TABLE DES MATIÈRES

Page

LISTE DES TABLEAUX

Page

LISTE DES FIGURES

Page

LISTE DES ABRÉVIATIONS, SIGLES ET ACRONYMES

API	Application Programming Interface (Interface de programmation)
IaaS	Infrastructure as a Service (Infrastructure comme un service)
IdP	Identity Provider (Fournisseur d'identité)
PDP	Policy Decision Point (Serveur de règles)
PEP	Policy Enforcement Point (Client de serveur de règles)
PIP	Policy Information Point (Point d'information de politique)
REST	Representational State Transfer
SOAP	Simple Object Access Protocol
SP	Service Provider (Fournisseur de services)
URI	Uniform Resource Identifier (identifiant uniforme de ressource)
VO	Virtual Organisation (Organisation Virtuelle)

INTRODUCTION

Le concept des nuages informatiques est relativement nouveau, mais se base sur des idées datant des années 60. En effet, en 1961, John McCarthy émettait l'hypothèse qu'un jour, l'informatique pourrait très bien être organisée comme un service public. Il prévoyait que les services informatiques pourraient devenir la base d'une industrie nouvelle (Foster *et al.*, 2008).

Une trentaine d'années plus tard, les premières manifestations de ces services informatiques virent le jour sous la forme de grilles informatiques. Une grille informatique est une infrastructure distribuée permettant de coordonner le partage de ressources et la résolution de problèmes (Foster *et al.*, 2001). Elle a comme but de fournir aux utilisateurs un accès transparent aux ressources, à l'image du réseau électrique (*Electric Grid*) qui offre un accès standardisé à l'électricité (Kesselman et Foster, 1998). Par exemple, dans le cadre d'un projet scientifique demandant une grande puissance de calcul, afin d'accélérer leur recherche, des scientifiques pourraient utiliser la puissance agrégée d'une grande quantité de machines distribuées à travers le monde et membres d'une grille.

Une dizaine d'années plus tard vint l'arrivée des premiers nuages informatiques. Il n'y a pas de consensus en ce qui concerne la définition des nuages informatiques. Par exemple, dans SYS-CON Media (2008), on cite 21 définitions. En fait, le terme *nuage informatique* fait référence à un ensemble de principes et technologies. Il ne prescrit pas des techniques rigides devant être employées. La définition suivante retient l'essentiel de ce qui est accepté dans la littérature :

> « A large-scale distributed computing paradigm that is driven by economies of scale, in which a pool of abstracted, virtualized, dynamically-scalable, managed computing power, storage, platforms, and services are delivered on demand to external customers over the Internet. » (Foster *et al.*, 2008)

Un nuage informatique partage beaucoup de caractéristiques avec une grille informatique. En réalité, l'architecture d'un nuage informatique peut très bien reposer sur des technologies de grilles informatiques. Dans les deux cas, on utilise des ressources distribuées à travers divers

sites afin de réaliser des objectifs applicatifs et des économies d'échelle. Les nuages informatiques vont par contre un peu plus loin puisqu'ils permettent la virtualisation de matériel et d'infrastructure. Cette technologie permet d'abstraire les composantes physiques d'un système, les rendant transparentes aux applications et services fournis par celui-ci. On peut donc du même coup bénéficier de partage et d'approvisionnement de ressource de manière beaucoup plus dynamique (Foster *et al.*, 2008; Zhang *et al.*, 2010).

Une des premières entreprises ayant basé son modèle d'affaires sur les nuages informatiques est la compagnie Salesforce.com (Salesforce.com, 2010), fondée en 1999. Celle-ci fournit des services et des applications à des entreprises. Ces services et applications sont accessibles sur Internet, en utilisant un fureteur et ne nécessitent pas l'installation de logiciels additionnels.

Par la suite vint l'arrivée d'entreprises fournissant l'accès à de l'infrastructure virtuelle. Par exemple, en 2006, Amazon a annoncé qu'elle utiliserait son infrastructure physique pour offrir des services de stockage de fichiers et de machines virtuelles au public (Businessweek, 2006). Il devenait ainsi possible de créer une infrastructure informatique complète sur du matériel physique loué à l'utilisation.

Finalement, depuis plus récemment, une organisation peut utiliser différentes plateformes afin de créer et gérer une infrastructure virtuelle qui est hébergée sur sa propre infrastructure physique. Parmi les plateformes permettant de réaliser une telle chose, mentionnons OpenStack (OpenStack, 2011), OpenNebula (OpenNebula, 2010), Eucalyptus (Eucalyptus Systems, 2010) et le IaaS Framewok (IaaS Framework Team, 2010), ce dernier est piloté par la compagnie Inocybe (Inocybe Technologies, 2007) et il est le sujet de ce mémoire. Une analyse préliminaire des travaux décrits dans ce mémoire a été publiée dans Tellier *et al.* (2010) et un article décrivant les travaux eux-mêmes a été soumis pour publication au *International Journal of Grid Computing and eScience*.

CHAPITRE 1

PROBLÉMATIQUE

1.1 Mise en contexte

Un des avantages clés des nuages informatiques vient du fait qu'ils peuvent être utilisés par des entreprises et des individus qui désirent externaliser certains de leurs besoins informatiques vers un fournisseur de nuage. Ce fournisseur peut ensuite fournir des services payés à l'utilisation à ses clients. D'un point de vue financier, cette relation est intéressante parce qu'elle permet à toutes les parties de réaliser des économies d'échelle. En effet, un fournisseur peut acquérir des ressources physiques en grande quantité pour servir ses nombreux clients.

Cependant, les offres actuelles de nuages informatiques comportent certains problèmes (Rochwerger *et al.*, 2009). L'un d'entre eux vient du manque d'interopérabilité entre les fournisseurs. La plupart des solutions de nuages informatiques n'ont pas été conçues pour être interopérables. Ceci empêche les fournisseurs de se regrouper. Des fournisseurs pourraient vouloir coopérer pour l'atteinte d'un objectif commun ou bien pour la conquête de nouveaux marchés. Par exemple, un ensemble de fournisseurs possédant des centres de données verts pourraient vouloir mettre leurs ressources en commun afin de bâtir un réseau plus propre. Un groupe de petits fournisseur pourraient aussi vouloir se regrouper afin de créer un nuage plus important que ce que chacun pourrait mettre en place individuellement. Finalement, le manque d'interopérabilité interfournisseur promeut le verrouillage des clients envers un vendeur. Notons toutefois que les regroupements interfournisseurs sont presque exclusivement bénéfiques aux petits et moyens fournisseurs. Des organisations de grande taille telles que Google et Amazon possèdent déjà des infrastructures et des nuages informatiques capables de rivaliser avec n'importe quel regroupement de fournisseurs de plus petite taille.

Les lacunes reliées à l'impossibilité des fournisseurs de se regrouper peuvent être résolues en permettant aux fournisseurs de fédérer leurs ressources et d'agréger leurs capacités. Ceci permet la création d'organisations virtuelles (VO) dynamiques sur lesquelles peuvent reposer des

nuages informatiques fédérés. De tels nuages ont le potentiel d'être constitués d'une quantité de ressources beaucoup plus importante que ce dont pourrait disposer n'importe quel fournisseur unique.

Ce mémoire présente une architecture de gestion de l'identité conçue pour les plateformes de nuages informatiques. Afin de la valider dans un contexte concret, l'architecture a été implémentée dans le IaaS Framework (IaaS Framework Team, 2010). Cette plateforme a comme but de gérer des infrastructures physiques afin de bâtir des infrastructures de nuages informatiques flexibles. À partir de l'infrastructure physique dont elle dispose, elle est en mesure de fournir plusieurs types de ressources virtuelles comme des machines virtuelles, des routeurs, des commutateurs, des unités de stockages, etc. En se regroupant en VO, des fournisseurs distincts peuvent collaborer et partager ces ressources entre eux de manière transparente pour leurs clients lorsqu'ils leur offrent des services. De plus, les entités participant à la VO peuvent la quitter à tout moment et de nouvelles entités peuvent la rejoindre. Ceci lui donne un caractère dynamique.

L'architecture de gestion de l'identité et du contrôle d'accès du IaaS Framework doit donc être en mesure de gérer des nuages informatiques traditionnels, mais aussi des nuages informatiques fédérés *dynamiques* qui reposent sur des VO. Ce concept provient en fait du monde des grilles informatiques, qui partage plusieurs caractéristiques avec le monde des nuages informatiques (Foster *et al.*, 2008). Il est fréquent pour des organisations participant à une grille de se regrouper temporairement, pour les besoins d'un projet donné. Une telle collaboration pourrait par exemple être formée spécifiquement pour les besoins d'un projet scientifique demandant un accès à une grande puissance de calcul.

Le dynamisme des nuages informatiques permis par le IaaS Framework touche trois axes. Premièrement, les services de ces nuages peuvent être créés sur des VOs dynamiques où de nouveaux membres peuvent s'ajouter ou quitter à tout moment. Dans un tel cas, les ressources virtuelles doivent être en mesure de migrer en temps réel et d'être hébergées chez n'importe quel fournisseur. Finalement, le parc d'utilisateurs de ces ressources peut être modifié à tout

moment et avoir des droits d'accès variables. Bien entendu, ces modifications de droits d'accès doivent se faire de manière cohérente. Par exemple, si des permissions sont ajoutées ou révoquées à des rôles définis dans une politique de sécurité d'un fournisseur, il est important que les modifications soient prises en compte le plus rapidement possible, et ce, pour toutes les requêtes visant n'importe quelles ressources du fournisseur.

Un autre défi de la solution concerne sa flexibilité en terme de gestion des politiques de sécurité. Étant donné que le IaaS Framework est un cadre logiciel, il est utilisé comme fondation pour d'autres projets. Par conséquent, il se doit d'être le plus flexible possible afin qu'un grand éventail de projets puissent l'utiliser comme point de départ. Sur le plan de la gestion des politiques de sécurité, ceci ce traduit en la possibilité de prendre des décisions de contrôle d'accès sur la base de n'importe quel attribut du sujet effectuant une requête, de la ressource étant la cible de la requête et de l'environnement dans lequel est effectué la requête.

Finalement, la gestion des politiques doit être en mesure de supporter efficacement le caractère dynamique des nuages informatiques et des VOs sur lesquels ils sont bâtis. Lors de l'ajout et la suppression de ressources et sujets, l'adaptation des politiques doit pouvoir se faire le plus rapidement et le plus facilement possible.

