CMS Security

Application Interfaces (API)

Version 1.0d11
April 21, 1999

Overview

This is a generic CMS (Cryptographic Message Standard formely PKCS7) signing/verification/encryption/decryption api. It can be used for creating S/MIME messages when combined with a MIME library. To do so use the SecSMIME domain for all operations this will assure that all attributes required by S/MIME are automatically added and or validated. Most email programs wanting to use this library to add secure email to their capabilities will already have MIME support.

When creating signed messages there is the option of creating detached signatures. Thisis useful because you can send along the original data in plain text (or differently encoded) form so that nonn S/MIME capably mailclients can still read the body of the message. When doing so the signing routines do not include the data passed in in the outputstream. Whenverifying with detached data the data is not extracted (since it is already seperate) thus no outputstream is required.

This library is designed to be Objected-Oriented-like just like CoreFoundation (CF). We srongly suggest reading the documentation that comes with CF before reading this document since we use the classes and concepts explained there. Even though C is not an object oriented language you should consider the different parts of this API as different classes of wich you can create instances (objects). All classes derived from a base "class" called SecType (a reference to it is called SecTypeRef).

Each class has it own typeID which you can use to determine which class it was that was passed in to a certain function. To do so use SecGetTypeID (object) and the Sec<classname>GetTypeID (). You can use == to compare typeID's.

There are a number of polymorphic functions that are defined for all SecType objects. These functions are described in the "Overloaded functions that work for all classes" section. The most notable one is SecRelease which should be used to get rid of objects you Created or Copied.

Whenever you create a SecType object using a Sec<classname>Create<arguments> function or Sec<classname>Copy<arguments> function you should call SecRelease of the object returned when you no longer need it. If you obtain an object reference though a different method you should call SecRetain on it until you no longer need it and then call SecRelease. This will prevent the object from going away while it is in use.

The API breaks down into the following classes:

Class Name	Used By	Description
CmsEncoder	caller	Message encoding object
CmsDecoder	caller	Message decoding object
Stream	all messages	Input or Output stream. Backed by file or memory
Signer	sign/verify	Someone who is signing or signed a message
Attributes	signer	The attributes of a signer

The functions on each class are covered in the sections below.

Example usage of the API for file signing

Example: creating a PKCS7 format signed file.

```
void SignFile (short infile, short outfile, KCItemRef signingCert)
{
      SecCmsEncoderRef cmsEncoder;
      SecStreamRef inStream, outStream;
      SecMutableSignerRef signer;
```

```
    CFMutableDataRef input;

    SecStreamCreateOutputFile (outfile, &outStream);
    SecCmsEncoderCreate (false, outStream, kSecRawPKCS7, &cmsEncoder);
    SecRelease (outStream);

    SecSignerCreateMutable (signingCert, kKCAlgSHA1, true, &signer);
    SecCmsEncoderAddSigner (cmsEncoder, signer, TRUE);
    SecRelease (signer);

    /* Create an inputFileStream on the open input file.  */
    SecStreamCreateInputFile (infile, &inStream);

    /* Create a mutableData object that will serve as a bridge
       between the inputSteam and the cmsEncoder.  */
    input = CFDataCreateMutable (NULL, 0);
    /* Clear out input.  */
    CFDataSetLength (input, 0);
    /* Read all available data from inStream into input.  */
    SecStreamReadData (inStream, 0, input);
    /* Hand off input to the cmsEncoder.   */
    SecCmsEncoderProcessData (cmsEncoder, input);
    /* We are done with input so release it. */
    CFRelease (input);

    /* We are done with inStream so release it. */
    SecRelease (inStream);

    /* Tell cmsEncoder we are done passing in data.  */
    SecCmsEncoderFinish (cmsEncoder);
    /* We are done with cmsEncoder so release it. */
    SecRelease (cmsEncoder);
}
```

TBD.

CMS routines

Creating a cms message

```
OSStatus SecCmsEncoderCreateForSign (Boolean includeData,
                                     Boolean contentOnly,
                                     SecCmsEncoderRef parentCmsEncoder,
                                     SecStreamRef outputStream,
                                     SecMsgDomain domain,
                                     SecCmsEncoderRef* pCmsEncoder)
```

`includeData`	Whether to create a external signature: - FALSE means create a external signature (do not include the data being signed in the generated PKCS7 data). - TRUE means create a contentInfo with the data embedded the data (the generated PKCS7 data will contain an embedded copy of the supplied data).

contentOnly	Whether to create only the inner content object: - FALSE means create a complete cms message. - TRUE means create only the content part of the cms message.
parentCmsEncoder	If non NULL outputStream must be NULL and the output will actually be the input to the parentCmsEncoder thus allowing creation of a signed and encrypted message in a single pass.
outputStream	This is where the resulting PKCS7 data will be written to.
domain	The domain this object operates on.
pCmsEncoder	Output: A pointer to an unintialized SecCmsEncoderRef.

DESCRIPTION

Creates a SecCmsEncoder object. This function will not put up any HI.

RESULT CODE

noErr	0	No error.
memFullErr	-108	Not enough memory to complete this operation.
errSecInvalidPointer	-TBD	cmsEncoder was not a valid pointer.
errSecInvalidStream	-TBD	The outputStream passed in is not valid.

```
OSStatus SecCmsEncoderCreateForEncrypt (SecAlgorithmId encryptionAlgorithm,
                                        UInt32 keySizeInBits,
                                        Boolean contentOnly,
                                        SecCmsEncoder parentCmsEncoder,
                                        SecStreamRef outputStream,
                                        SecStreamRef cipherTextStream,
                                        SecCmsEncoderRef* pCmsEncoder)
```

encryptionAlgorithm	Encryption algorithm to use.
keySizeInBits	Keysize in bits for session key.
contentOnly	Whether to create only the inner content object: - FALSE means create a complete cms message. - TRUE means create only the content part of the cms message.
parentCmsEncoder	If non NULL outputStream must be NULL and the output will actually be the input to the parentCmsEncoder thus allowing creation of an encrypted and signed message in a single pass.
outputStream	This is where the resulting PKCS7 data will be written to.
cipherTextStream	This is where the cipherText will be written to.
domain	The domain this object operates on.
pCmsEncoder	Output: A pointer to an unintialized SecCmsEncoderRef.

DESCRIPTION

Creates a SecCmsEncoder object. Generate a random session key of keySizeInBits bits using encryptionAlgorithm. If cipherTextStream is NULL the CMS message written to outputStream will contain the cipherText embedded. If it is non NULL a detached envelopedData will be created and the cipherText will go to ciperTextStream. This function write a PKCS7 ContentInfo of type EnvolopedData to outputStream as input is provided though further calls operating on cmsEncoder. This function will not put up any HI.

RESULT CODE

noErr	0	No error.
errSecInvalidStream	-TBD	The outputStream or ciperTextStream passed in is not valid.
errSecBadPointer	-TBD	cmsEncoder was not a valid pointer.

```
OSStatus SecCmsEncoderAddRecipient (SecCmsEncoderRef cmsEncoder,
                                    KCItemRef cert,
                                    CFArrayRef policies,
                                    SecValidateStopOn stopOn)
```

cmsEncoder	The SecCmsEncoder object we operate on.
cert	A KCItemRef that points to a valid certificate.
policies	Array of SecOID objects that represent cert validation policies.
stopOn	When to stop validation of policies.

DESCRIPTION

Adds recipient to the cmsEncoderbeing created. This function can be called multiple times, for the same cmsEncoderbeing. This function will not put up any HI.

