OpenSSL - cms

Utilitaire CMS (Cryptographic Message Syntax)

OPTIONS

-encrypt Chiffre un mail avec le certificat donné. Le fichier d'entrée est le message à chiffrer. Le fichier de sortie est le mail chiffré au format MIME. Le type de CMS actuel est **EnvelopedData**.

-decrypt Déchiffre un mail en utilisant le certificat et la clé privée spécifiée. Le fichier d'entrée est un mail au format MIME. le fichier de sortie est le mail déchiffré.

-sign Signe un mail en utilisant le certificat et la clé privée spécifiée. Le fichier d'entrée est le message à signer, le fichier de sortie est le message signé au format MIME.

-verify Vérifie le mail signé. L'entrée est le mail signé, le fichier de sortie et la sortie est la donnée signée.

-cmsout Prend un message en entrée et écrit une structure CMS encodée PEM

-resign Re-signe un message : prend un message existant et un ou plusieurs nouveaux signataires.

-data_create Créé un type CMS Data

-data_out Type une donnée et sort le contenu

-digest_create Crée un type CMS DigestedData

-digestVerify Vérifie un CMS DigestedData et sort le contenu

-compress Créé un CMS CompressedData. OpenSSL doit être compilé avec zlib pour que cette option fonctionne.

-uncompress Décompresse un CMS CompressedData.

-EncryptedData_encrypt Chiffre le contenu en utilisant une clé symétrique et un algorithme CMS EncryptedData et sort le contenu.

-sign_receipt Génère et sort un reçu signé pour le message spécifié. Le message d'entrée doit contenir une requête de reçu signée.

-verify_receipt receipt Vérifie un reçu signé dans le fichier spécifié. Le message d'entrée doit contenir une requête de reçu.

-in filename Fichier d'entrée

-inform SMIME|PEM|DER Format du fichier d'entrée, à utiliser avec -receipt_verify

-out filename Fichier de sortie

-outform SMIME|PEM|DER Spécifie le format du fichier de sortie (défaut SMIME)

-stream -indef -stream et indef sont équivalent et permettent le streaming I/O pour les opérations d'encodage. Cela permet un traitement en une passe de donnée sans nécessiter de maintenir tout le contenu en mémoire.

-noindef Désactive le streaming I/O.

-content filename spécifie un fichier contenant le contenu détaché. Seulement utile avec -verify et si la structure CMS utilise une signature détachée du contenu.

-text Ajoute des en-têtes text/plain au message fournis.

-noout Ne sort par la structure CMS parsée. Utile avec -print

-print pour -cmsout, affiche tous les champs de la structure CMS.

-CAfile file Un fichier contenant les certificats trustés. Utile avec -verify

-CApath dir Un répertoire contenant les certificats CA. Utile avec -verify

-md digest Algorithme digest à utiliser pour signer ou re-signer.

-[ciphers] L'algorithme de chiffrement à utiliser. Voir enc pour la liste des chiffrements supportés. (Défaut : 3DES)

-nointern En vérifiant un message, les certificats inclus dans le message sont utilisé pour vérifier la signature. Avec cette option, seul les certificats spécifiés avec -certfile sont utilisés.

-no_signer_cert_verify Ne vérifie pas le certificat du message signé.

-nocerts En signant un message, le certificat du signataire est inclus. Cette option l'empêche.

-noattr Normalement quand un message est signé, des attributs sont inclus dont la date de signature et les algorithmes symétriques supportés. Cette option ne les inclus pas.

-nosmimecap Exclus la liste des algorithmes supportés pour les attributs signés. D'autres options telles que la date de signature et le type de contenu sont inclus.

-binary Normalement, le message d'entrée est convertit au format canonique qui utilise CR et LF comme fin de ligne comme requis par la spécification S/MIME. Avec cette option, aucune conversion n'est faite.

-nodetach En signant un message en utilisant une signature opaque la forme est plus résistante aux translations par les relais, mais ne peut pas être lus par les agents qui ne supportent pas S/MIME. Sans cette option, la signature en texte clair est utilisée.

-certfile file Permet de spécifier des certificats additionnels, qui seront inclus dans le message. Ces certificats devraient être au format PEM.

-certsout file Tout certificat contenus dans le message sont écris dans le fichier spécifié

-signer file Un certificat de signature. Peut-être spécifiée plusieurs fois.

-recip file Le certificat pour le déchiffrement d'un message. Ce certificat doit matcher un des destinataires du message.

-keyid Utilise le subject key identifier pour identifier les certificats au lieu du nom et du numéro de série. Le certificat doit inclure une extension subject key identifier.

