RFC 6125 : Representation and Verification of Domain-Based Application Service Identity within Internet Public Key Infrastructure Using X.509 (PKIX) Certificates in the Context of Transport Layer Security (TLS)

Stéphane Bortzmeyer
<stephane+blog@bortzmeyer.org>

Première rédaction de cet article le 1 avril 2011

Date de publication du RFC : Mars 2011

https://www.bortzmeyer.org/6125.html

Un RFC long et complexe pour un problème difficile. En résumant violemment, il s'agit de savoir quelle **identité** utiliser lorsqu'un client TLS s'assure de l'authenticité du serveur auquel il se connecte. Par exemple, si un MTA veut transmettre du courrier à `bill@example.com`, que doit contenir le certificat X.509 du MX d'`example.com` pour que l'authentification réussisse ? `example.com` ? D'autres noms, plus spécifiques à ce service, sont-ils possibles ? Ce cas particulier est mentionné dans la section 4.1 du RFC 3207 [1] mais très peu détaillé, et renvoyé à des décisions locales. Comme de nombreux autres protocoles ont des problèmes de ce genre, l'idée avait germé de faire un RFC générique, définissant un cadre et des principes, RFC que les protocoles n'auraient plus qu'à citer. C'est notre RFC 6125 (depuis remplacé par le RFC 9525).

Traditionnellement, l'identité du pair avec lequel on se connecte était indiqué dans le champ "*Subject*" du certificat X.509. `openssl` permet de l'afficher :

```
% openssl s_client -starttls smtp -connect mail.bortzmeyer.org:25
...
Certificate chain
 0 s:/C=FR/ST=Ile-de-France/L=Paris/O=Moi/OU=Maison/CN=mail.bortzmeyer.org/emailAddress=postmaster@b
ortzmeyer.org
...
```

1. Pour voir le RFC de numéro NNN, `https://www.ietf.org/rfc/rfcNNN.txt`, par exemple `https://www.ietf.org/rfc/rfc3207.txt`

Le champ Sujet est formé de doublets attribut-valeur (par exemple, l'attribut C veut dire "*Country*", ici la France) notamment CN ("*Common Name*"), le plus utilisé. Mais l'identité pouvait aussi figurer dans une extension X.509, "*Subject Alternative Name*" (section 4.2.1.6 du RFC 5280). Cette extension est rare en pratique, bien que normalisée depuis de nombreuses années. Je ne sais pas comment l'afficher avec `openssl`. La seule solution que je connaissais était de copier/coller le certificat affiché par `openssl s_client -connect www.example.com:443` puis de faire un `openssl x509 -text -noout -in /le/certificat.pem`, ici avec le serveur HTTPS `www.digicert.com` :

```
Certificate:
...
        Subject: businessCategory=V1.0, Clause 5.(b)/1.3.6.1.4.1.311.60.2.1.3=US/1.3.6.1.4.1.311.60.2.1.2=Ut
               C=US, ST=Utah, L=Lindon, O=DigiCert, Inc.,
                CN=www.digicert.com
...
        X509v3 extensions:
...
            X509v3 Subject Alternative Name:
                DNS:www.digicert.com, DNS:www.origin.digicert.com,
                       DNS:content.digicert.com, DNS:digicert.com
...
```

On peut aussi combiner les deux commandes (`openssl s_client -connect www.digicert.com:443 | openssl x509 -text`) voire n'afficher que la partie qui nous intéresse (`openssl s_client -connect www.digicert.com:443 | openssl x509 -text | perl -n0777e 'print $1 if /(.*Subject Alternative Name:.*\n.*\n)/'`, merci à minimalist pour son aide). Mais les navigateurs Web permettent en général de montrer tout le certificat, y compris cette extension. Regardez par exemple sur `<https://www.cisco.com/>` ou bien `<https://www.digicert.com/>`. Un "*Subject Alternative Name*" peut contenir un `dNSName` (un nom de domaine), mais aussi un `uniformResourceIdentifier` (un URI) et, par des extensions de l'extension (RFC 4985), un `SRVName` (un nom de service, tel que présent en partie gauche d'un enregistrement SRV).

