치번호 (1 ~ 7 까지의 번호 사용 가능)	Integer	필수
TO_BE_SIGNED_DATA	서명할 데이터	ByteArray	필수
KEY_LENGTH	키 길이 0x20 = 1024bit(128 byte) 0x40 = 2048bit(256 byte)	Byte	필수
MECHANISM	매커니즘 0x01 = CKM_RSA_PKCS 0x0D = CKM_RSA_PKCS_PSS (현재 지원 안함)	Byte	필수

■ 응답 Bundle 구성

파라메타		데이터 형식	필수 여부
키	값		
REQUEST_ACTION	"GENERATE_SIGNATURE"	String	필수
RESULT_CODE	응답코드 표 참고	String	필수
SIGNATURE	전자서명 값	ByteArray	
ERROR_MESSAGE	에러 메시지	String	

3.13 복호화

보안토큰에 입력된 데이터에 대한 복호화 처리를 요청한다.

■ 요청 Bundle 구성

파라메타		데이터 형식	필수 여부
키	값		
REQUEST_ACTION	“DECRYPT”	String	필수
CERTIFICATE_INDEX	인증서 저장위치번호 (1 ~ 7 까지의 번호 사용 가능)	Integer	필수
ENC_DATA	암호화된 데이터	ByteArray	필수
MECHANISM	매커니즘 0x01 = CKM_RSA_PKCS 0x09 = CKM_RSA_PKCS_OAEP (현재 지원 안함)	Byte	필수

■ 응답 Bundle 구성

파라메타		데이터 형식	필수 여부
키	값		
REQUEST_ACTION	“DECRYPT”	String	필수
RESULT_CODE	응답코드 표 참고	String	필수
DEC_DATA	복호화된 데이터	ByteArray	
ERROR_MESSAGE	에러 메시지	String	

3.14 공개키/개인키 키 쌍 생성

보안토큰 내부에 공개키와 개인키 키 쌍의 생성을 요청한다. 키 쌍 생성 후 공개키와 생성된 키의 위치번호를 반환한다.

■ 요청 Bundle 구성

파라메타		데이터 형식	필수 여부
키	값		
REQUEST_ACTION	“GENERATE_KEY_PAIR”	String	필수
KEY_TYPE_LENGTH	키 종류 및 키 길이 코드 참조	Byte	필수
PURPOSE	키 생성 용도 0x01 = 인증서 발급 목적 0x02 = 인증서 갱신 목적	Byte	필수

■ 응답 Bundle 구성

파라메타		데이터 형식	필수 여부
키	값		
REQUEST_ACTION	“GENERATE_KEY_PAIR”	String	필수
RESULT_CODE	응답코드 표 참고	String	필수
MODULUS	Modulus	ByteArray	
EXPONENT	Public exponent	ByteArray	
KEY_INDEX	생성된 키의 저장위치번호 (1 ~ 7 까지의 번호 사용)	Integer	
ERROR_MESSAGE	에러 메시지	String	

※ 키 종류 및 키 길이 코드

코드	의미
0x01	RSA Private Key 1024bit
0x02	RSA Private Key 2048bit
0x11	RSA CRT Private Key 1024bit
0x12	RSA CRT Private Key 2048bit

6.15 개인키 주입

외부에서 생성된 개인키를 보안토큰 내부에 저장한다. 개인키 저장 후 키가 저장된 위치번호를 반환한다.

■ 요청 Bundle 구성

파라메타		데이터 형식	필수여부
키	값		
REQUEST_ACTION	“STORE_PRIVATE_KEY”	String	필수
KEY_TYPE_LENGTH	키 종류 및 키 길이 코드 참조	Byte	필수
MODULUS	Modulus 값	ByteArray	주1
PRIVATE_EXPONENT	Private exponent 값	ByteArray	
CRT_PRIME1	RSA CRT Private Key의 P값	ByteArray	주2
CRT_PRIME2	RSA CRT Private Key의 Q값	ByteArray	
CRT_EXPONENT1	RSA CRT Private Key의 DP값	ByteArray	
CRT_EXPONENT2	RSA CRT Private Key의 DQ값	ByteArray	
CRT_COEFFICIENT	RSA CRT Private Key의 QP값	ByteArray	

