# Gestion d'identité avec OpenID

Stéphane Bortzmeyer
AFNIC
Immeuble International
78181 Saint-Quentin-en-Yvelines
bortzmeyer@nic.fr

## Résumé

*La gestion des identités numériques est un sujet très actif aujourd'hui, qui a donné lieu à de nombreuses études et à plusieurs mises en œuvre.*

*Au sein de ce vaste domaine, OpenID est une proposition d'attribution d'un URL comme identité, et d'un mécanisme pour authentifier cette identité. Le fournisseur d'identité est séparé du vérificateur. Avec OpenID, un site Web peut authentifier un client sans que celui-ci ait besoin d'un compte sur le site Web, et sans qu'il existe de mécanisme central de gestion des identités.*

## Mots-clefs

OpenID Web identité SSO Passport Cardspace Liberty Alliance

# 1 Introduction

## 1.1 Un service d'identité

La question de l'identité numérique fait aujourd'hui l'objet d'une grande activité. On compte d'innombrables blogs sur le sujet, des grandes manœuvres industrielles ont commencé depuis un certain temps, plusieurs logiciels sont disponibles. Mais la question étant très complexe, une grande partie de la discussion est peu ou mal informée.

Revenons donc d'abord, avant de parler d'OpenID, à ce que fait un système d'identité. Il y a plusieurs services qu'il fournit :

1. Attribuer un identificateur (ce qui implique en général un registre, pour que l'identificateur soit unique),
2. Authentifier l'utilisateur ou la ressource qui prétend être désigné par cet identificateur,
3. Servir des données sur cet utilisateur ou cette ressource (son nom, son âge, son pays, sa langue),
4. Servir des données externes, à partir d'autres sources,
5. Servir des autorisations sur ce que l'utilisateur a le droit de faire ou pas.

Les points 3 et 4 sont proches, la principale différence étant que, dans le point 3, les méta-données sont supposées être sous le contrôle exclusif de l'utilisateur, alors que les points 4 et 5 concernent des données à propos de l'utilisateur, mais gérées par une tierce partie.

De même, la distinction entre 4 et 5 est subtile car une autorisation peut être vue comme une méta-donnée. La distinction est faite ici car l'autorisation est un important service en lui-même.

Tâchons de rendre cela plus concret avec quelques exemples.

Pour le point 1, l'attribution d'un identificateur, il existe de nombreux exemples. Pour une personne, cela peut être un numéro INSEE (le mien est 160109940400823), ou une combinaison du nom[1] et de la date de naissance (« Vous êtes bien Jean Durand, né à Chapelle-des-Bois le 1er février 1953 ? »). Avec OpenID, l'identificateur est un URL (le mien est http://www.bortzmeyer.org/).

Pour le point 2, il existe aussi de nombreuses méthodes, qui dépendent fortement de l'identificateur. Pour authentifier une personne physique, identifiée par son nom, prénom, date et lieu de naissance, on utilise souvent un document d'identité officiel, comme la Carte Nationale d'Identité. Avec OpenID, l'authentification est sous-traitée à un système tiers, nommé l'OP (OpenID Provider).

Notons tout de suite qu'on ne parle que d'authentification ici. L'authentification est uniquement la détermination de l'association entre une personne et un identificateur. En elle-même, elle ne fournit ni sécurité, ni confiance (un délinquant, un menteur, un imbécile, ont également des papiers d'identité authentiques). La confusion est souvent faite, beaucoup de gens pensant que l'authentification est un facteur de sécurité, ce qu'elle n'a jamais prétendu être[2].

[1] Le nom, même combiné au prénom, n'est pas unique, d'où cette méthode courante.

Le point 3 est fourni parfois par des extensions au mécanisme d'authentification (c'est ainsi que dans certains pays, la religion est indiquée sur les documents d'identité). Dans OpenID, c'est la Simple Registration Extension, qui permet de communiquer ce qu'on nomme souvent un profil (indiquant la langue de l'utilisateur, son pays, son nom complet, etc).

