This proposal updates PKCS#11 V2.40 AES Key Wrap specifications as discussed during the TC PKCS#11 F2F meeting held in February 2017. More precisely, it

* Clarifies that CKM\_AES\_KEY\_WRAP only accepts input data which length is a multiple of 8 bytes;
* Replaces “usual padding” by “PKCS#7 padding” for CKM\_AES\_KEY\_WRAP\_PAD; in the final version of PKCS#11 3.00 standard, definitions for CKM\_AES\_KEY\_WRAP\_PAD must still be moved from section 2.14 into the “Historical” part of the PKCS#11 specification.
* Introduces CKM\_AES\_KEY\_WRAP\_KWP for wrapping acc. SP800-38F section 6.3. As a consequence, sections RSA AES KEY WRAP and ECDH AES KEY WRAP must not anymore refer to CKM\_AES\_KEY\_WRAP\_PAD but to CKM\_AES\_KEY\_WRAP\_KWP, to produce results suitable for e.g. RFC 6033, and compatible with RFC 5649. Another consequence is that section 2.14.2 must allow for 8 byte IV or 4 byte IV, depending on the key wrap mechanism to be used.

In addition,

* The reference to the AES Key Wrap specification has been updated to NIST SP800-38F, making direct references to RFC 3394 and RFC 5649 obsolete. RFC 3394 is covered by [AES KEYWRAP] section 6.3, RFC 5649 by [AES KEYWRAP] section 6.3.
* Table 65, AES Key Wrap Mechanisms, has been updated to consistently allow the CKM\_AES\_... mechanisms for encrypt/decrypt and wrap/unwrap.

## 1.4 Non-Normative References

**[AES KEYWRAP]** National Institute of Standards and Technology, NIST Special Publication 800-38F, Recommendation for Block Cipher Modes of Operation: Methods for Key Wrapping, December 2012, <http://nvlpubs.nist.gov/nistpubs/SpecialPublications/NIST.SP.800-38F.pdf>

### 2.1.21 RSA AES KEY WRAP

The RSA AES key wrap mechanism, denoted **CKM\_RSA\_AES\_KEY\_WRAP** , is a mechanism based on the RSA public-key cryptosystem and the AES key wrap mechanism. It supports single-part key wrapping; and key unwrapping.

It has a parameter, a **CK\_RSA\_AES\_KEY\_WRAP\_PARAMS** structure.

The mechanism can wrap and unwrap a target asymmetric key of any length and type using an RSA key.

* A temporary AES key is used for wrapping the target key using CKM\_AES\_KEY\_WRAP\_KWP mechanism.
* The temporary AES key is wrapped with the wrapping RSA key using CKM\_RSA\_PKCS\_OAEP mechanism.

For wrapping, the mechanism -

* Generates a temporary random AES key of *ulAESKeyBits* length. This key is not accessible to the user - no handle is returned.
* Wraps the AES key with the wrapping RSA key using **CKM\_RSA\_PKCS\_OAEP** with parameters of *OAEPParams*.
* Wraps the target key with the temporary AES key using **CKM\_AES\_KEY\_WRAP\_KWP** ([AES KEYWRAP] section 6.3) .
* Zeroizes the temporary AES key.
* Concatenates two wrapped keys and outputs the concatenated blob. The first is the wrapped AES key, and the second is the wrapped target key.

The recommended format for an asymmetric target key being wrapped is as a PKCS8 PrivateKeyInfo

The use of Attributes in the PrivateKeyInfo structure is OPTIONAL. In case of conflicts between the object attribute template, and Attributes in the PrivateKeyInfo structure, an error should be thrown

For unwrapping, the mechanism -

* Splits the input into two parts. The first is the wrapped AES key, and the second is the wrapped target key. The length of the first part is equal to the length of the unwrapping RSA key.
* Un-wraps the temporary AES key from the first part with the private RSA key using **CKM\_RSA\_PKCS\_OAEP** with parameters of *OAEPParams*.
* Un-wraps the target key from the second part with the temporary AES key using **CKM\_AES\_KEY\_WRAP\_KWP** ([AES KEYWRAP] section 6.3) .
* Zeroizes the temporary AES key.
* Returns the handle to the newly unwrapped target key.

*Table 17, CKM\_RSA\_AES\_KEY\_WRAP Mechanisms vs. Functions*

|  |  |
| --- | --- |
|  | **Functions** |
| **Mechanism** | **Encrypt****&****Decrypt** | **Sign****&****Verify** | **SR****&****VR**1 | **Digest** | **Gen.****Key/****Key****Pair** | **Wrap****&****Unwrap** | **Derive** |
|  CKM\_RSA\_AES\_KEY\_WRAP |  |  |  |  |  |  |  |
| 1SR = SignRecover, VR = VerifyRecover |

### 2.3.12. ECDH AES KEY WRAP

The ECDH AES KEY WRAP mechanism, denoted **CKM\_ECDH\_AES\_KEY\_WRAP**, is a mechanism based on elliptic curve public-key crypto-system and the AES key wrap mechanism. It supports single-part key wrapping; and key unwrapping.

It has a parameter, a **CK\_ECDH\_AES\_KEY\_WRAP\_PARAMS** structure.

The mechanism can wrap and unwrap an asymmetric target key of any length and type using an EC key.

* A temporary AES key is derived from a temporary EC key and the wrapping EC key using the **CKM\_ECDH1\_DERIVE** mechanism.
* The derived AES key is used for wrapping the target key using the **CKM\_AES\_KEY\_WRAP\_KWP** mechanism.

