무선단말기에서의 공인인증서 저장 및 이용 기술규격

Certificate Management in Mobile Device

v1.60

2017년 11월

KISA 한국인터넷진흥원
KOREA INTERNET & SECURITY AGENCY

목 차

무선단말기에서의 공인인증서 저장 및 이용 기술규격

Certificate Management in Mobile Device

1. 개 요

본 규격은 무선단말기에서의 공인인증서비스 상호연동성 확보 및 이용 편의성을 고려하여 무선단말기내 공인인증서 관리 소프트웨어의 요구사항을 정의한다.

2. 규격의 구성 및 범위

본 규격은 무선단말기에 공인인증서 저장 및 이를 활용한 공인인증서비스 제공 시 사용자 편의성 및 상호연동성을 고려한 기술적 요구사항을 명시하고 있다.

첫 번째로 본문에서는 무선단말기에서의 공인인증서 저장매체, 저장방법 및 이용, 사용자 인터페이스 요구사항을 명시하고 있으며 두 번째로 부록에서는 구현에 필요한 상세 내용을 정의하고 있다.

3. 관련 표준 및 규격

3.1 국외 표준 및 규격

[WMLSCrypto] WAP Forum Approved Version 20-June-2001, WMLScript Crypto Library, June 2001

[RFC2119] IETF, RFC2119, Key words for use in RFCs to Indicate Requirement Levels, March 1997

[RFC2630] IETF RFC 2630, Cryptographic Message Syntax, June 1999

[PKCS5] RSA, PKCS#5 v1.5 & v2.0, Password-Based Cryptography Standard, 1993

[PKCS7] RSA Laboratories, PKCS#7 v1.5, Cryptographic Message Syntax, November 1993

[PKCS8] RSA, PKCS#8 v1.2, Private Key Information Syntax Standard, 1993

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

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

3.2 국내 표준 및 규격

[KCAC.TS.UI] KISA, KCAC.TS.UI v2.11, 공인인증기관간 상호연동을 위한 사용자 인터페이스 기술규격, 2015

[KCAC.TS.HSMU] KISA, KCAC.TS.HSMU v2.40, 보안토큰 기반의 공인인증서 이용기술 규격, 2016

[KCAC.TS.HSMS] KISA, KCAC.TS.HSMS v1.3, 보안토큰 기반의 공인인증서 저장형식 기술규격, 2016

[KCAC.TS.SIVID] KISA, KCAC.TS.SIVID, v1.21, 식별번호를 이용한 본인확인 기술규격, 2009

[KCAC.TS.DSIG] KISA, KCAC.TS.DSIG, v1.30, *전자서명 알고리즘 규격*, 2009

[KCAC.TS.HASH] KISA, KCAC.TS.HASH, v1.20, *해쉬 알고리즘 규격*, 2009

[KCAC.TS.ENC] KISA, KCAC.TS.ENC, v1.21, *암호 알고리즘 규격*, 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(과학기술정보통신부 고시)에 정의되어 있다.

가) 가입자

나) 공인인증서

다) 전자서명생성키

라) 전자서명

4.2 용어의 정의

가) 내장 메모리 : 무선단말기에 내장되어 있는 메모리로 사용자 데이터 저장이 가능한 저장소 영역

나) 외장 메모리 : 무선단말기에 장착하여 사용하는 분리 가능한 메모리

4.3 용어의 효력

본 규격에서 사용된 다음의 용어들은 무선단말기에서 공인인증서를 저장하고 이용하는 프로그램의 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 약어가 이용된다.

가) App : Mobile Application, 모바일 응용프로그램
나) eSE : Embedded Secure Element, 내장형 보안모듈
다) IC칩 : Integrated Circuit Chip, 집적회로칩
라) SD카드 : Secure Digital Card
마) USIM : Universal Subscriber Identity Module, 범용사용자식별모듈

바) TEE : Trusted Execution Environment

사) AP : Application Processor

6. 공인인증서의 저장

6.1 저장 공간 및 위치

최상위인증기관 및 인증기관 인증서는 기본적으로 무선단말기의 운영체제에서 제공하는 소프트웨어(SW) 기반방식을 통해 저장 공간에 저장한다.