Le point 4, par définition, nécessite l'accès à des bases extérieures. Il s'agit des informations qu'un tiers garde sur une personne. Par exemple "pb est un committeur FreeBSD", "AR41-FRNIC est le contact technique de `nic.fr`", "Jean Durand, né le 1er février 1959 à Paris, est docteur en médecine", "spokli98 a un indice de confiance de 6,75 sur /.").

Le point 5 est proche du 4. Les autorisations liées à une personne ne sont typiquement pas gérées par la personne elle-même[3] et sont distinctes de l'authentification. Si je montre une carte d'identité correcte à l'entrée du Palais de l'Élysée, je m'authentifie, mais cela ne me donnera pas automatiquement un accès au bureau de Sarkozy, puisque je ne suis pas autorisé.

Pour les ingénieurs système Unix, on peut rendre ces cinq points plus vivants en regardant comment Unix assurait ce service[4]

1. est assuré par le premier champ de `/etc/passwd` ou plutôt par l'ingénieur système qui l'édite (et qui est donc le registre). L'identificateur (le login) est donc une courte chaîne de caractères.
2. est assuré par `/etc/shadow` et la saisie d'un mot de passe,
3. est assuré par le champ GECOS de `/etc/passwd`, qui contient des informations comme le nom complet, le numéro de téléphone, etc,
4. n'était pas vraiment géré,
5. était simple : (id == 0) ou pas.

Comme il n'y avait pas de réseau et un seul vrai acteur, le kernel, tout était simple.

Si on a un compte Google ou Yahoo, l'identificateur est une courte chaîne de caractères, fournie par la société (pour éviter l'ambiguïté, on y ajoute parfois le nom de domaine, par exemple mon identificateur Google, `bortzmeyer`, devient `bortzmeyer@gmail.com`). L'authentification est assurée par un mot de passe. Les autres services dépendent également de la société qui fournit le compte. Comme avec `/etc/passwd`, ce système d'identité est centralisé et nécessite une confiance totale dans un fournisseur d'identité. Ces services d'identité centralisés inquiètent donc beaucoup d'utilisateurs, à juste titre.

## 1.2 L'identité

La section précédente présentait ma vision de l'identité. Mais le thème étant récent, il en existe d'autres approches. Une des plus connues est celle popularisée par Kim Cameron, architecte Identité chez Microsoft et connue sous le nom des « Sept lois de l'identité ».

Cameron présente l'identité comme une suite d'affirmations comme « Cet étudiant a la carte d'étudiant n° 234-12-81 » ou bien « Kim Cameron est de nationalité états-unienne » ou encore « Cette personne a plus de dix-huit ans ». Ces affirmations peuvent éventuellement être étayées, c'est l'authentification.

Cameron critique fortement les systèmes centralisés, y compris le projet Passport de sa société, Il plaide au contraire pour un métasystème d'identité, qui pourrait fédérer des systèmes divers, gérés par des organismes différents et proposant des services différents.

Ses principes sont résumés en sept lois :

1. Contrôle par l'utilisateur. La nécessité du consentement éclairé est posée en principe. Notons que Cameron, malgré l'utilisation qu'il fait du terme de « loi », n'envisage pas de traduction législative de ses principes. Cette première loi ressemble beaucoup à des lois comme Informatique et Libertés en France.
2. Divulgation minimale. Reprenons l'exemple de l'affirmation comme quoi le sujet a plus de dix-huit ans. Pour Cameron, devoir montrer un document d'identité pour prouver son âge dans un bar est excessif car ce document donne plus d'informations que celle qui compte, le fait d'être majeur.
3. Divulgation uniquement aux entités qui en ont besoin. Ainsi, que Google contrôle l'identité de ceux qui accèdent à Gmail est raisonnable mais on ne voit

[2] L'erreur a, par exemple, souvent été commise dans le contexte de la lutte anti-spam, où des gens ont annoncé que des technologies comme SPF ou DKIM étaient des échecs car les spammeurs les utilisent également. C'est un contre-sens complet. Ces technologies authentifient, les spammeurs comme les autres, elles ne fournissent pas en soi de garantie de confiance.

