Y.ZZ TLS 1.2
Details can be found in [TLS12].

Mechanism	Functions						
	Encrypt & Decrypt	**Sign & Verify**	**SR & VR**	**Digest**	**Gen. Key/ Key Pair**	**Wrap & Unwrap**	**Derive**
CKM_TLS12_MASTER_KEY_DERIVE							V
CKM_TLS12_MASTER_KEY_DERIVE_DH							V
CKM_TLS12_KEY_AND_MAC_DERIVE							V
CKM_TLS12_PRF							V
CKM_TLS12_SHA256_MAC		V					
CKM_TLS12_SHA384_MAC		V					

Y.ZZ.1 Definitions

Mechanisms:

```
CKM_TLS12_MASTER_KEY_DERIVE
CKM_TLS12_KEY_AND_MAC_DERIVE
CKM_TLS12_MASTER_KEY_DERIVE_DH
CKM_TLS12_PRF
CKM_TLS12_SHA256_HMAC
CKM_TLS12_SHA384_HMAC
```

Y.ZZ.2 TLS 1.2 mechanism parameters

♦ CK_TLS12_PRF_PARAMS; CK_TLS12_PRF_PARAMS_PTR

CK_TLS12_PRF_PARAMS is a structure, which provides the parameters to the **CKM_TLS12_PRF** mechanism. It is defined as follows:

```
typedef struct CK_TLS12_PRF_PARAMS {
     CK_BYTE_PTR         pSeed;
     CK_ULONG            ulSeedLen;
     CK_BYTE_PTR         pLabel;
     CK_ULONG            ulLabelLen;
     CK_BYTE_PTR         pOutput;
     CK_ULONG_PTR        pulOutputLen;
     CK_MECHANISM_TYPE   prfFunc;
} CK_TLS12_PRF_PARAMS;
```

The fields of the structure have the following meanings:

pSeed	pointer to the input seed
ulSeedLen	length in bytes of the input seed
pLabel	pointer to the identifying label
ulLabelLen	length in bytes of the identifying label

pOutput pointer receiving the output of the operation
pulOutputLen pointer to the length in bytes that the output to be created shall have, has to hold the desired length as input and will receive the calculated length as output

prfFunc PRF function identifier

CK_TLS12_PRF_PARAMS_PTR is a pointer to a **CK_TLS12_PRF_PARAMS**.

♦ **CK_TLS12_MASTER_KEY_DERIVE_PARAMS; CK_TLS12_MASTER_KEY_DERIVE_PARAMS_PTR**

CK_TLS12_MASTER_KEY_DERIVE_PARAMS is a structure that provides the parameters to the **CKM_TLS12_MASTER_KEY_DERIVE** mechanism. It is defined as follows:

```
typedef struct CK_TLS12_MASTER_KEY_DERIVE_PARAMS {
     CK_SSL3_RANDOM_DATA RandomInfo;
     CK_VERSION_PTR pVersion;
     CK_MECHANISM_TYPE prfFunc;
} CK_TLS12_MASTER_KEY_DERIVE_PARAMS;
```

The fields of the structure have the following meanings:

RandomInfo client's and server's random data information.
pVersion pointer to a **CK_VERSION** structure which receives the SSL protocol version information
prfFunc PRF identifier

CK_TLS12_MASTER_KEY_DERIVE_PARAMS_PTR is a pointer to a **CK_TLS12_MASTER_KEY_DERIVE_PARAMS**.

♦ **CK_TLS12_KEY_MAT_PARAMS; CK_TLS12_KEY_MAT_PARAMS_PTR**

CK_TLS12_KEY_MAT_PARAMS is a structure that provides the parameters to the **CKM_TLS12_KEY_AND_MAC_DERIVE** mechanism. It is defined as follows:

```
typedef struct CK_TLS12_KEY_MAT_PARAMS {
     CK_ULONG ulMacSizeInBits;
     CK_ULONG ulKeySizeInBits;
     CK_ULONG ulIVSizeInBits;
     CK_BBOOL bIsExport;
     CK_SSL3_RANDOM_DATA RandomInfo;
     CK_SSL3_KEY_MAT_OUT_PTR pReturnedKeyMaterial;
     CK_MECHANISM_TYPE prfFunc;
} CK_TLS12_KEY_MAT_PARAMS;
```

The fields of the structure have the following meanings:

ulMacSizeInBits	the length (in bits) of the MACing keys agreed upon during the protocol handshake phase
ulKeySizeInBits	the length (in bits) of the secret keys agreed upon during the protocol handshake phase
ulIVSizeInBits	the length (in bits) of the IV agreed upon during the protocol handshake phase. If no IV is required, the length should be set to 0
bIsExport	a Boolean value which indicates whether the keys have to be derived for an export version of the protocol
RandomInfo	client's and server's random data information.
pReturnedKeyMaterial	points to a **CK_SSL3_KEY_MAT_OUT** structures which receives the handles for the keys generated and the IVs
prfFunc	PRF identifier

CK_TLS12_KEY_MAT_PARAMS_PTR is a pointer to a **CK_TLS12_KEY_MAT_PARAMS**.

