RSA LABORATORIES

PKCS #11 Profiles For Mobile Devices – Final Draft

RSA Security

FINAL DRAFT – April 25, 2003

Editor's note: *This is the final draft of this conformance profile document, which is available for a 30-day public review period (deadline: May 25, 2003). Please send comments and suggestions, both technical and editorial, to cryptoki@rsasecurity.com*

Table of Contents

1. Introduction

Mobile devices such as phones, handsets, personal digital assistants, etc. are increasingly used for business-related data access and services. In this process, users expect and demand secure and seamless access to enterprise information. This implies mobilized applications with built-in business requirements for security services, often realized in the form of personal security tokens such as smart cards. Since the platform situation is heterogeneous, it creates a need for a standardized security interface, not just to ease the mobilization of applications, but also to utilize security functionality inherent in the mobile device to the best possible extent.

The above underscores the potential of PKCS #11 [2] ("Cryptoki") in this context. Because the environment also is constrained in terms of memory and computational resources, however, the sheer size and flexibility of PKCS #11 may complicate its use. This document presents conformance profiles of PKCS #11, based on the "Large Application Profile" of [1] that intends to solve this issue. Specifically, this document:

- identifies objects and attributes that must be supported;
- enumerates mechanisms that must be provided;
- lists the functional interface that must be provided; and
- specifies any other capabilities that must be met, in order for a Cryptoki library to be compliant with these profiles.

2. Terms and definitions

Terms and definitions are as in [2].

3. Common requirements

These requirements are common for both profiles in this document.

3.1 Session support

A Cryptoki library conformant with a profile in this document must support one R/W and at least ten simultaneous R/O sessions.

3.2 Thread handling

A Cryptoki library conformant with a profile in this document must support option 4) in [2], Section 6.5.2, i.e. "The application can specify that it will be accessing the library concurrently from multiple threads, and the library must use either the native operation system synchronization primitives or a set of application-supplied synchronization primitives to ensure proper thread-safe behavior."

4. Signature device profile

4.1 Cryptoki object classes

A Cryptoki library conformant with this profile must support (be able to create, search for, retrieve - when applicable, and use) the following objects defined in [2]:

- `CKO_CERTIFICATE` (at least as a token object)
- `CKO_PRIVATE_KEY` (both as a token and a session object)
- `CKO_PUBLIC_KEY` (both as a token and a session object)

As indicated above, it must be possible to create private keys both as session and token objects (assuming needed token permissions).

4.2 Cryptoki key types

A Cryptoki library conformant with this profile must support (be able to create, search for, retrieve - when applicable, and use) the following key types defined in [2]:

- `CKK_RSA`

4.3 Cryptoki certificate types

A Cryptoki library conformant with this profile must support (be able to create, search for, store, retrieve and make use of) the following certificate types defined in [2]:

- `CKC_X_509`

4.4 Cryptoki attributes

A Cryptoki library conformant with this profile must support (be able to create, set, search for, and retrieve) the following attributes defined in [2] for all supported objects:

Attribute	Applicable object class		
	CKO_CERTIFICATE	CKO_PRIVATE_KEY	CKO_PUBLIC_KEY
CKA_CLASS	✓	✓	✓
CKA_TOKEN	✓	✓	✓
CKA_PRIVATE	✓	✓	✓
CKA_MODIFIABLE	✓	✓	✓
CKA_LABEL	✓	✓	✓
CKA_CERTIFICATE_TYPE	✓		
CKA_TRUSTED	✓		✓
CKA_SUBJECT	✓[1]	✓	✓
CKA_ID	✓[1]	✓	✓
CKA_ISSUER	✓[1]		
CKA_SERIAL_NUMBER	✓		
CKA_VALUE	✓		
CKA_KEY_TYPE		✓	✓
CKA_LOCAL		✓	✓
CKA_KEY_GEN_MECHANISM		✓	✓
CKA_SENSITIVE		✓	
CKA_ALWAYS_SENSITIVE		✓	
CKA_EXTRACTABLE		✓	
CKA_NEVER_EXTRACTABLE		✓	
CKA_SIGN		✓	
CKA_VERIFY			✓
CKA_MODULUS		✓[2]	✓[2]
CKA_MODULUS_BITS		✓[2]	✓[2]
CKA_PUBLIC_EXPONENT		✓[2]	✓[2]

[1]Only when CKA_CERTIFICATE_TYPE is CKC_X_509
[2]Only when CKA_KEY_TYPE is CKK_RSA

Applications must be allowed to change the label of an object after creation. Applications must not expect to be able to modify any other attribute values after object creation.