RESULT CODE

noErr	0	No error.
memFullErr	-108	Not enough memory to complete this operation.
errSecInvalidCert	-TBD	One or more certificates in the signing certificates cert chain were invalid.
errSecMissingCert	-TBD	One or more certificates in the signing certificates cert chain were not found.
errSecInvalidStopOn	-TBD	stopOn was not a valid value.
errSecUnknownPolicy	-TBD	On of the policies passed in was unknown or unsupported.
errSecNotTrusted	-TBD	Cert chain was not trusted by one or more selected policies.
errSecInvalidReference	-TBD	Passed in object was invalid.
errSecInProgress	-TBD	Can no longer add recipients.

```
OSStatus SecCmsEncoderAddCert (SecCmsEncoderRef cmsEncoder, KCItemRef cert)
```

cmsEncoder	The SecCmsEncoder object we operate on.
cert	A KCItemRef that points to a valid certificate.

DESCRIPTION

Adds cert to the cmsEncoder being created. This function can be called multiple times, for the same cmsEncoder. This function will not put up any HI.

RESULT CODE

noErr	0	No error.
memFullErr	-108	Not enough memory to complete this operation.
errSecInvalidReference	-TBD	Passed in object was invalid.

```
OSStatus SecCmsEncoderAddCertChain (SecCmsEncoderRef cmsEncoder,
                                    KCItemRef cert)
```

cmsEncoder	The SecCmsEncoder object we operate on.
cert	A KCItemRef that points to a valid certificate.

DESCRIPTION

Adds cert and the certs that have signed it up to the root to the cmsEncoder being created. This function can be called multiple times for the same cmsEncoder. This function will not put up any HI.

RESULT CODE

noErr	0	No error.
memFullErr	-108	Not enough memory to complete this operation.
errSecInvalidReference	-TBD	Passed in object was invalid.

```
OSStatus SecCmsEncoderAddSigner (SecCmsEncoderRef cmsEncoder,
                                 SecSignerRef signer,
                                 Boolean addCertChain)
```

cmsEncoder	The SecCmsEncoder object we operate on.
signer	A SecSignerRef obtained by calling SecSignerCreate.
addCertChain	Whether to include the certificate chain for signer's cert in the message: - FALSE means just use this certificate but don't add any to the list of certificates being sent with this message. - TRUE means add the signer's certificate and it's issuer chain up to a root along with the message.

DESCRIPTION

Adds a signer to the cmsEncoder being created. This function can be called multiple times for the same cmsEncoder, but no calls to this function should be made after the first call to SecCmsEncoderProcessData. A copy of the signer is actually added to the message. This function will not put up any HI.

RESULT CODE

noErr	0	No error.
memFullErr	-108	Not enough memory to complete this operation.
errSecInvalidReference	-TBD	Passed in object was invalid.
errSecMissingAttribute	-TBD	Signer is missing one or more required attributes.
errSecInProgress	-TBD	Can no longer add signers.

```
OSStatus SecCmsEncoderProcessData (SecCmsEncoderRef cmsEncoder,
                                   CFDataRef data)
```

cmsEncoder	The SecCmsEncoder object we operate on.
data	A CFDataRef containing a block of data to be signed.

DESCRIPTION

Starts writing output to the outputStream of cmsEncoder. Computes the digests for given signers of the data passed in so far.. This function can be called multiple times. Once the first call to this function has been made no more signers can be added to this cmsEncoder. This function will not put up any HI.

RESULT CODE

noErr	0	No error.
dskFulErr	-34	Disk full.
ioErr	-36	I/O error.
fnOpnErr	-38	File not open.
posErr	-40	Attempt to position mark before start of file.
wPrErr	-44	Hardware volume lock.
fLckdErr	-45	File is locked.
vLckdErr	-46	Software volume lock.
rfNumErr	-51	Bad reference number.
wrPermErr	-61	Read/write permission doesn't allow writing.
memFullErr	-108	Not enough memory to complete this operation.

errSecInvalidReference	-TBD	Passed in object was invalid.
errSecFinished	-TBD	The Finish function has been invoked already.

```
OSStatus SecCmsEncoderFinish (SecCmsEncoderRef cmsEncoder)
```

cmsEncoder — The SecCmsEncoder object we operate on.

DESCRIPTION
Finishes writing output to the outputStream of cmsEncoder, writes out all certs and signerInfos. The cmsEncoder object may no longer be used after this function returns and should be relased. This function may put of HI needed to access the private key ascociated with one or more signers certificates.

RESULT CODE

noErr	0	No error.
dskFulErr	-34	Disk full.
ioErr	-36	I/O error.
fnOpnErr	-38	File not open.
posErr	-40	Attempt to position mark before start of file.
wPrErr	-44	Hardware volume lock.
fLckdErr	-45	File is locked.
vLckdErr	-46	Software volume lock.
rfNumErr	-51	Bad reference number.
wrPermErr	-61	Read/write permission doesn't allow writing.
memFullErr	-108	Not enough memory to complete this operation.
userCanceledErr	-128	The user canceled this operation.
errSecInvalidReference	-TBD	Passed in object was invalid.
errSecFinished	-TBD	The Finish function has been invoked already.

```
UInt32 SecCmsEncoderGetTypeID (void)
```

DESCRIPTION
Get the type id of SecCmsEncoder objects. This function will not put up any HI.

RESULT
Returns the typeID of SecCmsEncoder.

Decrypting or verifying a signed message or verifying a message with an external signature

```
OSStatus SecCmsDecoderCreate (UInt32 maxLevels,
                              void* reserved,
                              SecStreamRef outputStream,
                              SecMsgDomain domain,
                              SecCmsDecoderRef* pCmsDecoder)
```

externalSignature — Signature to verify data with:
- NULL means the data passed to this cmsDecoder object will be in PKCS7 format with a embedded data.
- A valid CFDataRef should contain a PKCS7 signature without embedded data. In this case the outputStream argument should be NULL.

`outputStream`	This is where the embedded data will be written to (see above).
`domain`	The domain this object operates on.
`pCmsDecoder`	`Output:` The caller must pass in a pointer to an unintialized `SecCmsDecoderRef`.

DESCRIPTION
Creates a `SecCmsDecoder` object. This function will not put up any HI.

RESULT CODE

`noErr`	0	No error.
`memFullErr`	-108	Not enough memory to complete this operation.
`errSecInvalidPointer`	-TBD	cmsDecoder was not a valid pointer.
`errSecInvalidStream`	-TBD	The outputStream passed in is not valid.

```
OSStatus SecCmsDecoderAddDigestAlgorithm (SecCmsDecoderRef cmsDecoder,
                                          SecAlgorithmId digestAlgorithmId)
```

`cmsDecoder`	The `SecCmsDecoder` object we operate on.
`digestAlgorithmId`	The digest algorithm to use for computing a digest of the data to be passed in.

DESCRIPTION
Adds a digestAlgortihm to a cmsDecoder object. This is useful when processing a multipart/signed mime body in a signle pass. The digest algortihm is known before the data or the external signature are. the caller can add the digestalgorithm used, then pass in the data and finally set the external signature. This function will not put up any HI.

RESULT CODE

`noErr`	0	No error.
`memFullErr`	-108	Not enough memory to complete this operation.
`errSecInvalidReference`	-TBD	cmsDecoder was not a valid pointer.

```
OSStatus SecCmsDecoderSetExternalSignature (SecCmsDecoderRef cmsDecoder,
                                            CFDataRef externalSignature)
```

`cmsDecoder`	The `SecCmsDecoder` object we operate on.
`externalSignature`	Signature to verify data with: - NULL means the data passed to this cmsDecoder object will be in PKCS7 format with a embedded data. - A valid CFDataRef should contain a PKCS7 signature without embedded data. In this case the outputStream argument should be NULL.

DESCRIPTION
Decodes an external signature and uses the information in it. You may call this function before passing in the data using SecCmsDecoderProcessData or after passing in the data provided SecCmsDecoderAddDigestAlgortihm was called at least once. This function will not put up any HI.