1.2 Importance de la recherche

Ce mémoire fait plusieurs contributions dans le domaine des nuages informatiques. Premièrement, il établit les besoins devant être considérés lors de la conception d'une architecture de gestion de l'identité et des droits d'accès pour une plateforme de nuages fédérés. Ensuite, une solution répondant à ces besoins est élaborée. L'implémentation de cette solution dans une plateforme de nuages fédérés est décrite et les standards et technologies qu'elle utilise sont exposés. Même si les travaux décrits dans ce mémoire sont reliés au IaaS Framework, ils sont assez généraux pour être pertinent dans plusieurs autres contextes de nuages informatiques fédérés.

Ensuite, la solution élaborée se base sur d'autres composants logiciels existants et développés par des tiers. De nouvelles fonctionnalités ont dû être ajoutées à certains de ces composants afin qu'ils puissent satisfaire les besoins du IaaS Framework. Deux contributions significatives ont été rendues publiques. Premièrement, une extension au fournisseur d'identité Shibboleth a été développée afin de lui permettre de communiquer avec une base de données OrientDB. En outre, la prise en charge du profil SAML ECP a été ajoutée au projet *Spring Security Extensions* (voir Sections 5.2.1 et 5.2.2).

Finalement, puisque les résultats de ces travaux sont intégrés au IaaS Framework, ils profiteront directement à plusieurs produits commerciaux et projets concrets. Le projet le plus important qui utilise cette plateforme actuellement est le réseau GreenStar (voir Section 3.2), mais étant donné que le IaaS Framework suit un modèle au code source libre, il pourrait être utilisé par n'importe quel autre projet.

1.3 Formalisation de métabesoins

Considérant la problématique exposée à la Section 1.1, voici une énumération des métabesoins généraux devant être satisfaits. Ceux-ci sont précisés et inscrits dans le contexte du IaaS Framework à la section 4.2 :

1. Autoriser la coopération entre des petits fournisseurs afin de leur permettre de (i) atteindre des objectifs communs et (ii) conquérir de nouveaux marchés et (iii) réduire le verrouillage envers les fournisseurs.

2. Permettre la création de VOs et d'environnements dynamiques pouvant être joints et quittés par leurs membres à tout moment. Ceci doit se faire avec le minimum d'adaptations aux politiques de sécurité en place.

3. Les utilisateurs et propriétaires des ressources doivent pouvoir modifier leurs droits d'accès à tout moment et ces modifications doivent être effectives immédiatement dans l'ensemble de la fédération.

4. Les décisions de contrôle d'accès concernant une requête doivent pouvoir être prises sur la base de n'importe quel attribut du sujet, de la cible et de l'environnement

1.4 Délimitations

Les travaux décrits dans ce mémoire se concentrent uniquement sur les problèmes de sécurité reliés à la gestion de l'identité et au contrôle d'accès. Aucun autre aspect de sécurité n'est traité. De plus, les aspects de gestion de l'identité touchant les questions de respects de la vie privée ne sont abordés que de manière théorique. Étant donné que la compagnie Inocybe, le gestionnaire du IaaS Framework, n'a pas jugé cette facette comme prioritaire, des efforts significatifs n'y ont pas été investis.

CHAPITRE 2

REVUE DE LA LITTÉRATURE ET NOTIONS THÉORIQUES

Ce chapitre explique les notions théoriques nécessaires à la compréhension du problème de la gestion de l'identité et du contrôle d'accès dans le IaaS Framework ainsi que celles nécessaires à la compréhension du contexte de ce projet. Différentes approches pouvant être employées pour aborder ce problème sont aussi introduites.

2.1 Le paradigme IaaS

Le modèle de nuage informatique qui est largement accepté est composé de quatre couches (Zhang *et al.*, 2010). La couche située au plus haut niveau contient les *applications* telles que les services de partage de vidéos ou de photos, les réseaux sociaux et les suites bureautiques en ligne. Ensuite vient la couche *plateforme*. Celle-ci est constituée de systèmes d'exploitation et de cadres applicatifs et a comme objectif de faciliter le déploiement d'applications. Deux exemples de solutions situés à ce niveau sont l'AppEngine (Google, 2011) de Google et Azure (Microsoft, 2011) de Microsoft. Ces solutions offrent des APIs permettant aux développeurs d'interagir avec différents services tels que l'accès à une base de données ou à une unité de stockage. Sans cette couche, les développeurs auraient à interagir directement avec la couche *infrastructure* et à déployer manuellement leurs applications et les services dont elles ont besoin. La couche *infrastructure* intègre quant à elle des technologies de virtualisation qui effectuent le partitionnement des ressources physiques en ressources virtuelles. Finalement, la couche *matériel* gère des ressources physiques telles que des hôtes, des routeurs ou des systèmes d'alimentation. Le paradigme IaaS opère aux couches infrastructure et matériel. En ce qui concerne les deux autres paradigmes des nuages informatiques, SaaS et PaaS, ils opèrent respectivement aux couches applicative et plateforme.

La Figure 2.1 illustre les différentes couches du modèle de nuage informatique ainsi que des exemples d'applications à chaque niveau.

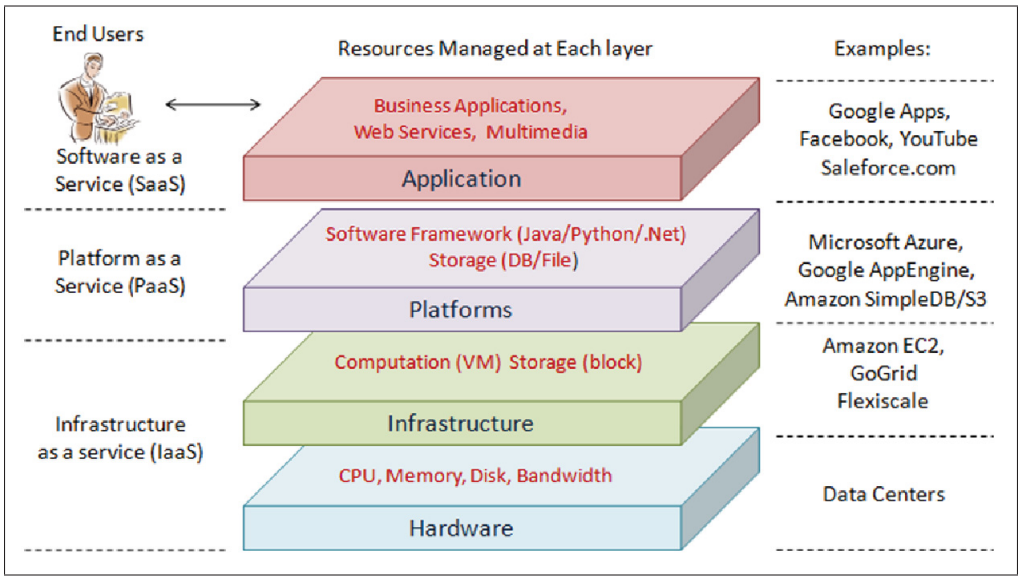

Figure 2.1 L'architecture en couche des nuages informatiques
Tiré de Zhang *et al.* (2010)

Une fois qu'une solution IaaS est en place, elle peut être employée pour créer les trois types de nuages informatiques décris dans Zhang *et al.* (2010) : les nuages publics, privés et hybrides. Un nuage public fournit des ressources au public en général. Par exemple, le service EC2 d'Amazon (Amazon, 2010) permet à n'importe qui de devenir propriétaire d'une machine virtuelle hébergée sur l'infrastructure d'Amazon. Un nuage privé est quant à lui conçu pour être utilisé exclusivement à l'intérieur d'une organisation. Ce concept peut être comparé aux centres de données internes à une organisation avec la principale différence que les ressources physiques sont virtualisées et partitionnées afin d'être utilisées d'une manière optimale. Finalement, un nuage hybride est une combinaison des deux autres types de nuage. Il comporte certaines composantes publiques et d'autres privées.

Eucalyptus (Eucalyptus Systems, 2010), OpenNebula (OpenNebula, 2010) et OpenStack (OpenStack, 2011) sont des exemples de solutions IaaS. Eucalyptus essaie de donner le pouvoir aux utilisateurs en ce qui concerne l'allocation de leurs ressources virtuelles, tant qu'elle reste dans les limites des ententes de service qu'ils ont avec leurs fournisseurs. D'un autre côté, avec OpenNebula, les administrateurs ont le contrôle sur l'allocation des ressources virtuelles. En ce qui concerne OpenStack, il s'agit d'une solution suivant la même philosophie qu'ObenNe-

bula, mais qui se veut plus complète. Elle offre en effet des services supplémentaires tels que l'archivage de données et la gestion d'image de disques virtuels. OpenNebula et OpenStack possèdent aussi des fonctionnalités qui sont absentes de chez Eucalyptus. Citons par exemple la migration en temps réel des ressources virtuelles, des capacités de mise à l'échelle avancées et la possibilité de définir des politiques complexes de placement de ressources. Les trois solutions procurent aussi un certain degré de compatibilité avec l'API (Amazon, 2011) de la solution de nuage informatique d'Amazon. En conséquence, ils sont interopérables avec ce vendeur. Une application interagissant avec le nuage d'Amazon pourrait donc communiquer avec un nuage Eucalyptus, OpenNebula ou OpenStack, sans avoir à être modifiée. Il n'est toutefois pas possible de migrer en temps réel une ressource du nuage d'Amazon vers un nuage OpenNebula ou OpenStack (Obertelli, 2010; Montero, 2009; OpenNebula Project Leads, 2011; Wolski, 2009; OpenNebula Project, 2010; Morgan, 2010; Computing, 2011).

Une autre solution IaaS disponible est le IaaS Framework (IaaS Framework Team, 2010). Celle-ci se distingue des autres plateformes mentionnées précédemment, car elle n'aborde pas le problème de la même manière. Alors qu'Eucalyptus, OpenNebula et OpenStack gèrent des ressources virtuelles appartenant à un seul fournisseur, le IaaS Framework permet à plusieurs fournisseurs de proposer des ressources de façon cohérente. Grâce à ses mécanismes de virtualisation d'équipement réseau, il permet de bâtir des réseaux virtuels reliant les ressources des différents fournisseurs. Ceci n'est pas possible avec les autres solutions. Comme le IaaS Framework permet à différents fournisseurs de se regrouper et d'interagir ensemble, ses besoins en gestion de l'identité et du contrôle d'accès sont plus complexes que ceux des autres plateformes IaaS.