Quel nom utiliser lorsque plusieurs sont présents ? C'est tout le but de ce RFC : lorsqu'un client TLS se connecte à un serveur TLS, et qu'une authentification X.509 prend place, le client a une idée sur l'identité du serveur ("*reference identity*", identité de référence, dit le RFC) et le serveur présente un certificat contenant un certain nombre d'identités ("*presented identity*", identité affichée). Comment comparer ? Aujourd'hui, la situation est plutôt confuse et deux approches sont largement utilisées :

— Utiliser uniquement le CN ("*Common Name*") du champ Sujet ("*subject field*"), puisque tout le monde doit le connaître, c'est le champ traditionnel,
— Répéter l'identité du serveur à plusieurs endroits dans le certificat en se disant que le vérificateur en trouvera au moins un.

Une partie de la difficulté est que le déploiement d'une méthode propre et standard nécessite la copération de plusieurs acteurs. Ce nouveau RFC 6125 est surtout destiné à l'usage interne de l'IETF, aux concepteurs de protocoles (section 1.2). Mais les autres acteurs impliqués, par exemple ceux qui écrivent les logiciels ou bien qui gèrent une CA ont tout intérêt à le lire également. Ils ne s'amuseront pas forcément (ce RFC est long, complexe et très abstrait, la section 1.3 donne quelques pistes pour le lire sans trop souffrir) mais ils apprendront des choses utiles. Avant de se mettre à la lecture, il fraut s'assurer d'avoir lu et bien compris le RFC 5280, dont les règles demeurent actives. Il faut également garder en mémoire les RFC spécifiques d'un protocole applicatif particulier comme le RFC 2595 pour IMAP, RFC 2818 pour HTTP, RFC 2830 pour LDAP, RFC 3207 pour SMTP, RFC 6120 pour XMPP, et plusieurs autres encore (liste en section 1.6)...

Pour le lecteur pressé, la section 1.5 résume les recommandations essentielles de ce RFC 6125 :

— Abandonner progressivement l'utilisation de noms de domaine dans le champ CN ("*Common Name*") du sujet du certificat. Aujourd'hui, c'est un vœu pieux, à en juger par les certificats HTTPS existants. Ceux-ci, soucieux de compatibilité, mettent presque tous le nom de la machine dans ce champ CN.

— Privilégier l'utilisation des champs de l'extension "*Subject Alternative Name*" du RFC 5280.
— Encore mieux, utiliser de préférence, dans cette extension, les champs typés comme celui qui permet d'indiquer l'URI du service ou celle qui permet d'indiquer le nom SRV (RFC 4985) car ils sont plus spécifiques et évitent de faire des certificats « attrape-tout ». (Personnellement, je n'ai jamais vu dans la nature des certificats d'un serveur HTTPS ou IMAP utilisant ces champs.)
— Pour le même souci de spécificité, abandonner petit à petit les certificats incluant un joker (`*.example.com`), un conseil qui plaira aux CA, car il augmentera leur chiffre d'affaires.

Plusieurs points sont exclus de ce RFC (section 1.7) :

— Les identités des clients finaux (comme l'adresse de courrier électronique représentée par l'identificateur `rfc822Name`), qui peuvent servir en cas de présentation d'un certificat client. Notre RFC ne considère que l'authentification du serveur.
— Les identificateurs qui ne sont pas des noms de domaines (par exemple les adresses IP, théoriquement possibles mais, en pratique, très peu utilisées).
— Les protocoles non-TLS, même si certains d'entre eux, comme IPsec, peuvent eux aussi utiliser X.509.
— Les certificats non-X.509 comme ceux d'OpenPGP (RFC 4880 et RFC 6091).

Une fois éliminé ces « non-sujets », le lecteur doit ensuite passer un peu de temps sur la section de terminologie en 1.8, car X.509 utilise beaucoup de vocabulaire. Il faut également connaître le vocabulaire standard de la sécurité, exposé dans le RFC 4949. Parmi les sigles dont on aura le plus besoin ici :

— CN-ID : un RDN ("*Relative Distinguished Name*" dans le champ sujet du certificat, qui contient une paire clé-valeur de type CN ("*Common Name*") et qui ait la forme d'un nom de domaine. Par exemple, dans le certificat actuel de `www.gmail.com` (après redirection vers `mail.google.com`), le champ sujet est `CN = mail.google.com, O = Google Inc, L = Mountain View, ST = California, C = US` et le CN-ID est `mail.google.com`. C'est de très loin le type le plus fréquent aujourd'hui.
— DNS-ID : un identificateur de type `dNSName` mis dans l'entrée `subjectAltName` du RFC 5280. Par exemple, dans le certificat actuel de `www.digicert.com`, le DNS-ID est `www.digicert.com` (le même que le CN-ID, ce qui est le cas de la quasi-totalité des certificats qu'on trouve sur l'Internet public).
— SRV-ID et URI-ID : des identificateurs trouvés également dans l'entrée `subjectAltName` du RFC 5280 (plus le RFC 4985 pour le SRV-ID), de types respectifs `SRVName` et `uniformResourceIdentifier`. On ne les trouve quasiment jamais dans les certificats disponibles sur l'Internet public.

