2.3 Elliptic Curve

The Elliptic Curve (EC) cryptosystem (also related to ECDSA) in this document is the one described in the ANSI X9.62 and X9.63 standards developed by the ANSI X9F1 working group.

Mechanism	Functions						
	Encrypt & Decrypt	**Sign & Verify**	**SR & VR**[1]	**Digest**	**Gen. Key/ Key Pair**	**Wrap & Unwrap**	**Derive**
CKM_EC_KEY_PAIR_GEN (CKM_ECDSA_KEY_PAIR_GEN)					✓		
CKM_ECDSA		✓[2]					
CKM_ECDSA_SHA1		✓					
CKM_ECDH1_DERIVE							✓
CKM_ECDH1_COFACTOR_DERIVE							✓
CKM_ECMQV_DERIVE							✓
CKM_ECDSA_FIPS_186_4		✓[2]					

Table 1, Mechanism Information Flags

CKF_EC_F_P	0x00100000	True if the mechanism can be used with EC domain parameters over F_p
CKF_EC_F_2M	0x00200000	True if the mechanism can be used with EC domain parameters over F_{2m}
CKF_EC_ECPARAMETERS	0x00400000	True if the mechanism can be used with EC domain parameters of the choice **ecParameters**
CKF_EC_NAMEDCURVE	0x00800000	True if the mechanism can be used with EC domain parameters of the choice **namedCurve**
CKF_EC_UNCOMPRESS	0x01000000	True if the mechanism can be used with elliptic curve point uncompressed
CKF_EC_COMPRESS	0x02000000	True if the mechanism can be used with elliptic curve point compressed

In these standards, there are two different varieties of EC defined:

1. EC using a field with an odd prime number of elements (i.e. the finite field F_p).

2. EC using a field of characteristic two (i.e. the finite field F_{2m}).

An EC key in Cryptoki contains information about which variety of EC it is suited for. It is preferable that a Cryptoki library, which can perform EC mechanisms, be capable of performing operations with the two varieties of EC, however this is not required. The **CK_MECHANISM_INFO** structure **CKF_EC_F_P** flag identifies a Cryptoki library supporting EC keys over F_p whereas the **CKF_EC_F_2M** flag identifies a Cryptoki library supporting EC keys over F_{2m}. A Cryptoki library that can perform EC mechanisms must set either or both of these flags for each EC mechanism.

In these specifications there are also three representation methods to define the domain parameters for an EC key. Only the **ecParameters** and the **namedCurve** choices are supported in Cryptoki. The **CK_MECHANISM_INFO** structure **CKF_EC_ECPARAMETERS** flag identifies a Cryptoki library supporting the **ecParameters** choice whereas the **CKF_EC_NAMEDCURVE** flag identifies a Cryptoki library supporting the **namedCurve** choice. A Cryptoki library that can perform EC mechanisms must set either or both of these flags for each EC mechanism.

In these specifications, an EC public key (i.e. EC point *Q*) or the base point *G* when the **ecParameters** choice is used can be represented as an octet string of the uncompressed form or the compressed form. The **CK_MECHANISM_INFO** structure **CKF_EC_UNCOMPRESS** flag identifies a Cryptoki library supporting the uncompressed form whereas the **CKF_EC_COMPRESS** flag identifies a Cryptoki library supporting the compressed form. A Cryptoki library that can perform EC mechanisms must set either or both of these flags for each EC mechanism.

Note that an implementation of a Cryptoki library supporting EC with only one variety, one representation of domain parameters or one form may encounter difficulties achieving interoperability with other implementations.

If an attempt to create, generate, derive, or unwrap an EC key of an unsupported variety (or of an unsupported size of a supported variety) is made, that attempt should fail with the error code CKR_TEMPLATE_INCONSISTENT. If an attempt to create, generate, derive, or unwrap an EC key with invalid or of an unsupported representation of domain parameters is made, that attempt should fail with the error code CKR_DOMAIN_PARAMS_INVALID. If an attempt to create, generate, derive, or unwrap an EC key of an unsupported form is made, that attempt should fail with the error code CKR_TEMPLATE_INCONSISTENT.