2.2 La gestion de l'identité fédérée

2.2.1 Explication du principe

L'identité numérique est la représentation numérique d'informations connues concernant un *sujet*. Un sujet est peut être une personne, une personne légale (p. ex., une compagnie), un artéfact virtuel (p. ex., une application), un objet tangible, un emplacement ou un regroupe-

ment d'entités représenté dans le domaine numérique et qui doit être pris en compte par un système d'identité (Commons, 2011). En fonction du contexte, un sujet peut être représenté par plusieurs identités distinctes qui sont créées à partir de leur propre ensemble d'*information d'identité* (Priem *et al.*, 2011; Bertino et Takahashi, 2010). Selon la recommandation ITU-T Y.2720 (ITU-T, 2009a), une information d'identité peut entrer dans l'une de catégories suivantes :

Identifiant Type de donné utilisé pour identifier un sujet. Un identifiant peut avoir une portée limitée dans le temps ou l'espace. Par exemple, un URI est globalement unique au fil du temps. Au contraire, un pseudonyme peut expirer et n'est généralement valide que pour un service donné.

Justificatif d'identité (*credential*) Type de donnée prouvant la valeur d'une revendication sur l'identité ou sur une partie de l'identité d'un sujet. Par exemple, un mot de passe prouve l'identité d'un sujet, et une assertion SAML (voir Section 2.4.1) peut prouver la valeur d'un ensemble restreint de ses attributs.

Attribut Propriété caractérisant un sujet et pouvant n'avoir aucune ou plusieurs valeurs (p. ex., prénom, nom de famille, adresse, numéro d'assurance sociale) Commons (2011). Dans cette catégorie, on retrouve aussi les informations biométriques. Une information biométrique est basée sur une caractéristique physique ou comportementale unique à un sujet et est généralement utilisée pour l'identifier (p. ex., empreinte digitale, ADN, ton de la voix). Finalement, un attribut peut faire référence aux activités d'un sujet. Ces activités révèlent les actions qu'il prend dans le monde virtuel (p. ex., sites Web visités, transactions en ligne, communications avec réseau de contacts en ligne) (Bertino et Takahashi, 2010).

Le principe d'un environnement fédéré réfère quant à lui au regroupement de plusieurs domaines administratifs régis par des entités distinctes. Un *domaine* est composé d'un groupe de ressources et de sujets qui sont gérés par une seule entité administrative pouvant être aussi bien une organisation qu'un individu. Les ressources et les sujets d'un domaine sont gouvernés par

un seul ensemble de politiques. L'entité administrative gérant un domaine est aussi responsable des actions de ses composantes (Alpár *et al.*, 2011; Demchenko *et al.*, 2009). Un exemple typique d'un environnement fédéré serait une grille informatique, où ses entités se réunissent afin de partager des données ou de la puissance de calcul.

Le partage de ressources et la fourniture de services sont les deux principaux buts des fédérations. Ils sont réalisés grâce à l'établissement de relations de confiance permettant de distribuer les responsabilités. Ceci autorise des sujets d'un domaine à accéder à des ressources et à des services gérés par un autre domaine. Les attributs d'un sujet ainsi que le processus d'authentification sont pris en charge par une entité généralement située dans son domaine d'appartenance et nommée *fournisseur d'identité* (IdP). Une fois qu'il y est authentifié, le sujet reçoit un justificatif d'identité contenant un sous-ensemble de ses attributs ou de ces droits d'accès, et une référence à son domaine d'appartenance ou à son IdP. Ce justificatif peut être présenté à tout *fournisseur de services* (SP) faisant confiance à ce que l'IdP l'ayant émis ait correctement authentifié l'usager et gère ses attributs de manière appropriée. Ce justificatif est présenté à un SP afin d'accéder aux ressources qu'il détient. Ce processus d'échange de messages d'authentification et d'autorisation est appelé *gestion de l'identité fédérée* lorsque l'IdP et le SP se situent dans des domaines administratifs différents (Chadwick, 2008; Suess et Morroney, 2009; Tellier *et al.*, 2010).

La Figure 2.2 présente l'échange d'information dans un contexte typique d'authentification et d'autorisation fédéré.

1. Le sujet s'authentifie avec son IdP en utilisant l'une des méthodes supportées par les politiques de ce dernier (p. ex., nom d'utilisateur et mot de passe, certificat X.509, biométrie, etc.). Si l'authentification réussit, l'IdP récupère l'information d'identité relative au sujet qui est présente dans sa base de données.

2. L'IdP retourne une preuve de l'authentification (c.-à-d., un justificatif ou un jeton de sécurité) au sujet.

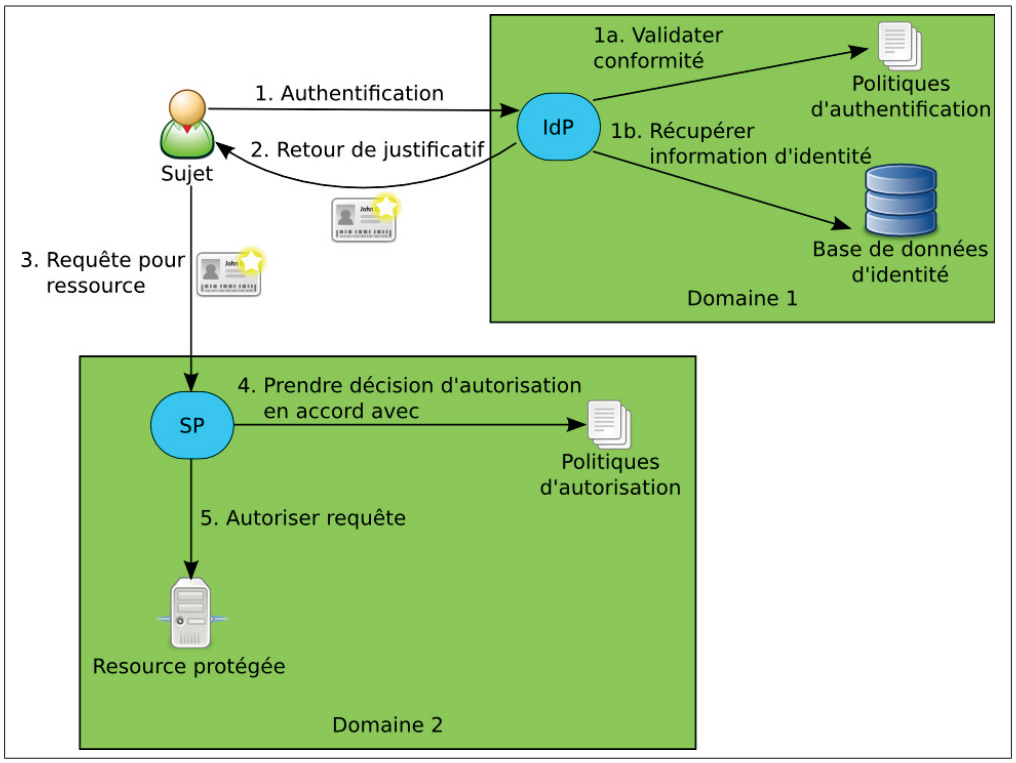

Figure 2.2 Authentification et autorisation dans un environnement fédéré

3. Le sujet fait une requête d'accès pour une ressource protégée auprès du SP. Le jeton émis à l'étape 3 est inclus dans la requête.

4. Le SP analyse le justificatif ainsi que toute autre information pertinente qu'il a à sa disposition. Il prend une décision d'autorisation en fonction de ces données et de ses politiques d'autorisation.

5. Si la requête est acceptée, le sujet est en mesure d'accéder à la ressource.

Avec la gestion de l'identité fédérée, un fournisseur de services peut se concentrer sur sa compétence première, c'est-à-dire, fournir un accès à des ressources. Il n'a pas à traiter avec la gestion de l'identité et n'est pas impliqué dans le processus d'authentification. Il est impliqué dans l'autorisation des sujets. Il réalise ce processus en analysant les justificatifs ou jetons de sécurité émis par les IdPs gérant l'identité des sujets désirant accéder à ces ressources. Dans un même ordre d'idées, un fournisseur d'identité peut se concentrer à gérer l'information et les

justificatifs de ses sujets de manière sécuritaire (Chadwick, 2008; Maler et Reed, 2008; Suess et Morroney, 2009).

2.2.2 Intérêts et rôles des parties prenantes

On peut distinguer quatre parties prenantes ayant chacun des intérêts distincts en ce qui concerne la gestion de l'identité fédérée. La Section 2.2.1 introduit les concepts de *sujet*, *fournisseur d'identité* et de *fournisseur de services*. Une quatrième partie prenante n'a pas été mentionnée jusqu'à présent. Il s'agit des *groupes de contrôle*.

Premièrement, comme il est expliqué à la Section 2.2.1, le fournisseur d'identité est l'entité qui se charge de l'émission et de la gestion des attributs d'un sujet ainsi que de l'émission d'assertions relatives à ces attributs. En outre, un fournisseur d'identité est aussi responsable des justificatifs permettant d'accéder à ces attributs. Il en est aussi de la discrétion du fournisseur d'identité d'émettre ou non un attribut pour un sujet en fonction du justificatif présenté. Par exemple, un IdP pourrait émettre l'attribut *numéro d'assurance sociale* d'un sujet, seulement si ce dernier se présente en personne avec une preuve de citoyenneté. Au contraire, pour un attribut tel que *numéro de passe d'autobus*, le processus de vérification pourrait être moins élaboré.

Les fournisseurs de services, quant à eux, sont les entités qui permettent l'accès à des ressources ou à des services après avoir reçu un justificatif d'un sujet. Il est de la discrétion du fournisseur de services si un justificatif émis par un fournisseur d'identité donné doit être accepté. L'utilisation première de l'identité numérique par les fournisseurs de services est de déterminer avec qui ils communiquent. Ceci est essentiel lorsqu'ils désirent régir l'accès à leurs ressources ou bien offrir des services personnalisés. Dans le cas des entreprises privées, fournir un service adapté aux besoins de leur clientèle peut constituer un avantage compétitif. Il est donc important pour eux de collecter un maximum d'attributs sur leurs clients (Bertino et Takahashi, 2010; Priem *et al.*, 2011).

