PKCS #11 Wrapped Template Specification (Draft v2 - 2016-05-10)

Cryptosense

pkcs11@cryptosense.com

Table of Contents

1 Introduction

This document specifies how to serialize secret and private key templates with the goal of wrapping them together with wrapped keys. Templates and keys are wrapped using authenticated encryption mechanisms. The unwrapping function may then know what the values of those attributes were originally, and make appropriate security decisions accordingly, such as preventing a critical attribute from taking a value which would compromise sensitive key values.

This document also specifies which attributes should be wrapped. Some attributes have more critical security impact than others, but we do not attempt to normalise that behaviour here.

This document does not specify how the **C_WrapKey** and **C_UnwrapKey** commands are extended to allow template wrapping, or if other commands are introduced instead. This is left for another document.

1.1 Terminology

The key words “MUST”, “MUST NOT”, “REQUIRED”, “SHALL”, “SHALL NOT”, “SHOULD”, “SHOULD NOT”, “RECOMMENDED”, “MAY”, and “OPTIONAL” in this document are to be interpreted as described in **[RFC2119]**.

1.2 Definitions

For the purposes of this standard, the following definitions apply:

Payload	A pair composed of a key value and a template. This pair is serialized and encrypted when wrapping.

1.4 Normative References

[KMIP]	OASIS, “Key Management Interoperability Protocol Specification Version 1.2”, 19 May 2015. URL: http://docs.oasis-open.org/kmip/spec/v1.2/kmip-spec-v1.2.html
[PKCS11]	OASIS, “PKCS #11 Cryptographic Token Interface Base Specification Version 2.40”, 14 April 2015. URL: http://docs.oasis-open.org/pkcs11/pkcs11-base/v2.40/os/pkcs11-base-v2.40-os.html
[RFC2119]	Bradner, S., “Key words for use in RFCs to Indicate Requirement Levels”, BCP 14, RFC 2119, March 1997. URL: http://www.ietf.org/rfc/rfc2119.txt.

2 Attributes to Wrap

This section lists which attributes MAY be wrapped together with key objects. As the **C_WrapKey** command SHALL only be used to wrap secret or private keys, we only consider attributes which apply to object types **CKO_SECRET_KEY** and **CKO_PRIVATE_KEY**. We also do not consider attributes which define the key value, such as **CKA_VALUE** for AES keys or **CKA_MODULUS** for RSA keys. Indeed, those are already part of the payload. Having them in both the key value and the template parts of the payload would be redundant.

The following attributes MAY be wrapped:

CKA_CLASS
CKA_TOKEN
CKA_PRIVATE

CKA_LABEL
CKA_APPLICATION
CKA_TRUSTED
CKA_CHECK_VALUE
CKA_KEY_TYPE
CKA_ID
CKA_SENSITIVE
CKA_ENCRYPT
CKA_DECRYPT
CKA_WRAP
CKA_UNWRAP
CKA_SIGN
CKA_SIGN_RECOVER
CKA_VERIFY
CKA_VERIFY_RECOVER
CKA_DERIVE
CKA_START_DATE
CKA_END_DATE
CKA_PUBLIC_KEY_INFO
CKA_EXTRACTABLE
CKA_LOCAL
CKA_NEVER_EXTRACTABLE
CKA_ALWAYS_SENSITIVE
CKA_KEY_GEN_MECHANISM
CKA_MODIFIABLE
CKA_COPYABLE
CKA_DESTROYABLE
CKA_ALWAYS_AUTHENTICATE
CKA_WRAP_WITH_TRUSTED
CKA_WRAP_TEMPLATE
CKA_UNWRAP_TEMPLATE
CKA_ALLOWED_MECHANISMS
CKA_VENDOR_DEFINED

3 Unwrapping Behavior

Unwrapping can be used to duplicate a key object. The duplicate may have different attribute values, so unwrapping may thus also be compared to **C_CopyObject** and **C_SetAttributeValue**. This means that the security issues which apply to **C_CopyObject** and **C_SetAttributeValue** also apply to unwrapping. Thanks to wrapped templates, unwrapping implementations may in particular check that changes to attribute values do not induce security breaches.

For instance, Section 4.1.2 of **[PKCS11]** states that **CKA_SENSITIVE** "can be changed from CK_FALSE to CK_TRUE, but not the other way around". If **CKA_SENSITIVE** is part of the payload, unwrapping SHALL check that the new requested value is not CK_FALSE if the payload value is CK_TRUE.

In general, we distinguish the following cases:

- an attribute is only specified in the payload;
- an attribute is only specified in the requested template;
- an attribute is specified both in the payload and in the requested template.

