## 2.8 AES

For the Advanced Encryption Standard (AES) see [FIPS PUB 197].

| Mechanism | Functions | | | | | | |
|---|---|---|---|---|---|---|---|
| | Encrypt & Decrypt | Sign & Verify | SR & VR[1] | Digest | Gen. Key/ Key Pair | Wrap & Unwrap | Derive |
| CKM_AES_KEY_GEN | | | | | ✓ | | |
| CKM_AES_ECB | ✓ | | | | | ✓ | |
| CKM_AES_CBC | ✓ | | | | | ✓ | |
| CKM_AES_CBC_PAD | ✓ | | | | | ✓ | |
| CKM_AES_MAC_GENERAL | | ✓ | | | | | |
| CKM_AES_MAC | | ✓ | | | | | |
| CKM_AES_OFB | ✓ | | | | | ✓ | |
| CKM_AES_CFB64 | ✓ | | | | | ✓ | |
| CKM_AES_CFB8 | ✓ | | | | | ✓ | |
| CKM_AES_CFB128 | ✓ | | | | | ✓ | |
| CKM_AES_XTS | ✓ | | | | | ✓ | |

### 2.8.1 Definitions

This section defines the key type “CKK_AES” for type CK_KEY_TYPE as used in the CKA_KEY_TYPE attribute of key objects.
Mechanisms:

- CKM_AES_KEY_GEN
- CKM_AES_ECB
- CKM_AES_CBC
- CKM_AES_MAC
- CKM_AES_MAC_GENERAL
- CKM_AES_CBC_PAD
- CKM_AES_OFB
- CKM_AES_CFB64
- CKM_AES_CFB8
- CKM_AES_CFB128
- CKM_AES_XTS

### 2.8.2 AES secret key objects

AES secret key objects (object class **CKO_SECRET_KEY,** key type **CKK_AES**) hold AES keys. The following table defines the AES secret key object attributes, in addition to the common attributes defined for this object class:

*Table 1, AES Secret Key Object Attributes*

| Attribute | Data type | Meaning |
|---|---|---|
| CKA_VALUE[1,4,6,7] | Byte array | Key value (16, 24, or 32 bytes) |
| CKA_VALUE_LEN[2,3,6] | CK_ULON | Length in bytes of key |

| Attribute | Data type | Meaning |
|---|---|---|
| | G | value |

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

The following is a sample template for creating an AES secret key object:

```
CK_OBJECT_CLASS class = CKO_SECRET_KEY;
CK_KEY_TYPE keyType = CKK_AES;
CK_UTF8CHAR label[] = “An AES secret key object”;
CK_BYTE value[] = {...};
CK_BBOOL true = CK_TRUE;
CK_ATTRIBUTE template[] = {
  {CKA_CLASS, &class, sizeof(class)},
  {CKA_KEY_TYPE, &keyType, sizeof(keyType)},
  {CKA_TOKEN, &true, sizeof(true)},
  {CKA_LABEL, label, sizeof(label)-1},
  {CKA_ENCRYPT, &true, sizeof(true)},
  {CKA_VALUE, value, sizeof(value)}
};
```

CKA_CHECK_VALUE: The value of this attribute is derived from the key object by taking the first three bytes of the ECB encryption of a single block of null (0x00) bytes, using the default cipher associated with the key type of the secret key object.

### 2.8.3 AES key generation

The AES key generation mechanism, denoted **CKM_AES_KEY_GEN**, is a key generation mechanism for NIST’s Advanced Encryption Standard.
It does not have a parameter.
The mechanism generates AES keys with a particular length in bytes, as specified in the **CKA_VALUE_LEN** attribute of the template for the key.
The mechanism contributes the **CKA_CLASS**, **CKA_KEY_TYPE**, and **CKA_VALUE** attributes to the new key. Other attributes supported by the AES key type (specifically, the flags indicating which functions the key supports) may be specified in the template for the key, or else are assigned default initial values.
For this mechanism, the *ulMinKeySize* and *ulMaxKeySize* fields of the **CK_MECHANISM_INFO** structure specify the supported range of AES key sizes, in bytes.

### 2.8.4 AES-ECB

AES-ECB, denoted **CKM_AES_ECB**, is a mechanism for single- and multiple-part encryption and decryption; key wrapping; and key unwrapping, based on NIST Advanced Encryption Standard and electronic codebook mode.
It does not have a parameter.
This mechanism can wrap and unwrap any secret key. Of course, a particular token may not be able to wrap/unwrap every secret key that it supports. For wrapping, the mechanism encrypts the value of the **CKA_VALUE** attribute of the key that is wrapped, padded on the trailing end with up to block size minus one null bytes so that the resulting length is a multiple of the block size. The output data is the same length as the padded input data. It does not wrap the key type, key length, or any other information about the key; the application must convey these separately.