Or, cette cueillette d'information et sa gestion doivent se faire selon des normes et des standards définis par les groupes de contrôle. Ceux-ci peuvent être des agences gouvernementales ou des organismes de règlementation. Ils doivent être en mesure d'analyser les pratiques des différentes parties prenantes et de recueillir de l'information sur les transactions d'identité à des fins d'audits et d'investigation (Bertino et Takahashi, 2010).

Jusqu'à ce jour, les systèmes de gestion de l'identité ont été principalement poussés par les besoins des entreprises. Comme ces dernières désirent principalement protéger l'accès à leurs ressources critiques, les fonctionnalités fondamentales des systèmes de gestion de l'identité gravitent autour du contrôle d'accès et de l'assignation de droits à des sujets. En outre, les entreprises désirent avoir accès à un ensemble d'information d'identité sur les sujets afin de personnaliser les services fournis et de mieux cibler leurs produits et publicités (Priem *et al.*, 2011).

De leur côté, les sujets ont des préoccupations différentes en ce qui concerne la gestion de leur identité numérique. Même s'il est vrai qu'ils désirent avoir accès à des services personnalisés et à des ressources protégées, ils sont aussi intéressés en la manière dont ils sont représentés. Ils souhaitent avoir un contrôle sur leur réputation et maitriser la divulgation de leurs informations d'identité en fonction d'un contexte précis. Tout comme c'est le cas dans le monde réel, un individu agissant dans le monde virtuel désire être perçu de manière différente en fonction de l'audience et de l'environnement. Ce contrôle sur la communication des informations d'identité est ce qui est généralement appelé *respect de la vie privée* (*privacy*). Ce concept est entre autres menacé par les fuites, vols et abus d'utilisation de l'information et par la collusion entre les entités possédant cette information (Bertino et Takahashi, 2010; Priem *et al.*, 2011).

On voit donc qu'il existe un clivage et des incompatibilités entre les besoins des fournisseurs de services et celui des sujets. D'un côté, les fournisseurs désirent avoir accès à un maximum d'information sur leurs consommateurs, et d'un autre côté, les sujets visent un contrôle maximal sur la divulgation de leurs informations d'identité. Ce problème de divergence des besoins n'a pas été réglé, et sa résolution constitue un défi de recherche actuel.

Mentionnons finalement qu'une partie prenante peut avoir plusieurs rôles. Le rôle de SP et celui d'IdP peut être assumé par la même organisation. Il est aussi possible qu'un sujet soit son propre IdP pour certains de ses attributs ou certaines de ses identités. Par exemple, un SP pourrait faire confiance au sujet lui-même pour lui fournir la véritable valeur d'attribut de faible importance tel que *couleur préférée*, mais pas pour un attribut plus important comme *dossier de conduite*.

2.3 Les modèles de contrôle d'accès

Les bases du contrôle d'accès furent posées formellement pour la première fois par Lampson (1969). Dans ses travaux, l'auteur introduit les concepts de *sujet*, l'entité qui désire effectuer une action, et d'*objet*, la ressource sur laquelle est effectuée une action. Il propose de lier ces concepts en utilisant une *matrice d'accès*. Dans ce paradigme, étant donné que les permissions sont directement liées à l'identité du sujet, on parle de contrôle d'accès basé du l'identité *(Identity Based Access Control – IBAC)*.

Par la suite, dans le début des années 1970, la défense américaine finança des travaux visant à mettre en place un modèle mathématique de contrôle d'accès (Schell *et al.*, 1973; LaPadula et Bell, 1973). Le modèle développé à cette époque impliquait des entités actives (des sujets) et des entités passives (des objets). Dans ce modèle, on propose que l'accès aux objets par les sujets soit protégé par des politiques de sécurité contenant deux éléments : le type d'accès demandé (lecture ou écriture) et le niveau de sécurité nécessaire pour l'effectuer. De leur côté, les sujets ont aussi un niveau de sécurité. L'accès est permis seulement si le niveau de sécurité du sujet est égal ou supérieur à celui de l'objet demandé. Ce modèle étant basé sur un ensemble ordonné (ou treillis) de niveaux de sécurité, on le nomme *contrôle d'accès basé sur les treillis (Lattice Based Access Control – LBAC)*. Il est approprié dans un contexte rigide comme dans le monde militaire par exemple, mais moins adéquat dans des systèmes où une plus grande flexibilité est nécessaire.

L'étape suivante dans l'évolution des modèles de contrôle d'accès est le modèle basé sur les rôles *(Role Based Access Control – RBAC)* (Ferraiolo et Kuhn, 1992; Bishop, 2004). Ce modèle

ajoute un niveau d'indirection supplémentaire. Chaque sujet se voit assigner un ou des rôles pouvant être reliés à des fonctions dans une entreprise par exemple. Ensuite, des permissions sur les objets sont liées à ces rôles. Les permissions ne sont donc jamais directement reliées à un sujet. Ceci permet une plus grande flexibilité que ce qui est possible avec LBAC, car un sujet peut se voir assigner plusieurs rôles en fonction des permissions qu'il doit détenir. De plus, un système dynamique ou d'une grande envergure utilisant RBAC est plus facile à gérer qu'un système basé sur l'identité, car il n'est pas nécessaire de lier explicitement chaque objet avec chaque sujet devant y avoir accès.

Finalement, vint l'arrivée des environnements orientés services où des sujets peuvent accéder à des objets provenant de plusieurs fournisseurs. Dans un tel scénario, le couplage entre un sujet et un fournisseur est très faible et des limitations avec les modèles de contrôle d'accès IBAC, LBAC et RBAC surgissent. Par exemple, avec ces modèles, il n'est pas possible de facilement autoriser un sujet en fonction de plusieurs attributs tels que son domaine d'appartenance et le type de forfait pour lequel il pourrait avoir payé. Afin de résoudre ce genre de problème, Yuan et Tong (2005) propose un modèle basé sur les attributs *(ABAC)* qui généralise les fonctionnalités des autres modèles. Le modèle ABAC permet la définition de droits d'accès basés sur n'importe quel attribut associé aux entités suivantes :

Sujet L'entité qui désire effectuer une action sur une ressource protégée. Un sujet peut avoir différents attributs tels qu'un *identifiant*, un *nom*, un *titre* ou une *adresse*.

Ressource ou objet L'objet d'une ressource peut avoir différents types d'attributs tels qu'un *nom de service Web*, un *auteur de fichier texte* ou une *dernière date d'accès*.

Environnement Cet élément est spécifique au modèle ABAC et n'est pas pris en compte par les autres modèles. Les attributs d'environnement peuvent inclure la *date et l'heure courante* ou bien *l'activité sur le réseau* par exemple. Le fait que ce type de paramètre soit pris en compte par le modèle ABAC et non par les autres modèles contribue à le rendre plus flexible que ses alternatives.

Le modèle ABAC apporte aussi un grand niveau de flexibilité, car il est assez général pour pouvoir être utilisé afin de mettre en place les autres modèles mentionnés précédemment. En effet, les éléments pris en compte par ces autres modèles d'accès, soit l'identité, le niveau de sécurité ou le rôle d'un sujet, sont en fait des attributs caractérisant un sujet. Ils peuvent par conséquent être considérés par le modèle ABAC lors des décisions de contrôle d'accès.

2.4 Standards

2.4.1 Security Assertion Markup Language

La recommandation SAML (OASIS, 2005) de l'organisme OASIS est un cadre applicatif basé sur XML qui permet de communiquer de l'information sur des politiques de sécurité ou des assertions sur des sujets entre des systèmes qui peuvent se trouver dans des domaines différents. La possibilité de réaliser des communications et de mettre en place des collaborations interdomaine est en fait un facteur clé derrière l'adoption de SAML.

Le standard SAML définit trois types d'assertions. Celles-ci sont émises par une autorité SAML (aussi appelée *partie affirmant (Asserting Party)*) et consultées par une *partie utilisatrice (Relying Party)*. Ces assertions concernent généralement un *sujet* et sont utilisées lorsqu'il désire accéder une ressource chez un *fournisseur de services* (voir Section 2.2.1). Lorsqu'un fournisseur de services effectue un contrôle d'accès en se basant sur des assertions, il devient une partie utilisatrice. Notons finalement que concrètement, l'entité qui agit en tant que partie affirmant est un fournisseur d'identité (voir Section 2.2.1) (OASIS, 2005).

Les trois types d'assertions pouvant être émises par une autorité SAML sont :

Authentification Indique qu'un sujet a été authentifié à un certain temps et en utilisant un mécanisme donné (nom d'utilisateur et mot de passe, certificat X.508, etc.).

Attribut Fourni des attributs associés à un sujet.

Décision d'autorisation Permet de transmettre une décision d'autorisation concernant une action effectuée sur une ressource, par un sujet. Notons que les travaux sur ce type d'as-

sertion sont complètement arrêtés depuis la version 2.0 de SAML et qu'il est maintenant recommandé de privilégier le standard XACML.

La recommandation SAML est utilisée dans plusieurs contextes d'authentification et permet de fournir des fonctionnalités d'authentification unique (*Single Sign-On*). Dans un tel scénario, un sujet s'authentifie chez son IdP et est subséquemment en mesure d'accéder à des ressources protégées chez différents SPs, sans avoir à explicitement s'authentifier chez ces derniers. Si les SPs et l'IdP sont dans des domaines différents, on parle alors d'authentification unique interdomaine, ou de gestion de l'identité fédérée tel que décrit à la Section 2.2.1. Au contraire, si les SPs et l'IdP sont dans le même domaine, on parle simplement d'authentification unique, mais le processus est le même que ce qui est décrit à la Section 2.2.1 (OASIS, 2005,?).

Notons que SAML peut être utilisé dans des contextes où l'identité réelle d'un sujet doit rester secrète pour des raisons de confidentialité, ou bien parce qu'elle n'est tout simplement pas pertinente dans un contexte donné. En effet, il est possible pour l'autorité SAML de ne fournir à la partie utilisatrice que les attributs nécessaires pour que cette dernière puisse prendre des décisions d'autorisation sur un sujet. Si parmi ces attributs on ne trouve pas de données identifiant uniquement un sujet, celui-ci peut conserver son anonymat et son droit à la vie privée, tout en étant en mesure de prouver qu'il s'est authentifié chez son fournisseur d'identité et qu'il possède certains attributs.