OpenID, comme toutes les techniques permettant l'authentification, doit être vu comme une fondation, sur laquelle on bâtit des services de confiance et d'autorisation.

[3] Certaines personnes estiment d'ailleurs qu'il ne s'agit pas ici d'identité, celle-ci étant, pour elles, uniquement ce qui est contrôlé par l'utilisateur.

[4] Avant PAM, Kerberos et systèmes équivalents.

pas pourquoi l'identité Google devrait être utilisée par une autre société.

4. Identité dirigée. Certains systèmes d'identité diffusent à tous (c'est le cas des passeports RFID et des passes Navigo, facilement lisibles à distance). Parfois, c'est souhaitable (il existe des services à vocation publique), parfois non et un système d'identité devrait être directionnel.
5. Pluralisme des acteurs et des techniques. Tirant les leçons de l'échec de Passport, Cameron affirme qu'il n'y aura pas de système unique.
6. Prise en compte des facteurs humains. Cameron fait remarquer que des techniques excellentes sur le papier, comme l'utilisation de certificats X509 pour authentifier les sessions SSL, ont échoué pour des raisons non techniques, par manque de prise en compte des utilisateurs (« Voulez-vous vraiment autoriser ce certificat signé par une autorité inconnue ? »).
7. Maintien d'une certaine cohérence, malgré la variété des acteurs et des technologies. Par exemple, ses différentes identités doivent, pour l'utilisateur, apparaître de manière similaire (ce que font les cartes de Cardspace).

# 2 OpenID

Voyons maintenant OpenID lui-même.

## 2.1 Bloguons

Commençons par un scénario d'usage très classique, celui du lecteur de blogs qui aime bien laisser des commentaires intelligents un peu partout. Aujourd'hui, il a le choix entre des commentaires anonymes et se créer un compte auprès de la plate-forme de blog.

Les commentaires anonymes (on peut souvent indiquer son nom mais rien n'est vérifié) empêchent la construction progressive d'une réputation. Rien ne distingue un contributeur régulier et apprécié du zozo qui va juste écrire une ligne en langage SMS.

Le fait d'avoir un compte permet de se construire une réputation progressive, mais ces comptes n'étant pas consolidés entre les plate-formes, il faudra en créer un à chaque fois, se souvenir des mots de passe, et donner des informations plus ou moins personnelles.

Le jour où tous les blogs auront OpenID, le commentateur n'aura à indiquer que son identité, son URL et le blog, rebaptisé RP pour Relaying Party, contactera le fournisseur d'identité (OP pour OpenID Provider[5]) qui lui garantira l'identité du commentateur et, éventuellement, lui passera quelques informations personnelles.

Comment va donc faire notre commentateur ? Dans le cas le plus simple, il va ouvrir un compte, un seul, auprès d'un fournisseur d'identité, l'OP. Il en existe plusieurs, souvent gratuits. Notons tout de suite que n'importe qui peut être OP, y compris sur sa petite machine Unix. Il n'y a pas besoin d'autorisation ou d'enregistrement pour cela, et c'est une des grosses forces d'OpenID par rapport à des techniques comme X509.

Une fois le compte ouvert, notre utilisateur a une identité, mettons `http://bortzmeyer.-myopenid.com/` Cette identité va fonctionner et je peux me connecter à tous les blogs qui acceptent OpenID et m'authentifier, via les serveurs de myOpenID.

C'est très simple et cela marche tout de suite. Mais il y a un problème, le nom de l'OP, ici myOpenID, apparaît dans l'URL. Cela veut dire que je suis lié à mon fournisseur d'identité, à l'OP, je ne peux pas en changer facilement. L'un des buts d'OpenID étant de mettre l'identité sous le contrôle de l'utilisateur, il serait préférable d'avoir un nom de domaine à soi et un identificateur OpenID qui en découle, indépendant de l'OP.

