

# Gestion des clés et des certificats

- [OpenSSL - Commandes Utiles](#)
- [acme.sh - Génération de certificat gratuit](#)
- [Principe des clés asymétriques](#)

# OpenSSL - Commandes Utiles

## Demande de certificat

Générer une demande de certificat avec la création d'une clé privée RSA de 2048 bits

```
openssl req -new -newkey rsa:2048 -nodes -sha256 -keyout mykey.key -out mycsr.csr -subj  
"/C=US/ST=Ohio/L=Columbus/O=Widgets Inc/OU=Some Unit/CN=myserver.com"
```

Générer une demande de certificat avec la création d'une clé privée ECDSA de 256 bits.

```
openssl req -newkey ec:<(openssl genpkey -genparam -algorithm ec -pkeyopt ec_paramgen_curve:P-  
256) -nodes -sha256 -keyout mykey.key -out mycsr.csr -subj "/C=US/ST=Ohio/L=Columbus/O=Widgets  
Inc/OU=Some Unit/CN=myserver.com"
```

Générer un certificat auto-signé

```
openssl req -newkey rsa:2048 -nodes -sha256 -keyout mykey.key -x509 -days 365 -out mycert.crt -  
subj "/C=US/ST=Ohio/L=Columbus/O=Widgets Inc/OU=Some Unit/CN=myserver.com"
```

Générer une demande de certificat à partir d'une clé et d'un certificat existants

```
openssl x509 -x509toreq -in mycert.crt -signkey mykey.key -out mycsr.csr
```

## Gestion des clés

Générer une nouvelle clé RSA

```
openssl genrsa 2048 > mykey.key
```

Générer une nouvelle clé ECC

```
openssl ecparam -name prime256v1 -genkey > mykey.key
```

Enlever la passphrase d'une clé privée

```
openssl rsa -in mykey-with-passphrase.key -out mykey.key
```

# Vérification des clés, des demandes et des certificats

Vérifier une clé

```
openssl rsa -noout -text -check -in mykey.key
```

Vérifier une demande

```
openssl req -noout -text -verify -in mycsr.csr
```

Vérifier un certificat

```
openssl x509 -noout -text -in mycrt.crt
```

Vérifier que la demande et le certificat sont bien associés à une clé :

- Le résultat des commandes doit retourner la même valeur

```
openssl x509 -noout -modulus -in mycrt.crt  
openssl req -noout -modulus -in mycsr.csr  
openssl rsa -noout -modulus -in mykey.key
```

Afficher le contenu d'un certificat en format PKCS#7:

```
openssl pkcs7 -print_certs -in www.server.com.p7b
```

Afficher le contenu d'un certificat et d'une clé en format PKCS#12:

```
openssl pkcs12 -info -in www.server.com.pfx
```

Contrôler une connection SSL et afficher tous les certificats intermédiaires:

```
openssl s_client -connect www.server.com:443
```

## Conversion des certificats

Conversion d'un fichier PKCS#12 vers le format PEM (clé privée + certificat + chaîne de certification)

```
openssl pkcs12 -nodes -in www.server.com.pfx -out www.server.com.crt
```

Conversion du format PEM vers le format PKCS#12:

```
openssl pkcs12 -export -in www.server.com.crt -inkey www.server.com.key -out www.server.com.pfx
```

Conversion du format PKCS#7 ( .p7b .p7c ) vers le format PEM:

```
openssl pkcs7 -print_certs -in www.server.com.p7b -out www.server.com.crt
```

Conversion du format PEM vers le format PKCS#7:

```
openssl crl2pkcs7 -nocrl -certfile www.server.com.crt -out www.server.com.p7b
```

Conversion du format DER vers le format PEM:

```
openssl x509 -inform der -in certificate.cer -out certificate.pem
```

## Références

- <https://www.kinamo.fr/base-de-connaissanse/openssl-commandes-utiles>

# acme.sh - Génération de certificat gratuit

## Présentation

"acme.sh" est un outil qui permet de générer des certificats à partir d'autorité gratuite. Il gère les autorités suivantes :

- [ZeroSSL.com CA](#)(default)
- Letsencrypt.org CA
- [BuyPass.com CA](#)
- [SSL.com CA](#)
- [Google.com Public CA](#)
- [Pebble strict Mode](#)
- Toutes autorités qui est compatible avec la [RFC8555](#)

Dépôt GIT : <https://github.com/acmesh-official/acme.sh>

## Installation de acme.sh

```
curl https://get.acme.sh | sh -s email=my@example.com
```

## Génération d'un certificat

Remarque : Utiliser de préférence la CA **Let's Encrypt**, par expérience l'API de ZeroSSL est parfois indisponible.

## Génération d'un certification en précisant le dossier "webroot"

```
/root/.acme.sh/acme.sh --issue -d example.com -d www.example.com -w /home/wwwroot/example.com  
--server letsencrypt
```

## Génération d'un certificat en mode server HTTP standalone

```
/root/.acme.sh/acme.sh --issue -d example.com -d www.example.com --standalone --httpport 8443  
--server letsencrypt
```

## Installation du certificat

### Installation du certificat sur HaProxy

```
/root/.acme.sh/acme.sh --install-cert -d example --reloadcmd "cat \${CERT_KEY_PATH}
\${CERT_FULLCHAIN_PATH} >/etc/pki/mycert.pem && systemctl reload haproxy"
```

## Installation du certificat sur Apache httpd

```
/root/.acme.sh/acme.sh --install-cert -d example.com --cert-file /etc/ssl/certs/mycert.pem --
key-file /etc/ssl/private/mykey.key --fullchain-file /etc/ssl/certs/fullchain.pem --reloadcmd
"systemctl reload apache2"
```

## Installation du certificat sur nginx

```
/root/.acme.sh/acme.sh --install-cert -d example.com --key-file
/etc/ssl/private/example.com.key --fullchain-file /etc/ssl/certs/example.com.pem --reloadcmd
"service nginx force-reload"
```

## Configuration des serveurs web pour accepter le challenge "acme"

### Configuration du server "apache httpd"

TODO

### Configuration du server "haproxy"

```
frontend extranet-ffsa-noproduct
    bind 193.41.223.33:443 ssl crt /etc/pki/moncertificat.pem no-sslv3 no-tlsv10 no-tlsv11 no-
    tls-tickets ciphers ECDHE-ECDSA-AES128-GCM-SHA256:ECDSA-AES128-GCM-SHA256:ECDSA-
    AES256-GCM-SHA384:ECDSA-AES256-GCM-SHA384:ECDSA-CHACHA20-POLY1305:ECDSA-
    CHACHA20-POLY1305:DHE-RSA-AES128-GCM-SHA256:DHE-RSA-AES256-GCM-
    SHA384:TLS_AES_128_GCM_SHA256:TLS_AES_256_GCM_SHA384:TLS_CHACHA20_POLY1305_SHA256
    bind 193.41.223.33:80

...

...

...

    #Let's encrypt
    acl begin_acme_challenge path_beg -i /.well-known/acme-challenge/

...

...

...

    use_backend acme_standalone_http if begin_acme_challenge

...

...
```

```
...  
backend acme_standalone_http  
    log global  
    mode http  
    server srv_letsencrypt 127.0.0.1:8443
```

## Configuration du server "nginx"

TODO

## Références

Site officiel de acme.sh : <https://github.com/acmesh-official/acme.sh>

# Principe des clés asymétriques

test

EE

- test

SSSSS