

Utilisation de `mutt` version ≥ 1.5 avec les fonctionnalités S/MIME de signature et chiffrement de messages

maurice.libes_at_com.univ-mrs.fr &
albert.shih_at_math.jussieu.fr

Oct. 2004 - v0.9

1 Introduction

Ce mini-Howto a pour objectif d'indiquer comment utiliser `mutt` avec la gestion des certificats s/mime. Il n'a pas vocation à être totalement objectif et complet. Si vous trouvez des erreurs ou souhaitez rajouter des compléments d'information, veuillez en informer les auteurs aux adresses ci-dessus, nous mettrons à jour cette documentation. On peut également trouver les informations basiques pour utiliser `mutt` avec s/mime ici :

```
http://cvs.gnupg.org/cgi-bin/viewcvs.cgi/mutt/  
doc/smime-notes.txt?cvsroot=Mutt&rev=3.5
```

ou dans le fichier `doc/smime-notes.txt` de la distribution source.

2 Configuration de base

Récupérer et compiler une distribution `mutt-1.5`

Les fonctionnalités de gestion des certificats S/MIME sont intégrées à partir de la version 1.5 de `mutt`. Si vous faites une installation de `mutt` à partir des sources, il ne faut pas oublier d'intégrer le support SSL lors de la configuration

```
./configure --prefix=/opt/mutt --enable-pop --enable-imap --with-ssl
```

Rajout de nouveaux paramètres pour s/mime dans `.muttrc`

Insérer le contenu du fichier¹ `./contrib/smime.rc` présent dans le paquetage `mutt-1.5`, à votre fichier de configuration de `mutt` qui se trouve dans votre `home-dir` sous le nom `.muttrc`. Cela ajoute à la configuration de `mutt` les nouveaux paramètres nécessaires pour la gestion des certificats,

```
set crypt_autosign = yes  
set crypt_replyencrypt = yes  
set crypt_replysign = yes  
set crypt_replysignencrypted = yes  
set crypt_verify_sig = yes  
set smime_keys=~/.smime/keys"
```

¹Durant cette documentation nous ferons souvent référence à des fichiers qui sont distribués avec les version ≥ 1.5 de `mutt`. Ces fichiers, suivant votre type d'installation, votre système, peuvent se trouver à d'autres endroits, seuls leurs noms sont inchangés

Initialisation des répertoires nécessaires pour s/mime

Lancer la commande `smime_keys init` (présente dans le répertoire `bin` du paquetage `mutt`). L'option «`init`» crée les répertoires personnels nécessaires pour stocker vos futurs certificats et clés:

```
~/.smime/certificates
~/.smime/keys
```

Vérifier dans `.muttrc` que ces répertoires dans lesquels `mutt` doit chercher les certificats et les clés privées sont correctement mentionnés:

```
set smime_certificates=~/.smime/certificates"
set smime_keys=~/.smime/keys"
```

3 Pour signer les messages

Intégration d'un certificat personnel et clé privée dans mutt

On suppose ici que vous avez déjà un certificat personnel, obtenu auprès d'une autorité de certification quelconque (Thawte, CNRS...). On suppose que vous avez déjà intégré votre certificat dans `mozilla` ou `netscape`. Il est probable que vous ayez déjà essayé et utilisé des certificats avec `mozilla` ou `netscape` avant de le faire avec `mutt`... et si vous êtes là c'est que vous appréciez `mutt`, et que vous voulez faire la même chose qu'avec les MUA à boutons ;-)

Il faut tout d'abord extraire votre certificat personnel de `mozilla` et l'intégrer dans `mutt`.² Voici une méthode pour exporter votre certificat avec `Mozilla`, il faut aller dans le menu :

- *Edition/Préférences/confidentialité et sécurité*
- *certificats et cliquer sur le bouton «gestion des certificats»*
- *exporter (ou backup)*