4.5 Cryptoki mechanisms

A Cryptoki library conformant with this profile must support the following mechanisms defined in [2]:

- CKM_RSA_KEY_PAIR_GEN
- CKM_RSA_PKCS
- CKM_MD5_RSA_PKCS
- CKM_SHA1_RSA_PKCS
- CKM_SHA_1
- CKM_MD5

4.6 Cryptoki functions

A Cryptoki library conformant with this profile must support the following functions defined in [2], in addition to the functions listed in the "Base APIs" of [1]:

- C_SetPIN
- C_GetSessionInfo
- C_Login
- C_Logout
- C_CreateObject
- C_DestroyObject
- C_SetAttributeValue
- C_DigestInit

- C_Digest
- C_SignInit
- C_Sign
- C_VerifyInit
- C_Verify
- C_GenerateKeyPair
- C_SeedRandom
- C_GenerateRandom

Note: C_Login must support both CKU_USER and CKU_SO.

5. Communication device profile

5.1 Cryptoki object classes

A Cryptoki library conformant with this profile must support (be able to create, search for, retrieve - when applicable, and use) the following objects defined in [2]:

- CKO_CERTIFICATE (at least as a token object)
- CKO_PRIVATE_KEY (both as a token and a session object)
- CKO_PUBLIC_KEY (both as a token and a session object)
- CKO_SECRET_KEY (at least as a session object)

As indicated above, it must be possible to create private keys both as session and token objects (assuming needed token permissions).

5.2 Cryptoki key types

A Cryptoki library conformant with this profile must support (be able to create, search for, retrieve - when applicable, and use) the following key types defined in [2]:

- CKK_RSA
- CKK_RC4 *or* CKK_DES3 (at least one of these two)

5.3 Cryptoki certificate types

A Cryptoki library conformant with this profile must support (be able to create, search for, store, retrieve and make use of) the following certificate types defined in [2]:

- CKC_X_509

5.4 Cryptoki attributes

A Cryptoki library conformant with this profile must support (be able to create, set, search for, and retrieve) the following attributes defined in [2]:

Attribute	Applicable object class			
	CKO_CERTIFICATE	CKO_PRIVATE_KEY	CKO_PUBLIC_KEY	CKO_SECRET_KEY
CKA_CLASS	✓	✓	✓	✓
CKA_TOKEN	✓	✓	✓	✓
CKA_PRIVATE	✓	✓	✓	✓
CKA_MODIFIABLE	✓	✓	✓	✓
CKA_LABEL	✓	✓	✓	✓
CKA_CERTIFICATE_TYPE	✓			
CKA_TRUSTED	✓		✓	
CKA_SUBJECT	✓[1]	✓	✓	
CKA_ID	✓[1]	✓	✓	✓
CKA_ISSUER	✓[1]			
CKA_SERIAL_NUMBER	✓			
CKA_VALUE	✓			✓
CKA_KEY_TYPE		✓	✓	✓
CKA_LOCAL		✓	✓	✓
CKA_KEY_GEN_MECHANISM		✓	✓	✓
CKA_SENSITIVE		✓		✓
CKA_ALWAYS_SENSITIVE		✓		✓
CKA_EXTRACTABLE		✓		✓
CKA_NEVER_EXTRACTABLE		✓		✓
CKA_SIGN		✓		✓
CKA_VERIFY			✓	✓
CKA_MODULUS		✓[2]	✓[2]	
CKA_MODULUS_BITS		✓[2]	✓[2]	
CKA_PUBLIC_EXPONENT		✓[2]	✓[2]	
CKA_ENCRYPT			✓	✓
CKA_DECR|YPT		✓		✓
CKA_WRAP			✓	✓
CKA_UNWRAP		✓		✓
CKA_DERIVE				✓
CKA_VALUE_LEN				✓

[1]Only when CKA_CERTIFICATE_TYPE is CKC_X_509
[2]Only when CKA_KEY_TYPE is CKK_RSA

Applications must be allowed to change the label of an object after creation. Applications must not expect to be able to modify any other attribute values after object creation.