가입자 인증서는 <표 1>과 같이 무선단말기의 운영체제 등에서 제공하는 SW 기반방식을 통해 저장 공간에 저장하거나, IC칩, AP 등의 하드웨어(HW)가 지원하는 저장 공간에 저장할 수 있다.

<표 1> 공인인증서 저장위치

구분		저장위치
SW 기반방식	앱 데이터	무선단말기용 앱 데이터 경로
	인증서 저장소	해당 운영체제 등에서 제공되는 규격 준용
	기타	해당 저장위치 및 방법 규격 준용
HW 지원방식	저장토큰	[KCAC.TS.UI]의 [부록 3. 스마트카드 파일 구성도 및 메모리맵] 참조
	보안토큰	[KCAC.TS.HSMU]와 [KCAC.TS.HSMS] 준용
	기타	해당 하드웨어 저장매체의 규격 준용

기타 방식으로 공인인증서 및 전자서명생성키를 저장하는 경우에는 한국인터넷진흥원이 인정하는 보안평가인증을 받아야 한다.

6.2 SW 기반방식

공인인증서 및 전자서명생성키를 무선단말기용 앱의 데이터 경로에 저장할 수 있으며, 기본적으로 무선단말기 운영체제에서 제공하는 인증서 저장방법을 따르며, 저장방법이 제공되지 않는 경우 앱 데이터 경로의 “./NPKI/(인증기관 식별자)” 하위에 전자서명생성키는 [PKCS5]를 준용하여 암호화 하고, [PKCS8] 저장형식을 준용하여 저장할 수 있다.

Google사의 Android를 사용하는 무선단말기의 경우, 공인인증서 및 전자서

명생성키를 Android에서 제공하는 인증서 저장소(KeyStore 등)에 저장할 수 있다. Apple사의 iOS를 사용하는 무선단말기의 경우, 공인인증서 및 전자서명생성키를 iOS에서 제공하는 인증서 저장소(KeyChain 등)에 저장할 수 있다. 공인인증서 및 전자서명생성키는 인증서 저장소의 저장 규격을 준용하여 저장할 수 있다.

전자서명생성키 유출방지 기능을 갖춘 SW 기반의 저장매체를 이용하고자 하는 경우 각 저장매체의 규격을 준용하여 저장할 수 있다.

Apple사의 iOS를 사용하는 무선단말기의 경우, 응용 프로그램 간 상호호환이 필요할 경우 App ID를 갖는 공인인증서 App 호출을 통해 가입자의 공인인증서와 전자서명생성키를 획득한다. App ID는 id368493398이며, 응용 프로그램과 공인인증서 App간 메시지 처리 및 기능구현은 [부록 4. iOS 운영체제 무선단말기에서의 공인인증서 저장방법]을 준용한다.

MS사의 윈도우 운영체제[1]를 사용하는 무선단말기의 경우, 공인인증서 이용 App만이 접근할 수 있도록 공인인증서와 전자서명생성키를 안전하게 저장하여야 한다. 응용 App 간 상호호환이 필요한 경우 공인인증서 App 호출을 통해 가입자의 공인인증서와 전자서명생성키를 획득할 수 있다. 공인인증서 App과 응용 App간 메시지 처리 및 기능구현은 [부록 5. 윈도우 운영체제 무선단말기에서의 공인인증서 호환성 확보]를 준용한다.

6.3 HW 지원방식

무선단말기의 USIM, eSE, microSD 카드 등 IC칩을 저장토큰과 보안토큰으로 이용하여 공인인증서와 전자서명생성키를 저장하는 경우, 각각 [KCAC.TS.UI], [KCAC.TS.HSMU]와 [KCAC.TS.HSMS]를 준용하여 저장한다.

TPM, TrustZone, TEE 등 전자서명생성키 유출방지 기능을 갖춘 하드웨어 저장매체를 보안토큰 및 저장토큰 이외의 방법으로 이용하고자 하는 경우 각

1) 윈도우폰을 포함하여 타일형태의 새로운 UI 스타일을 사용하는 윈도우 운영체제 환경을 무선단말기 환경으로 규정

저장매체의 규격을 준용하여 저장할 수 있다.