Lors de cette exportation il vous est tout d'abord demandé la passphrase que vous utilisez dans `mozilla` pour protéger votre coffre de certificats, ensuite une seconde passphrase vous sera demandé pour protéger la clé privée qui sera contenue dans le fichier exporté. Un nom de fichier vous sera naturellement demandé. L'exportation sous `mozilla` des certificats se fait au format `pkcs12`. L'extension de votre fichier sera `p12`. Pour la suite, nous supposons pour l'exemple que le nom du fichier contenant le certificat personnel sera `certiflibes-ssi.p12`

Après avoir sauvegardé votre certificat dans un fichier, il faut l'intégrer dans le coffre à certificats de `mutt`. Pour cela, lancer la commande :

```
smime_keys add_p12 certiflibes-ssi.p12
```

Si jamais ce script affiche une erreur en détectant que vous utilisez déjà une version précédente de `mutt` (1.4), il suffit de donner le chemin vers le binaire de la nouvelle version ≥ 1.5 , avec la variable d'environnement `MUTT_CMDLINE` initialisée avec le répertoire contenant la nouvelle version de `mutt`.

```
export MUTT_CMDLINE=/opt/mutt-1.5/bin/mutt
```

Le script `smime_keys` demande :

- la passphrase pour ouvrir le certificat (passphrase que vous avez donné lors de l'exportation depuis `mozilla`)

²On trouvera une autre documentation ici : http://www.verisignlabs.com/Projects/smime_docs/linux.html

- la passphrase pour sécuriser l'accès à la clé privée qui va être extraite. Cette passphrase sera demandée chaque fois³ qu'on signera un message avec `mutt`
- un label pour désigner la clé privée par une simple chaîne de caractères, «momo» est plus simple que `991c8ffb.0`⁴.

Ces opérations effectuées, votre certificat personnel et la clé privée se retrouvent dans les répertoires de `mutt` créés précédemment

```
~/.smime/certificates
~/.smime/keys
```

Vous devriez également avoir un fichier `.smime/key/.index` dans lequel on trouve la liste des clés qui ont déjà été intégrées à `mutt`, associées à leur label

```
maurice.libes@com.univ-mrs.fr 991c8ffb.0 momo
```

Enfin il faut maintenant indiquer à `mutt` quelle clé privée personnelle on va utiliser par défaut pour signer les mails. On utilise pour cela l'option `smime_default_key` dans `.muttrc`.

```
set smime_default_key="991c8ffb.0"
```

Avec ce qui précède, on peut déjà signer des messages avec notre clé privée. Cependant, on ne peut pas encore vérifier/authentifier les messages signés que l'on reçoit, car pour cela il faut donner à `mutt` les certificats des autorités de certification qui ont servi à signer les certificats des émetteurs de mail.

4 Intégration des certificats des autorités de certification

Pour authentifier/vérifier les signatures `s/mime` des mails que l'on reçoit, il faut commencer par intégrer dans `mutt` les certificats des Autorités de Certifications (CA) qui ont servi à signer les certificats des utilisateurs.

Certificats des Autorités de Certification du CNRS

Par exemple, avec les certificats émis par la PKI CNRS, il faut intégrer dans `mutt` l'ensemble des certificats «racines» des autorités de certifications du CNRS:

- CNRS,
- CNRS-Standard,
- CNRS-Projets,
- CNRS-PLus,
- CNRS-SSI

certificats que l'on trouvera sur le site de l'autorité de certification du CNRS,

```
http://igc.services.cnrs.fr/SSI/recherche.html
```

Il est nécessaire de récupérer les différents certificats dans un fichier. Ces derniers sont au format `pem`⁵, et peuvent alors être intégrés directement à `mutt` avec la commande :

³Suivant la configuration que vous ferez dans `mutt` il se peut que cette passphrase ne vous soit demandée qu'une fois par session de `mutt`