5.5 Cryptoki mechanisms

A Cryptoki library conformant with this profile must support the following mechanisms defined in [2]:

- `CKM_RSA_KEY_PAIR_GEN`
- `CKM_RSA_PKCS`
- `CKM_MD5_RSA_PKCS`
- `CKM_SHA1_RSA_PKCS`
- `CKM_SHA_1`
- `CKM_MD5`
- `CKM_SHA_1_HMAC`
- `CKM_RC4` *or* `CKM_DES3_CBC`
- `CKM_SSL3_PRE_MASTER_KEY_GEN`
- `CKM_SSL3_MASTER_KEY_DERIVE`
- `CKM_SSL3_KEY_AND_MAC_DERIVE`
- `CKM_TLS_PRE_MASTER_KEY_GEN`
- `CKM_TLS_MASTER_KEY_DERIVE`
- `CKM_TLS_KEY_AND_MAC_DERIVE`
- `CKM_SSL3_MD5_MAC`
- `CKM_SSL3_SHA1_MAC`

Note: At least one of `CKM_RC4` and `CKM_DES3_CBC` must be supported.

5.6 Cryptoki functions

A Cryptoki library conformant with this profile must support the following functions defined in [2], in addition to the functions listed in the "Base APIs" of [1]:

- `C_SetPIN`
- `C_GetSessionInfo`
- `C_Login`
- `C_Logout`
- `C_CreateObject`
- `C_DestroyObject`
- `C_SetAttributeValue`
- `C_DigestInit`
- `C_Digest`
- `C_SignInit`
- `C_Sign`
- `C_VerifyInit`
- `C_Verify`
- `C_EncryptInit`
- `C_Encrypt`
- `C_DecryptInit`
- `C_Decrypt`
- `C_GenerateKey`
- `C_GenerateKeyPair`
- `C_Wrap`
- `C_Unwrap`
- `C_SeedRandom`
- `C_GenerateRandom`
- `C_DeriveKey`

Note: `C_Login` must support both `CKU_USER` and `CKU_SO`.

A. Intellectual property considerations

RSA Security makes no patent claims on the general constructions described in this document, although specific underlying techniques may be covered.

License to copy this document is granted provided that it is identified as "RSA Security Inc. Public-Key Cryptography Standards (PKCS)" in all material mentioning or referencing this document.

RSA Security makes no representations regarding intellectual property claims by other parties. Such determination is the responsibility of the user.

B. References

[1] RSA Laboratories. *PKCS #11: Conformance Profile Specification*. October 2000. URL: http://www.rsasecurity.com/rsalabs/pkcs/pkcs-11/

[2] RSA Laboratories. *PKCS #11 v2.11 (Revision 1): Cryptographic Token Interface*. November 2001. URL: http://www.rsasecurity.com/rsalabs/pkcs/pkcs-11/

C. About PKCS

The *Public-Key Cryptography Standards* are specifications produced by RSA Laboratories in cooperation with secure systems developers worldwide for the purpose of accelerating the deployment of public-key cryptography. First published in 1991 as a result of meetings with a small group of early adopters of public-key technology, the PKCS documents have become widely referenced and implemented. Contributions from the PKCS series have become part of many formal and *de facto* standards, including ANSI X9 documents, PKIX, SET, S/MIME, and SSL.

Further development of PKCS occurs through mailing list discussions and occasional workshops, and suggestions for improvement are welcome. For more information, contact:

PKCS Editor
RSA Laboratories
174 Middlesex Turnpike
Bedford, MA 01730 USA
http://www.rsasecurity.com/rsalabs/pkcs/