Puisqu'on parle de ce qu'on trouve en pratique ou pas, je précise que mes avis là-dessus sont surtout tirés de quelques explorations faites sans méthode. Ce qui serait cool, ce serait d'avoir un Google des certificats X.509. Un robot récolterait les certificats, les analyserait et un moteur de recherches permettrait de trouver « Des certificats avec "*Subject Alternative Name*" » ou bien « Des certificats avec un condensat MD5 » ou aussi « Des certificats expirant après 2014 » ou encore « Des certificats Comodo émis par la Iranian Cyber Army <http://erratasec.blogspot.com/2011/03/comodo-hacker-releases-his-manifesto.html> :-) » Il existe deux projets allant dans cette direction : SSLiverse <http://www.eff.org/observatory>, qui a la récolte mais pas le moteur de recherche. L'archive fait 12 Go et est disponible en torrent. Il ne reste plus qu'à fouiller dedans (ce qui peut s'avérer assez long), la liste de diffusion du projet <https://mail1.eff.org/mailman/listinfo/observatory> devient utile dans ce cas (merci à Étienne Maynier pour ses conseils et informations). L'autre projet est Shodan <http://www.shodanhq.com/> qui, lui, dispose non seulement d'une telle base, mais d'une partie des fonctions de recherche souhaitées (mais pas celle qui me fallait, pour trouver les certificats avec "*Subject Alternative Name*").

Maintenant, passons aux noms. Un certificat contient les noms de services comme `www.example.org`. Certains noms sont **directs** : ils ont été tapés par un être humain ou sélectionnés à partir d'une page Web. D'autres, indirects, ont été obtenus, par exemple via une résolution DNS. Ainsi, pour envoyer du courrier à `sandra@example.com`, une résolution DNS de type MX permettra de trouver le relais de courrier de `example.com`, mettons `smtp.example.net`. (On peut noter que, en l'absence de résolution sécurisée, par exemple par DNSSEC, c'est le premier nom qu'il faut chercher dans le certificat, le second peut être le résultat d'un empoisonnement DNS.)

Les noms peuvent également être **restreints** ou pas. Un nom restreint ne peut être utilisé que pour un seul service. Par exemple, une CA peut émettre un certificat qui ne sera utilisable que pour de l'IMAP.

On voit donc qu'un CN-ID ou un DNS-ID sont directs et non restreints, qu'un SRV-ID peut être direct ou indirect (mais il est indirect la plupart du temps) mais est restreint (le nom de l'enregistrement SRV indique un certain type de service), qu'un URI-ID est direct et restreint.

Et le nom de domaine lui-même, a-t-il plusieurs catégories possibles ? Oui, dit la section 2.2 qui sépare les noms en :

— Traditionnels. Ce sont les noms historiques comme `www.mabanque.example`.
— Internationalisés. Ce sont les noms de domaine en Unicode définis par le RFC 5890.

Au tout début de X.509, les noms étaient censés être tirés d'un grand Annuaire officiel X.500 mais ce projet s'est cassé la figure, comme la plupart des autres éléphants blancs de l'UIT et il ne reste que des certificats isolés (section 2.3). Pour assurer l'unicité des noms, la méthode la plus courante sur l'Internet a donc rapidement été de mettre un nom de domaine dans le CN ("*Common Name*") du sujet du certificat, par exemple `C=CA,O=Example Internetworking,CN=mail.example.net`. Mais ce n'est qu'une convention : le CN n'est pas typé, il peut contenir n'importe quoi, par exemple un nom lisible par un humain comme `CN=A Free Chat Service,O=Example Org,C=GB` et, dans ce cas, la vérification automatique de ce nom par rapport à ce qu'attendait l'utilisateur n'est pas évidente. Voici pourquoi notre RFC 6125 recommande l'utilisation des champs typés de `subjectAltName` au lieu du CN (recommandation très peu suivie dans l'Internet d'aujourd'hui).