※ 주1) RSA Private Key의 경우 필수 입력, RSA CRT Private Key의 경우 입력 안 함
※ 주2) RSA CRT Private Key의 경우 필수 입력

■ 응답 Bundle 구성

파라메타		데이터 형식	필수 여부
키	값		
REQUEST_ACTION	“STORE_PRIVATE_KEY”	String	필수
RESULT_CODE	응답코드 표 참고	String	필수
KEY_INDEX	생성된 키의 저장위치번호 (1 ~ 7 까지의 번호 사용)	Integer	
ERROR_MESSAGE	에러 메시지	String	

※ 키 종류 및 키 길이 코드

코드	의미
0x01	RSA Private Key 1024bit
0x02	RSA Private Key 2048bit
0x11	RSA CRT Private Key 1024bit
0x12	RSA CRT Private Key 2048bit

6.16 공개키 주입

외부에서 생성된 공개키를 보안토큰 내부에 저장한다.

■ 요청 Bundle 구성

파라메타		데이터 형식	필수 여부
키	값		
REQUEST_ACTION	“STORE_PUBLIC_KEY”	String	필수
KEY_INDEX	키의 저장위치번호 (1 ~ 7 까지의 번호 사용 가능)	Integer	필수
KEY_LENGTH	키 길이 0x20 = 1024bit(128 byte) 0x40 = 2048bit(256 byte)	Byte	필수
MODULUS	Modulus 값	ByteArray	필수
PUBLIC_EXPONENT	Public exponent 값	ByteArray	필수

■ 응답 Bundle 구성

파라메타		데이터 형식	필수 여부
키	값		
REQUEST_ACTION	“STORE_PUBLIC_KEY”	String	필수
RESULT_CODE	응답코드 표 참고	String	필수
ERROR_MESSAGE	에러 메시지	String	

6.17 인증서 저장

보안토큰에 인증서를 저장한다.

■ 요청 Bundle 구성

파라메타		데이터 형식	필수 여부
키	값		
REQUEST_ACTION	“PUSH_CERTIFICATE”	String	필수
CERTIFICATE_INDEX	인증서 저장위치번호 (1 ~ 7 까지의 번호 사용 가능)	Integer	필수
CERTIFICATE_TYPE	인증서 종류 코드 참조	Byte	필수
CERTIFICATE_R	인증서 R 값	ByteArray	
CERTIFICATE_LENGTH	인증서 길이	Integer	필수
CERTIFICATE	저장할 인증서	ByteArray	필수

※ 인증서 종류 테이블

코드	의미
0x01	서명용 범용
0x02	서명용 용도 제한용(은행, 보험, 신용카드)
0x03	서명용 용도 제한용(조달청)
0x04	서명용 용도 제한용(증권, 보험)
0x05	키 분배용 범용
0xFF	종류 표기하지 않음

■ 응답 Bundle 구성

파라메타		데이터 형식	필수 여부
키	값		
REQUEST_ACTION	“PUSH_CERTIFICATE”	String	필수
RESULT_CODE	응답코드 표 참고	String	필수
ERROR_MESSAGE	에러 메시지	String	

6.18 인증서 R값 저장

지정된 위치에 인증서의 R 값을 저장한다.

■ 요청 Bundle 구성

파라메타		데이터 형식	필수 여부
키	값		
REQUEST_ACTION	“PUT_CERTIFICATE_R”	String	필수
CERTIFICATE_INDEX	인증서 저장위치번호 (1 ~ 7 까지의 번호 사용 가능)	Integer	필수
CERTIFICATE_R	인증서 R 값	ByteArray	필수

■ 응답 Bundle 구성

파라메타		데이터 형식	필수 여부
키	값		
REQUEST_ACTION	“PUT_CERTIFICATE_R”	String	필수
RESULT_CODE	응답코드 표 참고	String	필수
ERROR_MESSAGE	에러 메시지	String	

6.19 인증서 삭제

지정된 위치의 인증서와 해당 인증서와 관련된 모든 정보를 삭제한다.