2.4.1.1 Le profil SAML ECP

Un client ECP *(Enhanced Client of Proxy)* (OASIS, 2005, 2011c), ou client intelligent, est un système qui sait comment contacter l'IdP approprié en fonction d'un contexte donné. Dans la Figure 2.2, étant donné que le sujet contacte manuellement l'IdP, il doit nécessairement utiliser un client ECP. L'alternative à un client intelligent est un fureteur Web. Dans ce cas, le client contacte initialement le SP et se fait rediriger vers l'IdP grâce à des mécanismes de redirection HTTP. Ceci est démontré par la Figure 2.3 :

1. Le sujet fait une requête d'accès pour une ressource protégée auprès du SP.

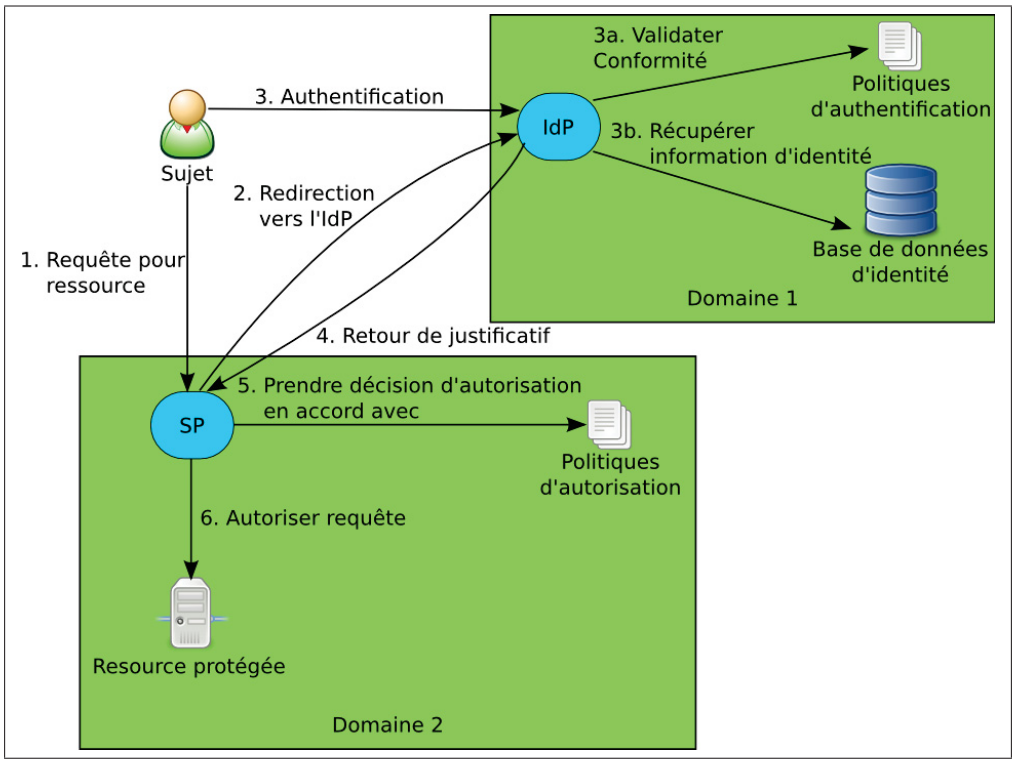

Figure 2.3 Authentification et autorisation dans un environnement fédéré lorsque le client est un fureteur Web, donc lorsque le profil ECP n'est pas nécessaire

2. Le SP détermine l'IdP avec qui le sujet fait affaire. Pour ce faire, plusieurs techniques peuvent être utilisées. Par exemple, le SP peut présenter une page avec la liste des IdPs qu'il supporte, permettant ainsi au sujet de sélectionner le sien. Le sujet est ensuite redirigé vers son IdP.

3. Le sujet s'authentifie avec son IdP en utilisant l'une des méthodes supportées par les politiques de ce dernier (nom d'utilisateur et mot de passe, certificat X.509, biométrie, etc.). Si l'authentification réussit, l'IdP récupère l'information d'identité relative au sujet qui est présente dans sa base de données.

4. L'IdP retourne une preuve de l'authentification (c.-à-d., un justificatif) au SP, en l'acheminant grâce au fureteur du sujet.

5. Le SP analyse le justificatif ainsi que toute autre information pertinente qu'il a à sa disposition. Il prend une décision d'autorisation en fonction de ces données et de ses politiques d'autorisation.

6. Si la requête est acceptée, le sujet est en mesure d'accéder à la ressource.

Alors que les fureteurs Web accèdent à des pages Web, les clients intelligents accèdent à des services Web et n'ont pas à supporter les mécanismes de redirections HTTP. Ils ne peuvent donc pas se faire rediriger par le SP vers un IdP et doivent nécessairement être en mesure de le contacter eux-mêmes s'ils veulent accéder à des services Web demandant une authentification par SAML. Le profil SAML ECP vient aborder ce cas et permet de sécuriser des services Web grâce à SAML.

Pour la liste complète des fonctionnalités devant être supportées par un client ECP et la syntaxe exacte des messages qu'il envoie et reçoit, voir la spécification (OASIS, 2005, 2011c).

2.4.2 WS-Federation

Le standard WS-Federation (Goodner *et al.*, 2007) est une spécification qui aborde le même problème que SAML (voir Section 2.4.1), mais le résout de manière différente. WS-Federation étend plusieurs standards de la série WS-* (OASIS, 2011b) afin de permettre à des systèmes basés sur des services Web de bénéficier de mécanismes de gestion de l'identité fédérée (voir Section 2.2.1). Les principaux standards sur lesquelles cette recommandation s'appuie sont WS-Security (OASIS, 2006), WS-Trust (OASIS, 2007) et WS-SecurityPolicy (OASIS, 2009). D'autres standards sont aussi nécessaires pour certaines fonctionnalités plus spécifiques. Par exemple, WS-Eventing est nécessaire afin de permettre une déconnexion globale (pour terminer une session établie avec un mécanisme d'authentification unique). Notons aussi que WS-Federation définit des règles d'encodage permettant d'utiliser la majorité de ses fonctionnalités avec des mécanismes HTTP standards. Bref, n'importe quelle application Web accessible à partir d'un fureteur Web est en mesure de bénéficier des avantages apportés par WS-Federation. SAML et WS-Federation peuvent donc tous deux être utilisés à la fois dans un contexte de ser-

vices Web avec un client intelligent que dans un contexte d'application Web avec un fureteur Web.

WS-Federation hérite des fonctionnalités des différents autres standards sur lesquels il se base. Par exemple, WS-Security fournit différents mécanismes permettant d'assurer la confidentialité, l'intégrité et authenticité de messages. De son côté, WS-SecurityPolicy permet de décrire les requis de sécurité nécessaires pour accéder à un service (p. ex. algorithmes de chiffrement à utiliser ou types de jetons acceptés). Enfin, WS-Trust apporte les bases de la gestion de l'identité fédérée. En effet, ce standard définit un modèle de service appelé STS *(Security Token Service)* et un protocole permettant d'émettre, de transporter et de traiter des jetons de sécurité entre des entités. Le modèle STS permet en fait de mettre en place un système de gestion de l'identité fédérée à l'image de ce qui est expliqué dans la Section 2.4.1. Ce modèle change toutefois quelque peut la nomenclature des différentes entités du processus. Un sujet est maintenant appelé un *demandeur* et le rôle de l'IdP est désigné par STS et non par partie affirmant.

WS-Federation vient ensuite augmenter ces fonctionnalités afin de rendre possibles les communications interdomaines entre les entités du modèle STS. WS-Federation permet premièrement aux participants d'une fédération d'identifier les différents services qu'elle offre et les politiques restreignant leur accès. Ceci est réalisé en définissant un format de document et un modèle pour les métadonnées d'une fédération. Ces métadonnées donnent des détails sur les services d'une fédération et étendent les autres formats de métadonnées utilisés dans un contexte de services Web tels que WSDL (W3C, 2001) et WS-Policy (W3C, 2006a).

Ensuite, WS-Federation définit diverses extensions à WS-Trust. Mentionnons la possibilité qu'un STS émette des jetons contenant des décisions d'autorisation en plus des jetons contenant des attributs sur un demandeur, ou bien la faculté qu'à un fournisseur de services de réclamer à ce qu'un demandeur lui fournisse des attributs additionnels, après que sa requête ait été approuvée. L'exécution des opérations par le fournisseur de services est donc suspendue jusqu'à ce que le fournisseur reçoive les informations demandées. WS-Federation définit

aussi un service de pseudonyme au sein du STS. Ce service peut être utilisé pour assigner un pseudonyme différent à un sujet en fonction du fournisseur de services avec qui il fait affaire. Mentionnons finalement que WS-Federation permet à un demandeur d'exprimer ses requis en termes de protection de la vie privée tels que l'identification d'attributs qu'un IdP devrait toujours chiffrer lorsqu'il les inclut dans une assertion par exemple.

2.4.3 XACML *eXtensible Access Control Markup Language*

Le standard XACML (OASIS, 2005b; Sun Microsystems Inc, 2003) de l'organisme OASIS a été conçu pour régler le problème des gros systèmes composés de plusieurs points d'accès hétérogènes. Dans de tels contextes, les diverses configurations des points où les politiques sont appliquées sont généralement gérées de manière indépendante et par des entités distinctes. Ceci fait en sorte qu'il est difficile d'avoir une vue globale des politiques et de mettre en place de bonnes pratiques dans l'ensemble d'un système.

