PKCS #11 v2.11 Amendment 1 - Draft 2

RSA Laboratories

Draft – 30 November, 2001

Editor's note: This is the second draft of PKCS #11 v2.11 Amendment 1, which is available for a 30-day public review period. Please send comments and suggestions, both technical and editorial, to pkcs-editor@rsasecurity.com or pkcs-tng@rsasecurity.com.

Table of Contents

1. INTRODUCTION 2

1.1 ABOUT PERSONAL TRUSTED DEVICES 2

2. CHANGES TO SECTION 3, "REFERENCES" 3

3. CHANGES TO SECTION 4, "DEFINITIONS" 4

4. CHANGES TO SECTION 9.4, "OBJECT TYPES" 4

5. CHANGES TO SECTION 9.5, "DATA TYPES FOR MECHANISMS" 4

6. CHANGES TO SECTION 10.3, "HARDWARE FEATURE OBJECTS" 5

7. CHANGES TO SECTION 12, "MECHANISMS" 7

8. OPEN ISSUES 10

8.1 NEW ATTRIBUTE 10

8.2 OTHER ISSUES 10

A. INTELLECTUAL PROPERTY CONSIDERATIONS 11

B. REFERENCES 11

C. ABOUT PKCS 11

1. Introduction

This amendment documents the changes to PKCS #11 v2.11 [4] needed to support:

- Personal Trusted Devices, i.e. devices or tokens capable not only of signing information but also of securely presenting that information to the user; and
- Tokens capable of forming CMS [1] (or PKCS #7 [3]) **SignedAttributes** by themselves.

It does so by defining:

- A new hardware feature object describing the presentation capabilities of the token; and
- A new signature mechanism, which allows the caller to submit the information to be signed, rather than the digest of the information.

1.1 About Personal Trusted Devices

1.1.1 Background

The term "personal trusted device" (PTD) is characterized in [2] as something which "is personal, controlled and used by one person and carried by that person most of the time…has an application platform with associated user interfaces for transaction related services such as banking, payment, bonus programs…[and] has the security functionality required for transaction related services: secure sessions, authentication and authorization." Further, the PTD "contains a Security Element, which is used for protecting its most critical data, such as private keys."

There is also built-in functionality to authenticate the user/owner to the PTD, and to store security-related objects, such as certificates.

A personal trusted device could be a PDA, a mobile phone or some other portable device. The important thing is that its owner can rightfully regard it as a trusted computing base. It is likely that, as mobile commerce evolves, PTDs will be an important enabler of applications that would otherwise not be feasible.

PTDs allow secure signatures to be made both in a personal environment and in more public environments, e.g. web cafés. The changes documented here will, when implemented, allow an application, informed of the fact that a PTD is available to sign the message, to provide the PTD with the message itself, and an indication of *desired* signed attributes. These attributes will then be compared against some configuration in the PTD before being accepted. The PTD may also add attributes of its own before returning the desired signature.

1.1.2 Security aspects

As indicated above, when a token acts as a personal trusted device, it does so thanks to certain characteristics, including:

- a trusted computing base;

- a user interface which is trusted;
- a security element which protects the signature keys; and
- a requirement of explicit user consent before each usage of the signature keys for non-repudiation purposes.

The combination of these characteristics provides users with a device following the "What You See Is What You Sign" paradigm. Users need therefore not trust software in personal computers in order to make signatures on transactions created in those personal computers – the transaction presented by the PTD is what needs to be authorized. While this provides an added level of security and assurance to signers, it does not, as the next sub-section will discuss, necessarily do so for receivers of these signatures.

1.1.3 Use of signature policies

A PTD may well add signed attributes of its own. One such attribute could indicate the particular signature policy it is working under. Another such attribute could identify what parts of a multi-part MIME message that has been presented by the PTD. The definition of these attributes is out of scope for this document.

A certificate-issuing authority may also elect to indicate in issued certificates the policies under which the private key may (or can) be used.

A signature-receiving application cannot in general trust signed attributes – a security element in the PTD may have been removed from the PTD and used in another environment which did not allow the user to view the information on a secure display before signing, for example. Such trust may however be asserted, when a certificate-issuing authority has vouched for the usage policies of the private key through, e.g. a certificate policy identifier or similar, subject to local policies of course.