RESULT CODE

`noErr`	0	No error.
`memFullErr`	-108	Not enough memory to complete this operation.
`errSecInvalidReference`	-TBD	cmsDecoder was not a valid pointer.

errSecUnsupported	-TBD	externalSignature contained an unsupported cms message.
errSecInvalidData	-TBD	externalSignature was not a valid cms message.
errSecTooMuchData	-TBD	externalSignature had extra data after the top level contentInfo.
errSecMissingData	-TBD	externalSignature was not a complete contentInfo.

```
OSStatus SecCmsDecoderProcessData (SecCmsDecoderRef cmsDecoder,
                                   CFDataRef data)
```

cmsDecoder	The SecCmsDecoder object we operate on.
data	A CFDataRef containing a block of data to be verified.

DESCRIPTION

Starts writing output to the outputStream of cmsDecoder if the signature contains embedded data. This function can be called multiple times for the same cmsDecoder. If niether SecCmsDecoderAddDigestAlgorithm nor SecCmsDecoderSetExternalSignature were called before the first call to this function the data is treated as a cms message. If one of those functions were called the data is treated as the raw data being verified. This function will not put up any HI.

RESULT CODE

noErr	0	No error.
dskFulErr	-34	Disk full.
ioErr	-36	I/O error.
fnOpnErr	-38	File not open.
posErr	-40	Attempt to position mark before start of file.
wPrErr	-44	Hardware volume lock.
fLckdErr	-45	File is locked.
vLckdErr	-46	Software volume lock.
rfNumErr	-51	Bad reference number.
wrPermErr	-61	Read/write permission doesn't allow writing.
memFullErr	-108	Not enough memory to complete this operation.
errSecInvalidReference	-TBD	Passed in object was invalid.
errSecUnsupported	-TBD	A valid but unsupported cms message was passed in.
errSecInvalidData	-TBD	Data passed did not add up to a valid cms message.
errSecTooMuchData	-TBD	Extra data was passed in after the top level contentInfo.
errSecFinished	-TBD	The Finish function has been invoked already.

```
OSStatus SecCmsDecoderGetStatus (SecCmsDecoderRef cmsDecoder)
```

cmsDecoder	The SecCmsDecoder object we operate on.

DESCRIPTION

Returns the status of this cmsDecoder object. This function will not put up any HI.

RESULT CODE

noErr	0	No error.
memFullErr	-108	Not enough memory to complete this operation.
errSecInvalidReference	-TBD	Passed in object was invalid.

```
OSStatus SecCmsDecoderGetContentType (SecCmsDecoderRef cmsDecoder,
                                      SecCmsType* pContentType)
```

cmsDecoder	The SecCmsDecoder object we operate on.
pContentType	Output: The type of the cms message decoded by cmsDecoder.

DESCRIPTION

Initializes pContentType with the type of the cms message decoded by cmsDecoder. This function will not put up any HI.

RESULT CODE

noErr	0	No error.
memFullErr	-108	Not enough memory to complete this operation.
errSecInvalidReference	-TBD	Passed in object was invalid.
errSecInvalidPointer	-TBD	pContentType was not a valid pointer.
errSecMissingData	-TBD	An imcomplete cms message was passed in.

```
OSStatus SecCmsDecoderGetContentTypeOID (SecCmsDecoderRef cmsDecoder,
                                         SecOIDRef* pContentTypeOID)
```

cmsDecoder	The SecCmsDecoder object we operate on.
pContentTypeOID	Output: The oid of the contentType of the cms message decoded by cmsDecoder.

DESCRIPTION

Initializes pContentTypeOID with the oid of the contentType of the cms message decoded by cmsDecoder. This function will not put up any HI.

RESULT CODE

noErr	0	No error.
memFullErr	-108	Not enough memory to complete this operation.
errSecInvalidReference	-TBD	Passed in object was invalid.
errSecInvalidPointer	-TBD	pContentTypeOID was not a valid pointer.
errSecMissingData	-TBD	An imcomplete cms message was passed in.

```
OSStatus SecCmsDecoderGetContentCms (SecCmsDecoderRef cmsDecoder,
                                     SecCmsDecoderRef* pContentCmsDecoder)
```

cmsDecoder	The SecCmsDecoder object we operate on.
pContentCmsDecoder	Output: The cms decoder for the message contained by the message decoded by cmsDecoder.

DESCRIPTION

Initializes pContentCmsDecoder with the cms decoder for the message contained by the message decoded by cmsDecoder. This function will not put up any HI.

RESULT CODE

noErr	0	No error.
memFullErr	-108	Not enough memory to complete this operation.
errSecInvalidReference	-TBD	Passed in object was invalid.
errSecInvalidPointer	-TBD	pContentTypeOID was not a valid pointer.
errSecMissingData	-TBD	An imcomplete cms message was passed in.

```
OSStatus SecCmsDecoderGetCertificates (SecCmsDecoderRef cmsDecoder,
                                       CFArrayRef* pCertificates)
```

`cmsDecoder`	The `SecCmsDecoder` object we operate on.
`pCertificates`	Output: A CFArrayRef containing the raw certificates included in this message encapsulated in CFDataRef objects.

DESCRIPTION

Initializes pCertificates with a CFArray of KCItemRef's representing the certificates contained in cmsDecoder. This may or may not contain the certificates used to sign (if they were sent along it will). The caller should call CFRetain on the pCertificates array if she wishes it to stick around after the cmsDecoder is released. By using this function the caller can control which certificates are added to the users keychain before calling SecCmsDecoderValidate. This function will not put up any HI.

RESULT CODE

`noErr`	0	No error.
`memFullErr`	-108	Not enough memory to complete this operation.
`errSecInvalidReference`	-TBD	Passed in object was invalid.
`errSecInvalidPointer`	-TBD	certificates was not a valid pointer.
`errSecMissingData`	-TBD	An imcomplete cms message was passed in.

```
OSStatus SecCmsDecoderSigners (SecCmsDecoderRef cmsDecoder,
                               SecOptions options, CFArrayRef policies,
                               SecValidateStopOn stopOn,
                               CFArrayRef* pSigners)
```

`cmsDecoder`	The `SecCmsDecoder` object we operate on.
`options`	Whether to add new certificates, whether to add new root certificates and whether to add all other certificates.
`policies`	An optional list of policies to use for validation.
`stopOn`	When to stop validation.
`pSigners`	Output: CFArrayRef of SecSigner objects that were included in this message.

DESCRIPTION

Only the kSecOptionAddRootCerts, kSecOptionAddLeafCerts and kSecOptionAddOtherCerts options are supported. The policies provided add to the default policies for the selected domain. The caller should call CFRetain on the pSigners array if she wishes it to stick around after the cmsDecoder is released. The cmsEncoder object should be released with SecRelease when it is no longer needed. This function may put up HI needed to query the user whether or not to add certificates contained in the message, but only if one of the options mentinoed before are passed in.

RESULT CODE

`noErr`	0	No error, all signers validated ok.
`memFullErr`	-108	Not enough memory to complete this operation.
`errSecInvalidReference`	-TBD	Passed in object was invalid.
`errSecInvalidPointer`	-TBD	certificates was not a valid pointer.
`errSecInvalidData`	-TBD	The cms message passed in was invalid.
`errSecMissingData`	-TBD	An imcomplete cms message was passed in.
`errSecNoSigners`	-TBD	There were no signers.
`errSecSignerFailed`	-TBD	One of more of the signers validated with an error.
`errSecInvalidPolicy`	-TBD	On of the policies passed in was invalid.
`errSecUnknownPolicy`	-TBD	On of the policies passed in was not supported by any TP.

errSecInvalidStopOn	-TBD	stopOn was not a valid value.

```
UInt32 SecCmsDecoderGetTypeID (void)
```

DESCRIPTION
Get the type id of SecCmsDecoder objects. This function will not put up any HI.

RESULT
Returns the typeID of SecCmsDecoder.