7. 전자서명 생성

무선단말기 내의 공인인증서 저장매체에 저장된 전자서명생성키를 이용하여 전자서명을 생성하는 경우, 전자서명은 [PKCS7], [RFC2630] 등의 전자서명문 형식을 이용할 수 있다. 사용자가 전자서명 원문을 확인해야 하는 경우, 원문 확인 기능을 제공하여야 한다.

8. 전자서명 전송

8.1 전자서명 요청 및 생성 주체 인증

데스크톱 환경 또는 무선단말기에서 타 무선단말기로 전자서명을 요청하여 전자서명 결과를 전송받는 경우, 전자서명 요청 주체(예, 데스크톱 PC)와 전자서명 생성 주체(예, 스마트폰) 간의 인증을 수행하여야 한다.

전자서명 요청 주체와 전자서명 생성 주체 사이에 중계서비스가 이용되는 경우, 전자서명 요청 주체 및 전자서명 생성 주체는 중계서버 인증서 등을 사용하여 중계서버의 신뢰성을 검증할 수 있는 방법을 제공하여야 한다.

8.2 전자서명 전송

전자서명 요청 주체와 전자서명 생성 주체 간 전송채널은 반드시 암호화되어야 한다. 전송채널의 암호화는 RSA 1024비트 이상의 안전성에 준하여야 하며, 전자서명의 전송 시마다 매번 새로운 암호화키를 사용하여야 한다. 이때 알고리즘은 [KCAC.TS.DSIG], [KCAC.TS.HASH], [KCAC.TS.ENC]에 규정된 암호 알고리즘을 사용한다.

암호화된 전송채널은 안전한 난수생성함수의 이용, 키분배용 인증서 이용 등을 통해 안전하게 생성되어야 한다.

전자서명 전송서비스 제공을 위한 중계서버는 출입통제, 물리적 침입감시, 시스템 및 네트워크 보호 등의 사항에 대해 보호설비를 갖추어야 한다. 또한 중계서버는 전자서명문 전달 후 기억장치 및 임시파일에서 즉시 해당 내용을 삭제하여야 한다.

9. 무선단말기에서의 공인인증서 사용자 인터페이스

무선단말기내 공인인증서 가입자 소프트웨어는 사용자가 <표 1>의 저장매체를 선택할 수 있는 기능을 제공하여야 한다. 또한, 저장매체 선택 시 해당 저장매체에 저장되어 있는 모든 인증서를 화면에 보여주고 인증서를 선택할 수 있는 기능을 제공하여야 한다. 다만, 무선단말기에서 외장메모리, IC칩 등 저장매체를 지원하지 않을 경우에는 해당 저장매체를 표시하지 않을 수 있다.

저장위치 표시를 위한 명칭은 저장매체 명칭과 저장매체 분류명칭을 조합하여 사용할 수 있다. 예로, USIM 내의 보안토큰을 이용하는 경우, 저장매체 표시를 “USIM 보안토큰”으로 표시한다.

<표 2> 무선단말기 내 공인인증서 저장매체 표시 명칭 예시

저장매체		표시 명칭	저장매체		표시 명칭
내장 메모리		내장 메모리	외장 메모리		외장 메모리
USIM	저장토큰	USIM 저장토큰	microSD	저장토큰	microSD 저장토큰
	보안토큰	USIM 보안토큰		보안토큰	microSD 보안토큰

또한, 가입자 소프트웨어는 가입자가 인증서의 정보를 알 수 있도록 인증서 소유자명, 인증서 유효기간, 발급기관명, 인증서 용도 등을 표시할 수 있다. 다만, 무선단말기 화면의 제약으로 모든 정보를 한 화면에 보여주기 어려운 경우에는 초기화면에서 인증서 소유자만을 표시하고 문자열의 횡 스크롤이나 인증서 보기 메뉴를 통한 별도 화면을 통해 기타 정보를 표시할 수 있다.