1.1.4 Authentication of PCs to PTDs

In certain environments or scenarios, there might be a need for the PTD to authenticate the requestor. Since the authentication will be carried out beneath the PKCS #11 interface, it is however out of scope for this document.

2. Changes to Section 3, "References"

[Add the following references, maintaining the alphabetical ordering of references:]

CC/PP Struct W3C. *Composite Capability/Preference Profiles (CC/PP): Structure and Vocabularies*. World Wide Web Consortium, Working Draft 15 March 2001. URL: http://www.w3.org/TR/CCPP-struct-vocab/

MeT-PTD MeT. *MeT PTD Definition – Personal Trusted Device Definition*, Version 1.0, 21 February 2001. URL: http://www.mobiletransaction.org

RFC 2045 Freed, N., and N. Borenstein. *IETF RFC 2045: Multipurpose Internet Mail Extensions (MIME) Part One: Format of Internet Message Bodies*. November 1996. URL: http://ietf.org/rfc/rfc2045.txt

RFC 2534 Masinter, L., Wing, D., Mutz, A., and K. Holtman. *IETF RFC 2534: Media Features for Display, Print, and Fax*. March 1999. URL: http://ietf.org/rfc/rfc2534.txt

RFC 2630 R. Housley. *IETF RFC 2630: Cryptographic Message Syntax*. June 1999. URL: http://ietf.org/rfc/rfc2630.txt.

3. Changes to Section 4, "Definitions"

[Add the following new definitions to PKCS #11 2.11 Section 4, maintaining the alphabetical order of definitions:]

CMS Cryptographic Message Syntax (see RFC 2630)
PTD Personal Trusted Device, as defined in MeT-PTD

4. Changes to Section 9.4, "Object types"

[Add the following hardware feature type definition to the listing under the heading "CK_HW_FEATURE_TYPE":]

```
#define CKH_USER_INTERFACE           0x00000TBA
#define CKH_CMS_ATTRIBUTES           0x00000TBA
```

[Add the following attribute type definitions to the listing under the heading "CK_ATTRIBUTE_TYPE":]

```
#define CKA_PIXEL_X                  0x00000TBA
#define CKA_PIXEL_Y                  0x00000TBA
#define CKA_RESOLUTION               0x00000TBA
#define CKA_CHAR_ROWS                0x00000TBA
#define CKA_CHAR_COLUMNS             0x00000TBA
#define CKA_COLOR                    0x00000TBA
#define CKA_CHAR_SETS                0x00000TBA
#define CKA_ENCODING_METHODS         0x00000TBA
#define CKA_CONTENT_TYPES            0x00000TBA
#define CKA_REQUIRED_CMS_ATTRIBUTES  0x00000TBA
#define CKA_DEFAULT_CMS_ATTRIBUTES   0x00000TBA
#define CKA_SUPPORTED_CMS_ATTRIBUTES 0x00000TBA
```

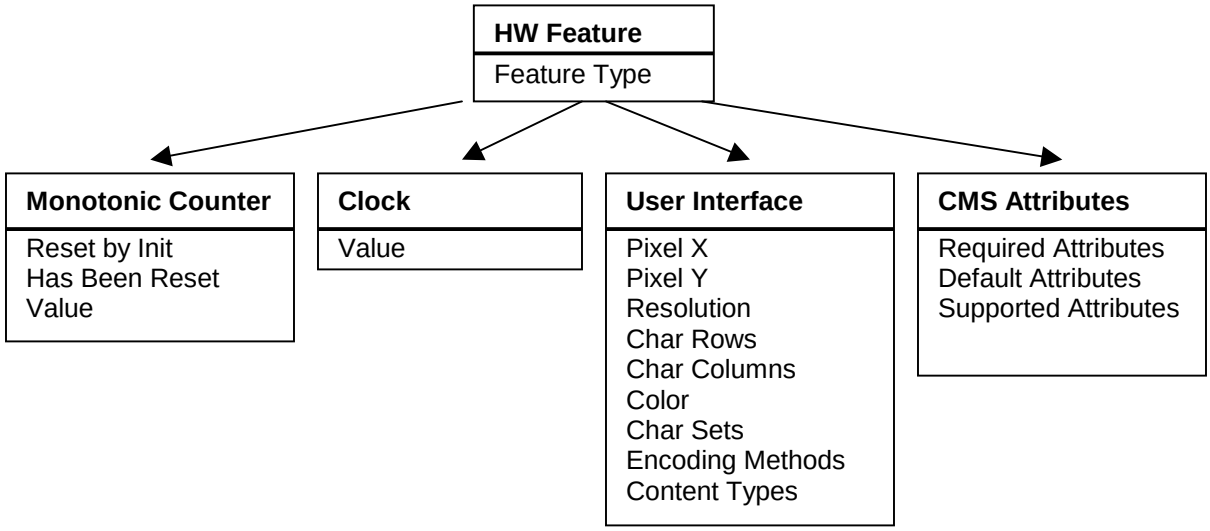
5. Changes to Section 9.5, "Data types for mechanisms"