⁴Il est parfaitement normal si vous n'avez pas le même numéro

⁵Attention, même s'ils sont au format `pem`, ils ont une extension `.crt`

```
smime_keys add_cert CNRS.crt
smime_keys add_cert CNRS-Projets.crt
smime_keys add_cert SSI.crt
etc...
```

Les clés publiques des certificats racines des autorités de certification sont alors placées dans `~/smime/certificates`. Vérifier le fichier de configuration `.muttrc` qui indique à `mutt` le chemin où sont installés les clés publiques des autorités de certification:

```
set smime_ca_location="~/smime/certificates"
```

Certificats des Autorités de Certification du CRU (Comité Réseau Université)

On trouvera sur les sites suivants

```
http://igc.cru.fr/ac-utilisateur/
http://igc.cru.fr/ac-utilisateur-plus/
```

les certificats racines des autorités de certification du CRU que l'on peut récupérer et intégrer dans `mutt` de la même façon que ci dessus.

Certificats d'Autorités de Certification commerciaux et privés

Afin de pouvoir authentifier les signatures `s/mime` composées avec des certificats délivrés par des opérateurs privés comme Thawte, Verisign, etc... Il faut se procurer les certificats racines de ces Autorités de Certification (CA) . Ces certificats sont la plupart du temps intégrés dans des navigateurs connus comme mozilla, netscape, internet explorer... On les trouve également dans `mutt` sous la forme d'un fichier "ca-bundle.crt" qui contient un grand nombre de certificats de diverses CA

```
mutt-1.5.6/contrib/ca-bundle.crt
```

Il est alors possible d'éditer ce fichier, extraire (par `vi`, `script shell` ou `perl...`) les lignes concernant le certificat d'une CA particulière. On sauve ces lignes dans un fichier `thawte_freemail.pem` par exemple. Ce fichier peut alors directement être intégré dans `mutt` avec la commande indiquée précédemment

```
smime_keys add_cert thawte_freemail.pem
```

NB: le script `perl smime_keys` utilise en fait simplement `openssl`, et lance la commande

```
/usr/bin/openssl x509 -noout -hash -in thawte_freemail.pem -inform PEM
```

5 Pour chiffrer les messages

Récupération et intégration des clés publiques des destinataires

On chiffre les messages avec la clé publique du destinataire, lequel destinataire déchiffre le message avec sa propre clé privée. Pour chiffrer un message pour un certain correspondant, il nous faut donc pouvoir récupérer sa clé publique et l'intégrer dans `mutt`. Pour cela il faut que votre correspondant vous envoie un premier message comportant une signature `s/mime`. Cette signature `s/mime` contient la clé publique de votre correspondant. On peut alors récupérer cette clé de la façon suivante:

- Dans mutt, se placer sur l'attachement `s/mime` «smime.p7s» du message signé, reçu en tapant «v» (pour visualiser les attachments)
- sauvegarder cet attachement dans un fichier en tapant «s» .

De cette manière on a enregistré le certificat de notre correspondant dans un fichier au format `pkcs7`, par exemple `albert.p7s`. Cependant pour que mutt puisse utiliser la clef publique contenu dans ce certificat il faut l'intégrer dans le «coffre» à certificats de mutt . Pour cela il faut lancer la commande `openssl` suivante :

```
openssl pkcs7 -inform DER -in ./albert.p7s -print_certs -text > albert.pem
smime_keys add_cert albert.pem
```

Comme pour les exemples précédents, la commande vous demande un label désignant la clé publique de ce correspondant

```
Enter label: albert
certificate fb4602dd.0 (albert) for shih@math.jussieu.fr added.
==> about to verify certificate of shih@math.jussieu.fr
/home/libes/.smime/certificates/fb4602dd.0: OK
```

Une fois cela fait on peut envoyer un mail chiffré à `shih@math.jussieu.fr` puisque mutt connaît désormais sa clé publique.

