## Public key objects

Public key objects (object class **CKO\_PUBLIC\_KEY**) hold public keys. The following table defines the attributes common to all public keys, in addition to the common attributes defined for this object class:

Table , Common Public Key Attributes

| **Attribute** | **Data type** | **Meaning** |
| --- | --- | --- |
| CKA\_SUBJECT8 | Byte array | DER-encoding of the key subject name (default empty) |
| CKA\_ENCRYPT8 | CK\_BBOOL | CK\_TRUE if key supports encryption9 |
| CKA\_VERIFY8 | CK\_BBOOL | CK\_TRUE if key supports verification where the signature is an appendix to the data9 |
| CKA\_VERIFY\_RECOVER8 | CK\_BBOOL | CK\_TRUE if key supports verification where the data is recovered from the signature9 |
| CKA\_WRAP8 | CK\_BBOOL | CK\_TRUE if key supports wrapping (*i.e.*, can be used to wrap other keys)9 |
| CKA\_TRUSTED10 | CK\_BBOOL | The key can be trusted for the application that it was created.  The wrapping key can be used to wrap keys with CKA\_WRAP\_WITH\_TRUSTED set to CK\_TRUE. |
| CKA\_WRAP\_TEMPLATE | CK\_ATTRIBUTE\_PTR | For wrapping keys. The attribute template to match against any keys wrapped using this wrapping key. Keys that do not match cannot be wrapped. The number of attributes in the array is the *ulValueLen* component of the attribute divided by the size of CK\_ATTRIBUTE. |
| CKA\_PUBLIC\_KEY\_INFO2,4,6 | Byte array | DER-encoding of the SubjectPublicKeyInfo for this public key. (MAY be empty, DEFAULT derived from the underlying public key data) |

- Refer to table Table 15 for footnotes

It is intended in the interests of interoperability that the subject name and key identifier for a public key will be the same as those for the corresponding certificate and private key. However, Cryptoki does not enforce this, and it is not required that the certificate and private key also be stored on the token.

To map between ISO/IEC 9594-8 (X.509) **keyUsage** flags for public keys and the PKCS #11 attributes for public keys, use the following table.

Table , Mapping of X.509 key usage flags to cryptoki attributes for public keys

|  |  |
| --- | --- |
| **Key usage flags for public keys in X.509 public key certificates** | **Corresponding cryptoki attributes for public keys.** |
| dataEncipherment | CKA\_ENCRYPT |
| digitalSignature, keyCertSign, cRLSign | CKA\_VERIFY |
| digitalSignature, keyCertSign, cRLSign | CKA\_VERIFY\_RECOVER |
| keyAgreement | CKA\_DERIVE |
| keyEncipherment | CKA\_WRAP |
| nonRepudiation | CKA\_VERIFY |
| nonRepudiation | CKA\_VERIFY\_RECOVER |

The value of the CKA\_PUBLIC\_KEY\_INFO attribute is the DER encoded value of SubjectPublicKeyInfo:

SubjectPublicKeyInfo ::= SEQUENCE {

algorithm AlgorithmIdentifier,

subjectPublicKey BIT STRING }

The encodings for the subjectPublicKey field are specified in the descriptions of the public key types in the appropriate [Mechanisms] document for the key types defined within this specification.

## Private key objects

Private key objects (object class **CKO\_PRIVATE\_KEY**) hold private keys. The following table defines the attributes common to all private keys, in addition to the common attributes defined for this object class:

Table , Common Private Key Attributes

| **Attribute** | **Data type** | **Meaning** |
| --- | --- | --- |
| CKA\_SUBJECT8 | Byte array | DER-encoding of certificate subject name (default empty) |
| CKA\_SENSITIVE8,11 | CK\_BBOOL | CK\_TRUE if key is sensitive9 |
| CKA\_DECRYPT8 | CK\_BBOOL | CK\_TRUE if key supports decryption9 |
| CKA\_SIGN8 | CK\_BBOOL | CK\_TRUE if key supports signatures where the signature is an appendix to the data9 |
| CKA\_SIGN\_RECOVER8 | CK\_BBOOL | CK\_TRUE if key supports signatures where the data can be recovered from the signature9 |
| CKA\_UNWRAP8 | CK\_BBOOL | CK\_TRUE if key supports unwrapping (*i.e.*, can be used to unwrap other keys)9 |
| CKA\_EXTRACTABLE8,12 | CK\_BBOOL | CK\_TRUE if key is extractable and can be wrapped 9 |
| CKA\_ALWAYS\_SENSITIVE2,4,6 | CK\_BBOOL | CK\_TRUE if key has *always* had the CKA\_SENSITIVE attribute set to CK\_TRUE |
| CKA\_NEVER\_EXTRACTABLE2,4,6 | CK\_BBOOL | CK\_TRUE if key has *never* had the CKA\_EXTRACTABLE attribute set to CK\_TRUE |
| CKA\_WRAP\_WITH\_TRUSTED11 | CK\_BBOOL | CK\_TRUE if the key can only be wrapped with a wrapping key that has CKA\_TRUSTED set to CK\_TRUE.  Default is CK\_FALSE. |
| CKA\_UNWRAP\_TEMPLATE | CK\_ATTRIBUTE\_PTR | For wrapping keys. The attribute template to apply to any keys unwrapped using this wrapping key. Any user supplied template is applied after this template as if the object has already been created. The number of attributes in the array is the *ulValueLen* component of the attribute divided by the size of  CK\_ATTRIBUTE. |
| CKA\_ALWAYS\_AUTHENTICATE | CK\_BBOOL | If CK\_TRUE, the user has to supply the PIN for each use (sign or decrypt) with the key. Default is CK\_FALSE. |
| CKA\_PUBLIC\_KEY\_INFO8 | Byte Array | DER-encoding of the SubjectPublicKeyInfo for the associated public key (MAY be empty, DEFAULT derived from the underlying private key data, MAY be manually set for specific key types, SHOULD be consistent with the private key) |

- Refer to table Table 15 for footnotes

It is intended in the interests of interoperability that the subject name and key identifier for a private key will be the same as those for the corresponding certificate and public key. However, this is not enforced by Cryptoki, and it is not required that the certificate and public key also be stored on the token.

If the **CKA\_SENSITIVE** attribute is CK\_TRUE, or if the **CKA\_EXTRACTABLE** attribute is CK\_FALSE, then certain attributes of the private key cannot be revealed in plaintext outside the token. Which attributes these are is specified for each type of private key in the attribute table in the section describing that type of key.

The **CKA\_ALWAYS\_AUTHENTICATE** attribute can be used to force re-authentication (i.e. force the user to provide a PIN) for each use of a private key. “Use” in this case means a cryptographic operation such as sign or decrypt. This attribute may only be set to CK\_TRUE when **CKA\_PRIVATE** is also CK\_TRUE.

Re-authentication occurs by calling **C\_Login** with *userType* set to **CKU\_CONTEXT\_SPECIFIC** immediately after a cryptographic operation using the key has been initiated (e.g. after **C\_SignInit**). In this call, the actual user type is implicitly given by the usage requirements of the active key. If **C\_Login** returns CKR\_OK the user was successfully authenticated and this sets the active key in an authenticated state that lasts until the cryptographic operation has successfully or unsuccessfully been completed (e.g. by **C\_Sign**, **C\_SignFinal**,..). A return value CKR\_PIN\_INCORRECT from **C\_Login** means that the user was denied permission to use the key and continuing the cryptographic operation will result in a behavior as if **C\_Login** had not been called. In both of these cases the session state will remain the same, however repeated failed re-authentication attempts may cause the PIN to be locked. **C\_Login** returns in this case CKR\_PIN\_LOCKED and this also logs the user out from the token. Failing or omitting to re-authenticate when CKA\_ALWAYS\_AUTHENTICATE is set to CK\_TRUE will result in CKR\_USER\_NOT\_LOGGED\_IN to be returned from calls using the key. **C\_Login** will return CKR\_OPERATION\_NOT\_INITIALIZED, but the active cryptographic operation will not be affected, if an attempt is made to re-authenticate when CKA\_ALWAYS\_AUTHENTICATE is set to CK\_FALSE.

The **CKA\_PUBLIC\_KEY\_INFO** attribute represents the public key associated with this private key. The data it represents may either be stored as part of the private key data, or regenerated as needed from the private key. If this attribute is supplied as part of a call to **C\_CreateObject** or C**\_SetAttributeValue** for a private key, the token SHOULD verify correspondence between the private key and the public key as supplied in **CKA\_PUBLIC\_KEY\_INFO**. If there is a mismatch, **C\_CreateObject** shall return **CKR\_TEMPLATE\_INCONSISTENT.** The client is encouraged to do its own verification of correspondence between this attribute and the associated private key.

As a general guideline, private keys of any type SHOULD store sufficient information to retrieve the public key information. In particular, the RSA private key description has been modified in <this version> to add the CKA\_PUBLIC\_EXPONENT to the list of attributes required for an RSA private key. All other private key types described in this specification contain sufficient information to recover the associated public key.