[Add the following mechanism definition in the listing under the heading "CK_MECHANISM_TYPE; CK_MECHANISM_TYPE_PTR," maintaining the numerical ordering of definitions:]

```
#define CKM_CMS_SIG                  0x00000TBA
```

6. Changes to Section 10.3, "Hardware Feature Objects"

[Replace Figure 6 with the following figure:]



[Replace the paragraph directly after Table 16 with the following paragraph:]

This version of Cryptoki supports the following values for **CKA_HW_FEATURE_TYPE**: **CKH_MONOTONIC_COUNTER**, **CKH_CLOCK**, **CKH_USER_INTERFACE**, and **CKH_CMS_ATTRIBUTES**.

[Add new sub-subsections 10.3.3 and 10.3.4 as follows:]

10.3.3 User Interface Objects

User interface objects represent the presentation capabilities of the device.

Attribute	Data type	Meaning
CKA_PIXEL_X	CK_ULONG	Screen resolution (in pixels) in X-axis (e.g. 1280)
CKA_PIXEL_Y	CK_ULONG	Screen resolution (in pixels) in Y-axis (e.g. 1024)
CKA_RESOLUTION	CK_ULONG	DPI, pixels per inch
CKA_CHAR_ROWS	CK_ULONG	For character-oriented displays; number of character rows (e.g. 24)
CKA_CHAR_COLUMNS	CK_ULONG	For character-oriented displays: number of character columns (e.g. 80). If display is of proportional-font type, this is the width of the display in "em"-s (letter "M"), see CC/PP Struct.
CKA_COLOR	RFC 2279 string	Color support; value shall be in accordance with one of the values defined in RFC 2534, e.g. "`full`"
CKA_CHAR_SETS	RFC 2279 string	List of integers indicating supported character sets, as defined by IANA MIBenum sets (www.iana.org). Supported character sets are separated with ";". E.g. a token supporting iso-8859-1 and us-ascii would set the attribute value to "`4;3`".
CKA_ENCODING_METHODS	RFC 2279 string	String indicating supported content transfer encoding methods, as defined by IANA (www.iana.org). Supported methods are separated with ";". E.g. a token supporting 7bit, 8bit and base64 could set the attribute value to "`7bit;8bit;base64`".
CKA_CONTENT_TYPES	RFC 2279 string	String indicating supported (presentable) content-types; see IANA. Supported content-types are separated with ";". E.g. a token supporting content types a/b, a/c and a/d would set the attribute value to "`a/b;a/c;a/d`".

The selection of attributes, and associated data types, has been done in an attempt to stay as aligned with RFC 2534 and CC/PP Struct as possible. The special value CK_UNAVAILABLE_INFORMATION may be used for CK_ULONG-based attributes when information is not available or applicable.

None of the attribute values may be set by an application.

The value of the **CKA_ENCODING_METHODS** attribute shall be used when the application needs to send MIME objects with encoded content to the token.

10.3.4 CMS Attributes

CMS Attribute objects represent information about supported CMS signature attributes in the token. They are only present on tokens supporting the **CKM_CMS_SIG** mechanism, but must be present on those tokens.

Attribute	Data type	Meaning
CKA_REQUIRED_CMS_ATTRIBUTES	Byte array	Attributes that the token always will include in the set of CMS signed attributes
CKA_DEFAULT_CMS_ATTRIBUTES	Byte array	Attributes that will the token will include in the set of CMS signed attributes in the absence of any attributes specified by the application
CKA_SUPPORTED_CMS_ATTRIBUTES	Byte array	Attributes that the token may include in the set of CMS signed attributes upon request by the application

The contents of each byte array will be a DER-encoded list of CMS **Attributes** with optional accompanying values. Any attributes in the list shall be identified with its object identifier, and any values shall be DER-encoded. The list of attributes is defined in ASN.1 as:

```
Attributes ::= SET SIZE (1..MAX) OF Attribute

Attribute ::= SEQUENCE {
    attrType    OBJECT IDENTIFIER,
    attrValues SET OF ANY DEFINED BY OBJECT IDENTIFIER
}
```

The client may not set any of the attributes.