For unwrapping, the mechanism decrypts the wrapped key, and truncates the result according to the **CKA_KEY_TYPE** attribute of the template and, if it has one, and the key type supports it, the **CKA_VALUE_LEN** attribute of the template. The mechanism contributes the result as the **CKA_VALUE** attribute of the new key; other attributes required by the key type must be specified in the template.
Constraints on key types and the length of data are summarized in the following table:

*Table 2, AES-ECB: Key And Data Length*

| Function | Key type | Input length | Output length | Comments |
|---|---|---|---|---|
| C_Encrypt | AES | multiple of block size | same as input length | no final part |
| C_Decrypt | AES | multiple of block size | same as input length | no final part |
| C_WrapKey | AES | any | input length rounded up to multiple of block size | |
| C_UnwrapKey | AES | multiple of block size | determined by type of key being unwrapped or CKA_VALUE_LEN | |

For this mechanism, the *ulMinKeySize* and *ulMaxKeySize* fields of the **CK_MECHANISM_INFO** structure specify the supported range of AES key sizes, in bytes.

### 2.8.5 AES-CBC

AES-CBC, denoted **CKM_AES_CBC**, is a mechanism for single- and multiple-part encryption and decryption; key wrapping; and key unwrapping, based on NIST's Advanced Encryption Standard and cipher-block chaining mode.
It has a parameter, a 16-byte initialization vector.
This mechanism can wrap and unwrap any secret key. Of course, a particular token may not be able to wrap/unwrap every secret key that it supports. For wrapping, the mechanism encrypts the value of the **CKA_VALUE** attribute of the key that is wrapped, padded on the trailing end with up to block size minus one null bytes so that the resulting length is a multiple of the block size. The output data is the same length as the padded input data. It does not wrap the key type, key length, or any other information about the key; the application must convey these separately.
For unwrapping, the mechanism decrypts the wrapped key, and truncates the result according to the **CKA_KEY_TYPE** attribute of the template and, if it has one, and the key type supports it, the **CKA_VALUE_LEN** attribute of the template. The mechanism contributes the result as the **CKA_VALUE** attribute of the new key; other attributes required by the key type must be specified in the template.
Constraints on key types and the length of data are summarized in the following table:

*Table 3, AES-CBC: Key And Data Length*

| Function | Key type | Input length | Output length | Comments |
|---|---|---|---|---|
| C_Encrypt | AES | multiple of | same as input length | no final part |

| Function | Key type | Input length | Output length | Comments |
|---|---|---|---|---|
| | | block size | | |
| C_Decrypt | AES | multiple of block size | same as input length | no final part |
| C_WrapKey | AES | any | input length rounded up to multiple of the block size | |
| C_UnwrapKey | AES | multiple of block size | determined by type of key being unwrapped or CKA_VALUE_LEN | |

For this mechanism, the *ulMinKeySize* and *ulMaxKeySize* fields of the **CK_MECHANISM_INFO** structure specify the supported range of AES key sizes, in bytes.

### 2.8.6 AES-CBC with PKCS padding

AES-CBC with PKCS padding, denoted **CKM_AES_CBC_PAD**, is a mechanism for single- and multiple-part encryption and decryption; key wrapping; and key unwrapping, based on NIST's Advanced Encryption Standard; cipher-block chaining mode; and the block cipher padding method detailed in PKCS #7.
It has a parameter, a 16-byte initialization vector.
The PKCS padding in this mechanism allows the length of the plaintext value to be recovered from the ciphertext value. Therefore, when unwrapping keys with this mechanism, no value should be specified for the **CKA_VALUE_LEN** attribute.
In addition to being able to wrap and unwrap secret keys, this mechanism can wrap and unwrap RSA, Diffie-Hellman, X9.42 Diffie-Hellman, EC (also related to ECDSA) and DSA private keys (see Section Error: Reference source not found for details). The entries in the table below for data length constraints when wrapping and unwrapping keys do not apply to wrapping and unwrapping private keys.
Constraints on key types and the length of data are summarized in the following table:

*Table 4, AES-CBC with PKCS Padding: Key And Data Length*

| Function | Key type | Input length | Output length |
|---|---|---|---|
| C_Encrypt | AES | any | input length rounded up to multiple of the block size |
| C_Decrypt | AES | multiple of block size | between 1 and block size bytes shorter than input length |
| C_WrapKey | AES | any | input length rounded up to multiple of the block size |
| C_UnwrapKey | AES | multiple of block size | between 1 and block length bytes shorter than input length |

For this mechanism, the *ulMinKeySize* and *ulMaxKeySize* fields of the

**CK_MECHANISM_INFO** structure specify the supported range of AES key sizes, in bytes.

### 2.8.7 AES-OFB

AES-OFB, denoted CKM_AES_OFB. It is a mechanism for single and multiple-part encryption and decryption with AES. AES-OFB mode is described in [NIST sp800-38a].
It has a parameter, an initialization vector for this mode. The initialization vector has the same length as the blocksize.