Stream object manipulation

```
OSStatus SecStreamCreateOutputData (CFMutableDataRef data,
                                    SecStreamRef* pStream)
```

data	The backing store for this stream.
pStream	Output: A pointer to an uninitialized SecStreamRef.

DESCRIPTION
Create a new memory based stream object. This is a write only stream. Read from it by looking at data. This function will not put up any HI.

RESULT CODE

noErr	0	No error.
memFullErr	-108	Not enough memory to complete this operation.
errSecInvalidReference	-TBD	Passed in data object was invalid.
errSecInvalidPointer	-TBD	stream was not a valid pointer.

```
OSStatus SecStreamCreateFifo (SecStreamRef* pStream)
```

pStream	Output: A pointer to an uninitialized SecStreamRef.

DESCRIPTION
Create a new memory based stream object. Being a fifoStream makes it possible to write to and then read from this stream. This function will not put up any HI.

RESULT CODE

noErr	0	No error.
memFullErr	-108	Not enough memory to complete this operation.
errSecInvalidPointer	-TBD	stream was not a valid pointer.

```
OSStatus SecStreamCreateOutputFile (short fileRefNum,
                                    SecStreamRef* pStream)
```

fileRefNum	A valid open fileRefNum.
pStream	Output: A pointer to an uninitialized SecStreamRef.

DESCRIPTION

Create a new stream object. It is not possible to read from this stream, only write. This function will not put up any HI.

RESULT CODE

noErr	0	No error.
memFullErr	-108	Not enough memory to complete this operation.
errSecInvalidPointer	-TBD	signer was not a valid pointer.

```
OSStatus SecStreamCreateInputFile (short fileRefNum,
                                   SecStreamRef* pStream)
```

fileRefNum	A valid open fileRefNum.
pStream	Output: A pointer to an uninitialized SecStreamRef.

DESCRIPTION

Create a new stream object. It is not possible to write to this stream, only read. This function will not put up any HI.

RESULT CODE

noErr	0	No error.
memFullErr	-108	Not enough memory to complete this operation.
errSecInvalidPointer	-TBD	signer was not a valid pointer.

```
OSStatus SecStreamWriteData (SecStreamRef stream,
                             CFDataRef data)
```

stream	The SecStream object we operate on.
data	The data to write to stream.

DESCRIPTION

This function writes data to a stream. This function will not put up any HI.

RESULT CODE

noErr	0	No error.
dskFulErr	-34	Disk full.
ioErr	-36	I/O error.
fnOpnErr	-38	File not open.
posErr	-40	Attempt to position mark before start of file.
wPrErr	-44	Hardware volume lock.
fLckdErr	-45	File is locked.
vLckdErr	-46	Software volume lock.
rfNumErr	-51	Bad reference number.
wrPermErr	-61	Read/write permission doesn't allow writing.
memFullErr	-108	Not enough memory to complete this operation.
errSecInvalidStream	-TBD	The stream passed in is not valid.
errSecInvalidReference	-TBD	Passed in data object was invalid.

```
OSStatus SecStreamReadData (SecStreamRef stream,
                            SInt32 length,
                            CFMutableDataRef data)
```

`stream`	The `SecStream` object we operate on.
`length`	The amount of data wanted from the stream.
`data`	The data read from the stream will be appended to this.

DESCRIPTION
This function reads data from a stream. The length is just a hint of how much data is wanted less could be read without returning an error. The data read is appended to the passed in CFMutableData object data. This function will not put up any HI.

RESULT CODE

`noErr`	0	No error.
`ioErr`	-36	I/O error.
`fnOpnErr`	-38	File not open.
`eofErr`	-39	Logical end-of-file reached.
`posErr`	-40	Attempt to position mark before start of file.
`fLckdErr`	-45	File is locked.
`rfNumErr`	-51	Bad reference number.
`gfpErr`	-52	Error during GetFPos.
`memFullErr`	-108	Not enough memory to complete this operation.
`afpAccessDenied`	-5000	User does not have the correct access to the file.
`errSecInvalidStream`	-TBD	The stream passed in is not valid.
`errSecInvalidReference`	-TBD	Passed in data object was invalid.

```
UInt32 SecStreamGetTypeID (void)
```

DESCRIPTION
Get the type id of SecStream objects. This function will not put up any HI.

RESULT
Returns the typeID of SecStream.

Signer object manipulation

```
OSStatus SecSignerCreateMutable (KCItemRef cert,
                                 SecDigestAlgorithm digAlg,
                                 SecMutableSignerRef* pSigner)
```

`cert`	A KCItemRef that points to a valid certificate.
`digAlg`	The algorithm to use for digesting this signers signature.
`pSigner`	`Output:` A pointer to an uninitialized `SecSignerRef.`

DESCRIPTION
Create a new signer object. To add attributes to a signer use the SecSignerAttributes function call. The signer object created by this call must be released at some point by calling SecSignerRelease. This function will not put up any HI.

RESULT CODE

`noErr`	0	No error.
`memFullErr`	-108	Not enough memory to complete this operation.
`errSecInvalidPointer`	-TBD	`signer` was not a valid pointer.

```
OSStatus SecSignerGetAttributes (SecSignerRef signer,
                                 Boolean signed,
                                 SecAttributesRef* pMsgAttibutes)
```

`signer`	The `SecMutableSignerRef` object we operate on.
`signed`	Whether to return the signed or unsigned attribute list: - FALSE means return the unsigned attibute list. - TRUE means return the signed attibute list.
`pMsgAttibutes`	Output: Pointer to an uninitialized SecAttributes.

DESCRIPTION

This function returns a reference to a attributes object contains either the signed or unsigned attributes of a signer. This function will not put up any HI.

RESULT CODE

`noErr`	0	No error.
`memFullErr`	-108	Not enough memory to complete this operation.
`errSecInvalidStream`	-TBD	The outputStream passed in is not valid.
`errSecInvalidReference`	-TBD	Passed in object was invalid.
`errSecInvalidPointer`	-TBD	`signer` was not a valid pointer.

```
OSStatus SecSignerSetAttributes (SecMutableSignerRef signer,
                                 Boolean signed,
                                 SecAttributesRef msgAttibutes)
```

`signer`	The `SecMutableSignerRef` object we operate on.
`signed`	Whether to set the signed or unsigned attribute list: - FALSE means set the unsigned attibute list. - TRUE means set the signed attibute list.
`msgAttibutes`	An initialized SecAttributes object.

DESCRIPTION

The function sets the signed or unsigned attributes of a signer. This function will not put up any HI.

RESULT CODE

`noErr`	0	No error.
`memFullErr`	-108	Not enough memory to complete this operation.
`errSecInvalidReference`	-TBD	Passed in object was invalid.
`errSecInvalidStream`	-TBD	The outputStream passed in is not valid.

```
OSStatus SecSignerGetCertificate (SecSignerRef signer, KCItemRef* pCert)
```

`signer`	The `SecMutableSignerRef` object we operate on.
`pCert`	Output: Pointer to an uninitialized KCItemRef.

DESCRIPTION

Returns the signing certificate in cert. The caller should not call KCReleaseItem on the returned pCert. This function will not put up any HI.

RESULT CODE

noErr	0	No error.
errSecInvalidPointer	-TBD	signer was not a valid pointer.
errSecInvalidReference	-TBD	cert was an invalid pointer.
errSecMissingCert	-TBD	The signing certificate was not found. (Was not included in the message and is not in any available Keychain.)

```
OSStatus SecSignerGetCertificateChain (SecSignerRef signer,
                                       CFArrayRef* pCertChain)
```

signer	The SecMutableSignerRef object we operate on.
pCertChain	Output: Pointer to an uninitialized CFArrayRef.

DESCRIPTION

Returns the signing certificate in cert. The caller should not call CFRelease on the returned pCertChain. This function will not put up any HI.