XAML décrit à la fois un langage de politiques et un langage permettant d'effectuer des requêtes et de communiquer des réponses de contrôle d'accès de manière uniforme au travers des différents composants d'un système. Le langage de politiques permet de décrire des exigences générales en termes de contrôle d'accès et peut aussi facilement être étendu afin de définir de nouveaux éléments tels que des types de données et des algorithmes de combinaison. Un algorithme de combinaison est appelé lorsqu'une requête peut être évaluée par plusieurs règles d'une politique ou par plusieurs politiques d'un ensemble de politiques. Un tel algorithme permet de combiner les décisions d'autorisation retournées par les multiples règles ou les multiples politiques qui ont été évaluées. Par exemple, il est possible de déterminer le comportement à adopter si une politique est composée de deux règles et que pour une requête donnée, l'évaluation de la première règle retourne *Permit* et la seconde *Deny*. En ce qui concerne le langage de requêtes et de réponse défini par le standard XACML, il permet de formuler une demande pour une action spécifique et de répondre avec une décision de contrôle d'accès. Une telle décision a l'une de ces trois valeurs : *Permit*, *Deny*, *Indeterminate* (impossible de prendre une

décision, car une erreur est survenue ou bien des données nécessaires n'ont pas été fournies) et *Not Applicable* (le service auquel la requête a été effectuée n'est pas en mesure d'y répondre).

2.4.3.1 Le langage de politiques

Le modèle des politiques XACML est composé des principaux éléments illustrés hiérarchiquement dans la Figure 2.4.

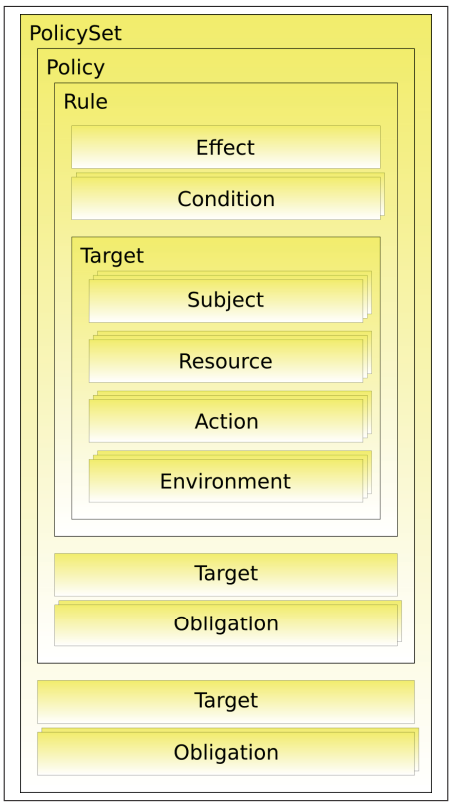

Figure 2.4 Le modèle hiérarchique des politiques XACML

PolicySet Peut être l'élément racine d'une politique. Consiste en un conteneur pouvant renfermer des *Policy* ou des *PolicySet*. Si l'élément *Target* d'un *PolicySet* correspond au contexte d'une requête donnée, ce *PolicySet* peut être utilisé afin de prendre une décision d'autorisation pour cette requête. En ce qui concerne l'élément *Obligation*, il contient des exigences devant impérativement être remplies par le PEP (voir Section 2.4.3.3). Si le PEP n'est pas en mesure de remplir ces obligations, il doit agir comme si la réponse de contrôle d'accès résultant de l'évaluation de cette politique était *Deny*.

Policy Doit être l'élément racine d'une politique si l'élément *PolicySet* n'est pas utilisé. Les éléments *Target* et *Obligation* ont la même signification que ceux du *PolicySet*.

Rule Une règle évaluée par une politique sur une *Target* donnée. Peut avoir un *Effect* de *Permit* ou *Deny*. L'élément *Condition* indique un prédicat devant être satisfait afin que l'*Effect* ait une valeur.

Target Identifie l'élément ou l'ensemble des éléments devant être évalués par la politique. Ceux-ci peuvent être des *Subjects*, des *Resources*, des *Actions* et des caractéristiques d'*Environment*.

2.4.3.2 La syntaxe des requêtes et des réponses

La norme XACML définit aussi le formats des messages à utiliser pour communiquer des requêtes et des réponses de contrôle d'accès. Ceux-ci sont appelés *Request Context* et *Response Context* et leurs principaux éléments sont représentés à la Figue 2.5.

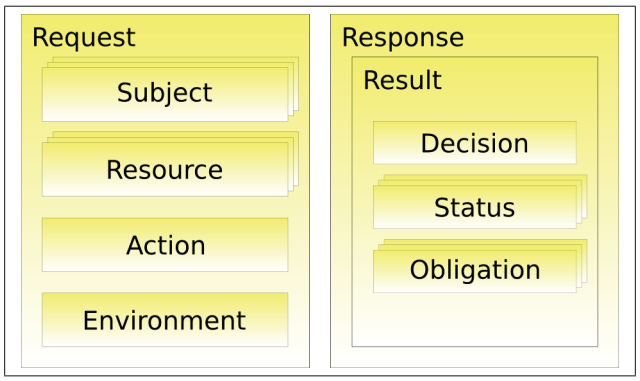

Figure 2.5 Le modèle hiérarchique des politiques XACML

Subject Représente le sujet effectuant la requête ainsi que ses attributs. Il est possible qu'une requête contienne plusieurs *Subject* afin de pouvoir représenter, par exemple, l'être humain effectuant une requête et l'application logicielle qu'il utilise.

Resource La ressource ou les ressources étant la cible de la requête.

Action L'action à effectuer sur la ressource.

Environment Les attributs décrivant l'environnement dans lequel la requête est effectuée.

Result Représente une décision d'autorisation. Il peut y avoir plusieurs *Results* dans une seule *Response* si celle-ci répond à une requête concernant plusieurs ressources par exemple.

Decision La réponse de contrôle d'accès pouvant prendre une valeur de *Permit*, *Deny*, *Indeterminate* ou *Not Applicable*.

Status Contient de l'information sur les erreurs qui ont pu se produire lors de l'évaluation des politiques.

Obligation Exigences devant être remplies par le PEP.

2.4.3.3 Composantes d'une architecture XACML

Le standard XACML décrit le rôle des quatre composants suivants lors de l'évaluation de politiques. Notons que ces principes sont repris du RFC 2904 (Vollbrecht *et al.*, 2000) de l'IETF :

Client de serveur de règles *(Policy Enforcement Point – PEP)* Système qui effectue le contrôle d'accès en émettant des requêtes d'accès et en appliquant des décisions d'autorisation. Lorsqu'un sujet effectue une requête pour commettre une action sur une ressource, il communique en fait avec le PEP la protégeant, et non directement avec la ressource. Le PEP effectue alors une requête auprès du PDP et met en place la décision d'autorisation que le PDP lui retourne.

Serveur de règles *(Policy Decision Point – PDP)* Système qui évalue des politiques de sécurité afin de prendre une décision de contrôle d'accès. Le PDP a ainsi besoin des différentes informations qu'il reçoit du PEP, c'est-à-dire, l'action à commettre ainsi que les attributs concernant le sujet, la ressource cible et l'environnement.

Point d'information de politique *(Policy Information Point – PIP)* Système qui agit en tant que source de valeurs d'attributs. Étant donné que le PEP communique directement avec

le sujet, il peut facilement obtenir ses attributs pour ensuite les faire connaitre au PDP. Pour obtenir toute autre information nécessaire au PDP (attributs sur la ressource et sur l'environnement), le PEP consulte le PIP.

Point d'administration de politique *(Policy Administration Point – PAP)* Système qui crée les politiques, les gère et les stocke dans un référentiel.

2.5 Approches

La section qui suit explique le fonctionnement de trois systèmes de gestion de l'identité fédérée. Premièrement, GSI est un système de sécurité utilisé dans les grilles informatiques. Celles-ci constituent en fait le premier type d'environnements fédérés. Ensuite, le Métasystème d'identité est une solution mettant l'usager au premier plan et utilisant la métaphore des *Information Cards* pour représenter les différentes identités relatives à un sujet. Finalement, Shibboleth est un système qui a ses racines dans le partage de ressources Web entre différentes institutions académiques.

2.5.1 GSI (*Grid Security Infrastructure*)

Une des premières solutions de gestion de l'identité dans un environnement fédéré est GSI, une composante du système de gestion de grilles informatiques Globus (Foster *et al.*, 1998; University of Chicago, 2010). Le contexte typique d'utilisation de cette solution est une grille informatique utilisée pour collaborer sur un projet scientifique. Par exemple, un scientifique pourrait vouloir exécuter des simulations sur des données quelconques. Or, le scientifique pourrait se trouver dans un domaine, les données qu'il désire analyser, dans un second domaine, et les ressources effectuant les calculs et simulations, dans un troisième. Dans un tel contexte, GSI a été développé avec trois principaux objectifs :

- Permettre des communications authentifiées et au besoin, confidentielles (en chiffrant le contenu des communications et non en masquant l'identité des interlocuteurs), entre les éléments d'une grille.

- Afin de ne pas avoir besoin de recourir à un système central de sécurité, la sécurité doit pouvoir se faire entre les frontières des domaines organisationnels.
- Supporter des mécanismes d'authentification unique incluant la délégation des justificatifs lorsque des opérations impliquent plusieurs ressources et sites.

Afin de réaliser ses objectifs, GSI authentifie les sujets grâce à une infrastructure à clés publiques (ArticSoft Technologies Limited, 2011), à des certificats X.509 (ITU-T, 2009b) et à des mécanismes d'autorisation principalement basés sur des listes de contrôle d'accès. Avec une telle cette approche, chaque utilisateur dispose d'un certificat X.509 ayant été émis par une autorité de certification. Ceci lui permet d'avoir un identifiant unique reconnu dans l'ensemble du système distribué. Les mécanismes d'autorisation étant basés sur des listes de contrôle d'accès donnant différentes permissions à différents sujets en fonction de leur identité, on parle alors de IBAC (voir Section 2.3).

Cette approche comporte plusieurs lacunes. Premièrement, étant donné que l'authentification des sujets repose sur une infrastructure à clés publiques, les sujets doivent gérer leur paire de clés et s'assurer que leur clé privée soit conservée de manière sécuritaire. Cette contrainte n'est pas réaliste si les sujets ne sont pas familiers avec les principes de la sécurité informatique. Un autre problème vient du modèle de contrôle d'accès utilisé. Comme celui-ci se base sur le fait qu'un sujet est en mesure de prouver qu'il est bel et bien le sujet qu'il prétend être, toute forme d'anonymat est impossible. Puisque les fournisseurs de services doivent connaitre l'identité des sujets effectuant des requêtes, il est possible de bâtir un historique complet des actions d'un sujet dans un système en combinant l'information détenue par les fournisseurs de services (Chadwick, 2008; Muppavarapu, 2009; Sinnott *et al.*, 2008; Cameron, 2005).