■ 요청 Bundle 구성

파라메타		데이터 형식	필수 여부
키	값		
REQUEST_ACTION	“DELETE_CERTIFICATE”	String	필수
CERTIFICATE_INDEX	삭제할 인증서 위치번호 (1 ~ 7 까지의 번호 사용 가능)	Integer	필수

■ 응답 Bundle 구성

파라메타		데이터 형식	필수 여부
키	값		
REQUEST_ACTION	“DELETE_CERTIFICATE”	String	필수
RESULT_CODE	응답코드 표 참고	String	필수
ERROR_MESSAGE	에러 메시지	String	

※ 응답코드 정의

응답코드	의미
“0000”	정상
“0E01”	지원하지 않는 보안토큰 제어방식
“0E02”	미지원 기능 (표준API 서비스 제공 기능 목록 참고)
“0E03”	NFC 기능을 사용할 수 없음
“0F00”	알 수 없는 오류
“0F01”	지원하지 않는 보안토큰 표준화 버전
“0F02”	입력 파라메타 오류
“0F03”	PIN 인증 실패
“0F04”	인증서 또는 키의 저장위치번호 오류
“0F05”	PIN 인증 오류로 인한 카드 잠김 상태
“0F06”	Security 조건(PIN 인증, 상호인증 등) 불충족
“0FC1”	사용 가능한 보안토큰이 존재하지 않음
“0FC2”	보안토큰과 NFC 칩 사이의 통신문제 발생 (스마트폰에 NFC 카드를 다시 접촉 시도)
“0FC3”	보안토큰과의 통신 에러
“0FC4”	보안토큰 내부명령 수행 실패
“0FC5”	인증서가 존재하지 않음

※ 에러 응답 시 보안토큰(스마트카드) 내부의 오류코드는 응답번들에 포함된 “ERROR_MESSAGE”에 “[]” 구분 기호 안에 명시한다.

부록 10. 규격 연혁

버전	제·개정일	제·개정내역
v1.00	2003년 9월	o “암호토큰을 위한 PKCS#11 프로파일 규격”으로 제정
v1.20	2007년 3월	o TTA 정보통신 용어사전의 토큰 표준용어를 준용하기 위해 규격 내 “암호토큰”을 “보안토큰”으로 용어 변경 o “보안토큰 기반 공인인증서 이용기술 규격”으로 규격명 변경 o 본 규격이 보안토큰 관련 표준API로 이용될 수 있도록 7.2.1 및 7.2.3에서 개인키 외부 유출관련 규정 삭제 등
v1.50	2007년 8월	o 공인인증서 가입자 S/W가 보안토큰 구동프로그램을 이용하여 객체를 생성하거나 처리할 경우, 해당 객체 속성 템플릿이 이용되도록 [부록 1.] 준용 규정 신설 o 공인인증서 가입자 S/W가 보안토큰에서 발생하는 비정상적인 오류상황에 대해 적절히 대처할 수 있도록, 구동프로그램이 지원해야 하는 반환값 프로파일 준용 규정 신설
v1.70	2007년 11월	o [부록4] 환경파일을 이용한 보안토큰 구동프로그램 위치정보 관리의 환경파일 구성내용 변경
v1.80	2008년 10월	o 보안토큰 보안기능 강화를 위해 암호프로세서 탑재 및 차분전력분석기법 등 공격에 대비할 수 있는 기능 추가
v1.90	2009년 9월	o 공인인증서 암호체계 고도화에 따른 알고리즘 변경 사항 반영
v2.00	2010년 7월	o 바이오 보안토큰 정의 o 바이오 보안토큰 등록정보 및 사용자 등록정보를 위한 데이터 객체 추가 o 바이오 보안토큰 API 반환값 프로파일 추가
v2.10	2012년 11월	o 모바일 플랫폼 환경의 위치정보 환경파일 및 검증정보 처리 내용 추가 o PKCS#11 구현 예시 추가 o 자바를 이용한 PKCS#11 구현 예시 추가
v2.20	2013년 9월	o 모바일토큰 정의 및 위치정보 관리 내용 추가 o 금융IC카드 표준 개정사항 반영
v2.30	2014년 4월	o ‘모바일토큰’을 ‘스마트인증’으로 명칭 변경

v2.40	2016년 1월	o 무선통신 지원 보안토큰 표준 API 적용 반영