7. Changes to Section 12, "Mechanisms"

[Add the following entry to Table 63, just after the "CKM_TLS_KEY_AND_MAC_DERIVE" entry, indicating that the new mechanism supports signatures with and without message recovery:]

CKM_CMS_SIG		✓	✓				

[Add new sub-sections 12.44 and 12.45 as follows:]

12.44 CMS mechanism parameters

♦ CK_CMS_SIG_PARAMS, CK_CMS_SIG_PARAMS_PTR

CK_CMS_SIG_PARAMS is a structure that provides the parameters to the **CKM_CMS_SIG** mechanism. It is defined as follows:

```
typedef struct CK_CMS_SIG_PARAMS {
      CK_MECHANISM_PTR      pSigningMechanism;
      CK_MECHANISM_PTR      pDigestMechanism;
      CK_UTF8CHAR_PTR       pContentType;
      CK_BYTE_PTR           pRequestedAttributes;
      CK_ULONG              ulRequestedAttributesLen;
      CK_BYTE_PTR           pRequiredAttributes;
      CK_ULONG_PTR          ulRequiredAttributesLen;
      CK_BYTE_PTR           pSignedAttributes;
      CK_ULONG_PTR          pulSignedAttributesLen
} CK_CMS_SIG_PARAMS;
```

The fields of the structure have the following meanings:

pSigningMechanism	Mechanism to use when signing a constructed CMS **SignedAttributes** value. E.g. **CKM_SHA1_RSA_PKCS**.
pDigestMechanism	Mechanism to use when digesting the data. Value shall be NULL_PTR when the digest mechanism to use follows from the *pSigningMechanism* parameter.
pContentType	NULL-terminated string indicating complete MIME content-type of message to sign. If *pContentType* has the value NULL_PTR, then either *pData* is itself a MIME object or the message shall not be presented. Note that this string shall conform to the syntax specified in RFC 2045, i.e. any parameters needed for correct presentation of the content by the token (such as, for example, a non-default "`charset`") must be present. The token must follow rules and procedures defined in RFC 2045 when presenting the content.
pRequestedAttributes	Pointer to DER-encoded list of CMS **Attributes** the caller requests to be included in the signed attributes. Token may freely ignore this list or modify any supplied values.
ulRequestedAttributesLen	Length in bytes of the value pointed to by *pRequestedAttributes*
pRequiredAttributes	Pointer to DER-encoded list of CMS **Attributes** (with accompanying values) required to be included in the resulting signed attributes. Token must not modify any supplied values. If the token does not support one or more of the attributes, or does not accept provided values, the signature operation will fail. The token will use its own default attributes when signing if both the *pRequestedAttributes* and *pRequiredAttributes* field are set to NULL_PTR.
ulRequiredAttributesLen	Length in bytes, of the value pointed to by *pRequiredAttributes*.
pSignedAttributes	Points to a memory area that receives the resulting DER-encoded CMS **SignedAttributes** value
pulSignedAttributesLen	Pointer to the length, in bytes, of the value of the *pSignedAttributes* area.

12.45 CMS mechanisms

12.45.1 CMS signatures

The CMS mechanism, denoted **CKM_CMS_SIG**, is a multi-purpose mechanism based on the structures defined in PKCS #7 and RFC 2630. It supports single- or multiple-part signatures with and without message recovery. The mechanism is intended for use with, e.g., PTDs (see MeT-PTD) or other capable tokens. The token will construct a CMS (PKCS #7) **SignedAttributes** value and compute a signature on this value. The content of the **SignedAttributes** value is decided by the token, however the caller can suggest some attributes in the parameter *pRequestedAttributes*. The caller can also require some attributes to be present through the parameters *pRequiredAttributes*. The signature is computed in accordance with the parameter *pSigningMechanism*.

