

## NAME

CMS\_verify, CMS\_get0\_signers - verify a CMS SignedData structure

## SYNOPSIS

```
#include <openssl/cms.h>
```

```
int CMS_verify(CMS_ContentInfo *cms, STACK_OF(X509) *certs, X509_STORE *store, BIO *indata,
```

```
STACK_OF(X509) *CMS_get0_signers(CMS_ContentInfo *cms);
```

## DESCRIPTION

*CMS\_verify()* verifies a CMS SignedData structure. **cms** is the CMS\_ContentInfo structure to verify. **certs** is a set of certificates in which to search for the signing certificate(s). **store** is a trusted certificate store used for chain verification. **indata** is the detached content if the content is not present in **cms**. The content is written to **out** if it is not NULL.

**flags** is an optional set of flags, which can be used to modify the verify operation.

*CMS\_get0\_signers()* retrieves the signing certificate(s) from **cms**, it must be called after a successful *CMS\_verify()* operation.

## VERIFY PROCESS

Normally the verify process proceeds as follows.

Initially some sanity checks are performed on **cms**. The type of **cms** must be SignedData. There must be at least one signature on the data and if the content is detached **indata** cannot be NULL.

An attempt is made to locate all the signing certificate(s), first looking in the **certs** parameter (if it is not NULL) and then looking in any certificates contained in the **cms** structure itself. If any signing certificate cannot be located the operation fails.

Each signing certificate is chain verified using the **smimesign** purpose and the supplied trusted certificate store. Any internal certificates in the message are used as untrusted CAs. If CRL checking is enabled in **store** any internal CRLs are used in addition to attempting to look them up in **store**. If any chain verify fails an error code is returned.

Finally the signed content is read (and written to **out** if it is not NULL) and the signature's checked.

If all signature's verify correctly then the function is successful.

Any of the following flags (ored together) can be passed in the **flags** parameter to change the default verify behaviour.

If **CMS\_NOINTERN** is set the certificates in the message itself are not searched when locating the signing certificate(s). This means that all the signing certificates must be in the **certs** parameter.

If **CMS\_NOCRL** is set and CRL checking is enabled in **store** then any CRLs in the message itself are ignored.

If the **CMS\_TEXT** flag is set MIME headers for type **text/plain** are deleted from the content. If the content is not of type **text/plain** then an error is returned.

If **CMS\_NO\_SIGNER\_CERT\_VERIFY** is set the signing certificates are not verified.

If **CMS\_NO\_ATTR\_VERIFY** is set the signed attributes signature is not verified.

If **CMS\_NO\_CONTENT\_VERIFY** is set then the content digest is not checked.

## NOTES

One application of **CMS\_NOINTERN** is to only accept messages signed by a small number of certificates. The acceptable certificates would be passed in the **certs** parameter. In this case if the signer is not one of the certificates supplied in **certs** then the verify will fail because the signer

cannot be found.

In some cases the standard techniques for looking up and validating certificates are not appropriate; for example an application may wish to lookup certificates in a database or perform customised verification. This can be achieved by setting and verifying the signers certificates manually using the signed data utility functions.

Care should be taken when modifying the default verify behaviour, for example setting **CMS\_NO\_CONTENT\_VERIFY** will totally disable all content verification and any modified content will be considered valid. This combination is however useful if one merely wishes to write the content to **out** and its validity is not considered important.

Chain verification should arguably be performed using the signing time rather than the current time. However since the signing time is supplied by the signer it cannot be trusted without additional evidence (such as a trusted timestamp).

## RETURN VALUES

*CMS\_verify()* returns 1 for a successful verification and zero if an error occurred.

*CMS\_get0\_signers()* returns all signers or NULL if an error occurred.

The error can be obtained from [ERR\\_get\\_error\(3\)](#)

## BUGS

The trusted certificate store is not searched for the signing certificate, this is primarily due to the inadequacies of the current **X509\_STORE** functionality.

The lack of single pass processing means that the signed content must all be held in memory if it is not detached.

## SEE ALSO

[ERR\\_get\\_error\(3\)](#), [CMS\\_sign\(3\)](#)

## HISTORY

*CMS\_verify()* was added to OpenSSL 0.9.8