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

## Certificate objects

### Definitions

This section defines the object class CKO\_CERTIFICATE for type CK\_OBJECT\_CLASS as used in the CKA\_CLASS attribute of objects.

### Overview

Certificate objects (object class **CKO\_CERTIFICATE**) hold public-key or attribute certificates. Other than providing access to certificate objects, Cryptoki does not attach any special meaning to certificates. The following table defines the common certificate object attributes, in addition to the common attributes defined for this object class:

Table , Common Certificate Object Attributes

| **Attribute** | **Data type** | **Meaning** |
| --- | --- | --- |
| CKA\_CERTIFICATE\_TYPE1 | CK\_CERTIFICATE\_TYPE | Type of certificate |
| CKA\_TRUSTED10 | CK\_BBOOL | The certificate can be trusted for the application that it was created. |
| CKA\_CERTIFICATE\_CATEGORY | CK\_ULONG | Categorization of the certificate: 0 = unspecified (default value), 1 = token user, 2 = authority, 3 = other entity |
| CKA\_CHECK\_VALUE | Byte array | Checksum |
| CKA\_START\_DATE | CK\_DATE | Start date for the certificate (default empty) |
| CKA\_END\_DATE | CK\_DATE | End date for the certificate (default empty) |
| CKA\_PUBLIC\_KEY\_INFO | Byte array | DER-encoding of the SubjectPublicKeyInfo for the public key contained in this certificate (default empty) |

- Refer to table Table 15 for footnotes

Cryptoki does not enforce the relationship of the CKA\_PUBLIC\_KEY\_INFO to the public key in the certificate, but does recommend that the key be extracted from the certificate to create this value.

### RSA private key objects

RSA private key objects (object class **CKO\_PRIVATE\_KEY,** key type **CKK\_RSA**) hold RSA private keys. The following table defines the RSA private key object attributes, in addition to the common attributes defined for this object class:

Table 3, RSA Private Key Object Attributes

| **Attribute** | **Data type** | **Meaning** |
| --- | --- | --- |
| CKA\_MODULUS1,4,6 | Big integer | Modulus *n* |
| CKA\_PUBLIC\_EXPONENT1,4,6 | Big integer | Public exponent *e* |
| CKA\_PRIVATE\_EXPONENT1,4,6,7 | Big integer | Private exponent *d* |
| CKA\_PRIME\_14,6,7 | Big integer | Prime *p* |
| CKA\_PRIME\_24,6,7 | Big integer | Prime *q* |
| CKA\_EXPONENT\_14,6,7 | Big integer | Private exponent *d* modulo *p*-1 |
| CKA\_EXPONENT\_24,6,7 | Big integer | Private exponent *d* modulo *q*-1 |
| CKA\_COEFFICIENT4,6,7 | Big integer | CRT coefficient *q*-1 mod *p* |

- Refer to [PKCS #11-B] table 15 for footnotes

Depending on the token, there may be limits on the length of the key components. See PKCS #1 for more information on RSA keys.

Tokens vary in what they actually store for RSA private keys. Some tokens store all of the above attributes, which can assist in performing rapid RSA computations. Other tokens might store only the **CKA\_MODULUS** and **CKA\_PRIVATE\_EXPONENT** values. Effective with version <this version>, tokens MUST also store CKA\_PUBLIC\_EXPONENT. This permits the retrieval of sufficient data to reconstitute the associated public key.

Because of this, Cryptoki is flexible in dealing with RSA private key objects. When a token generates an RSA private key, it stores whichever of the fields in Table 3 it keeps track of. Later, if an application asks for the values of the key’s various attributes, Cryptoki supplies values only for attributes whose values it can obtain (*i.e.*, if Cryptoki is asked for the value of an attribute it cannot obtain, the request fails). Note that a Cryptoki implementation may or may not be able and/or willing to supply various attributes of RSA private keys which are not actually stored on the token. *E.g.*, if a particular token stores values only for the **CKA\_PRIVATE\_EXPONENT, CKA\_PUBLIC\_EXPONENT**, **CKA\_PRIME\_1**, and **CKA\_PRIME\_2** attributes, then Cryptoki is certainly *able* to report values for all the attributes above (since they can all be computed efficiently from these three values). However, a Cryptoki implementation may or may not actually do this extra computation. The only attributes from Table 3 for which a Cryptoki implementation is *required* to be able to return values are **CKA\_MODULUS, CKA\_PRIVATE\_EXPONENT, and CKA\_PUBLIC\_EXPONENT**. A token SHOULD also be able to return **CKA\_PUBLIC\_KEY\_INFO** for an RSA private key. See the general guidance for Private Keys above.