Y.ZZ.3 TLS 1.2 PRF (pseudorandom function)

PRF (pseudo random function) in TLS 1.2, denoted **CKM_TLS12_PRF**, is a mechanism used to produce a securely generated pseudo-random output of arbitrary length. The keys it uses are generic secret keys.

It has a parameter, a **CK_TLS12_PRF_PARAMS** structure, which allows for the passing of the input seed and its length, the passing of an identifying label and its length and the passing of the length of the output to the token and for receiving the output.

This mechanism produces securely generated pseudo-random output of the length specified in the parameter.

This mechanism departs from the other key derivation mechanisms in Cryptoki in not using the template sent along with this mechanism during a **C_DeriveKey** function call, which means the template shall be a NULL_PTR. For most key-derivation mechanisms, **C_DeriveKey** returns a single key handle as a result of a successful completion.

However, since the **CKM_TLS12_PRF** mechanism returns the requested number of output bytes in the **CK_TLS_PRF12_PARAMS** structure specified as the mechanism parameter, the parameter *phKey* passed to **C_DeriveKey** is unnecessary, and should be a NULL_PTR.

If a call to **C_DeriveKey** with this mechanism fails, then no output will be generated.

Y.ZZ.4 Master key derivation

Master key derivation in TLS 1.2, denoted **CKM_TLS12_MASTER_KEY_DERIVE**, is a mechanism used to derive one 48-byte generic secret key from another 48-byte generic secret key. It is used to produce the "master_secret" key used in the TLS 1.2 protocol from the "pre_master" key. This mechanism returns the value of the client version, which is built into the "pre_master" key as well as a handle to the derived "master_secret" key.

It has a parameter, a **CK_TLS12_MASTER_KEY_DERIVE_PARAMS** structure, which allows for the passing of random data to the token as well as the returning of the protocol version number which is part of the pre-master key.

The mechanism contributes the **CKA_CLASS**, **CKA_KEY_TYPE**, and **CKA_VALUE** attributes to the new key (as well as the **CKA_VALUE_LEN** attribute, if it is not supplied in the template). Other attributes may be specified in the template, or else are assigned default values.

The template sent along with this mechanism during a **C_DeriveKey** call may indicate that the object class is **CKO_SECRET_KEY**, the key type is **CKK_GENERIC_SECRET**, and the **CKA_VALUE_LEN** attribute has value 48. However, since these facts are all implicit in the mechanism, there is no need to specify any of them.

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 both indicate 48 bytes.

Note that the **CK_VERSION** structure pointed to by the **CK_SSL3_MASTER_KEY_DERIVE_PARAMS** structure's *pVersion* field will be modified by the **C_DeriveKey** call. In particular, when the call returns, this structure will hold the SSL version associated with the supplied pre_master key.

Note that this mechanism is only useable for cipher suites that use a 48-byte "pre_master" secret with an embedded version number. This includes the RSA cipher suites, but excludes the Diffie-Hellman cipher suites.

Y.ZZ.5 Master key derivation for Diffie-Hellman

Master key derivation for Diffie-Hellman in TLS 1.2, denoted **CKM_TLS12_MASTER_KEY_DERIVE_DH**, is a mechanism used to derive one 48-byte generic secret key from another arbitrary length generic secret key. It is used to produce the "master_secret" key used in the TLS 1.2 protocol from the "pre_master" key.

It has a parameter, a **CK_TLS12_MASTER_KEY_DERIVE_PARAMS** structure, which allows for the passing of random data to the token. The *pVersion* field of the structure must be set to NULL_PTR since the version number is not embedded in the "pre_master" key as it is for RSA-like cipher suites.

The mechanism contributes the **CKA_CLASS**, **CKA_KEY_TYPE**, and **CKA_VALUE** attributes to the new key (as well as the **CKA_VALUE_LEN** attribute, if it is not supplied in the template). Other attributes may be specified in the template, or else are assigned default values.

The template sent along with this mechanism during a **C_DeriveKey** call may indicate that the object class is **CKO_SECRET_KEY**, the key type is **CKK_GENERIC_SECRET**, and the **CKA_VALUE_LEN** attribute has value 48. However, since these facts are all implicit in the mechanism, there is no need to specify any of them.