구분	내용
인증서 상태	o 인증서 검증 결과(유효, 폐지 등) 표시
소유자명	o 사용자의 인증서 CN(인증서 정보 이용) o 기관인 경우 기관의 실명(또는 인증기관 식별자)으로 표시
발급기관	o 공인인증기관의 실명(인증서 정보 이용)으로 표시
만료일	o 인증서 만료일(또는 유효기간) 표시
인증서 용도	o 공인인증서의 용도(범용, 용도제한용)

부록 1. iOS 운영체제 무선단말기에서의 공인인증서 저장방법

1. 개요

Apple사의 iOS 운영체제를 갖는 무선단말기에서의 응용 App간 공인인증서 저장 위치 호환성 확보를 위해 공인인증서 App은 공인인증서 저장과 공인인증서 획득 기능을 갖춰야 한다. 또한, 응용 App은 공인인증서 App을 통해 가입자의 공인인증서를 내보내고 가져올 수 있는 기능을 구현해야 한다.

2. 공인인증서 App URL Scheme

공인인증서 App의 URL Scheme은 다음과 같다.

```
o URL Scheme : "kisa-cert-exchange"
o iTunes link  : http://itunes.apple.com/kr/app/id368493398?mt=8
```

3. 공인인증서 전송 메시지 형식

응용 App과 공인인증서 App간 전송되는 공인인증서 및 전자서명생성키의 형태는 [PKCS12]의 방법을 이용하여 전송 메시지의 무결성을 보장해야 한다. 가입자가 전자서명용과 키분배용 공인인증서를 함께 소유한 경우에는 반드시 전자서명용과 키분배용 인증서를 함께 전송해야 한다. 또한, 필요한 경우에는 공인인증기관 인증서를 함께 전송할 수 있다.

4. 공인인증서 App의 인증서 저장기능

가입자가 타 응용 App에서 공인인증서를 사용하기 위해 응용 App의 내보내기 기능을 활성화할 경우 공인인증서 App이 처리해야 하는 공인인증서 저장 기능 요구사항을 정의한다. 응용 App은 가입자가 내보내기 할 공인인증서를 선택할 수 있는 기능을 제공해야 한다.

4.1 메시지 파라미터

응용 APP이 공인인증서 App을 호출할 때 전송되는 파라미터는 명령어(cmd,

필수정보)인 'CertPush', 응용 App의 URL Scheme(caller_url_scheme, 선택정보), 응용 App의 현재 상태정보 문자열(callback, 선택정보), 전송하고자 하는 인증서(cert, 필수정보) 정보인 Base64 인코딩된 [PKCS12] 문자열이다.

```
o 공인인증서 APP 저장 예시
  kisa-cert-exchange://?cmd=certpush&caller_url_scheme=kbapp&callback=01&c
  ert=(base64 String)
```

4.2 반환값

공인인증서 App은 공인인증서 저장기능 완료 후 자신을 호출한 응용 App을 호출한다. 이때 전송하는 호출 파라미터는 결과 문자열(00 : 성공, 11 : 실패)과 응용 App로 전송받은 현재 상태정보 문자열이다.

```
o 공인인증서 APP 저장 반환 예시
  kbapp://?cmd=certpush&callback=01&result=00
```

5. 공인인증서 App의 인증서 획득기능

가입자가 타 응용 App에서 공인인증서를 사용하기 위해 응용 App의 가져오기 기능을 활성화할 경우 공인인증서 App이 처리해야 하는 공인인증서 획득 기능 요구사항을 정의한다. 공인인증서 App은 가입자가 내보내기할 공인인증서를 선택할 수 있는 기능을 제공해야 한다.

5.1 메시지 파라미터

응용 App이 공인인증서 App를 호출할 때 전송하는 파라미터는 명령어(cmd, 필수정보)인 'CertGet', 응용 App의 URL Scheme(caller_url_scheme, 선택정보), 응용 App의 현재 상태정보 문자열(callback, 선택정보)이다.

```
o 공인인증서 APP 저장 반환 예시
  kisa-cert-exchange://?cmd=certget&caller_url_scheme=kbapp&callback=01
```

5.2 반환값

공인인증서 App은 공인인증서 획득기능 완료 후 자신을 호출한 응용 App을 호출한다. 이때 전송하는 호출 파라미터는 결과 문자열(00 : 성공, 11 : 실패), Base64 Encoding된 [PKCS12] 문자열(필수 정보), 응용 App로 전송받은 현재 상태정보 문자열이다.

o 공인인증서 반환 예시
kisa-cert-exchange://?cmd=certget&callback=01&result=00&cert=base64(pfx)

부록 2. 윈도우 운영체제 무선단말기에서의 공인인증서 저장방법

1. 개요

MS사의 윈도우 운영체제를 탑재한 무선단말기에서의 공인인증서 저장방법과 공인인증서를 이용하는 응용 App(이하 '응용 App')에서의 연동방법을 설명한다.

