RFC 7512 : The PKCS#11 URI Scheme

Stéphane Bortzmeyer
<stephane+blog@bortzmeyer.org>

Première rédaction de cet article le 23 avril 2015

Date de publication du RFC : Avril 2015

https://www.bortzmeyer.org/7512.html

La norme technique PKCS#11 décrit une interface d'accès à des fonctions cryptographiques. Par exemple, elle permet à un logiciel de signature de signer en utilisant une clé privée stockée dans une base de données, un HSM, une carte à puce ("*smart card*") ou autres dépôts. Jusqu'à présent, la désignation des objets auxquels on pouvait accéder par PKCS#11 n'était pas normalisée et chaque logiciel avait sa méthode. Désormais, il existe un plan d'URI standard pour cela, spécifié dans ce RFC. Bienvenue à `pkcs11:object=cle-de-machin;type=public;id=%69%95%3E%5C%F4%BD%EC%91` et à ses copains.

Prenons l'exemple de l'outil de manipulation de certificats `certtool`, de GnuTLS. Son option `load-ca-certificate` permet de charger le certificat d'une AC. La valeur de cette option peut être le nom d'un fichier local mais cela peut aussi être un URI PKCS#11, autorisant ainsi l'accès à tous les mécanismes de stockage ayant une interface PKCS#11. Le principe de ces URI PKCS#11 est de permettre l'accès à des objets (comme les certificats, les clés publiques, les clés privées, etc), en donnant une série d'**attributs** qui identifient l'objet (dans mon exemple au premier paragraphe, les attributs étaient `object`, `type` et `id`). Le but de ces URI est limité à la récupération d'objets existants, on ne s'en sert pas pour créer de nouveaux objets.

Donc, la définition formelle de ce plan d'URI (section 3 de notre RFC) : le plan est `pkcs11` (et les URI commencent donc par `pkcs11:` et ce plan a été réservé dans le registre IANA <https://www.iana.org/assignments/uri-schemes/uri-schemes.xml>), et l'URI est ensuite composé de paires attribut-valeur, séparées par des points-virgules. L'attribut `id` a une valeur binaire (et qui est donc pourcent-encodée dans l'exemple au premier paragraphe), tous les autres sont du texte en UTF-8. Voici les principaux attributs, avec leur nom dans les URI et l'équivalent dans l'API PKCS#11 :

- `id` (`CKA_ID`) est l'identificateur d'un objet. C'est, comme on l'a vu, une valeur binaire opaque.
- `object` (`CKA_LABEL`) est le nom d'un objet. Contrairement à l'`id`, c'est du texte.
- `serial` (`serialNumber` dans `CK_TOKEN_INFO`) est le numéro de série du "*token*", le dispositif de stockage des objets (un HSM, par exemple).
- `token` (`label` dans le `CK_TOKEN_INFO`), le nom du dispositif de stockage.

— `type` (`CKA_CLASS`), désigne le type d'objet stocké qu'on veut récupérer. Cela peut être `cert` (un certificat), `public` (une clé publique), `private` (une clé privée en cryptographie asymétrique), `secret-key` (une clé secrète en cryptographie symétrique).

Il est également possible d'ajouter des requêtes dans l'URI, après le point d'interrogation. Par exemple, si le dispositif de stockage réclame un PIN, on peut mettre quelque chose comme `?pin-source=file:/etc/dnssec` `pin` dans l'URI (ici, cela dit que le PIN se trouve dans le fichier `/etc/dnssec/token_pin`).

La section 4 contient plusieurs exemples d'URI PKCS#11, désignant :

— Une clé publique, identifiée uniquement par son nom : `pkcs11:object=my-pubkey;type=public`.
— Un certificat, en indiquant le nom du dépôt matériel (`token`) utilisé, et l'`id` du certificat : `pkcs11:token=The%`
Notez les attributs `manufacturer` et `model`, que je n'avais pas cité dans la liste des attributs les plus utilisés.
— Une clé privée, en utilisant une nouvelle requête, `module-name`, qui permet d'indiquer la bibliothèque dynamique à charger pour parler au dépôt des clés (PKCS#11 est une API, pas un protocole, et il faut donc une bibliothèque différente par type de HSM ou "*smart card*") : `pkcs11:object=my-sign`
Ici, sur une machine Unix, l'application PKCS#11 va donc tenter de charger la bibliothèque `snakeoil-pkcs11`

Il existe plusieurs implémentations des URI PKCS#11, le document était en développement depuis longtemps. Ainsi, OpenConnect `<http://www.infradead.org/openconnect/>` documente « "*Objects from PKCS#11 tokens are specified by a PKCS#11 URI.*" `<http://www.infradead.org/openconnect/pkcs11.html>` ». Le projet Fedora tente de standardiser tous ses logiciels qui font de la cryptographie autour de ces URI « "*Currently, there are many different ways to tell each application how to find the certificate. [...] where PKCS#11 objects are specified in a textual form which is visible to the user (e.g. on the command line or in a config file), objects SHOULD be specified in the form of a PKCS#11 URI as as described*" `<https://fedoraproject.org/wiki/PackagingDrafts/PKCS11>` ». Pour un programme développé avec GnuTLS :

```
/* In addition the following functions can be used to load PKCS #11
key and certificates by specifying a PKCS #11 URL instead of a
filename. */
int gnutls_certificate_set_x509_trust_file (gnutls_certificate_credentials_t cred, const char * cafile, gnut
...
```

Mais, parmi les programmes qui reconnaissent ces URI PKCS#11, il y a aussi p11-kit `<http://p11-glue.freedesktop.org/p11-kit.html>`, Free IPA `<https://www.freeipa.org/>`, etc.