Constraints on key types and the length of data are summarized in the following table:

*Table 5, AES-OFB: Key And Data Length*

| Function | Key type | Input length | Output length | Comments |
|---|---|---|---|---|
| C_Encrypt | AES | any | same as input length | no final part |
| C_Decrypt | AES | any | same as input length | no final part |

For this mechanism the CK_MECHANISM_INFO structure is as specified for CBC mode.

### 2.8.8 AES-CFB

Cipher AES has a cipher feedback mode, AES-CFB, denoted CKM_AES_CFB8, CKM_AES_CFB64, and CKM_AES_CFB128. It is a mechanism for single and multiple-part encryption and decryption with AES. AES-OFB mode is described [NIST sp800-38a].
It has a parameter, an initialization vector for this mode. The initialization vector has the same length as the blocksize.

Constraints on key types and the length of data are summarized in the following table:

*Table 6, AES-CFB: Key And Data Length*

| Function | Key type | Input length | Output length | Comments |
|---|---|---|---|---|
| C_Encrypt | AES | any | same as input length | no final part |
| C_Decrypt | AES | any | same as input length | no final part |

For this mechanism the CK_MECHANISM_INFO structure is as specified for CBC mode.

### 2.8.9 General-length AES-MAC

General-length AES-MAC, denoted **CKM_AES_MAC_GENERAL**, is a mechanism for single- and multiple-part signatures and verification, based on NIST Advanced Encryption Standard as defined in FIPS PUB 197 and data authentication as defined in FIPS PUB 113.
It has a parameter, a **CK_MAC_GENERAL_PARAMS** structure, which specifies the output length desired from the mechanism.
The output bytes from this mechanism are taken from the start of the final AES cipher block produced in the MACing process.
Constraints on key types and the length of data are summarized in the following table:

*Table 7, General-length AES-MAC: Key And Data Length*

| Function | Key type | Data length | Signature length |
|---|---|---|---|
| C_Sign | AES | any | 0-block size, as specified in parameters |
| C_Verify | AES | any | 0-block size, as specified in parameters |

For this mechanism, the *ulMinKeySize* and *ulMaxKeySize* fields of the **CK_MECHANISM_INFO** structure specify the supported range of AES key sizes, in bytes.

### 2.8.10 AES-MAC

AES-MAC, denoted by **CKM_AES_MAC**, is a special case of the general-length AES-MAC mechanism. AES-MAC always produces and verifies MACs that are half the block size in length.
It does not have a parameter.
Constraints on key types and the length of data are summarized in the following table:

*Table 8, AES-MAC: Key And Data Length*

| Function | Key type | Data length | Signature length |
|---|---|---|---|
| C_Sign | AES | any | ½ block size (8 bytes) |
| C_Verify | AES | any | ½ block size (8 bytes) |

For this mechanism, the *ulMinKeySize* and *ulMaxKeySize* fields of the **CK_MECHANISM_INFO** structure specify the supported range of AES key sizes, in bytes.

### 2.8.11 AES-XTS

AES-XTS (XEX-based Tweaked CodeBook mode with CipherText Stealing), denoted **CKM_AES_XTS**, is a mechanism for single- and multiple-part encryption and decryption; based on NIST Advanced Encryption Standard [NIST sp800-38e]. It is also an IEEE standard [IEEE Std. 1619-2007].

Its single parameter is a Data Unit Sequence Number 8 bytes long. Supported key lengths are 256 bits and 512 bits. Keys are internally split into half-length sub-keys of 128 and 256 bits respectively.

Constraints on key types and the length of data are summarized in the following table:

*Table 9, AES-XTS: Key And Data Length*

| Function | Key type | Input length | Output length | Comments |
|---|---|---|---|---|
| C_Encrypt | AES | any | same as input length | no final part |
| C_Decrypt | AES | any | same as input length | no final part |

For this mechanism, the *ulMinKeySize* and *ulMaxKeySize* fields of the **CK_MECHANISM_INFO** structure specify the supported range of AES key sizes, in bytes.