2. 공인인증서 App

MS 윈도우 운영체제를 갖는 무선단말기에서의 응용 App간 공인인증서 저장위치 호환성 확보를 위해 윈도우 공인인증서 관리 App(이하 '공인인증서 App')은 공인인증서 저장과 공인인증서 획득기능을 갖춰야 한다. 또한 응용 App은 공인인증서 App을 통해 가입자의 공인인증서를 내보내고 가져올 수 있는 기능을 구현해야 한다.

공인인증서 App은 가입자가 내보내기할 공인인증서를 선택할 수 있는 기능을 제공해야 한다. 가입자가 전자서명용과 키분배용 인증서를 함께 소유한 경우에는 반드시 전자서명용과 키분배용 인증서를 함께 전송해야 한다. 또한, 필요한 경우에는 공인인증기관 인증서를 함께 전송할 수 있다.

응용 App과 공인인증서 App간 전송되는 공인인증서와 전자서명생성키에 대한 무결성을 보장해야 한다.

3. 공인인증서 App 연동방법

공인인증서 App과 응용 App 간의 연동은 URL 프로토콜을 이용하여 해당 App을 호출하고 공인인증서와 전자서명생성키는 별도의 데이터 전송방법에 의해 전달한다. 공인인증서 App의 URL 프로토콜은 다음과 같다.

```
o URL 프로토콜 : "kisa-cert-exchange"
```

공인인증서 App과 응용 App 간에 전송되는 파라미터는 다음의 [표 5]와 같다.

4. 공인인증서 App의 인증서 저장기능

4.1 공인인증서 App 호출

파라미터 명	파라미터 값	기본값	처리
cmd	o 인증서와 개인키의 내보내기 및 가져오기 요청 – certpush(내보내기), certget(가져오기)	–	M
caller_protocol	o 호출 App의 프로토콜	–	O
callback	o 호출 App의 콜백 아이디	–	O
cert	o 인증서 및 개인키의 전송방식 – share(공유기능), clip(클립보드)	clip	M
type	o 전송 데이터 형식 – pfx : PKCS#12 형식, der : DER 인코딩 형식	der	O
pki	o NPKI, GPKI, MPKI, EPKI 중 1개 선택	NPKI	O
result	o 처리결과 – 00(성공), 11(실패)	–	M
reason	o 처리결과에 따른 오류메시지	–	O

[표 5] URL 프로토콜 파라미터 정의

응용 App이 공인인증서 App에 공인인증서를 저장(내보내기)하는 경우 cmd 파라미터의 값이 'CertPush', 응용 App이 공인인증서 App으로부터 공인인증서를 획득(가져오기)하는 경우 'CertGet'를 이용한다.

따라서, 응용 APP이 공인인증서 App을 호출할 때 전송되는 파라미터는 명령어(cmd, 필수정보)인 'CertPush' 또는 'CertGet', 응용 App의 URL 프로토콜(caller_protocol, 선택정보), 응용 App의 현재 상태정보 문자열(callback, 선택정보), 인증서 전송방식(cert, 필수정보), 전송 데이터 형식(type, 선택사항)이다.

전송 데이터 형식(type)은 [PKCS12] 형식(pfx)과 DER 인코딩된 인증서 및 전자서명생성키 형식(der) 선택 가능하다.