If an attribute is only specified in the payload, it SHOULD be copied as-is. Attributes **CKA_LOCAL**, **CKA_NEVER_EXTRACTABLE** and **CKA_ALWAYS_SENSITIVE** are exceptions to this rule, as those attributes SHALL always be set to CK_FALSE.

If an attribute is only specified in the requested template, the behavior SHALL be equivalent to the behavior of **C_UnwrapKey** when wrapped templates are not used. Particular implementations MAY require that some attributes are always specified in the payload.

If an attribute is specified both in the payload and in the requested template, the result SHALL be equivalent to running **C_UnwrapKey** using the payload template as the requested template, followed by **C_SetAttributeValue** using the requested template as the new template. Attributes such as **CKA_TOKEN** and **CKA_PRIVATE** which are read-only but which may nevertheless change value when copied or unwrapped are an exception to this rule; the requested value takes precedence.

4 Payload Serialization

This section specifies how payloads SHALL be serialized. Payloads are composed of the key value, which is serialized in the same way that **C_WrapKey** serializes it without wrapped templates, and of the wrapped template, which is the list of wrapped attributes and their values.

In order to provide a bridge between PKCS #11 and KMIP, serialization uses the TTLV encoding defined in **[KMIP]**. Below sections follow the format used in **[KMIP]**.

4.1 Tags

This document defines a number of new tags which do not overlap with existing KMIP tags. Each tag corresponds to a field of a structure defined in one of the next sections. Tag values are defined in hexadecimal.

Object	**Tag Value**
PKCS #11 Payload	541100
PKCS #11 Key Value	541101
PKCS #11 Template	541102
PKCS #11 Attribute	541103
PKCS #11 Attribute Type	541104
PKCS #11 Attribute Value	541105
PKCS #11 Mechanism Type	541106

4.2 Payload Structure

The main data structure is the Payload Structure. It is TTLV-encoded and then encrypted. It contains the key value, encoded as a byte string, and the wrapped template. The latter MAY be omitted if there are no attributes to wrap.

Object	Encoding	REQUIRED
PKCS #11 Payload	Structure	
PKCS #11 Key Value	Byte string	Yes
PKCS #11 Template	Structure, see Section 4.3	No

4.3 Template Structure

The wrapped template in the payload is a structure which contains any number of attributes.

Object	Encoding	REQUIRED
PKCS #11 Template	Structure	
PKCS #11 Attribute	Structure, see Section 4.4	No. MAY be repeated

4.4 Attribute Structure

Each attribute is encoded as a structure containing its type (e.g. **CKA_SENSITIVE**) and its value (e.g. **CKA_TRUE**).

Object	Encoding	REQUIRED
PKCS #11 Attribute	Structure	
PKCS #11 Attribute Type	Enumeration, see Section 4.5	Yes
PKCS #11 Attribute Value	Depends, see Section 4.6	Yes

4.5 Attribute Type Enumeration

Attribute types (e.g. **CKA_SENSITIVE**) are encoded as enumerations. The actual value of each attribute (e.g. **CKA_SENSITIVE** is defined as 0x00000103 in pkcs11t.h) is used for the value in the TTLV encoding. In particular, vendor-defined attributes always have their left-most bit set (0x80000000). This is compatible with the “Extensions” case used in KMIP for all enumerations.

4.6 Attribute Values

The type used to encode an attribute value depends on the attribute type. The following table sums up the conversion rules the types of attributes which can be wrapped (see Section 2).

PKCS #11 Type	TTLV Type
CK_BBOOL	Boolean, see Section 4.6.1
CK_OBJECT_CLASS	Enumeration, see Section 4.6.2
CK_KEY_TYPE	Enumeration, see Section 4.6.3
CK_MECHANISM_TYPE	Enumeration, see Section 4.6.4
Byte array	Byte String, see Section 4.6.5
RFC2279 string	Text String, see Section 4.6.6

CK_DATE	Date-Time, see Section 4.6.7
CK_ATTRIBUTE_PTR	Structure, see Section 4.6.8 or Section 4.3
CK_MECHANISM_TYPE_PTR	Structure, see Section 4.6.9
Types for CKA_VENDOR_DEFINED attributes	Vendor-defined

4.6.1 CK_BBOOL

Values of type CK_BBOOL are encoded as Booleans. CK_FALSE (which is defined as 0 in pkcs11t.h) is encoded as 0x0000000000000000 and CK_TRUE (which is defined as 1 in pkcs11t.h) is encoded as 0x0000000000000001.

4.6.2 CK_OBJECT_CLASS

Just like attribute types (see Section 4.5), object classes are encoded as enumerations. The actual value of each object class is used for the value in the TTLV encoding.