RESULT CODE

noErr	0	No error.
errSecInvalidPointer	-TBD	signer was not a valid pointer.
errSecInvalidReference	-TBD	cert was an invalid pointer.
errSecMissingCert	-TBD	The signing certificate was not found. (Was not included in the message and is not in any available Keychain.)

```
OSStatus SecSignerGetStatus (SecSignerRef signer)
```

signer	The SecMutableSignerRef object we operate on.

DESCRIPTION

Returns an error code indicating whether this particular signers signature was valid. This function will not put up any HI.

RESULT CODE

noErr	0	No error.
errSecInvalidReference	-TBD	Passed in object was invalid.
errSecUnsupported	-TBD	This signer uses an unsupported cms message, digest algorithm or signature algorithm.
errSecMissingCert	-TBD	The signing certificate used was not found. (Was not included in the message and is not in any available Keychain.)
errSecCreateChainFail	-TBD	Cert chain could not be constructed up to a root an incomplete chain will be returned when SecSignerGetCertificateChain is called.
errSecInvalidCert	-TBD	One or more certificates in the signing certificate chain were invalid, check the certs to find out which one and why (expired revoked etc.)
errSecNotSigner	-TBD	Signer certificate is not signer of subject somewhere in the chain.
errSecNotTrusted	-TBD	Cert chain was not trusted by one or more selected policies.
errSecMissingAttribute	-TBD	An attribute required by the specified domain or by cms itself is not present in the signerInfo.
errSecMissingDigest	-TBD	The digest used by this signer was not calculated so it cannot be verified.
errSecDigestMismatch	-TBD	The digest in the signedAttributes did not match the digest of

		the data computed.
errSecInvalidSignature	-TBD	The signature was invalid because the digest of the signedAttributes did not match the decrypted data.
errSecAlgMismatch	-TBD	The algorithm specified in the SignerInfo did not match the algorithm of the public key in the signing certificate (treat this like errSecInvalidData).

```
UInt32 SecSignerGetTypeID (void)
```

DESCRIPTION
Get the type id of SecSigner objects. This function will not put up any HI.

RESULT
Returns the typeID of SecSigner.

Generic attributes object manipulation

```
OSStatus SecAttributesCreateMutable (SecMutableAttributesRef* pAttributes)
```

attributes	Output: A pointer to an uninitialized SecMutableAttributesRef.

DESCRIPTION
Create a new attributes object. This function will not put up any HI.

RESULT CODE

noErr	0	No error.
memFullErr	-108	Not enough memory to complete this operation.
errSecInvalidPointer	-TBD	Passed in object was invalid.

```
OSStatus SecAttributesAdd (SecMutableAttributesRef attributes,
                           SecOIDRef attibuteOID,
                           CFDataRef value)
```

attributes	The SecAttributesRef object we operate on.
attibuteOID	The OID (object identifier) of the attibute being added.
value	DER encoded value of the attribute being added.
isSigned	Iff true this is an signed attribute.

DESCRIPTION
Adds an attribute to a signer object. This function should only be called on a signer object obtained from a still valid cmsEncoder. This function will not put up any HI.

RESULT CODE

noErr	0	No error.
memFullErr	-108	Not enough memory to complete this operation.
errSecInvalidStream	-TBD	The outputStream passed in is not valid.
errSecInvalidReference	-TBD	Passed in object was invalid.

```
OSStatus SecAttributesCount (SecAttributesRef attributes,
                             UInt32* pCount)
```

attributes	The SecAttributesRef object we operate on.
pCount	Output: The number of attributes this signer provided.

DESCRIPTION
Returns the number of attributes in the attributes. This function will not put up any HI.

RESULT CODE

noErr	0	No error.
errSecInvalidReference	-TBD	Passed in object was invalid.

```
OSStatus SecAttributesGetAtIndex (SecAttributesRef attributes,
                                  UInt32 index,
                                  SecOIDRef* pAttributeOID,
                                  CFDataRef* pValue)
```

attributes	The SecAttributesRef object we operate on.
pAttributeOID	Output: The OID (object identifier) of the attibute.
pValue	Output: Value of the attribute.

DESCRIPTION
Return the attributeOID and value for a given attribute of a given type. This function will not put up any HI.

RESULT CODE

noErr	0	No error.
memFullErr	-108	Not enough memory to complete this operation.
errSecInvalidIndex	-TBD	The attribute index is invalid.
errSecInvalidReference	-TBD	Passed in object was invalid.

```
UInt32 SecAttributesGetTypeID (void)
```

DESCRIPTION
Get the type id of SecAttributes objects. This function will not put up any HI.

RESULT
Returns the typeID of SecAttributes.

S/Mime specific attributes object manipulation

```
OSStatus SecAttributesGetMessageDigest (SecAttributesRef attributes,
                                        CFDataRef* pDigest)
```

attributes	The SecAttributesRef object we operate on.
pDigest	Output: Value of the attribute.

DESCRIPTION
Return the raw octets of the message digest. The caller is responsible for releasing the returned value. This

function will not put up any HI.

RESULT CODE

noErr	0	No error.
memFullErr	-108	Not enough memory to complete this operation.
errSecNoSuchAttribute	-TBD	The attribute is not in the list.
errSecInvalidReference	-TBD	Passed in object was invalid.

```
OSStatus SecAttributesSetMessageDigest (SecMutableAttributesRef attributes,
                                        CFDataRef digest)
```

attributes	The SecAttributesRef object we operate on.
digest	New value of the attribute.

DESCRIPTION

Set the raw octets of the message digest. This function will not put up any HI.

RESULT CODE

noErr	0	No error.
memFullErr	-108	Not enough memory to complete this operation.
errSecNoSuchAttribute	-TBD	The attribute is not in the list.
errSecInvalidReference	-TBD	Passed in object was invalid.

```
OSStatus SecAttributesGetMsgSigDigest (SecAttributesRef attributes,
                                       CFDataRef* pSigDigest)
```

attributes	The SecAttributesRef object we operate on.
pSigDigest	Output: Value of the attribute.

DESCRIPTION

Return the raw octets of the message signature digest. The caller is responsible for releasing the returned value. This function will not put up any HI.

RESULT CODE

noErr	0	No error.
memFullErr	-108	Not enough memory to complete this operation.
errSecNoSuchAttribute	-TBD	The attribute is not in the list.
errSecInvalidReference	-TBD	Passed in object was invalid.

```
OSStatus SecAttributesSetMsgSigDigest (SecMutableAttributesRef attributes,
                                       CFDataRef sigDigest)
```

attributes	The SecAttributesRef object we operate on.
sigDigest	New value of the attribute.

DESCRIPTION

Set the raw octets of the message signature digest. This function will not put up any HI.

RESULT CODE

noErr	0	No error.

memFullErr	-108	Not enough memory to complete this operation.
errSecNoSuchAttribute	-TBD	The attribute is not in the list.
errSecInvalidReference	-TBD	Passed in object was invalid.

```
OSStatus SecAttributesCreateSigningTime (SecAttributesRef attributes,
                                         CFDateRef* pSigningTime)
```

attributes	The SecAttributesRef object we operate on.
pSigningTime	Output: Values of the attribute.

DESCRIPTION

Return an CFDateRef representing the signing time. The caller is responsible for releasing the returned value. This function will not put up any HI.

RESULT CODE

noErr	0	No error.
memFullErr	-108	Not enough memory to complete this operation.
errSecNoSuchAttribute	-TBD	The attribute is not in the list.
errSecInvalidData	-TBD	The attributes value was not a valid date.
errSecInvalidReference	-TBD	Passed in object was invalid.

```
OSStatus SecAttributesSetSigningTime (SecMutableAttributesRef attributes,
                                      CFDateRef signingTime)
```

attributes	The SecAttributesRef object we operate on.
signingTime	New value of the attribute.

DESCRIPTION

Set the signing time. This function will not put up any HI.