<공인인증서 APP 호출 예시>

1. 응용 App(msapp)은 공인인증서 App(kisa-cert-exchange)에 윈도우8의 클립보드를 이용하여 PKCS#12형식으로 인증서를 전송하기 위해 호출한다.
   kisa-cert-exchange://?cmd=certpush&caller_protocol=msapp&callback=01&cert=clip&type=pfx

2. 응용 App(msapp)은 공인인증서 App(kisa-cert-exchange)에 윈도우8의 클립보드를 이용하여 DER형식으로 인증서를 전송하기 위해 호출한다.
   kisa-cert-exchange://?cmd=certpush&caller_protocol=msapp&callback=01&cert=clip&type=der

4.2 데이터 전송

전송형식	파라미터 명	파라미터 값	
DER	cert	전자서명용 공인인증서	
	key	전자서명용 전자서명생성키	
	kmsign	키분배용 공인인증서	
	kmkey	키분배용 전자서명생성키	
[PKCS12]	pfx	Base64 형식으로 인코딩된 [PKCS12] 문자열	

[표 7] 데이터 전송 파라미터 정의

URL 프로토콜을 통한 공인인증서 App 호출 후 데이터 전송방법에 따른 데이터 전송을 하여야 한다. 인증서 전송방식(cert)이 clip인 경우, 클립보드를 이용하여 데이터를 전송하며, 인증서 전송방식(cert)이 share인 경우, 공유기능을 활용하여 데이터를 전송한다.

전송 데이터 형식(type)을 pfx로 호출한 경우 공인인증서와 전자서명생성키를 [PKCS12] 형식으로 변경하여야 하며, der로 호출한 경우는 공인인증서와 전자서명생성키의 형식변경 없이 전송한다.

클립보드를 활용하여 [PKCS12] 형식으로 인증서를 전송하고자 하는 경우, 응용 App은 클립보드 초기화 후 “pfx=[Base64 형식으로 인코딩된 PKCS#12 문자열]”을 클립보드에 복사하고, 공인인증서 App은 클립보드에서 해당 데이터를 획득한 후 바로 클립보드를 초기화한다. der 형식으로 인증서를 전송하고자 하는 경우 “cert=[Base64 형식으로 인코딩된 인증서 파일(SignCert.der) 문자열]&key=[Base64 형식으로 인코딩된 개인키 파일(SignCert.key) 문자열]”을 사용한다. 키분배용 인증서를 함께 전송하고자 하는 경우에는 kmsign, kmkey 파라미터를 함께 사용한다.

공유기능을 활용하여 인증서를 전송하는 경우, 클립보드 방식과 동일한 데이터 형식을 공유기능을 통해 공인인증서 App에 인증서를 전송한다. 공유기능에 대한 자세한 구현내용은 MS사의 윈도우 운영체제 관련 개발문서를 참고한다.

4.3 반환값

공인인증서 App은 공인인증서 저장기능 완료 후 자신을 호출한 응용 App을 호출한다. 이때 전송하는 호출 파라미터는 결과 명령어(cmd, 필수정보)인 ‘CertPush’, 전송결과(result, 필수정보) 문자열(00 : 성공, 11 : 실패), 응용 App로 전송받은

현재 상태정보 문자열(callback, 선택정보) 및 오류메시지(reason, 선택정보)이다.

<공인인증서 APP 반환 예시>

1. 공인인증서 App(kisa-cert-exchange)은 인증서 저장 후 응용 App(msapp)을 호출하여 전송결과(성공)를 전달한다.
   msapp://?cmd=certpush&callback=01&result=00
2. 공인인증서 App(kisa-cert-exchange)은 인증서 저장 후 응용 App(msapp)을 호출하여 전송결과(실패)와 오류메시지를 전달한다.
   msapp://?cmd=certpush&callback=01&result=11&reason=오류메시지

5. 예시

5.1 공인인증서 App에서 응용 App으로 인증서 가져오기

공인인증서 App에서 응용 App으로 공인인증서와 전자서명생성키를 클립보드 및 윈도우 공유기능을 이용하여 DER 형식으로 가져오기 하는 예시는 다음과 같다.