Cela se fait en indiquant dans une page HTML les coordonnées de l'OP. Prenons l'exemple de mon identité `http://www.bortzmeyer.org/`. À cette adresse écoute un serveur HTTP qui sert une page qui contient :

```
<link rel="openid.server"
href="http://www.myopenid.com/server/"/>
<link rel="openid.delegate"
href="http://bortzmeyer.myopenid.com/"/>
```

Ces élements XHTML indiquent quelle est l'URL du serveur OpenID à utiliser et quelle identité utiliser auprès de ce dernier.

Ainsi, je dispose d'une identité stable, et je peux changer d'OP à ma guise.

Lorsque je m'authentifie auprès d'un site Web (le RP) qui accepte l'OpenID, ce RP se connecte à `www.bortzmeyer.org` puis, lisant les éléments `<link>` ci-dessus, se connecte à l'OP, me redirige vers lui pour l'authentification (myOpenID utilise des mots de passe transmis sur HTTPS) puis l'OP me redirige vers la page Web du site, auprès duquel je suis désormais authentifié.

[5] Le RP se nommait autrefois *consumer* et l'OP *server*. OpenID 2 a adopté une terminologie plus proche de celle utilisée par les autres systèmes.

Je peux désormais me connecter sur des sites OpenID, des plus sérieux comme http://www.livejournal.com/ aux plus rigolos comme http://jyte.com/.

Rappelons que j'ai juste été authentifié, autrement dit que le RP est désormais convaincu que je contrôle bien http://www.bortzmeyer.org/. Il me reste à bâtir une réputation sur cette identité mais c'est une autre histoire, qui ne dépend pas d'OpenID...

## 2.2 Identité

Après cette vision orientée utilisateur, voyons plus précisément ce que fait OpenID des cinq points présentés p. 1.

Le point 1, l'attribution d'un identificateur, est assuré par l'URL. Ce doit être un URL et pas n'importe quel URI car il doit être résolvable, pour que le site Web qui veut une identité, le RP, puisse aller chercher la page. Notons aussi que OpenID dépend de l'infrastructure du DNS (l'ICANN, les registres, etc). Avec OpenID, vous n'êtes donc pas un numéro, vous êtes un URL.

Le point 2, l'authentification, n'est pas réellement fait par OpenID, puisque chaque OP est libre d'authentifier comme il le souhaite. OpenID joue surtout un rôle de médiateur entre le RP et l'OP.

Le point 3, l'envoi de méta-données, est assuré par des extensions au protocole comme OpenID Simple Registration Extension.

Les points 4 et 5 sont complètement en dehors du domaine d'OpenID. OpenID peut servir à établir une identité, qui servira alors d'entrée dans des registres, genre liste blanche ou liste noire. (Jyte est un exemple pour rire d'un tel registre.)

## 2.3 Mettre OpenID sur un RP

Si on veut permettre aux visiteurs de son site Web de s'authentifier via OpenID, quelles sont les solutions ? D'abord, notons que certains logiciels ont déjà un support OpenID. Par exemple, le moteur de blog PyBlosxom a déjà un greffon OpenID.

Mais si on a tout programmé soi-même ou bien si on utilise un logiciel qui n'a pas encore OpenID, est-ce compliqué à ajouter ?

Le protocole est beaucoup plus simple que tous ses concurrents et écrire une bibliothèque OpenID est souvent une tâche réaliste. Toutefois, plutôt que de réinventer la roue, il est sans doute préférable de partir d'une bibliothèque existante, il y en a dans plusieurs langages de programmation (Ruby, Python, PHP, Perl, Lua, C#, Java et bien d'autres). Les exemples ci-dessous utilisent la bibliothèque Python de JanRain.