4.6.3 CK_KEY_TYPE

Just like attribute types (see Section 4.5), key types are encoded as enumerations. The actual value of each key type is used for the value in the TTLV encoding.

4.6.4 CK_MECHANISM_TYPE

Just like attribute types (see Section 4.5), mechanism types are encoded as enumerations. The actual value of each mechanism type is used for the value in the TTLV encoding.

4.6.5 Byte array

Byte arrays are encoded as Byte Strings. They are left unchanged.

4.6.6 RFC2279 string

RFC2279 strings are encoded as Text Strings. Note that Text Strings SHALL NOT be null-terminated, as the length is given by the TTLV encoding itself.

4.6.7 CK_DATE

Dates are encoded as Date-Time. Date-Times are POSIX time values encoded as Long Integers. As CK_DATE is a structure containing strings, a conversion MUST be made.

4.6.8 CK_ATTRIBUTE_PTR

Attributes **CKA_WRAP_TEMPLATE** and **CKA_UNWRAP_TEMPLATE** have type **CK_ATTRIBUTE_PTR**: their value is a pointer to an array of attributes. They SHALL be encoded as a PKCS #11 Template structure as defined in Section 4.3.

4.6.9 CK_MECHANISM_TYPE_PTR

Attribute **CKA_ALLOWED_MECHANISM** has type **CK_MECHANISM_TYPE_PTR**: its value is a pointer to an array of mechanism types. It SHALL be encoded using the following structure.

Object	Encoding	REQUIRED
PKCS #11 Mechanism Types	Structure	
PKCS #11 Mechanism Type	Enumeration, see Section 4.6.4	No. MAY be repeated

Appendix A. Attribute Encoding

This appendix lists all attributes, their PKCS #11 type and the section of this document which specifies their TTLV encoding if they can be wrapped.

CKA_CLASS	CK_OBJECT_CLASS	4.6.2
CKA_TOKEN	CK_BBOOL	4.6.1
CKA_PRIVATE	CK_BBOOL	4.6.1
CKA_LABEL	RFC2279 string	4.6.6
CKA_APPLICATION	RFC2279 string	4.6.6
CKA_VALUE	Cannot be wrapped	
CKA_OBJECT_ID	Cannot be wrapped	
CKA_CERTIFICATE_TYPE	Cannot be wrapped	
CKA_ISSUER	Cannot be wrapped	
CKA_SERIAL_NUMBER	Cannot be wrapped	
CKA_AC_ISSUER	Cannot be wrapped	
CKA_OWNER	Cannot be wrapped	
CKA_ATTR_TYPES	Cannot be wrapped	
CKA_TRUSTED	CK_BBOOL	4.6.1
CKA_CERTIFICATE_CATEGORY	Cannot be wrapped	
CKA_JAVA_MIDP_SECURITY_DOMAIN	Cannot be wrapped	
CKA_URL	Cannot be wrapped	
CKA_HASH_OF_SUBJECT_PUBLIC_KEY	Cannot be wrapped	
CKA_HASH_OF_ISSUER_PUBLIC_KEY	Cannot be wrapped	
CKA_NAME_HASH_ALGORITHM	Cannot be wrapped	
CKA_CHECK_VALUE	Byte array	4.6.5
CKA_KEY_TYPE	CK_KEY_TYPE	4.6.3
CKA_SUBJECT	Cannot be wrapped	
CKA_ID	Byte array	4.6.5
CKA_SENSITIVE	CK_BBOOL	4.6.1
CKA_ENCRYPT	CK_BBOOL	4.6.1
CKA_DECRYPT	CK_BBOOL	4.6.1
CKA_WRAP	CK_BBOOL	4.6.1
CKA_UNWRAP	CK_BBOOL	4.6.1
CKA_SIGN	CK_BBOOL	4.6.1
CKA_SIGN_RECOVER	CK_BBOOL	4.6.1