o 공인인증서 App 호출

```
string strProtocol = "kisa-cert-exchange:cmd=certget&caller_protocol=msapp&callback=01&cert
=clip&type=der";  // 또는 cert=share
var uri = new Uri(strProtocol);
var success = await Windows.System.Launcher.LaunchUriAsync(uri);
```

o 공인인증서 및 전자서명생성키 수신(URL 프로토콜 방식)

```
// App,xaml.cs
// 프로토콜로 호출된 경우 진입점
protected override void OnActivated(IActivatedEventArgs args)
{
    if (args.Kind == ActivationKind.Protocol)
    {
        ProtocolActivatedEventArgs protocolArgs = args as ProtocolActivatedEventArgs;
        String strProtocol = "";
        strProtocol = protocolArgs.Uri.PathAndQuery;
//"cmd=certget&caller_ protocol=kisa-cert-exchange&callback=1&cert=clip &type=der&pki=NPK
```

```
I"
            GerReceivedData(strProtocol);
        }
        Window.Current.Activate();
}

private void GetReceivedData(string protocol)
{
        //protocol 파싱 처리
        //cmd가 clip인경우
        var dataPackageView = Clipboard.GetContent(); //클립보드 이용
        if (dataPackageView.Contains(StandardDataFormats.Text))
        {
            var text = await dataPackageView.GetTextAsync();
            //데이터는 "signcert=data1&sigkey=data2&kmcert=data3&kmkey=data4"형식으로 수신 됨
            //암호용 인증서(kmcert,kmkey)가 없을 경에는 "null"로 수신
            Clipboard.Clear();
        }
}
```

o 공인인증서 및 전자서명생성키 수신(윈도우 공유 방식)

```
// App,xaml.cs
// 공유로 호출된 경우 진입점
protected override void OnShareTargetActivated(ShareTargetActivatedEventArgs args)
{
        //공유로(share)로 수신시 별도의 공유 페이지 필요
        var rootFrame = new Frame();
        rootFrame.Navigate(typeof(sharedpage), args.ShareOperation);
        Window.Current.Content = rootFrame;
        Window.Current.Activate();
}

// sharedpage.xaml.cs
ShareOperation shareOperation;
private IReadOnlyList<IStorageItem> sharedStorageItems; //공유된 항목(파일리스트 형식)
async protected override void OnNavigatedTo(NavigationEventArgs e)
```

```
{
    this.shareOperation = (ShareOperation)e.Parameter;
    await Task.Factory.StartNew(async () =>
    {
        if (this.shareOperation.Data.Contains(StandardDataFormats.StorageItems))
        {
            try
            {
                this.sharedStorageItems = await this.shareOperation.Data.GetStorageItemsAs
ync();
            }
            catch (Exception ex)
            {
            }
        }
        await Dispatcher.RunAsync(CoreDispatcherPriority.Normal, async () =>
        {
            if (this.sharedStorageItems != null)
            {
                // Display the name of the files being shared.
                StorageFolder localFolder = ApplicationData.Current.LocalFolder;
                String fileName = "";
                byte[] tmpDer = null;
                byte[] tmpKey = null;
                for (int index = 0; index < this.sharedStorageItems.Count; index++)
                {
                    StorageFile file = this.sharedStorageItems[index] as StorageFile;
                    fileName = file.Name.ToLower();
                    Windows.Storage.Streams.IBuffer buffer = null;
                    buffer = await Windows.Storage.FileIO.ReadBufferAsync(file);
                    if (buffer != null)
                    {
                        var datareader = Windows.Storage.Streams.DataReader.FromBuffer(bu
ffer);
                        int fsize = (int)buffer.Length;
                        if (fileName.IndexOf("signcert.der") >= 0)
```

```
                        {
                            tmpDer = new byte[fsize];
                            datareader.ReadBytes(tmpDer);
                        }
                        else if (fileName.IndexOf("signpri.key") >= 0)
                        {
                            tmpKey = new byte[fsize];
                            datareader.ReadBytes(tmpKey);
                        }
                        else if (fileName.IndexOf("kmcert.der") >= 0)
                        {
                            tmpDer = new byte[fsize];
                            datareader.ReadBytes(tmpkmDer);
                        }
                        else if (fileName.IndexOf("kmpri.key") >= 0)
                        {
                            tmpKey = new byte[fsize];
                            datareader.ReadBytes(tmpkmKey);
                        }
                    }
                }
                String strTo = ApplicationData.Current.LocalFolder.Path;
                if (tmpDer != null && tmpKey != null)
                {
                    //인증서 저장
                }
            }
        });
    });
}
```

5.2 공인인증서 App으로 인증서 내보내기

응용 App에서 공인인증서 App으로 공인인증서와 전자서명생성키를 클립보드 및 윈도우 공유기능을 이용하여 DER 형식으로 내보내기 하는 예시는 다음과 같다.

