# Key Certification Mechanism

CKM\_CERTIFY\_KEY

This is a signing only mechanism. It may be used by any private key to certify the CKA\_PUBLIC\_KEY\_INFO (SubjectPublicKeyInfo) value of any other private key on the same token. When used with a Token Identity private key (see [CKA\_GLOBAL section]), this can be used to certify that a key exists on a specific token, is not exportable, was generated on the token, etc. If a key has this mechanism as a permitted mechanism, it MUST NOT have any other signing mechanism as a permitted mechanism.

This mechanism uses the following mechanism parameter structure:

typedef struct CK\_CERTIFY\_KEY\_PARAMS {  
 CK\_OBJECT\_HANDLE hKeyToBeSigned;  
 CK\_MECHANISM\_PTR pSigningMechanism;  
 CK\_ULONG ulAttributeCount;  
 CK\_ATTTIBUTE\_TYPE[] attributesToCertify;  
}

A call to C\_SignUpdate is optional, but allows additional information (e.g. a challenge) to be included under the signature.

hHeyToBeSigned - a handle to the private key to be certified.

pSigningMechanism a pointer to the mechanism used to create the signature for the certification information. This should be any signature mechanism compatible with the signing key.

ulAttributeCount the number of attributes in the attributesToCertify array

attributesToCertify the specific attributes for the private key that shall be included under the certification. They may be specified in any order, and will be included in the signature in that order.

The data to be signed consists of the concatenation of:

* The DER encoded SubjectPublicKeyInfo for the private key being signed.
* The attribute count expressed as a 32 bit Network Byte Order (NBO) quantity.
* For each attribute in the order specified in the mechanism:
  + The attribute type as a 32 bit NBO
  + The size of the attribute in bytes as a 32 bit NBO
  + The encoded bytes of the attribute. If the value is an integer quantity, it must be expressed as an NBO for the purposes of the signature.
* Any data provided by C\_SignUpdate

If any attribute specified in attributesToCertify does not exist, the attribute shall be included the signature calculation, but with a length of 0 and no encoded bytes.

Note that verifications will not use this mechanism, as the key to be verified will not exist on the token doing the verification. The verifier will need to be provided all of the data being signed, as well as the public key of the signer to perform its verification.

CKA\_LOCAL should generally be one of the attributes included for certification. If this attribute is FALSE, the recipient should not trust that value of SubjectPublicKeyInfo has a valid cryptographic relationship with the private key being certified. I.e., it should not trust that the public key represented by the info in the SPKI blob is in the same key pair as the private key.

Other attributes of interest may include CKA\_EXTRACTABLE, CKA\_NEVER\_EXTRACTABLE, CKA\_SENSITIVE, CKA\_ALWAYS\_SENSITIVE, and CKA\_COPYABLE. These, along with CKA\_LOCAL and knowledge about the specific type of security module can be used to determine whether the sensitive information of the private key being certified has been or can ever be seen outside the module.

C\_SignUpdate can be used to inject additional data into the certification process. A possible use of this function is for the party who will rely on the certification to provide a random nonce to the party certifying the key.

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

Questions: Should the array attributes (e.g. wrap and unwrap template) be allowed here? If so, what is the encoding for the purposes of generating a signature?