When this mechanism is used in successful calls to **C_Sign** or **C_SignFinal**, the *pSignature* return value will point to the resulting signature value, and the *pSignedAttributes* field of the **CK_CMS_SIG_PARAMS** structure provided in the associated **C_SignInit** call will point to the resulting DER-encoded **SignedAttributes**. The mechanism shall not be used in calls to **C_Verify** or **C_VerifyFinal** (use the *pSigningMechanism* mechanism instead).

In order for an application to find out what attributes are supported by a token, what attributes that will be added by default, and what attributes that always will be added, it shall analyze the contents of the **CKH_CMS_ATTRIBUTES** hardware feature object. Applications setting *pSignature* to NULL_PTR in calls to **C_Sign** or **C_SignFinal** will also receive the resulting length of the *pSignedAttributes* value in *pulSignedAttributesLen*.

For the *pRequiredAttributes* field, the token may have to interact with the user to find out whether to accept a proposed value or not. The token should never accept any proposed attribute values without some kind of confirmation from its owner (but this could be through, e.g., configuration or policy settings and not direct interaction).

When possible, applications should always use the **CKM_CMS_SIG** mechanism when generating CMS-compatible signatures rather than lower-level mechanisms such as **CKM_SHA1_RSA_PKCS**. This is especially true when the signatures are to be made on content that the token is able to present to a user. Exceptions may include those cases where the token does not support a particular signing attribute. Note however that the token may refuse usage of a particular signature key unless the content to be signed is known (i.e. the **CKM_CMS_SIG** mechanism is used).

When a token does not have presentation capabilities, the PKCS #11-aware application may avoid sending the whole message to the token by electing to use a suitable signature mechanism (e.g. **CKM_RSA_PKCS**) as the *pSigningMechanism* value in the **CKM_CMS_SIG_PARAMS** structure, and digesting the message itself before passing it to the token.

PKCS #11-aware applications making use of PTDs, should always attempt to provide messages to be signed by the PTD in a format possible for the PTD to present to the user. PTDs that receive multipart MIME-messages for which only certain parts are possible to present may fail the signature operation with a return value of **CKR_DATA_INVALID**,

but may also choose to add a signing attribute indicating which parts of the message that were possible to present.

8. Open Issues

8.1 New attribute

It is proposed that an amendment is made to PKCS #9, defining the following attributes:

- **allegedContentType**

This attribute will convey the content type (*pContentType*) values supplied in the **CK_CMS_SIG_PARAMS** structure in the **C_Sign** call. The purpose is to protect against attacks where an application disguises a message using, for example, a bogus content type or character set. This attribute should always be present when *pContentType* is set in a **CK_CMS_SIG_PARAMS** structure. It need not be present otherwise. Note that the CMS **contentType** attribute is not possible to use directly in this scenario.

- **tokenCapabilities**

This attribute will convey information about the token's capabilities (i.e. basically an enumeration of the hardware feature object defined herein). This is to protect users in case an application tries to take advantage, e.g., of a token's display limitations to conceal information in a message the user is about to sign, for example by using special colors.

8.2 Other issues

If an application calls **C_Sign** with a NULL *pSignature* parameter, and subsequently receives upper bounds for the *pulSignatureLen* and *pulSignedAttributesLen* fields, must it then call **C_SignInit** again with a suitable, non-NULL_PTR value for *pSignedAttributes*? Could one instead let *pSignedAttributes* be a *ppSignedAttributes*, i.e. a pointer to a pointer to signed attributes? This way, an application should be able to update the **pSignedAttributes* value from NULL_PTR to an actual value without having to call **C_SignInit** again, should it not?

Should the CKM_CMS_SIG mechanism be possible to use also in calls to **C_Verify**?

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] R. Housley. *IETF RFC 2630: Cryptographic Message Syntax*. June 1999. URL: http://ietf.org/rfc/rfc2630.txt.

[2] "*MeT PTD Definition – Personal Trusted Device Definition*," Version 1.0, 21 February 2001. URL: http://www.mobiletransaction.org

[3] RSA Laboratories. *PKCS #7 v1.5: Cryptographic Message Syntax Standard*. November 1, 1993. URL: http://www.rsalabs.com/pkcs/

[4] RSA Laboratories. *PKCS #11 v2.11: Cryptographic Token Interface*. June 2001. URL: http://www.rsalabs.com/pkcs/

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
20 Crosby Drive
Bedford, MA 01730 USA
pkcs-editor@rsasecurity.com
http://www.rsasecurity.com/rsalabs/