

NAME

CMS_decrypt, CMS_decrypt_set1_pkey_and_peer, CMS_decrypt_set1_pkey,
CMS_decrypt_set1_password – decrypt content from a CMS envelopedData structure

SYNOPSIS

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

int CMS_decrypt(CMS_ContentInfo *cms, EVP_PKEY *pkey, X509 *cert,
                BIO *dcont, BIO *out, unsigned int flags);
int CMS_decrypt_set1_pkey_and_peer(CMS_ContentInfo *cms,
                                   EVP_PKEY *pk, X509 *cert, X509 *peer);
int CMS_decrypt_set1_pkey(CMS_ContentInfo *cms, EVP_PKEY *pk, X509 *cert);
int CMS_decrypt_set1_password(CMS_ContentInfo *cms,
                              unsigned char *pass, ossl_ssize_t passlen);
```

DESCRIPTION

CMS_decrypt() extracts the decrypted content from a CMS EnvelopedData or AuthEnvelopedData structure. It uses **CMS_decrypt_set1_pkey()** to decrypt the content with the recipient private key *pkey* if *pkey* is not NULL. In this case, the associated certificate is recommended to provide in *cert* – see the NOTES below. *out* is a BIO to write the content to and *flags* is an optional set of flags. If *pkey* is NULL the function assumes that decryption was already done (e.g., using **CMS_decrypt_set1_pkey()** or **CMS_decrypt_set1_password()**) and just provides the content unless *cert*, *dcont*, and *out* are NULL as well. The *dcont* parameter is used in the rare case where the encrypted content is detached. It will normally be set to NULL.

CMS_decrypt_set1_pkey_and_peer() decrypts the CMS_ContentInfo structure *cms* using the private key *pkey*, the corresponding certificate *cert*, which is recommended but may be NULL, and the (optional) originator certificate *peer*. On success, it also records in *cms* the decryption key *pkey*, and then should be followed by **CMS_decrypt(cms, NULL, NULL, dcont, out, flags)**. This call deallocates any decryption key stored in *cms*.

CMS_decrypt_set1_pkey() is the same as **CMS_decrypt_set1_pkey_and_peer()** with *peer* being NULL.

CMS_decrypt_set1_password() decrypts the CMS_ContentInfo structure *cms* using the secret *pass* of length *passlen*. On success, it also records in *cms* the decryption key used, and then should be followed by **CMS_decrypt(cms, NULL, NULL, dcont, out, flags)**. This call deallocates any decryption key stored in *cms*.

NOTES

Although the recipients certificate is not needed to decrypt the data it is needed to locate the appropriate (of possible several) recipients in the CMS structure.

If *cert* is set to NULL all possible recipients are tried. This case however is problematic. To thwart the MMA attack (Bleichenbacher's attack on PKCS #1 v1.5 RSA padding) all recipients are tried whether they succeed or not. If no recipient succeeds then a random symmetric key is used to decrypt the content: this will typically output garbage and may (but is not guaranteed to) ultimately return a padding error only. If **CMS_decrypt()** just returned an error when all recipient encrypted keys failed to decrypt an attacker could use this in a timing attack. If the special flag **CMS_DEBUG_DECRYPT** is set then the above behaviour is modified and an error is returned if no recipient encrypted key can be decrypted **without** generating a random content encryption key. Applications should use this flag with **extreme caution** especially in automated gateways as it can leave them open to attack. See **EVP_PKEY_decrypt(3)** for more details.

It is possible to determine the correct recipient key by other means (for example looking them up in a database) and setting them in the CMS structure in advance using the CMS utility functions such as **CMS_set1_pkey()**, or use **CMS_decrypt_set1_password()** if the recipient has a symmetric key. In these cases both *cert* and *pkey* should be set to NULL.

To process KEKRecipientInfo types **CMS_set1_key()** or **CMS_RecipientInfo_set0_key()** and **CMS_RecipientInfo_decrypt()** should be called before **CMS_decrypt()** and *cert* and *pkey* set to NULL.

The following flags can be passed in the *flags* parameter.

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.

RETURN VALUES

CMS_decrypt(), **CMS_decrypt_set1_pkey_and_peer()**, **CMS_decrypt_set1_pkey()**, and **CMS_decrypt_set1_password()** return either 1 for success or 0 for failure. The error can be obtained from **ERR_get_error**(3).

BUGS

The **set1_** part of these function names is misleading and should better read: **with_**.

The lack of single pass processing and the need to hold all data in memory as mentioned in **CMS_verify()** also applies to **CMS_decrypt()**.

SEE ALSO

ERR_get_error(3), **CMS_encrypt**(3), **EVP_PKEY_decrypt**(3)

HISTORY

CMS_decrypt_set1_pkey_and_peer() and **CMS_decrypt_set1_password()** were added in OpenSSL 3.0.

COPYRIGHT

Copyright 2008–2026 The OpenSSL Project Authors. All Rights Reserved.

Licensed under the Apache License 2.0 (the "License"). You may not use this file except in compliance with the License. You can obtain a copy in the file `LICENSE` in the source distribution or at [<https://www.openssl.org/source/license.html>](https://www.openssl.org/source/license.html).