Supposons donc qu'un `<form>` HTML permette de se loguer et que l'URL `/verify` soit appelé ensuite. Notre logiciel va devoir appeler le code `do_verify` lorsque cet URL est demandé. Ce code va :

```
# Se connecter au serveur HTTP pour connaître l'OP
openid_url = self.query.get('openid_identifier')
request = oidconsumer.begin(openid_url)
# Renvoyer l'utilisateur à l'OP
redirect_url = request.redirectURL(trust_root, return_to)
self.redirect(redirect_url)
```

Puis, lorsque le code sera à nouveau appelé par l'utilisateur après la redirection initiée par l'OP, on vérifie la réponse :

```
info = oidconsumer.complete(self.query)
if info.status == consumer.FAILURE:
   fmt = "Verification of %s failed."
elif info.status == consumer.SUCCESS:
   fmt = "You have successfully verified %s"
```

L'essentiel du travail, notamment la gestion des associations (voir section suivante) est faite par la bibliothèque.

## 2.4 Le protocole

Que se passe t-il sur le câble lors d'une authentification avec OpenID ? On l'a vu, le site Web, le RP, doit d'abord se connecter, en HTTP normal et classique, avec un GET simple. Il récupère et analyse la page pour y trouver les coordonnées du serveur OpenID.

OpenID utilise le cadre REST[6] pour toutes les interactions. Il se connecte ensuite au serveur OpenID, en ajoutant à l'URL quelques paramètres standard comme `openid.identity` ou `openid.return_to`. Ce dernier est fabriqué par le RP et doit être protégé contre le rejeu, par exemple en y incorporant un identificateur de session unique. Voici un exemple de requête à l'OP :

```
GET
/openidserver?openid.assoc_handle=%7BH-
MACSHA1%7D%7B46f0c538%7D%7BHgdwIQ%3D%3D%
7D&openid.identity=http%3A%2F%2Fwww.exam
ple.org%3A8000%2Fstephane&openid.mode=ch
eckid_setup&openid.return_to=http%3A%2F%
2Fwww.example.org%3A8001%2Fprocess%3F-
nonce%3DTGq6UNHM&openid.trust_root=http%
3A%2F%2Fwww.example.org%3A8001%2F
```

L'OP authentifie alors l'utilisateur. Il est important de noter que le mécanisme utilisé n'est pas spécifié par OpenID. L'OP peut utiliser des traditionnels mots de passe, un certificat X509,

[6] REST (REpresentational State Transfer) est une formalisation de l'architecture Web+HTTP classique, utile notamment pour construire des services Web sans le lourd attirail de SOAP.

une clé PGP, ou même l'adresse IP de l'utilisateur combinée avec les phases de la lune.

C'est l'occasion de rappeler qu'OpenID concerne l'identité, pas la confiance. C'est l'utiisateur qui choisit son OP et le RP doit, soit faire une confiance aveugle à l'OP, soit gérer une liste blanche ou noire des bons ou mauvais OP.

Une fois l'authentification faite, l'OP envoie une redirection (statut HTTP 302) à l'utilisateur, qui est alors ramené à la page du RP, authentifié. Voici un exemple de la dernière requête à l'OP :

```
GET /process?nonce=ou0A2TRO&openid.assoc_handle=%7BHMAC-SHA1%7D%7B46f0c538%7D%7BHgdwIQ%3D%3D%7D&openid.identity=http%3A%2F%2Fwww.example.org%3A8000%2Fstephane&openid.mode=id_res&openid.return_to=http%3A%2F%2Fwww.example.org%3A8001%2Fprocess%3Fnonce%3Dou0A2TRO&openid.sig=Vqei7fEkmY6zzQwZG5iXBYwJrZ0%3D&openid.signed=return_to%2Cmode%2Cidentity
```

Avez-vous noté le paramètre `openid.sig` ? C'est la signature cryptographique de l'OP, permettant d'être sûr qu'un OP a bien signé cette identité (n'oubliez pas que ce dernier GET a été envoyé par le navigateur de l'utilisateur et peut donc être fabriqué de toutes pièces). Mais comment la vérifier ? A priori, un RP quelconque ne connait pas la clé de l'OP, et nous ne voulons pas bâtir une usine à gaz genre X509 pour qu'il la connaisse.