-receipt_request_all -receipt_request_first Pour que l'option -sign inclue une requête de reçu. Indique que les requêtes devraient être fournies par tous les destinataires ou le premier tiers des destinataires.

-receipt_request_to_emailaddress Ajoute une adresse mail où les reçus signés devraient être envoyés. Doit être fournis si un reçu signé est demandé.

-receipt_request_print Pour -verify, affiche le contenu des demandes de reçus signés.

-secretkey key Spécifie la clé symétrique à utiliser. La clé doit être fournie en hexa.

-secretkeyid id L'identifiant de clé pour la clé symétrique fournies pour le type KEKRecipentInfo.

-econtent_type type Définis le type de contenu encapsulé. Peut-être un OID soit en texte soit sous forme numérique.

-inkey file La clé privée à utiliser pour signer ou déchiffrer.

-passin arg Le mot de passe pour la clé privée.

-rand file(s) Un ou plusieurs fichiers contenant des données aléatoires utilisée pour le générateur de nombre aléatoires, ou un socket EGD.

cert.pem... Un ou plusieurs certificats de destinataire de message à utiliser pour chiffrer un message

-to, -from -subject Les en-têtes du mail.

-purpose, -ignore_critical, -issuer_checks, -crl_check, -crl_check_all, -policy_check, -extended_crl, -x509_strict, -policy -check_ss_sig Diverse options de validation de la chaîne de certificat. Voir verify.

Codes de sortie

0 succès de l'opération

1 Erreur en parsant les options

2 Un des fichiers d'entrée ne peut pas être lu

3 Une erreur s'est produite en créant le fichier CMS ou en lisnt le message MIME

4 Erreur durant la vérification ou le déchiffrement du message

5 Le message a été vérifié correctement mais une erreur s'est produite en écrivant les certificats des signeurs.

Compatibilité avec le format PKCS #7

l'utilitaire **smime** peut seulement traiter l'ancien format pkcs #7. L'utilitaire **cms** supporte le format CMS.

L'utilisation de **-keyid** avec **-sign** ou **-encrypt**

L'option **-outform PEM**

L'option **-compress**

L'option **-secretkey** avec **-encrypt**

-EncryptedData_create et **-data_create** ne peuvent être traitée par les commandes **smime**.

Exemples

Créer un message signé en texte clair :
openssl cms -sign -in message.txt -text -out mail.msg -signer mycert.pem
Créer un message signé opaque :
openssl cms -sign -in message.txt -text -out mail.msg -nodetach -signer mycert.pem
Créer un message signé, incluant des certificats additionnels et lire la clé privée depuis un autre fichier :
openssl cms -sign -in in.txt -text -out mail.msg -signer mycert.pem -inkey mykey.pem -certfile mycerts.pem
Créer un message signé avec 2 signataires, utiliser un identifier de clé :
openssl cms -sign -in message.txt -text -out mail.msg -signer mycert.pem -signer othercert.pem -keyid
Envoyer un message signé sous Unix directement à sendmail, incluant les en-têtes :
openssl cms -sign -in in.txt -text -signer mycert.pem -from steve@openssl.org -to someone@somewhere -subject "Signed message" | sendmail someone@somewhere
Vérifier un message et extraire le certificat du signataire en cas de réussite :
openssl cms -verify -in mail.msg -signer user.pem -out signedtext.txt
Envoyer un mail chiffré avec 3DES :
openssl cms -encrypt -in in.txt -from steve@openssl.org -to someone@somewhere -subject "Encrypted message" -des3 user.pem -out mail.msg
Signer et chiffrer un mail :
openssl cms -sign -in ml.txt -signer my.pem -text | openssl cms -encrypt -out mail.msg -from steve@openssl.org -to someone@somewhere -subject "Signed and Encrypted message" -des3 user.pem
Déchiffrer un mail :
openssl cms -decrypt -in mail.msg -recip mycert.pem -inkey key.pem
La sortie de la Netscape est une structure PKCS #7 avec le format de signature détachée. Pour vérifier la signature, mettre la structure codé en base 64 entre :
—–BEGIN PKCS7—–
—–END PKCS7—–
Et utiliser la commande :
openssl cms -verify -inform PEM -in signature.pem -content content.txt
Alternativement vous pouvez décoder la signature et utiliser :
openssl cms -verify -inform DER -in signature.der -content content.txt
Créer et chiffrer un message en utilisant Camellia 128 bits :
openssl cms -encrypt -in plain.txt -camellia128 -out mail.msg cert.pem
Ajouter un signataire à un message existant :
openssl cms -resign -in mail.msg -signer newsign.pem -out mail2.msg