For wrapping, the mechanism -

* Generates a temporary random EC key (transport key) having the same parameters as the wrapping EC key (and domain parameters). Saves the transport key public key material.
* Performs ECDH operation using **CKM\_ECDH1\_DERIVE** with parameters of kdf, ulSharedDataLen and pSharedData using the private key of the transport EC key and the public key of wrapping EC key and gets the first ulAESKeyBits bits of the derived key to be the temporary AES key.
* Wraps the target key with the temporary AES key using **CKM\_AES\_KEY\_WRAP\_KWP (**[AES KEYWRAP] section 6.3).
* Zeroizes the temporary AES key and EC transport private key.
* Concatenates public key material of the transport key and output the concatenated blob. The first part is the public key material of the transport key and the second part is the wrapped target key.

The recommended format for an asymmetric target key being wrapped is as a PKCS8 PrivateKeyInfo

The use of Attributes in the PrivateKeyInfo structure is OPTIONAL. In case of conflicts between the object attribute template, and Attributes in the PrivateKeyInfo structure, an error should be thrown.

For unwrapping, the mechanism -

* Splits the input into two parts. The first part is the public key material of the transport key and the second part is the wrapped target key. The length of the first part is equal to the length of the public key material of the unwrapping EC key.

*Note: since the transport key and the wrapping EC key share the same domain, the length of the public key material of the transport key is the same length of the public key material of the unwrapping EC key.*

* Performs ECDH operation using **CKM\_ECDH1\_DERIVE** with parameters of kdf, ulSharedDataLen and pSharedData using the private part of unwrapping EC key and the public part of the transport EC key and gets first ulAESKeyBits bits of the derived key to be the temporary AES key.
* Un-wraps the target key from the second part with the temporary AES key using **CKM\_AES\_KEY\_WRAP\_KWP** **(**[AES KEYWRAP] section 6.3).
* Zeroizes the temporary AES key.

*Table 35, CKM\_ECDH\_AES\_KEY\_WRAP Mechanisms vs. Functions*

|  |  |
| --- | --- |
|  | **Functions** |
| **Mechanism** | **Encrypt****&****Decrypt** | **Sign****&****Verify** | **SR****&****VR**1 | **Digest** | **Gen.****Key/****Key****Pair** | **Wrap****&****Unwrap** | **Derive** |
| CKM\_ECDH\_AES\_KEY\_WRAP |  |  |  |  |  | ✓ |  |
| 1SR = SignRecover, VR = VerifyRecover |

## 2.14. AES Key Wrap

*Table 65, AES Key Wrap Mechanisms vs. Functions*

|  |  |
| --- | --- |
|  | **Functions** |
| **Mechanism** | **Encrypt****&****Decrypt** | **Sign****&****Verify** | **SR****&****VR**1 | **Digest** | **Gen.** **Key/****Key****Pair** | **Wrap****&****Unwrap** | **Derive** |
| CKM\_AES\_KEY\_WRAP | ✓ |  |  |  |  | ✓ |  |
| CKM\_AES\_KEY\_WRAP\_PAD  | ✓ |  |  |  |  | ✓ |  |
| CKM\_AES\_KEY\_WRAP\_KWP  | ✓ |  |  |  |  | ✓ |  |
| 1SR = SignRecover, VR = VerifyRecover |

### 2.14.1 Definitions

Mechanisms:

CKM\_AES\_KEY\_WRAP

CKM\_AES\_KEY\_WRAP\_PAD

CKM\_AES\_KEY\_WRAP\_KWP

### 2.14.2 AES Key Wrap Mechanism parameters

The mechanisms will accept an optional mechanism parameter as the Initialization vector which, if present, must be a fixed size array of 8 bytes for CKM\_AES\_KEY\_WRAP and CKM\_AES\_KEY\_WRAP\_PAD, resp. 4 bytes for CKM\_AES\_KEY\_WRAP\_KWP; and, if NULL, will use the default initial value defined in Section 4.3 resp. 6.2 / 6.3 of [AES KEYWRAP].

The type of this parameter is CK\_BYTE\_PTR and the pointer points to the array of bytes to be used as the initial value. The length shall be either 0 and the pointer NULL; or 8 for CKM\_AES\_KEY\_WRAP / CKM\_AES\_KEY\_WRAP\_PAD, resp. 4 for CKM\_AES\_KEY\_WRAP\_KWP, and the pointer non-NULL.

### 2.14.3 AES Key Wrap

The mechanisms support only single-part operations, single part wrapping and unwrapping, and single-part encryption and decryption.

The CKM\_AES\_KEY\_WRAP mechanism can only wrap a key resp. encrypt a block of data whose size is an exact multiple of the AES Key Wrap algorithm block size. Wrapping / encryption is done as defined in Section 6.2 of [AES KEYWRAP].

The CKM\_AES\_KEY\_WRAP\_PAD mechanism can wrap a key or encrypt a block of data of any length. It does the padding detailed in PKCS #7 of inputs (keys or data blocks), always producing wrapped output that is larger than the input key/data to be wrapped. This padding is done by the token before being passed to the AES key wrap algorithm, which then wraps / encrypts the padded block of data as defined in Section 6.2 of [AES KEYWRAP].

The CKM\_AES\_KEY\_WRAP\_KWP mechanism can wrap a key or encrypt block of data of any length. It does the padding of inputs that are not multiples of the AES Key Wrap algorithm block size, and then wraps / encrypts the padded block of data as defined in Section 6.3 of [AES KEYWRAP], which produces same results as RFC 5649.