Pour pallier ces problèmes, plusieurs ajouts et projets, étant à différents stades de maturité, ont été ajoutés à GSI. Mentionnons par exemple le projet GridShib (Board of Trustees of the University of Illinois, 2009) qui vise à intégrer Shibboleth (voir Section 2.5.3) à Globus afin d'y inclure des fonctionnalités de gestion de l'identité fédérée.

2.5.2 Le Métasystème d'identité

Avant de parler du Métasystème d'identité (Cameron et Jones, 2006; Bhargavan *et al.*, 2008), il est important de discuter de son précurseur, le Passeport .NET de Microsoft (Chadwick, 2008). Ce système permettait à des utilisateurs d'accéder à plusieurs fournisseurs de services avec les mêmes informations d'authentification. Il s'agissait en fait d'un système de gestion de l'identité fédérée composé d'un seul IdP géré par Microsoft. Les fournisseurs déléguaient donc tout le processus d'authentification à Microsoft et utilisaient les informations fournies par le Passeport .NET afin de mettre en place leur propre politique d'autorisation. Lorsqu'un utilisateur s'enregistrait à un fournisseur de services, il voyait toutes les informations concernant son profil enregistrées de manière centralisée dans les serveurs de Microsoft. Les organisations utilisant ce service devaient donc avoir une confiance absolue envers Microsoft pour conserver les données utilisateurs de manière sécuritaire et pour effecteur les opérations d'authentification de manière appropriée.

L'adoption de cette technologie a été très limitée. En fait, peu de services hors de ceux fournis par Microsoft l'ont utilisé. Plusieurs raisons sont derrières cet échec. La plus évidente est son fonctionnement centralisé et la notion d'entière confiance qu'il implique. En effet, toute transaction devait nécessairement inclure Microsoft. Lorsque deux entreprises ont des relations d'affaires importantes ou confidentielles, elles ne veulent pas nécessairement impliquer une autre entité dans la transaction, et lorsqu'elles sont prêtes à le faire, elles veulent généralement pouvoir décider quelles informations elles acceptent de divulguer à un tiers et quelles opérations elles désirent impartir. Un autre point négatif affligeant cette solution est le fait que les sujets n'avaient que peu de contrôle sur les éléments d'identité qu'ils désiraient partager avec un fournisseur de services. Ils devaient en effet faire un choix parmi trois options de configuration reliées à cet aspect. Un sujet pouvait choisir de divulguer :

- son adresse de courriel
- son nom et prénom

- Tous les autres éléments de son profil, incluant sa date de fête, son pays ou région de résidence, son nom, son prénom, son emploi, son code postal, sa langue préférée, son état et son fuseau horaire.

Après l'échec du Passeport .NET, l'architecte en chef de la division identité chez Microsoft, Kim Cameron, s'est penché sur le problème et a essayé de concevoir un système de gestion de l'identité plus distribué et ne commettant pas les erreurs ayant amené à l'insuccès du Passeport .NET. Ses réflexions ont eu deux résultats : les sept lois de l'identité (Cameron, 2005) et le Métasystème d'identité (Cameron et Jones, 2006; Bhargavan *et al.*, 2008). Les lois de l'identité ont dirigé les choix de conception du Métasystème d'identité.

2.5.2.1 Les lois de l'identité

Ces lois ont été présentées par Kim Cameron de Microsoft sur son blog (Cameron). Elles sont régulièrement cités dans de nombreux articles de recherche.

2.5.2.1.1 Contrôle et consentement de l'utilisateur

Les systèmes techniques de gestion de l'identité ne doivent révéler de l'information identifiant un utilisateur que lorsqu'ils ont l'approbation de ce dernier.

Afin d'être adopté, un système de gestion de l'identité doit détenir la confiance de ses utilisateurs. Une façon d'obtenir cette confiance est en donnant le contrôle à l'usager. Il doit avoir la certitude que les informations qu'il fournit à un système sont acheminées là où elles devraient l'être. Il doit aussi être informé des conditions d'utilisation de ces données et être en mesure d'arrêter le processus s'il le désire.

Tel qu'indiqué à la Section 2.5.2, cette règle n'était pas respectée par le Passeport .NET, car celui-ci ne donnait presque aucun contrôle à l'utilisateur concernant la divulgation de ses informations personnelles.

2.5.2.1.2 Divulgation minimale pour une utilisation contrainte

La solution qui divulgue le moins d'information pouvant identifier un individu et qui limite le plus son utilisation est la solution la plus stable à long terme.

Cette loi vient du fait qu'il est très fréquent que des systèmes voient leurs bases de données perdues ou volées. Si celles-ci contiennent peu d'informations personnelles, l'impact de ce genre d'évènements est minimisé ce qui rend le système plus robuste. De plus, un système conservant une plus petite quantité d'informations personnelles est une cible moins intéressante pour un attaquant désirant commettre un vol d'identité.

Cette loi demande non seulement qu'une plus petite quantité d'attributs soit transférée et conservée, mais aussi que les attributs à communiquer soient choisis judicieusement. Ceci a pour but de réduire la probabilité qu'un sujet soit identifié à travers plusieurs contextes. Par exemple, si un fournisseur de services veut connaitre l'âge d'un sujet, il est moins approprié de récupérer sa date de naissance que de directement récupérer son âge. En effet, une date de naissance peut plus facilement identifier uniquement un sujet qu'un âge, et ce, indépendamment du contexte.

2.5.2.1.3 Parties justifiables

Les systèmes numériques de gestion de l'identité doivent être conçus de sorte que les informations pouvant identifier un individu ne soient divulguées qu'à des parties ayant une fonction nécessaire et justifiable dans une relation d'identité.

Cet énoncé est similaire à la première loi. Il requiert qu'un système d'identité informe le sujet des entités avec lesquelles il communique et interagit. De plus, la présence de ces entités doit être justifiable. Ce principe n'était pas respecté par le Passeport .NET, car dans cette architecture, toutes les informations étaient traitées par Microsoft. Or, les utilisateurs n'apprécient pas que leurs informations soient données à un tiers n'ayant aucun rôle à jouer dans une transaction numérique. La présence de Microsoft lors d'une transaction visant à accéder un service fourni

par Microsoft est justifiable, mais au contraire, elle ne l'est pas si la transaction touche une autre compagnie.

2.5.2.1.4 Identité dirigée

Un système universel de gestion de l'identité doit permettre aussi bien le concept d'identifiant omnidirectionnel *pour les entités publiques que le concept d'identifiant* unidirectionnel *pour les entités privées. Ceci facilite la découverte d'entités publiques, mais empêchant la corrélation d'identifiants.*

L'hypothèse derrière cette loi est que les utilisateurs préfèrent généralement conserver leurs identifiants privés afin de garder un certain degré d'anonymat. D'un autre côté, les entités publiques telles que les organisations commerciales désirent être facilement reconnaissables et joignables. Ainsi, lorsqu'un utilisateur fait affaire avec un fournisseur de services, on devrait lui assigner un identifiant unique et seulement valable chez ce fournisseur. Ceci rend la corrélation et la collusion entre les fournisseurs beaucoup plus difficiles. Des fournisseurs pourraient vouloir comploter afin de mettre en commun l'information qu'ils détiennent sur leurs sujets pour bâtir des profils personnels globaux. Ce genre de pratique est beaucoup plus difficile s'il n'y a pas de moyens simples pour déduire qu'un sujet ayant fait affaire avec un fournisseur de services donné et un second sujet ayant fait affaire avec un autre fournisseur réfère en fait au même individu.

2.5.2.1.5 Pluralisme des opérateurs et des technologies

Dans un système universel de gestion de l'identité, de multiples fournisseurs d'identité doivent être en mesure d'utiliser différentes technologies fonctionnant entre elles.

En fonction du contexte, les fonctionnalités nécessaires à un système de gestion de l'identité et les informations qu'il doit acheminer aux différentes entités peuvent être différentes, voire contradictoires. L'approche universelle qui devrait être employée consiste donc en un métasystème pouvant encapsuler d'autres systèmes. À l'image des standards tels que RSS ou HTML,

il est important de s'entendre sur un protocole de communication pouvant être utilisé pour transporter de l'information de différente nature.

Le Passeport .NET ne respectait pas cette règle, car la seule implémentation du système ne supportait qu'un seul fournisseur d'identité et n'était pas adaptable à différents contextes.

2.5.2.1.6 Intégration humaine

Un métasystème universel de gestion de l'identité doit considérer l'utilisateur humain comme composant même du système, doit interagir avec lui avec des mécanismes ne comportant aucune ambigüité et offrant des dispositifs empêchant les attaques sur l'identité.

Cette hypothèse conçoit que le maillon faible des chaines de communications est généralement l'utilisateur et non les technologies. Par exemple, la sécurisation du canal entre un fureteur Web et un serveur Web est un problème résolu, mais pas celui de la sécurisation du canal entre un fureteur Web et l'utilisateur. En effet, les attaques d'hameçonnage ont habituellement plus de succès que celles dirigées sur les protocoles de communication sécurisés tels que SSH, ou bien sur les algorithmes de chiffrement tels que AES. Un autre point impliqué par cette loi est que le processus d'authentification doit être simple, familier pour l'utilisateur et difficilement imitable par une entité malveillante. De cette manière, un usager peut facilement déterminer si une attaque est en cours.

2.5.2.1.7 Une expérience cohérente entre les contextes

Un métasystème de gestion de l'identité unificateur doit garantir à ses utilisateurs une expérience simple et cohérente et permettre une séparation de contexte par l'entremise de plusieurs opérateurs et technologies.

Cette loi est très similaire à l'énoncé précédent et fait un retour sur d'autres lois mentionnées plus haut. Elle rappelle la nécessité d'un processus d'authentification familier, peu importe la technologie utilisée ou le pseudonyme choisi. Du point de vue de l'usager, les identités doivent être représentées sous forme d'objets clairs, facilement reconnaissables, lui permettant de com-

prendre à quoi elles réfèrent et donc de lui permettre de faire un choix éclairé lorsque vient le temps de les divulguer à diverses parties prenantes. C'est cette loi qui a mené au concept des *Information Cards* (Cameron et Jones, 2006; Bhargavan *et al.*, 2008), une abstraction numérique des cartes d'identité que tout le monde conserve dans son portefeuille. Ceci est la notion centrale dans le Métasystème d'identité.