Il y a deux méthodes principales. Dans l'une, le RP va recontacter l'OP une nouvelle fois, pour lui demander de vérifier la signature. Cette méthode est entièrement REST et donc sans état.

Dans l'autre, le RP et l'OP vont d'abord s'associer, se mettre d'accord sur un secret partagé, suivant la méthode de Diffie-Hellman. Ce secret partagé permettra au RP de vérifier les signatures de l'OP.

## 2.5 Créer un OP

De même qu'il est assez simple de doter un site Web de capacités OpenID, on peut devenir son propre fournisseur d'identité sans trop de mal. Une des solutions est d'utiliser un serveur OpenID tout fait, comme Gracie, qui authentifie en utilisant le service PAM d'Unix et transforme donc tous les comptes locaux en identificateurs OpenID.

Mais on peut aussi vouloir écrire le code soi-même. Là encore, les bibliothèques existantes facilitent nettement le travail. Reprenant la bibliothèque Python de JanRain :

```
request = self.server.openid.decodeRequest(query)
if request.mode in
["checkid_immediate", "checkid_setup"]:
    if (self.user and
        self.isAuthorized(request.identity,
                          request.trust_root)):
        response = request.answer(True)
    elif request.immediate:
        # Pas d'utilisateur. Refus.
        response = request.answer(
            False,
           server_url=self.server.base_url + \
               'openidserver')
    else:
         # On demande à l'utilisateur
        self.server.lastCheckIDRequest[self.user]
                          = request
        self.showDecidePage(request)
else:
    # Demande d'établissement d'une association.
    # On sous-traite.
    response = self.server.openid.handleRequest
                              (request)
```

Notons que ce code ne montre pas l'authentification elle-même, puisqu'elle dépend de chaque OP (mots de passe en clair dans une base de données, examen d'un certificat, etc). Ce qu'il gère, par contre, c'est la rémanence (si l'utilisateur a déjà été authentifié et qu'il a accepté que son authentification soit réutilisé, alors, on peut répondre Oui tout de suite).

## 2.6 Le risque de hameçonnage

Une des faiblesses possibles d'OpenID est le risque d'hameçonnage[7], c'est-à-dire de détournement de l'utilisateur vers un autre site que son OP habituel. Ainsi trompé, l'utilisateur donnerait son mot de passe au méchant.

Quelle est l'ampleur du risque et comment le limiter ?

Le risque existe pour toutes les technologies d'identification lorsqu'un mot de passe est réutilisable. L'expérience montre la vanité qu'il y a à espérer que l'utilisateur scrute le nom de domaine ou le certificat X509, ou sache distinguer le chrome[8] du navigateur de la page Web.

Mais OpenID a une vulnérabilité particulière, qui est la redirection à laquelle procède le RP. Le RP n'est pas a priori digne de confiance et il pourrait automatiquement rediriger vers un méchant OP. OpenID, comme beaucoup de techniques réseaux, fait face ici à la contradiction entre la facilité d'usage (pour laquelle on a la redirection) et la sécurité.

---

[7] *Phishing* en anglais

[8] Le chrome désigne les éléments de l'interface utilisateur qui ne sont pas sous le contrôle du site Web qu'on est en train de regarder, la barre de menus, par exemple. Une technique courante d'hameçonnage est de mettre dans la page Web des éléments rassurants (un cadenas solidement fermé, par exemple), en comptant que l'utilisateur ne voit pas la contradiction avec les éléments du chrome.

Comment limiter les risques ? Plusieurs approches sont étudiées. L'une est de supprimer ou de limiter la redirection en obligeant l'utilisateur à taper l'URL de son OP, ou bien en lui faisant utiliser un signet. Cela rendrait l'usage d'OpenID plus difficile.

Les techniques anti-hameçonnage classiques peuvent aussi être utilisés. Par exemple, l'OP peut demander, à la création du compte, que l'utilisateur choisisse une image, qu'il lui montrera à chaque tentative d'authentification. Cette image est un secret partagé entre l'utilisateur et l'OP, un éventuel méchant ne pourrait pas la connaître et donc pas l'afficher. Comme toutes les techniques reposant sur l'éducation de l'utilisateur, on peut craindre que son déploiement soit long et difficile.