RESULT CODE

noErr	0	No error.
memFullErr	-108	Not enough memory to complete this operation.
errSecNoSuchAttribute	-TBD	The attribute is not in the list.
errSecInvalidReference	-TBD	Passed in object was invalid.

```
OSStatus SecAttributesGetCounterSignatures (SecAttributesRef attributes,
                                            CFArrayRef* pSignatures)
```

attributes	The SecAttributesRef object we operate on.
pSignatures	Output: Values of the attribute.

DESCRIPTION

Return an array of SecSignerRef's. The caller is responsible for releasing the returned value. This function will not put up any HI.

RESULT CODE

noErr	0	No error.
memFullErr	-108	Not enough memory to complete this operation.
errSecNoSuchAttribute	-TBD	The attribute is not in the list.

errSecInvalidReference	-TBD	Passed in object was invalid.

```
OSStatus SecAttributesSetCounterSignatures (SecMutableAttributesRef
attributes,
                                            CFArrayRef signatures)
```

attributes	The SecAttributesRef object we operate on.
signatures	New value of the attribute.

DESCRIPTION
Set an array of SecSignerRef's. This function will not put up any HI.

RESULT CODE

noErr	0	No error.
memFullErr	-108	Not enough memory to complete this operation.
errSecNoSuchAttribute	-TBD	The attribute is not in the list.
errSecInvalidReference	-TBD	Passed in object was invalid.

```
OSStatus SecAttributesGetContentHints (SecAttributesRef attributes,
                                       CFDataRef* pContentHints)
```

attributes	The SecAttributesRef object we operate on.
pContentHints	Output: Values of the attribute.

DESCRIPTION
Return a the contentHints. The caller is responsible for releasing the returned value. This function will not put up any HI.

RESULT CODE

noErr	0	No error.
memFullErr	-108	Not enough memory to complete this operation.
errSecNoSuchAttribute	-TBD	The attribute is not in the list.
errSecInvalidReference	-TBD	Passed in object was invalid.

```
OSStatus SecAttributesSetContentHints (SecMutableAttributesRef attributes,
                                       CFDataRef contentHints)
```

attributes	The SecAttributesRef object we operate on.
contentHints	New value of the attribute.

DESCRIPTION
Set the contentHints. This function will not put up any HI.

RESULT CODE

noErr	0	No error.
memFullErr	-108	Not enough memory to complete this operation.
errSecNoSuchAttribute	-TBD	The attribute is not in the list.
errSecInvalidReference	-TBD	Passed in object was invalid.

```
OSStatus SecAttributesGetContentReference (SecAttributesRef attributes,
                                           CFDataRef* pContentReference)
```

attributes	The SecAttributesRef object we operate on.
pContentReference	Output: Values of the attribute.

DESCRIPTION

Return a the contentReference. The caller is responsible for releasing the returned value. This function will not put up any HI.

RESULT CODE

noErr	0	No error.
memFullErr	-108	Not enough memory to complete this operation.
errSecNoSuchAttribute	-TBD	The attribute is not in the list.
errSecInvalidReference	-TBD	Passed in object was invalid.

```
OSStatus SecAttributesSetContentReference (SecMutableAttributesRef attributes,
                                           CFDataRef contentReference)
```

attributes	The SecAttributesRef object we operate on.
contentReference	New value of the attribute.

DESCRIPTION

Set the contentReference. This function will not put up any HI.

RESULT CODE

noErr	0	No error.
memFullErr	-108	Not enough memory to complete this operation.
errSecNoSuchAttribute	-TBD	The attribute is not in the list.
errSecInvalidReference	-TBD	Passed in object was invalid.

```
OSStatus SecAttributesGetSecurityLabel (SecAttributesRef attributes,
                                        CFDataRef* pSecurityLabel)
```

attributes	The SecAttributesRef object we operate on.
pSecurityLabel	Output: Values of the attribute.

DESCRIPTION

Return a the securityLabel. The caller is responsible for releasing the returned value. This function will not put up any HI.

RESULT CODE

noErr	0	No error.
memFullErr	-108	Not enough memory to complete this operation.
errSecNoSuchAttribute	-TBD	The attribute is not in the list.
errSecInvalidReference	-TBD	Passed in object was invalid.

```
OSStatus SecAttributesSetSecurityLabel (SecMutableAttributesRef attributes,
                                        CFDataRef securityLabel)
```

attributes	The SecAttributesRef object we operate on.
securityLabel	New value of the attribute.

DESCRIPTION
Set the securityLabel. This function will not put up any HI.

RESULT CODE

noErr	0	No error.
memFullErr	-108	Not enough memory to complete this operation.
errSecNoSuchAttribute	-TBD	The attribute is not in the list.
errSecInvalidReference	-TBD	Passed in object was invalid.

```
OSStatus SecAttributesGetSMIMECapabilities (SecAttributesRef attributes,
                                            CFArrayRef* pSmimeCapabilities)
```

attributes	The SecAttributesRef object we operate on.
pSmimeCapabilities	Output: Values of the attribute.

DESCRIPTION
Return an array of CFData's. The caller is responsible for releasing the returned value. This function will not put up any HI.

RESULT CODE

noErr	0	No error.
memFullErr	-108	Not enough memory to complete this operation.
errSecNoSuchAttribute	-TBD	The attribute is not in the list.
errSecInvalidReference	-TBD	Passed in object was invalid.

```
OSStatus SecAttributesSetSMIMECapabilities (SecMutableAttributesRef
attributes,
                                            CFArrayRef smimeCapabilities)
```

attributes	The SecAttributesRef object we operate on.
smimeCapabilities	New value of the attribute.

DESCRIPTION
Set an array of CFData. This function will not put up any HI.

RESULT CODE

noErr	0	No error.
memFullErr	-108	Not enough memory to complete this operation.
errSecNoSuchAttribute	-TBD	The attribute is not in the list.
errSecInvalidReference	-TBD	Passed in object was invalid.

```
OSStatus SecAttributesGetEquivalentLabels (SecAttributesRef attributes,
                                           CFArrayRef* pEquivalentLabels)
```

attributes	The SecAttributesRef object we operate on.
pEquivalentLabels	Output: Values of the attribute.

DESCRIPTION
Return an array of CFData's. The caller is responsible for releasing the returned value. This function will not put up any HI.

RESULT CODE

noErr	0	No error.
memFullErr	-108	Not enough memory to complete this operation.
errSecNoSuchAttribute	-TBD	The attribute is not in the list.
errSecInvalidReference	-TBD	Passed in object was invalid.

```
OSStatus SecAttributesSetEquivalentLabels (SecMutableAttributesRef attributes,
                                           CFArrayRef equivalentLabels)
```

attributes	The SecAttributesRef object we operate on.
equivalentLabels	New value of the attribute.

DESCRIPTION
Set an array of CFData. This function will not put up any HI.

RESULT CODE

noErr	0	No error.
memFullErr	-108	Not enough memory to complete this operation.
errSecNoSuchAttribute	-TBD	The attribute is not in the list.
errSecInvalidReference	-TBD	Passed in object was invalid.

```
OSStatus SecAttributesGetMailList (SecAttributesRef attributes,
                                   CFArrayRef* pMailList)
```

attributes	The SecAttributesRef object we operate on.
pMailList	Output: Values of the attribute.

DESCRIPTION
Return an array of CFData's. The caller is responsible for releasing the returned value. This function will not put up any HI.

RESULT CODE

noErr	0	No error.
memFullErr	-108	Not enough memory to complete this operation.
errSecNoSuchAttribute	-TBD	The attribute is not in the list.
errSecInvalidReference	-TBD	Passed in object was invalid.

```
OSStatus SecAttributesSetMailList (SecMutableAttributesRef attributes,
                                   CFArrayRef mailList)
```

attributes	The SecAttributesRef object we operate on.
mailList	New value of the attribute.