CKA_VERIFY	CK_BBOOL	4.6.1
CKA_VERIFY_RECOVER	CK_BBOOL	4.6.1
CKA_DERIVE	CK_BBOOL	4.6.1
CKA_START_DATE	CK_DATE	4.6.7
CKA_END_DATE	CK_DATE	4.6.7
CKA_MODULUS	Cannot be wrapped	
CKA_MODULUS_BITS	Cannot be wrapped	
CKA_PUBLIC_EXPONENT	Cannot be wrapped	
CKA_PRIVATE_EXPONENT	Cannot be wrapped	
CKA_PRIME_1	Cannot be wrapped	
CKA_PRIME_2	Cannot be wrapped	
CKA_EXPONENT_1	Cannot be wrapped	
CKA_EXPONENT_2	Cannot be wrapped	
CKA_COEFFICIENT	Cannot be wrapped	
CKA_PUBLIC_KEY_INFO	Byte array	4.6.5
CKA_PRIME	Cannot be wrapped	
CKA_SUBPRIME	Cannot be wrapped	
CKA_BASE	Cannot be wrapped	
CKA_PRIME_BITS	Cannot be wrapped	
CKA_SUBPRIME_BITS	Cannot be wrapped	
CKA_SUB_PRIME_BITS	Deprecated	
CKA_VALUE_BITS	Cannot be wrapped	
CKA_VALUE_LEN	Cannot be wrapped	
CKA_EXTRACTABLE	CK_BBOOL	4.6.1
CKA_LOCAL	CK_BBOOL	4.6.1
CKA_NEVER_EXTRACTABLE	CK_BBOOL	4.6.1
CKA_ALWAYS_SENSITIVE	CK_BBOOL	4.6.1
CKA_KEY_GEN_MECHANISM	CK_MECHANISM_TYPE	4.6.4
CKA_MODIFIABLE	CK_BBOOL	4.6.1
CKA_COPYABLE	CK_BBOOL	4.6.1
CKA_DESTROYABLE	CK_BBOOL	4.6.1
CKA_ECDSA_PARAMS	Cannot be wrapped	
CKA_EC_PARAMS	Cannot be wrapped	
CKA_EC_POINT	Cannot be wrapped	
CKA_SECONDARY_AUTH	Deprecated	
CKA_AUTH_PIN_FLAGS	Deprecated	
CKA_ALWAYS_AUTHENTICATE	CK_BBOOL	4.6.1
CKA_WRAP_WITH_TRUSTED	CK_BBOOL	4.6.1
CKA_WRAP_TEMPLATE	CK_ATTRIBUTE_PTR	4.6.8, 4.3
CKA_UNWRAP_TEMPLATE	CK_ATTRIBUTE_PTR	4.6.8, 4.3
CKA_OTP_FORMAT	Cannot be wrapped	

CKA_OTP_LENGTH	Cannot be wrapped	
CKA_OTP_TIME_INTERVAL	Cannot be wrapped	
CKA_OTP_USER_FRIENDLY_MODE	Cannot be wrapped	
CKA_OTP_CHALLENGE_REQUIREMENT	Cannot be wrapped	
CKA_OTP_TIME_REQUIREMENT	Cannot be wrapped	
CKA_OTP_COUNTER_REQUIREMENT	Cannot be wrapped	
CKA_OTP_PIN_REQUIREMENT	Cannot be wrapped	
CKA_OTP_COUNTER	Cannot be wrapped	
CKA_OTP_TIME	Cannot be wrapped	
CKA_OTP_USER_IDENTIFIER	Cannot be wrapped	
CKA_OTP_SERVICE_IDENTIFIER	Cannot be wrapped	
CKA_OTP_SERVICE_LOGO	Cannot be wrapped	
CKA_OTP_SERVICE_LOGO_TYPE	Cannot be wrapped	
CKA_GOSTR3410_PARAMS	Cannot be wrapped	
CKA_GOSTR3411_PARAMS	Cannot be wrapped	
CKA_GOST28147_PARAMS	Cannot be wrapped	
CKA_HW_FEATURE_TYPE	Cannot be wrapped	
CKA_RESET_ON_INIT	Cannot be wrapped	
CKA_HAS_RESET	Cannot be wrapped	
CKA_PIXEL_X	Cannot be wrapped	
CKA_PIXEL_Y	Cannot be wrapped	
CKA_RESOLUTION	Cannot be wrapped	
CKA_CHAR_ROWS	Cannot be wrapped	
CKA_CHAR_COLUMNS	Cannot be wrapped	
CKA_COLOR	Cannot be wrapped	
CKA_BITS_PER_PIXEL	Cannot be wrapped	
CKA_CHAR_SETS	Cannot be wrapped	
CKA_ENCODING_METHODS	Cannot be wrapped	
CKA_MIME_TYPES	Cannot be wrapped	
CKA_MECHANISM_TYPE	Cannot be wrapped	
CKA_REQUIRED_CMS_ATTRIBUTES	Cannot be wrapped	
CKA_DEFAULT_CMS_ATTRIBUTES	Cannot be wrapped	
CKA_SUPPORTED_CMS_ATTRIBUTES	Cannot be wrapped	
CKA_ALLOWED_MECHANISMS	CK_MECHANISM_TYPE_PTR	4.6.9
CKA_VENDOR_DEFINED	Vendor-defined	