Une autre solution serait de rendre l'hameçonnage inutile en supprimant les mots de passe réutilisables. Si l'OP utilise une authentification à clé cryptographiques asymétriques, un OP méchant ne pourrait récupérer aucune information utile.

# 3 Systèmes analogues

Il existe de nombreux systèmes d'identité. Il est difficile de les comparer avec OpenID car la plupart n'ont pas le même cahier des charges et ne traitent pas les mêmes points (voir la liste des composants d'un système d'identité en p. 1).

Éliminons tout d'abord les systèmes centralisés comme Passport de Microsoft. Ces systèmes ne peuvent pas convenir à tous les besoins et, même s'ils le pouvaient, il serait très dangereux de n'avoir qu'un seul fournisseur d'identité.

Parmi les systèmes décentralisés (on dit parfois « Centrés sur l'utilisateur », les systèmes centralisés étant centrés sur le fournisseur d'identité), notons Cardspace de Microsoft, Shibboleth et Liberty Alliance.

Plusieurs de ces systèmes utilisent SAML[9], une norme OASIS permettant d'exprimer des assertions de sécurité en XML. SAML spécifie un format et aussi un protocole, utilisant SOAP, pour interroger un fournisseur d'identité. Voici un exemple du format SAML, indiquant que John Smith est authentifié (naturellement, l'assertion elle-même est authentifiée, par exemple via XML Signature) :

```
<saml:Assertion
   <saml:Conditions NotBefore="2004-06-19T17:00:37.795Z"
        NotOnOrAfter="2004-06-19T17:10:37.795Z"/>
    <saml:AuthenticationStatement
     AuthenticationMethod="urn:oasis:names:tc:SAML:1.0:am:
password"
   <saml:Subject>
       <saml:NameIdentifier>John_Smith
                              </saml:NameIdentifier>
   </saml:Subject>
 </saml:AuthenticationStatement>
</saml:Assertion>
```

Cardspace est un système très riche et très complexe, qui utilise toutes les capacités de SAML. Il est surtout connu, par sa face visible, les jolies cartes d'identité qu'on voit sur les copies d'écran, et que l'utilisateur peut choisir de présenter aux RP. Mais les détails techniques manquent.

Cardspace peut utiliser OpenID, comme technique sous-jacente (et Microsoft participe activement à la conception d'OpenID).

Shibboleth est bâti, comme Cardspace, sur SAML. Très complexe également, mais disposant d'une implémentation en logiciel libre, il fait l'objet d'un gros projet d'Internet 2, de déploiement dans plusieurs universités états-uniennes.

Il existe une fédération Shibboleth en France, au CRU.

Liberty Alliance était surtout un contre-feu aux prétentions de Microsoft avec Passport. Ce projet, qui avait contribué à populariser l'idée de métasystème d'identité, semble aujourd'hui suspendu.

# 4 Futures directions

La version 1 d'OpenID se caractérisait par son extrême simplicité, surtout par rapport à des projets comme Shibboleth. OpenID pouvait donc proclamer suivre un principe traditionnel de l'Internet, faire simple, afin d'être largement déployé. Les protocoles vainqueurs sur Internet ont souvent été les plus simples, plutôt que les grosses architectures conçues en détail par des comités.

Mais OpenID 2, encore non officiel, s'écarte de cette voie, en intégrant bien d'autres choses comme les XRI ou Yadis.

XRI est une norme Oasis qui relève, selon certains, du syndrome du « Not Invented Here ». À partir de demandes fréquentes (notamment la permanence des identificateurs ou leur internationalisation) mais aussi d'une analyse contestée des forces et faiblesses des URL, Oasis a produit une nouvelle famille d'URI. Parmi les XRI, les plus connus sont les i-names. Si j'en avais un, il s'écrirait `=Stéphane.Bortzmeyer`. Un protocole permet de résoudre un XRI en URL.

