



# PKCS #11 Profiles For Mobile Devices

*RSA Laboratories*

*June 10, 2003*

## Table of Contents

<b>1. INTRODUCTION .....</b>	<b>2</b>
<b>2. TERMS AND DEFINITIONS .....</b>	<b>2</b>
<b>3. COMMON REQUIREMENTS .....</b>	<b>2</b>
3.1 SESSION SUPPORT .....	2
3.2 THREAD HANDLING .....	2
<b>4. SIGNATURE DEVICE PROFILE .....</b>	<b>3</b>
4.1 CRYPTOKI OBJECT CLASSES .....	3
4.2 CRYPTOKI KEY TYPES .....	3
4.3 CRYPTOKI CERTIFICATE TYPES .....	3
4.4 CRYPTOKI ATTRIBUTES .....	3
4.5 CRYPTOKI MECHANISMS .....	4
4.6 CRYPTOKI FUNCTIONS .....	4
<b>5. COMMUNICATION DEVICE PROFILE .....</b>	<b>5</b>
5.1 CRYPTOKI OBJECT CLASSES .....	5
5.2 CRYPTOKI KEY TYPES .....	5
5.3 CRYPTOKI CERTIFICATE TYPES .....	5
5.4 CRYPTOKI ATTRIBUTES .....	5
5.5 CRYPTOKI MECHANISMS .....	6
5.6 CRYPTOKI FUNCTIONS .....	7
5.7 PRIVATE KEY ARCHIVAL .....	7
<b>A. INTELLECTUAL PROPERTY CONSIDERATIONS.....</b>	<b>8</b>
<b>B. REFERENCES.....</b>	<b>8</b>
<b>C. ABOUT PKCS.....</b>	<b>8</b>

## 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	✓ <sup>1</sup>	✓	✓
CKA_ID	✓ <sup>1</sup>	✓	✓
CKA_ISSUER	✓ <sup>1</sup>		
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		✓ <sup>2</sup>	✓ <sup>2</sup>
CKA_MODULUS_BITS		✓ <sup>2</sup>	✓ <sup>2</sup>
CKA_PUBLIC_EXPONENT		✓ <sup>2</sup>	✓ <sup>2</sup>

<sup>1</sup>Only when CKA\_CERTIFICATE\_TYPE is CKC\_X\_509  
<sup>2</sup>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	✓ <sup>1</sup>	✓	✓	
CKA_ID	✓ <sup>1</sup>	✓	✓	✓
CKA_ISSUER	✓ <sup>1</sup>			
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		✓ <sup>2</sup>	✓ <sup>2</sup>	
CKA_MODULUS_BITS		✓ <sup>2</sup>	✓ <sup>2</sup>	
CKA_PUBLIC_EXPONENT		✓ <sup>2</sup>	✓ <sup>2</sup>	
CKA_ENCRYPT			✓	✓
CKA_DECRYPT		✓		✓
CKA_WRAP			✓	✓
CKA_UNWRAP		✓		✓
CKA_DERIVE				✓
CKA_VALUE_LEN				✓

<sup>1</sup>Only when CKA\_CERTIFICATE\_TYPE is CKC\_X\_509  
<sup>2</sup>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\_WrapKey
- C\_UnwrapKey
- C\_SeedRandom
- C\_GenerateRandom
- C\_DeriveKey

Note: C\_Login must support both CKU\_USER and CKU\_SO.

## 5.7 Private key archival

The following text applies only to private keys used for decrypt operations, and not to private keys used for signature operations.

Cryptoki libraries conformant with this profile are *recommended* to support encryption key archival either through wrapping of private encryption keys by C\_WrapKey() or key injection through C\_CreateObject(). In the latter case, when dealing with RSA private keys, the library must also support attribute CKA\_PRIVATE\_EXPONENT, and should support attributes CKA\_PRIME\_1, CKA\_PRIME\_2, CKA\_EXPONENT\_1, CKA\_EXPONENT\_2, and CKA\_COEFFICIENT.

## 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/>