Ces (longs) préliminaires étant achevés, la section 3 commence enfin la partie pratique du RFC, destinée aux auteurs de protocoles. Ceux-ci doivent d'abord faire une petite liste des caractéristiques pertinentes de leur protocole : utilise-t-il les enregistrements SRV ? Utilise-t-il des URI ? A-t-il besoin de jokers dans les noms de domaine ? Armé de cette connaissance, le concepteur de protocoles peut passer à la section 4 qui définit les règles de production d'identité (les règles pour les autorités de certification, donc).

Il est donc recommandé de mettre dans les certificats un `subjectAltName` DNS-ID. Si le protocole utilise les SRV, il est recommandé de mettre un SRV-ID (même conseil évident pour les URI). J'aimerais bien savoir combien de CA permettent de mettre un tel identificateur. Je parierais bien qu'il n'y en a aucune. Le RFC conseille également de ne **pas** inclure de nom de domaine dans le CN, pour encourager la migration (notons que le premier exemple donné, en section suivante, ne suit pas ce conseil...).

Voyons quelques exemples concrets :

— Pour un serveur HTTP, protocole qui n'utilise pas les SRV et n'a pas prévu de mettre des URI dans les certificats (cf. RFC 2818), le certificat pour `www.example.com` doit inclure un DNS-ID `www.example.com` et peut inclure un CN-ID `www.example.com` pour la compatibilité avec les vieilles implémentations.
— Pour un serveur IMAP, qui sert `user@example.net` et se nomme `mail.example.net`, comme le protocole peut utiliser les SRV (cf. RFC 6186), il faudra mettre le SRV-ID `_imap.example.net`, ainsi que les DNS-ID `example.net` et `mail.example.net` (là encore, on pourra ajouter un nom de domaine dans le CN, si on tient à gérer les vieux clients IMAP).
— C'est presque pareil pour XMPP, avec en prime l'extension X.509 spécifique à ce protocole, `XmppAddr` (section 13.7.1.4 du RFC 6120).
— Pour un service SIP qui gère `user@voice.example.edu`, comme SIP peut utiliser des URI (cf. RFC 5922), il faudra au moins un URI-ID `sip:voice.example.edu` et un DNS-ID `voice.example.edu`.

Ces spécifications de la section 4 étaient pour les autorités de certification. Et les gérants des services sécurisés, que doivent-ils mettre dans leurs demandes de certificats (CSR, "*Certificate Signing Requests*") ? La section 5 précise qu'ils doivent demander tous les identificateurs nécessaires (ce qui ne veut pas dire que la CA leur donnera...), donc, par exemple, le SRV-ID et le DNS-ID dans le cas de IMAP. Si le certificat sera utilisé pour des services différents, il vaut mieux ne demander que le DNS-ID (les autres étant restreints, rappelez-vous la section 2.1).

Les gérants des services ont demandé les certificats (section 6), les CA ont émis des certificats (section 5), reste à les vérifier dans le logiciel client (section 6, qui concerne donc surtout les programmeurs). Le principe est le suivant :

— L'application construit une liste des **identités de référence**. Typiquement, c'est ce que l'utilisateur a tapé au clavier ou bien sélectionné explicitement. S'il a configuré son compte de messagerie comme étant `thierry.breton@iloveemail.example`, le nom `iloveemail.example` sera une identité de référence (rappelez-vous que notre RFC ne se préoccupe que d'authentifier des serveurs, pas des personnes, donc le nom de l'individu n'est pas pris en compte). Si l'utilisateur a sélectionné `https://www.dns-oarc.net/` sur une page Web, `www.dns-oarc.net` est une identité de référence.
— Le serveur TLS envoie un certificat, qui contient des **identités affichées**.
— Le client TLS compare les identités de référence aux identités affichées, râlant s'il n'y a pas correspondance.

Le reste de la section 6 formalise cette méthode. Quelques points particuliers, par exemple :