6 Utilisation de mutt version ≥ 1.5

Envoi de message chiffré

Dans le menu `s/mime`, après avoir tapé la commande «s», il suffit de taper la lettre «c» dans mutt pour demander le chiffrement du message. mutt demande avec quelle clé on veut chiffrer. Il suffit de taper le label «albert» avec lequel on a désigné la clé publique de notre correspondant, précédemment intégrée. Si on a oublié le label avec lequel on a désigné la clé, on peut lister les clés qui ont été ajoutées à mutt avec la commande

- `smime_keys list`

Réception de message chiffré

Lors de la réception d'un message chiffré, (on déchiffre avec sa propre clé privée), mutt demande la *passphrase* permettant d'accéder à sa clé privée personnelle. Si vous avez donné la bonne passphrase, mutt rajoute les informations suivantes lors de la lecture du message.

```
[-- Les données suivantes sont chiffrées avec S/MIME --]
[-- La sortie OpenSSL suit (heure courante : mer 22 sep 2004 17:00:20 CEST) --]
Verification successful
[-- Fin de sortie OpenSSL --]
[-- Les données suivantes sont signées --]
```

Ce qui signifie que la vérification s'est bien déroulée.

Pour lire/déchiffrer un fichier p7s

Le format standard utilisé par les certificats, le `pkcs7`, n'est pas en format ASCII, et donc pas directement lisible par un être humain. Pour décoder et lire le contenu du certificat, vous pouvez utiliser la commande :

```
openssl pkcs7 -inform DER -in ./albert.p7s -print_certs -text -noout
```

Si le fichier `pkcs7` contient aussi la clé privée, donc en général chiffrée, il vous sera demandé de taper la passphrase utilisée lors de la création du fichier (en général exportation depuis mozilla).

Pour lister l'ensemble des certificats que mutt connaît

Il est parfois utile de connaître la liste des certificats que l'on a intégré à mutt, ainsi que leur label. Pour cela vous pouvez utiliser la commande

```
smime_keys list
```

Pour authentifier les clés publiques par rapport aux certificats racines des autorités de certification

Lorsque vous intégrez la première fois les certificats utilisateurs, mutt ne fait pas leur vérification par rapport aux certificats racines des CA. Pour authentifier un certificat utilisateur, vous devez posséder le certificat de la CA qui a servi à le signer, comme vu précédemment. Cela étant fait, vous pouvez lancer la commande de vérification :

```
smime_keys verify all
```

Si le certificat racine d'autorité de certification, ayant servi à signer un certificat utilisateur est absent, ce certificat utilisateur ne sera pas authentifié

```
/home/libes/.smime/certificates/a4a1f3ba.0:  
/C=FR/O=CNRS/OU=UMR7586/CN=Noel Larchand/emailAddress=nla@univ-loin.fr  
error 20 at 0 depth lookup:unable to get local issuer certificate
```

On récupère le certificat de l'autorité de certification qui manquait, et on l'intègre à mutt avec les commandes vues précédemment

```
smime_keys add_cert CNRS-Standard.crt
```

et le certificat utilisateur peut alors être authentifié

```
==> about to verify certificate of jma@math.jussieu.fr  
/home/libes/.smime/certificates/a4a1f3ba.0: OK}
```

Notez que si vous n'avez pas accès au certificat racine de l'autorité de certification ayant servi à signer un certificat utilisateur, cela ne vous empêche pas d'envoyer un mail signé (et/ou chiffré) à cet utilisateur. Seulement vous ne pourrez pas authentifier un message venant de cette personne.

Pour déchiffrer/lire automatiquement un certificat s/mime intégré dans un mail

Il suffit de rajouter la ligne de commande suivante dans votre fichier .mailcap

```
application/x-pkcs7-signature;openssl pkcs7 -in \%s -inform der  
-noout \textbackslash{} -print_certs -text | less; needsterminal
```

A la lecture du message avec mutt, il suffit de se placer sur l'attachment s/mime et de taper "retour". La commande openssl sera lancée et le certificat sera visualisé en clair.