o 공인인증서 및 전자서명생성키 송신(클립보드 방식)

```
byte[] cert = certdata;
byte[] pKey = keydata
string certB64 = Convert.ToBase64String(cert);
string keyB64 = Convert.ToBase64String(pKey);
//클립 보드에 데이터 저장
DataPackage dataPackage = new DataPackage();
dataPackage.SetText(string.Format("signcert={0}&signkey={1}", certB64, keyB64));
Clipboard.SetContent(dataPackage);
//프로토콜 생성
string strProtocol =
 "kisa-cert-exchange:cmd=certpush&caller_ protocol=msapp&callback=01&cert=clip&type=der";
var uri = new Uri(strProtocol);
//인증서 관리 앱 호출
var success = await Windows.System.Launcher.LaunchUriAsync(uri);
```

o 공인인증서 및 전자서명생성키 송신(윈도우 공유 방식)

```
private static IReadOnlyList<StorageFile> storageItems;
private static DataTransferManager dataTransferManager;
async public void ExportCertification()
{
    dataTransferManager = DataTransferManager.GetForCurrentView();
    dataTransferManager.DataRequested += new TypedEventHandler<DataTransferManager, DataReq
uestedEventArgs>(OnDataRequested);

    StorageFolder f = ApplicationData.Current.LocalFolder;
    storageItems = //내보낼 인증서 파일 리스트 설정
    DataTransferManager.ShowShareUI(); //공유 창 호출
}

/// 공유시 데이터 전송
private static void OnDataRequested(DataTransferManager sender, DataRequestedEventArgs e)
{
    DataRequest request = e.Request;
```

```
    if (storageItems != null)
    {
        if (storageItems.Count == 0)
        {
            return;
        }
        DataPackage requestData = request.Data;
        requestData.Properties.Title = "Title";
        requestData.Properties.Description = "Description."; //The description is optional.
        requestData.SetStorageItems(storageItems);
        // Unregister the current page as a share source.
        if (dataTransferManager != null)
            dataTransferManager.DataRequested -= new TypedEventHandler<DataTransferManage
r, DataRequestedEventArgs>(OnDataRequested);

    }
    else
    {
        request.FailWithDisplayText("Select the files you would like to share and try agai
n.");
    }
    if (dataTransferManager != null)
    {
        dataTransferManager.DataRequested -= new TypedEventHandler<DataTransferManager, Dat
aRequestedEventArgs>(OnDataRequested);
        dataTransferManager = null;
        storageItems = null;
    }
}
```

부록 3. 규격 연혁

버전	제·개정일	제·개정내역
v1.00	2007년 4월	o “무선단말기내 공인인증서 저장 및 이용 기술 규격”으로 제정
v1.10	2008년 10월	o 무선단말기에서의 공인인증서 저장매체를 기존의 내장형·외장형 메모리에 USIM을 추가하고 저장매체별 저장위치 및 저장형식을 표준화 o USIM 추가에 따른 저장 방법 변경 및 공인인증서 관리 프로그램의 저장매체 표시 방법 변경
v1.11	2009년 9월	o 공인전자서명인증체계 기술규격 개정에 따라 본문 내용 중 관련 기술규격 참조 변경 사항 개정
v1.12	2010년 3월	o 스마트폰 환경에서의 공인인증서 저장위치 및 형식 표준화
v1.20	2010년 10월	o 바다폰 등 내장형 메모리에 표준 인터페이스 함수(PKCS#11)를 구현하기 힘든 스마트폰에 대한 공인인증서 저장위치 마련
v1.30	2012년 11월	o 윈도우8 UI 환경에서의 공인인증서 이용방법 o 무선단말기에서의 전자서명 생성 및 전송 표준화
v1.40	2013년 10월	o 무선단말기 앱 데이터 경로 내 저장 등 공인인증서 저장위치 및 방법 변경
v1.50	2015년 4월	o 무선단말기 내·외장메모리에서의 공인인증서 저장위치 및 방법 변경
v1.60	2017년 11월	o 무선단말기에서 공인인증서 저장 공간, 위치 및 방법 변경