— Le client doit faire attention à ne pas dériver des identités à partir d'autres identités, sauf si cela peut être fait de façon sûre. Ainsi, analyser un URI, pour en extraire le nom de domaine, si c'est fait proprement, est une opération sûre. Résoudre un nom en un autre (par exemple le serveur de messagerie à partir du MX du domaine) via le DNS est non sûr. Si on utilise DNSSEC, cela redevient sûr.
— Mais, d'une manière générale, le RFC décourage la dérivation : seuls les noms entrés directement par l'utilisateur sont vraiment fiables comme identité de référence puisque, par définition, ils expriment la volonté directe de l'utilisateur (« je veux communiquer avec `gmail.com` »). La règle n'est toutefois pas absolue car, par exemple, l'utilisateur a pu « épingler » un autre nom précédemment (« je sais que le serveur XMPP de `example.org` est `jabber.toto.example`, authentifie ce dernier nom », cf. section 7.1).
— La liste des identités de référence doit comprendre un DNS-ID. S'il y a eu une résolution SRV, elle devrait comprendre un SRV-ID (même raisonnement pour les URI-ID).
— On peut y ajouter un CN-ID, pour réussir l'authentification avec les vieux certificats (le RFC note bien que, en l'état actuel des choses, cette possibilité est de facto une obligation, les certifiats avec CN-ID étant largement dominants).

Le RFC donne quelques exemples : une requête Web pour `https://www.dns-oarc.net/` produira une liste d'identités de référence comportant un DNS-ID, `www.dns-oarc.net` et un CN-ID optionnel identique. Une tentative de connexion IMAP pour le domaine `example.net`, dont l'enregistrement SRV indiquera que le serveur est `mail.example.net`, donnera une liste comportant les SRV-ID `_imap.example.net` et `_imaps.example.net` ainsi que les DNS-ID `example.net` et `mail.example.net` (sans compter le CN-ID) optionnel. Le RFC ne rappelle apparemment pas que l'étape de résolution DNS de `example.net` en `mail.example.net` n'est pas forcément sûre... Le fait qu'ils soient sous le même domaine `example.net` n'est pas une protection parfaite (car on ne connait pas les frontières des zones DNS) et, de toute façon, le serveur n'est pas forcément dans le même domaine que celui qu'il sert. Et le RFC semble avoir oublié ici son conseil précédent de se méfier des noms dérivés...

Ensuite, il reste à comparer les deux listes (section 6.3). Le principe est celui de la **première correspondance**. Dès qu'une identité présentée correspond à une des identités de référence, on s'arrête, on a authentifié le serveur (section 6.6.1) et cette identité authentifiée est celle qui doit pêtre montrée à l'utilisateur. (Le cas où on voudrait que **toutes** les identités correspondent est laissé à une future norme.) Attention, le CN-ID dans les identités de référence est traité un peu à part : il ne doit pas être utilisé si le certificat inclus d'autres identités (comme un DNS-ID).

Cette correspondance nécessite quelques précisions (section 6.4). Ainsi, le RFC rappelle que les noms de domaine traditionnels doivent être comparés de manière insensible à la casse (RFC 4343), que les noms de domaine internationalisés doivent être transformés en "*A-labels*" (cf. RFC 5890) avant comparaison, que les jokers (indiqués par le signe *) nécessitent de l'attention particulière (par exemple, ils ne sont considérés comme spéciaux que s'ils apparaissent seuls, dans le composant le plus à gauche du nom). Globalement, faire correctement cette comparaison n'est pas trivial pour le programmeur <https://www.bortzmeyer.org/openssl-hostname-check.html>.

Pour les CN, il est rappelé qu'un CN dans la liste des identités de référence n'est là que pour pouvoir authentifier avec les « vieux » certificats et ne doit pas être utilisé avec un certificat moderne, incluant des DNS-ID, SRV-ID ou URI-ID. S'il n'y a pas le choix (pas de "*Subject Alternative Name*" dans le certificat), l'application peut comparer ses identités de référence au CN, en suivant les règles de comparaison des noms de domaine.

Bon, si l'authentification du serveur a réussi, on continue et on affiche à l'utilisateur, sur un joli fond vert, l'identité authentifiée. Et si ça a raté, que doit-on faire ? La section 6.6.4 propose des stratégies de repli : si le logiciel est interactif, avec un humain pas loin, la méthode recommandée est de mettre fin à la connexion, avec un message d'erreur (conseil que je trouve déplorable : vu le nombre de certificats invalides, cela diminuerait nettement l'utilisabilité du Web et cela encouragerait les sites à ne **pas** proposer HTTPS). La possibilité de permettre à l'utilisateur de passer outre le problème et de contnuer (ce que font tous les navigateurs Web, autrement ils perdraient tous leurs clients) est mentionnée mais très découragée.

Si l'application n'est pas interactive, la démarche recommandée est d'afficher un message d'erreur et de s'arrêter. Une option pour ignorer le certificat est possible mais ne doit pas être activée par défaut. C'est ainsi que wget, fidèle à cette règle, a son option `--no-check-certificate`, alors que fetchmail, qui ne suit pas le RFC, a son `--sslcertck` pour **activer** la vérification.