DESCRIPTION
Set an array of CFData. This function will not put up any HI.

RESULT CODE

noErr	0	No error.
memFullErr	-108	Not enough memory to complete this operation.
errSecNoSuchAttribute	-TBD	The attribute is not in the list.
errSecInvalidReference	-TBD	Passed in object was invalid.

Overloaded functions that work for all classes

These functions may be replaced by CFRetain, CFRelease, CFRetainCount, CFEquals and CFHash respectivaly if possible.

```
UInt32 SecGetTypeID (SecTypeRef sec)
```

sec — The SecType object we operate on.

DESCRIPTION
Get the id of the passed in SecTypeRef. This function will not put up any HI.

RESULT
Returns the typeID of sec.

```
CFStringRef SecCopyTypeIDDescription (UInt32 typeID)
```

typeID — The typeID of the object we want a description of.

DESCRIPTION
Create a copy of the string contains the name of the SecType object with typeID. This function will not put up any HI.

RESULT
Returns the string created.

```
SecTypeRef SecRetain (SecTypeRef sec)
```

sec — The SecType object we operate on.

DESCRIPTION
Increases the retain count of the sec object. This function will not put up any HI.

RESULT
Returns sec for convinience.

```
void SecRelease (SecTypeRef sec)
```

sec — The SecType object we operate on.

DESCRIPTION

Decreases the retain count of the sec object. When it reaches zero the object is deallocated. Any object created with a Sec<object>Create<something> call will have a retain count of one. Any other object will be owned by the object who returned it and might go away unless the caller uses SecRetain to take an additional referenmce to it. See the CF (CoreFoundation) documentation for details. This function will not put up any HI.

```
UInt32 SecRetainCount (SecTypeRef sec)
```

`sec`	The `SecType` object we operate on.

DESCRIPTION
Returns the retain count of the sec object. This function will not put up any HI.

```
Boolean SecEquals (SecTypeRef sec1, SecTypeRef sec2)
```

`sec1`	The `SecType` object we operate on.
`sec2`	The `SecType` object to compare with.

DESCRIPTION
Compare the values of type and other. This function will not put up any HI.

RESULT
Returns true if sec1 and sec2 are equal. Returns false otherwise.

```
UInt32 SecHash (SecTypeRef sec)
```

`sec`	The `SecType` object we operate on.

DESCRIPTION
Computes a hash of sec. This function will not put up any HI.

RESULT
Returns the computed hash value.

```
CFStringRef SecCopyDescription (SecTypeRef sec)
```

`sec`	The `SecType` object we operate on.

DESCRIPTION
Creates a human readable description of the sec object. This function will not put up any HI.

RESULT
Returns the copy of the description of sec (caller should CFRelease the result) or NULL if there was not enough memory availble to complete the operation.

Data structures and types

```
typedef struct __SecTypeRef*              SecTypeRef;
typedef struct __SecCmsEncoderRef*        SecCmsEncoderRef;
typedef struct __SecCmsDecoderRef*  SecCmsDecoderRef;
typedef struct __SecCmsEncoderRef*  SecCmsEncoderRef;
typedef struct __SecCmsDecoderRef*  SecCmsDecoderRef;
typedef struct __SecStreamRef*            SecStreamRef;
typedef const struct __SecSignerRef*      SecSignerRef;
typedef struct __SecSignerRef*            SecMutableSignerRef;
typedef const struct __SecAttributesRef*  SecAttributesRef;
typedef struct __SecAttributesRef*        SecMutableAttributesRef;
typedef const CFDataRef                   SecOIDRef;

extern const CFArrayCallBacks             kSecTypeArrayCallBacks;
extern const CFBagCallBacks               kSecTypeBagCallBacks;
extern const CFDictionaryKeyCallBacks     kSecTypeDictionaryKeyCallBacks;
extern const CFDictionaryValueCallBacks   kSecTypeDictionaryValueCallBacks;
extern const CFSetCallBacks               kSecTypeSetCallBacks;

enum {
      kSecStopOnPolicy        = 0,
      kSecStopOnNone          = 1,
      kSecStopOnFirstPass     = 2,
      kSecStopOnFirstFail     = 3
};
typedef UInt16    SecValidateStopOn;

enum
{
      kSecOptionProgress            = 1 << 0,
      kSecOptionAddRootCerts        = 1 << 1,
      kSecOptionAddLeafCerts        = 1 << 2, /* Leaf and intermidiate  */
      kSecOptionAddOtherCerts       = 1 << 3  /* Encryption or other certs */
};
typedef UInt32    SecOptions;

enum {
      kSecRawPKCS7            = 0,
      kSeciSign               = 1,
      kSecSMIME               = 2
};
typedef UInt32    SecMsgDomain;

enum {
      kSecCmsTypeData               = 0,
      kSecCmsTypeSignedData         = 1,
      kSecCmsTypeEnvelopedData      = 2,
      kSecCmsTypeDigestedData       = 3,
      kSecCmsTypeEncryptedData      = 4,
      kSecCmsTypeCtAuthData         = 5,
      kSecCmsTypeUnknown            = 6
};
typedef UInt32    SecMsgDomain;

enum {
      errSecInvalidPointer          = -13840,
```

```
        errSecInvalidReference          = -13841,
        errSecInvalidStream             = -13842,
        errSecUnsupported               = -13843,
        errSecInvalidData               = -13844,
        errSecTooMuchData               = -13845,
        errSecMissingData               = -13846,
        errSecNoSigners                 = -13847,
        errSecSignerFailed              = -13848,
        errSecInvalidPolicy             = -13849,
        errSecUnknownPolicy             = -13850,
        errSecInvalidStopOn             = -13851,
        errSecMissingCert               = -13852,
        errSecInvalidCert               = -13853,
        errSecNotSigner                 = -13854,
        errSecNotTrusted                = -13855,
        errSecMissingAttribute          = -13856,
        errSecMissingAttribute          = -13857,
        errSecDigestMismatch            = -13858,
        errSecInvalidSignature          = -13859,
        errSecAlgMismatch               = -13860,
        errSecInProgress                = -13861,
        errSecFinished                  = -13862,
        errSecNoSuchAttribute           = -13863,
        errSecCreateChainFail           = -13869
};
```

Creating a cms message

```
UInt32 SecCmsEncoderGetTypeID (void);

OSStatus SecCmsEncoderCreateForSign (Boolean includeData,
                                     Boolean contentOnly,
                                     SecCmsEncoderRef parentCmsEncoder,
                                     SecStreamRef outputStream,
                                     SecMsgDomain domain,
                                     SecCmsEncoderRef* pCmsEncoder);

OSStatus SecCmsEncoderCreateForEncrypt (SecAlgorithmIdentifier algorithm,
                                        Boolean contentOnly,
                                        SecCmsEncoder parentCmsEncoder,
                                        SecStreamRef outputStream,
                                        SecMsgDomain domain,
                                        SecCmsEncoderRef* pCmsEncoder);

OSStatus SecCmsEncoderAddRecipient (SecCmsEncoderRef cmsEncoder,
                                    KCItemRef cert,
                                    CFArrayRef policies,
                                    SecValidateStopOn stopOn);

OSStatus SecCmsEncoderAddCert (SecCmsEncoderRef cmsEncoder, KCItemRef cert);

OSStatus SecCmsEncoderAddCertChain (SecCmsEncoderRef cmsEncoder, KCItemRef
cert);

OSStatus SecCmsEncoderAddSigner (SecCmsEncoderRef cmsEncoder,
                                 SecSignerRef signer,
                                 Boolean addCertChain);

OSStatus SecCmsEncoderProcessData (SecCmsEncoderRef cmsEncoder, CFDataRef
data);

OSStatus SecCmsEncoderFinish (SecCmsEncoderRef cmsEncoder);
```