2.3.1 EC Signatures

For the purposes of these mechanisms, an ECDSA signature is an octet string of even length which is at most two times *nLen* octets, where *nLen* is the length in octets of the base point order *n*. The signature octets correspond to the concatenation of the ECDSA values *r* and *s*, both represented as an octet string of equal length of at most *nLen* with the most significant byte first. If *r* and *s* have different octet length, the shorter of both must be padded with leading zero octets such that both have the same octet length. Loosely spoken, the first half of the signature is *r* and the second half is *s*. For signatures created by a token, the resulting signature is always of length 2*nLen*. For signatures passed to a token for verification, the signature may have a shorter length but must be composed as specified before.

If the length of the hash value is larger than the bit length of *n*, only the leftmost bits of the hash up to the length of *n* will be used. Any truncation is done by the token.

Note: For applications, it is recommended to encode the signature as an octet string of length two times *nLen* if possible. This ensures that the application works with PKCS#11 modules which have been implemented based on an older version of this document. Older versions required all signatures to have length two times *nLen*. It may be impossible to encode the signature with the maximum length of two times *nLen* if the application just gets the integer values of *r* and *s* (i.e. without leading zeros), but does not know the base point order *n*, because *r* and *s* can have any value between zero and the base point order *n*.

2.3.2 Definitions

This section defines the key type “CKK_ECDSA” and “CKK_EC” for type CK_KEY_TYPE as used in the CKA_KEY_TYPE attribute of key objects.

Mechanisms:

Note: CKM_ECDSA_KEY_PAIR_GEN is deprecated in v2.11
CKM_ECDSA_KEY_PAIR_GEN
CKM_EC_KEY_PAIR_GEN
CKM_ECDSA
CKM_ECDSA_SHA1
CKM_ECDH1_DERIVE
CKM_ECDH1_COFACTOR_DERIVE
CKM_ECMQV_DERIVE
CKM_ECDSA_FIPS_186_4

CKD_NULL
CKD SHA1_KDF

2.3.11 Elliptic curve Menezes-Qu-Vanstone key derivation

The elliptic curve Menezes-Qu-Vanstone (ECMQV) key derivation mechanism, denoted **CKM_ECMQV_DERIVE**, is a mechanism for key derivation based the MQV version of the elliptic curve key agreement scheme, as defined in ANSI X9.63, where each party contributes two key pairs all using the same EC domain parameters.

It has a parameter, a **CK_ECMQV_DERIVE_PARAMS** structure.

This mechanism derives a secret value, 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 truncation removes bytes from the leading end of the secret value.) 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.

This mechanism has the following rules about key sensitivity and extractability:

- The **CKA_SENSITIVE** and **CKA_EXTRACTABLE** attributes in the template for the new key can both be specified to be either CK_TRUE or CK_FALSE. If omitted, these attributes each take on some default value.

- If the base key has its **CKA_ALWAYS_SENSITIVE** attribute set to CK_FALSE, then the derived key will as well. If the base key has its **CKA_ALWAYS_SENSITIVE** attribute set to CK_TRUE, then the derived key has its **CKA_ALWAYS_SENSITIVE** attribute set to the same value as its **CKA_SENSITIVE** attribute.

- Similarly, if the base key has its **CKA_NEVER_EXTRACTABLE** attribute set to CK_FALSE, then the derived key will, too. If the base key has its **CKA_NEVER_EXTRACTABLE** attribute set to CK_TRUE, then the derived key has its **CKA_NEVER_EXTRACTABLE** attribute set to the *opposite* value from its **CKA_EXTRACTABLE** attribute.

For this mechanism, the *ulMinKeySize* and *ulMaxKeySize* fields of the **CK_MECHANISM_INFO** structure specify the minimum and maximum supported number of bits in the field sizes, respectively. For example, if a Cryptoki library supports only EC using a field of characteristic 2 which has between 2^{200} and 2^{300} elements, then *ulMinKeySize* = 201 and *ulMaxKeySize* = 301 (when written in binary notation, the number 2^{200} consists of a 1 bit followed by 200 0 bits. It is therefore a 201-bit number. Similarly, 2^{300} is a 301-bit number).

2.3.12 FIPS 186-4

CKM_ECDSA_FIPS_186_4 is identical to CKM_ECDSA except that curves must either be NIST recommended curves (with a fixed set of domain parameters) or curves with domain parameters generated as specified by ANS X9.64. The NIST recommended curves are:

P-192, P-224, P-256, P-384, P-521

K-163, B-163, K-233, B-233

K-283, B-283, K-409, B-409

K-571, B-571