D'ailleurs, une section entière, la 7, est consacrée à la discussion détaillée de certains problèmes de sécurité. Ainsi, les jokers font l'objet de la section 7.2. Ils sont déconseillés (pour les raisons expliquées en « "*New Tricks for Defeating SSL in Practice*" <http://www.blackhat.com/presentations/bh-dc-09/Marlinspike/BlackHat-DC-09-Marlinspike-Defeating-SSL.pdf> » et « "*HTTPS Can Byte Me*" <https://media.blackhat.com/bh-ad-10/Hansen/Blackhat-AD-2010-Hansen-Sokol-HTTPS-Ca pdf> ») mais, s'ils sont présents dans un certificat, le client devrait se préparer à les gérer, avec prudence.

Autre cas complexe, celui où un même serveur est accédé par plusieurs noms ("*virtual hosting*") par exemple un serveur IMAP qui hébergerait plusieurs domaines de messagerie, ce qui est courant. Il existe plusieurs méthodes <https://www.bortzmeyer.org/auth-x509-plusieurs-noms.html> pour gérer ce cas, comme de mettre tous les noms possibles dans le certificat (pas pratique...), ou bien d'utiliser SNI ("*Server Name Indication*", section 3 du RFC 6066). Si on met plusieurs noms dans le certificat, le RFC recommande que ce soit plusieurs DNS-ID (ou SRV-ID ou URI-UID) pas plusieurs CN.

Voilà, arrivé ici, le concepteur d'un protocole utilisant l'authentification avec X.509 sait normalement tout ce qu'il faut savoir. Il ne lui reste plus qu'à inclure dans sa spécification le texte exact et, pour lui faciliter la vie, l'annexe A contient un tel texte (pris dans le RFC 6120, section 13.7.1.2), qui peut être copié/collé avant d'être adapté à ce protocole particulier. De nombreux RFC avaient des règles spécifiques, avant la publication de notre RFC 6125 et la variété de ces règles était justement une motivation pour le développement du nouveau RFC. L'annexe B rappelle et cite les principaux RFC qui avaient traité le sujet comme le RFC 2818 pour HTTPS ou le RFC 2830 pour LDAP.

Ce RFC 6125 avait été chaudement discuté à l'IETF, ce qui explique qu'il ait mis si longtemps à être publié. Le lecteur courageux doit relire toutes les archives de la liste certid <http://www.ietf.

org/mail-archive/web/certid/current/maillist.html> pour suivre toutes ces discussions (je recommande celles de septembre 2010 et décembre 2010).

Quelques petits trucs techniques pour ajouter des extensions comme subject alternative name avec les logiciels existants (merci à Dany Mello et Christian Pélissier ; on peut aussi consulter mon article « Plusieurs noms dans un certificat X.509 <https://www.bortzmeyer.org/plusieurs-noms-dans-certificat.html> »). Avec OpenSSL, on peut aussi utiliser un fichier texte où on place l'extension. Pour une extension de type IP, le fichier ext.txt contient :

```
subjectAltName="IP:1.1.1.1"
```

puis on ajoute l'argument -extfile dans la ligne de commande d'openssl :

```
% openssl x509 -req -days 7300 -in moncsr.csr -CA ca.crt -CAkey ca.key \
    -set_serial 01 -out moncrt.crt -extfile ext.txt
```

Avec l'utilitaire certpatch (qui ne semble pas très documenté) disponible dans le paquetage isakmpd <http://www.openbsd.org/cgi-bin/man.cgi?query=isakmpd&apropos=0&sektion=0&manpath=OpenBSD+Current&arch=i386&format=html>, c'est :

```
% certpatch -t fqdn -i www.example.org -k /etc/ssl/private/ca.key \
    oldcert.crt nwcert.crt
```

Avec l'utilitaire certutil (des outils de sécurité NSS <http://www.mozilla.org/projects/security/pki/nss/tools/>), c'est :

```
% certutil -R -s "cn=dapou.exemple.fr,ou=DI,o=Exemple,...,c=FR" \
         -a -o $CADIR/$host.$domain.pem.csr -d $CADIR \
         -8 "cn=ldap1.exemple.fr" \
         -z random -f password-file
```

où ldap1.exemple.fr sera le nom alternatif.