Decrypting or verifying a signed message or verifying a message with an external signature

```
OSStatus SecCmsDecoderCreate (UInt32 maxLevels,
                              void* reserved,
                              SecStreamRef outputStream,
                              SecMsgDomain domain,
                              SecCmsDecoderRef* pCmsDecoder);

OSStatus SecCmsDecoderAddDigestAlgorithm (SecCmsDecoderRef cmsDecoder,
                                          SecAlgorithmId digestAlgorithmId);

OSStatus SecCmsDecoderSetExternalSignature (SecCmsDecoderRef cmsDecoder,
                                            CFDataRef externalSignature);

OSStatus SecCmsDecoderProcessData (SecCmsDecoderRef cmsDecoder,
```

```
                                    CFDataRef data);

OSStatus SecCmsDecoderGetStatus (SecCmsDecoderRef cmsDecoder);

OSStatus SecCmsDecoderGetContentType (SecCmsDecoderRef cmsDecoder,
                                      SecCmsType* pContentType);

OSStatus SecCmsDecoderGetContentTypeOID (SecCmsDecoderRef cmsDecoder,
                                         SecOIDRef* pContentTypeOID);

OSStatus SecCmsDecoderGetContentCms (SecCmsDecoderRef cmsDecoder,
                                     SecCmsDecoderRef* pContentCmsDecoder);

OSStatus SecCmsDecoderGetCertificates (SecCmsDecoderRef cmsDecoder,
                                       CFArrayRef* pCertificates);

OSStatus SecCmsDecoderSigners (SecCmsDecoderRef cmsDecoder,
                               SecOptions options, CFArrayRef policies,
                               SecValidateStopOn stopOn,
                               CFArrayRef* pSigners);

UInt32 SecCmsDecoderGetTypeID (void);

OSStatus SecCmsDecoderCreate (CFDataRef externalSignature,
                              SecStreamRef outputStream,
                              SecMsgDomain domain,
                              SecCmsDecoderRef* pCmsDecoder);

OSStatus SecCmsDecoderProcessData (SecCmsDecoderRef cmsDecoder, CFDataRef
data);

OSStatus SecCmsDecoderGetCertificates (SecCmsDecoderRef cmsDecoder,
                                       CFArrayRef* pCertificates);

OSStatus SecCmsDecoderValidate (SecCmsDecoderRef cmsDecoder,
                                SecOptions options, CFArrayRef policies,
                                SecValidateStopOn stopOn,
                                CFArrayRef* pSigners);
```

Stream object manipulation

```
UInt32 SecStreamGetTypeID (void);

OSStatus SecStreamCreateOutputData (CFMutableDataRef data,
                                    SecStreamRef* pStream);

OSStatus SecStreamCreateFifo (SecStreamRef* pStream);

OSStatus SecStreamCreateOutputFile (short fileRefNum,
                                    SecStreamRef* pStream);

OSStatus SecStreamCreateInputFile (short fileRefNum,
                                   SecStreamRef* pStream);

OSStatus SecStreamWriteData (SecStreamRef stream,
```

```
                              CFDataRef data);

OSStatus SecStreamReadData (SecStreamRef stream,
                            SInt32 length,
                            CFMutableDataRef data);
```

Signer object manipulation

```
UInt32 SecSignerGetTypeID (void);

OSStatus SecSignerCreateMutable (KCItemRef cert,
                                 SecDigestAlgorithm digAlg,
                                 SecMutableSignerRef* pSigner);

OSStatus SecSignerGetAttributes (SecSignerRef signer,
                                 Boolean signed,
                                 SecAttributesRef* pMsgAttibutes);

OSStatus SecSignerSetAttributes (SecMutableSignerRef signer,
                                 Boolean signed,
                                 SecAttributesRef pMsgAttibutes);

OSStatus SecSignerGetCertificate (SecSignerRef signer, KCItemRef* pCert);

OSStatus SecSignerGetCertificateChain (SecSignerRef signer,
                                       CFArrayRef* pCertChain);

OSStatus SecSignerGetStatus (SecSignerRef signer);
```

Generic attributes object manipulation

```
UInt32 SecAttributesGetTypeID (void);

OSStatus SecAttributesCreateMutable (SecMutableAttributesRef* pAttributes);

OSStatus SecAttributesAdd (SecMutableAttributesRef attributes,
                           SecOIDRef attibuteOID,
                           CFDataRef value);

OSStatus SecAttributesCount (SecAttributesRef attributes,
                             UInt32* pCount);

OSStatus SecAttributesGetAtIndex (SecAttributesRef attributes,
                                  UInt32 index,
                                  SecOIDRef* pAttributeOID,
                                  CFDataRef* pValue);
```

S/Mime specific attributes object manipulation

```
OSStatus SecAttributesGetMessageDigest (SecAttributesRef attributes,
                                        CFDataRef* pDigest);

OSStatus SecAttributesSetMessageDigest (SecMutableAttributesRef attributes,
```

```
                                        CFDataRef digest);

OSStatus SecAttributesGetMsgSigDigest (SecAttributesRef attributes,
                                        CFDataRef* pSigDigest);

OSStatus SecAttributesSetMsgSigDigest (SecMutableAttributesRef attributes,
                                        CFDataRef sigDigest);

OSStatus SecAttributesCreateSigningTime (SecAttributesRef attributes,
                                         CFDateRef* pSigningTime);

OSStatus SecAttributesSetSigningTime (SecMutableAttributesRef attributes,
                                      CFDateRef signingTime);

OSStatus SecAttributesGetCounterSignatures (SecAttributesRef attributes,
                                            CFArrayRef* pSignatures);

OSStatus SecAttributesSetCounterSignatures (SecMutableAttributesRef
attributes,
                                            CFArrayRef signatures);

OSStatus SecAttributesGetContentHints (SecAttributesRef attributes,
                                        CFDataRef* pContentHints);

OSStatus SecAttributesSetContentHints (SecMutableAttributesRef attributes,
                                        CFDataRef contentHints);

OSStatus SecAttributesGetContentReference (SecAttributesRef attributes,
                                           CFDataRef* pContentReference);

OSStatus SecAttributesSetContentReference (SecMutableAttributesRef attributes,
                                           CFDataRef contentReference);

OSStatus SecAttributesGetSecurityLabel (SecAttributesRef attributes,
                                        CFDataRef* pSecurityLabel);

OSStatus SecAttributesSetSecurityLabel (SecMutableAttributesRef attributes,
                                        CFDataRef securityLabel);

OSStatus SecAttributesGetSMIMECapabilities (SecAttributesRef attributes,
                                            CFArrayRef* pSmimeCapabilities);

OSStatus SecAttributesSetSMIMECapabilities (SecMutableAttributesRef
attributes,
                                            CFArrayRef smimeCapabilities);

OSStatus SecAttributesGetEquivalentLabels (SecAttributesRef attributes,
                                           CFArrayRef* pEquivalentLabels);

OSStatus SecAttributesSetEquivalentLabels (SecMutableAttributesRef attributes,
                                           CFArrayRef equivalentLabels);

OSStatus SecAttributesGetMailList (SecAttributesRef attributes,
                                   CFArrayRef* pMailList);

OSStatus SecAttributesSetMailList (SecMutableAttributesRef attributes,
```

```
                                    CFArrayRef mailList);
```

Overloaded functions that work for all classes

```
UInt32 SecGetTypeID (SecTypeRef sec);

CFStringRef SecCopyTypeIDDescription (UInt32 typeID);

SecTypeRef SecRetain (SecTypeRef sec);

void SecRelease (SecTypeRef sec);

UInt32 SecRetainCount (SecTypeRef sec);

Boolean SecEqual (SecTypeRef sec1, SecTypeRef sec2);

UInt32 SecHash (SecTypeRef sec);

CFStringRef SecCopyDescription (SecTypeRef sec);
```