[9] Security assertion markup language

Yadis, lui, est un mécanisme permettant d'exprimer plusieurs fournisseurs d'identité pour un URL donné[10]. Ces fournisseurs peuvent utiliser des protocoles différents, Yadis n'est pas spécifique à OpenID.

# 5 Conclusion

OpenID est aujourd'hui un standard stable, largement mise en œuvre dans beaucoup de logiciels (ce qui est un tribut à sa simplicité), mais encore relativement peu déployé.

Il ne bénéficie pas du travail d'une organisation de normalisation mais il a l'avantage de fonctionner aujourd'hui et de fournir une solution simple aux problèmes des identificateurs multiples, enfermés dans des silos.

Son avenir est probablement d'être un composant parmi d'autres, des futurs système d'identité décentralisés, pluralistes, et permettant aux utilisateurs de garder ou de reprendre le contrôle de leurs identités.

# Bibliographie


[1] *OpenID: an actually distributed identity system* http://openid.net/ Le site officiel

[2] *OpenID Authentication 1.1*. 2006. Le standard actuel.

[3] *OpenID Simple Registration Extension* 1.0. 2006.

[4] *OpenID Enabled*. http://www.openidenabled.com/ Le site des mises en œuvre, pour tous les langages.

[5] *OpenID Phishing Brainstorm* http://openid.net/wiki/index.php/OpenID_Phishing_Brainstorm

[6] S. Bortzmeyer *Authentifier et autoriser, deux choses différentes.* 2007. http://www.bortzmeyer.org/authentifier-et-autoriser.html

[7] Tom Coates *Social whitelisting with OpenID...* 2007 http://www.plasticbag.org/archives/2007/01/social_whitelisting_w/ Une proposition de partage des autorisations, une fois les personnes authentifiées avec OpenID

[8] Ben Laurie et Mary Rundle *Identity Management as a Cybersecurity Case Study*. 2005. Oxford Internet Institute Conference « Safety and Security in a Networked World: Balancing Cyber-Rights and Responsibilities »

[9] Kim Cameron *Identity Weblog.* http://www.identityblog.com/ Le plus riche blog de discussions sur ls questions d'identité, par le gourou Identité de Microsoft.

[10] Kim Cameron *The laws of identity* http://www.identityblog.com/stories/2004/12/09/thelaws.html Les fameuses « sept lois ».

[11] Serge Aumont *Faut-il brûler vos certificats ?* http://2003.jres.org/actes/paper.21.pdf 2003. Ou pourquoi les certificats X509 n'ont-ils pas été le sucès escompté.

[12] *myOpenID.* https://www.myopenid.com/. Le fournisseur OpenID que j'utilise actuellement.

[13] terrell.myopenid.com *OpenID flow* http://www.lifewiki.net/openid/OpenIDFlow Très bonne description du protocole

[14] *Shibboleth* http://shibboleth.internet2.edu/ Un système de SSO bâti sur SAML

[15] David Chappell *Introducing Windows CardSpace* http://msdn2.microsoft.com/en-us/library/aa480189.aspx Cardspace (ex-Infocard) est le système de gestion d'identité de Microsoft

[16] *Windows Live ID* https://accountservices.passport.net/ppnetworkhome.srf?lc=1033 Live ID (ex-Passport) est un autre système de gestion d'identité de Microsoft, probablement concurrent de Cardspace

[17] *Liberty Alliance* http://www.projectliberty.org/

[18] Yadis *The Identity and Accountability Foundation for Web 2.0* http://yadis.org/

[19] *Higgins* http://www.eclipse.org/higgins/

[20] OASIS *SAML, Defining and maintaining a standard, XML-based framework for creating and exchanging security information between online partners* http://www.oasis-open.org/committees/tc_home.php?wg_abbrev=security


[10] Yadis permet aussi de trouver les fournisseurs d'identité par d'autres méthodes qu'une modification de la page HTML correspond à l'URL-identité. Cela peut être un avantage dans les cas où on n'a pas une maîtrise complète du code HTML de ses pages.