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 both indicate 48 bytes.

Note that this mechanism is only useable for cipher suites that do not use a fixed length 48-byte "pre_master" secret with an embedded version number. This includes the Diffie-

Hellman cipher suites, but excludes the RSA cipher suites.

Y.ZZ.6 Key and MAC derivation

Key, MAC and IV derivation in TLS 1.2, denoted **CKM_TLS12_KEY_AND_MAC_DERIVE**, is a mechanism used to derive the appropriate cryptographic keying material used by a "CipherSuite" from the "master_secret" key and random data. This mechanism returns the key handles for the keys generated in the process, as well as the IVs created.

It has a parameter, a **CK_TLS12_KEY_MAT_PARAMS** structure, which allows for the passing of random data as well as the characteristic of the cryptographic material for the given CipherSuite and a pointer to a structure which receives the handles and IVs which were generated.

This mechanism contributes to the creation of four distinct keys on the token and returns two IVs (if IVs are requested by the caller) back to the caller. The keys are all given an object class of **CKO_SECRET_KEY**.

The two MACing keys ("client_write_MAC_secret" and "server_write_MAC_secret") are always given a type of **CKK_GENERIC_SECRET**. They are flagged as valid for signing, verification, and derivation operations.

The other two keys ("client_write_key" and "server_write_key") are typed according to information found in the template sent along with this mechanism during a **C_DeriveKey** function call. By default, they are flagged as valid for encryption, decryption, and derivation operations.

IVs will be generated and returned if the *ulIVSizeInBits* field of the **CK_SSL_KEY_MAT_PARAMS** field has a nonzero value. If they are generated, their length in bits will agree with the value in the *ulIVSizeInBits* field.

All four keys inherit the values of the **CKA_SENSITIVE**, **CKA_ALWAYS_SENSITIVE**, **CKA_EXTRACTABLE**, and **CKA_NEVER_EXTRACTABLE** attributes from the base key. The template provided to **C_DeriveKey** may not specify values for any of these attributes which differ from those held by the base key.

Note that the **CK_SSL3_KEY_MAT_OUT** structure pointed to by the **CK_SSL3_KEY_MAT_PARAMS** structure's *pReturnedKeyMaterial* field will be modified by the **C_DeriveKey** call. In particular, the four key handle fields in the **CK_SSL3_KEY_MAT_OUT** structure will be modified to hold handles to the newlycreated keys; in addition, the buffers pointed to by the **CK_SSL3_KEY_MAT_OUT** structure's *pIVClient* and *pIVServer* fields will have IVs returned in them (if IVs are requested by the caller). Therefore, these two fields must point to buffers with sufficient space to hold any IVs that will be returned.

This mechanism departs from the other key derivation mechanisms in Cryptoki in its

returned information. For most key-derivation mechanisms, **C_DeriveKey** returns a single key handle as a result of a successful completion. However, since the **CKM_SSL3_KEY_AND_MAC_DERIVE** mechanism returns all of its key handles in the **CK_SSL3_KEY_MAT_OUT** structure pointed to by the **CK_SSL3_KEY_MAT_PARAMS** structure specified as the mechanism parameter, the parameter *phKey* passed to **C_DeriveKey** is unnecessary, and should be a NULL_PTR.

If a call to **C_DeriveKey** with this mechanism fails, then *none* of the four keys will be created on the token.

Y.ZZ.7 SHA256 MACing in TLS 1.2

SHA-256 MACing in TLS1.2, denoted **CKM_TLS12_SHA256_MAC**, is a mechanism for single- and multiple-part signatures (data authentication) and verification using SHA-256, based on the TLS 1.2 protocol. This technique is very similar to the HMAC technique.

It has a parameter, a **CK_MAC_GENERAL_PARAMS**, which specifies the length in bytes of the signatures produced by this mechanism.

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

Table NN, SHA-256 MACing in TLS 1.2: Key And Data Length

Function	Key type	Data length	Signature length
C_Sign	generic secret	any	32
C_Verify	generic secret	any	32

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

Y.ZZ.8 SHA384 MACing in TLS 1.2

SHA-384 MACing in TLS1.2, denoted **CKM_TLS12_SHA384_MAC**, is a mechanism for single- and multiple-part signatures (data authentication) and verification using SHA-384, based on the TLS 1.2 protocol. This technique is very similar to the HMAC technique.

It has a parameter, a **CK_MAC_GENERAL_PARAMS**, which specifies the length in bytes of the signatures produced by this mechanism.

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

Table NN, SHA-384 MACing in TLS 1.2: Key And Data Length

Function	Key type	Data	Signature length

		length	
C_Sign	generic secret	any	48
C_Verify	generic secret	any	48

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