2.5.2.2 Fonctionnement du Métasystème d'identité

Le Métasystème d'identité défini le concept des *Information Cards* (Cameron et Jones, 2006; Bhargavan *et al.*, 2008) (représentation numérique d'une carte d'identité) émises pas un IdP et contenant des *claims* (revendications) indiquant la valeur des attributs d'un sujet. Comme tout système de gestion de l'identité fédéré (voir Section 2.2.1), la valeur des revendications dépend de l'IdP les émettant. Dans le cas du Métasystème d'identité, un sujet peut émettre lui-même des *Information Cards* contenant des revendications. Dans ce cas, on parle de cartes autodélivrées *(self-issued)*. Ce type de carte a potentiellement moins de valeur auprès d'un fournisseur de services qu'une carte gérée *(managed)* émises par un IdP externe envers qui le fournisseur de services fait confiance pour émettre des revendications véridiques. Un IdP externe peut en effet effectuer un processus additionnel de vérification avant d'émettre des revendications indiquant la valeur d'attributs critiques.

La Figure 2.6 fait une description générale des interactions et échanges de messages entre les différents composants du Métasystème d'identité. Voici les différents éléments du modèle :

User (U) Le sujet du système. Un utilisateur humain possédant des cartes d'identité et désirant accéder à une ressource d'un fournisseur de services.

Identity Selector (IS) Le sélecteur d'identité est un sous-système permettant à l'usager de gérer ses cartes et de consulter l'information qu'elles contiennent. Il s'agit de l'interface entre ce dernier et le fournisseur d'identité. De plus, dans le cas des cartes autodélivrées, ce module agit comme fournisseur d'identité, c'est-à-dire qu'il contient les données privées associées aux revendications qu'elles contiennent.

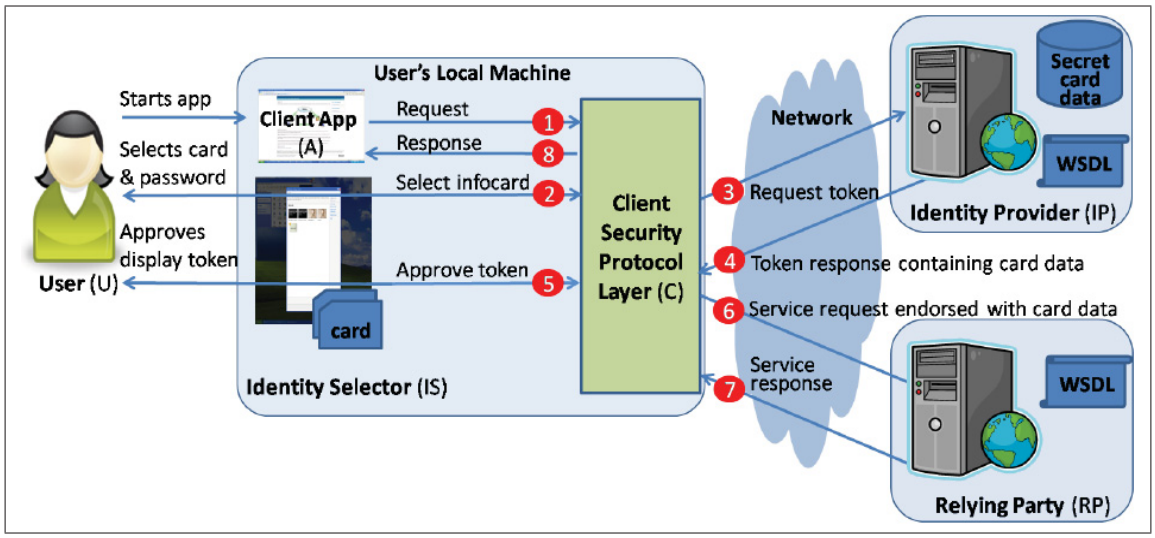

Figure 2.6 Séquence des opérations dans le Métasystème d'identité
Tiré de Bhargavan *et al.* (2008)

Client App (A) L'application client est l'intermédiaire entre l'utilisateur et le fournisseur de services. Il peut s'agir, par exemple, d'un applet exécuté à l'intérieur du fureteur Web ou bien d'un client à un service Web. Ce module n'a aucun accès aux jetons de sécurité contenant les revendications contenues dans les cartes d'identité.

Client Security Protocol Layer (C) Afin de garantir un certain de niveau de sécurité dans les transactions, l'application client et le sélecteur de carte communiquent avec le monde extérieur grâce à une couche gérant la sécurité. Celle-ci se charge entre autres d'effectuer le chiffrement. Notons toutefois qu'en fonction de la configuration du protocole utilisé, les données pourraient circuler dans le clair.

Relying Party (RP) Tel qu'il est indiqué dans les Sections 2.2.1, 2.4.1 et 2.4.2, un fournisseur de service ou partie utilisatrice *(Relying Party)* est l'entité qui détient des ressources et des services protégés auxquels un sujet désire accéder. Avant d'accorder un accès à ses actifs, un RP doit recevoir un jeton de sécurité représentant une *Information Card* contenant des informations d'identité. Cet acteur doit publier sa politique de sécurité sur sa page Web ou bien sur un serveur de métadonnées. Cette politique peut contenir différents éléments tels que le ou les fournisseurs d'identité duquel doivent provenir les

jetons de sécurité, les revendications concernant le sujet qui sont réclamées ou bien les détails quant à la rétention et l'utilisation des données personnelles du sujet.

Identity Provider (IP) (aussi noté IdP dans d'autres systèmes de gestion de l'identité fédérée)
Un fournisseur d'identité émet et gère les cartes d'identité d'un utilisateur. Cette entité est généralement implémentée sous la forme d'un service Web permettant de générer des jetons de sécurité. Comme dans tout système de gestion de l'identité fédérée, les IPs sont les éléments centraux dans le Métasystème d'identité. Tous les autres acteurs doivent en effet leur faire confiance. Premièrement, les utilisateurs doivent faire confiance à ce qu'un IP gère ses informations correctement. Ensuite, les RPs doivent avoir confiance en ce qu'un IP authentifie correctement ses sujets et génère des jetons de sécurité exacts. Les IPs doivent aussi publier leur politique de sécurité. Celle-ci peut spécifier, par exemple, qu'un utilisateur doit protéger sa requête de jeton de sécurité grâce à un mot de passe.

Voici ensuite la séquence d'opérations démontrée dans la Figure 2.6 et expliquée dans Bhargavan *et al.* (2008). Notons que préalablement à ses opérations, la couche de sécurité C doit obtenir les politiques de sécurité du fournisseur de services RP et du fournisseur d'identité IP.

1. L'application A envoie une requête pour RP. Cette requête est interceptée par la couche de sécurité C.

2. Comme la politique de sécurité de RP est connue par C, ce dernier connait les revendications que l'usager doit fournir. Il peut donc déclencher le sélecteur de carte en lui fournissant les informations appropriées (revendications demandées, identité de RP, etc.). L'usager doit ensuite choisir une carte appropriée et fournir le mot de passe qu'il partage avec IP.

3. C envoie une requête de jeton de sécurité à IP en y incluant :

   - la référence à la carte choisie par l'usager
   - l'identité de RP

- les revendications réclamées

Ce message est authentifié grâce au mot de passe fourni par l'utilisateur.

4. IP authentifie et autorise la requête. Si la requête est valide, il émet le jeton de sécurité demandé et le retourne à C. Une représentation, graphique du jeton est aussi retournée. Cette représentation est utilisée uniquement à des fins d'affichage à l'intention de l'utilisateur humain. Le jeton de sécurité contenant les revendications est quant à lui chiffré à l'intention de RP.

5. La représentation graphique du jeton est affichée à l'usager par l'entremise du sélecteur de carte. Une vérification peut donc être effectuée par U avant qu'il confirme qu'il veut bel et bien que le jeton soit envoyé à RP.

6. C envoie une requête de service à RP. Il s'agit en fait de la requête originelle effectuée par A au tout début de l'opération. Cette requête contient le jeton de sécurité émis par IP.

7. RP autorise la requête en fonction de sa politique de sécurité et des informations contenues dans le jeton de sécurité. Si la requête est acceptée, RP retourne sa réponse à C.

8. C achemine la réponse de RP vers l'application A.

Ce processus est très similaire avec ce qui est possible de réaliser avec le profil SAML ECP (voir Section 2.4.1.1), mais en positionnant l'utilisateur humain au centre du système. Il est en effet informé et sollicité à deux endroits dans le processus (étapes 2 et 5). À ces étapes, on l'informe des politiques de RP et des données qui lui seront divulguées. Il est donc en mesure d'arrêter le processus si ces modalités ne le satisfont pas.

2.5.2.3 Le futur du Métasystème d'identité

En février 2011, Microsoft a annoncé qu'il abandonnait le développement de CardSpace, son implémentation du Métasystème d'identité (Identity and Access Team, 2011). Même s'il existe plusieurs autres projets visant à implémenter le Métasystème d'identité dans son intégralité ou

en partie, le concept des *Information Cards* a eu droit à une adoption très mitigée. Plusieurs facteurs pouvant expliquer l'insuccès de ce système ont été révélés. Les experts s'entendent sur le fait que l'industrie ne perçoit pas le Métasystème d'identité comme une solution à un problème immédiat, mais que tôt ou tard, le problème que ce système aborde devra être résolu. Ensuite, le fait que ce ne sont pas tous les fureteurs et systèmes d'exploitation qui disposent d'un sélecteur de carte a ralenti l'adoption des *Information Cards* par les fournisseurs. Mentionnons aussi que le processus d'authentification avec des *Information Cards* est bien différent de celui qui est familier de tous et qui est effectué avec un nom d'usager et un mot de passe. Bien des fournisseurs de services n'ont pas voulu prendre le risque de déstabiliser leurs utilisateurs (Cameron, 2011b; Peterson, 2011; Jones, 2011; Kassaei, 2011; Cameron, 2011a).

Toutefois, même si Microsoft a abandonné CardSpace, plusieurs autres projets et organisations reliés au Metasystème d'identité travaillent toujours sur des implémentations et autres initiatives reliées à ce projet. Mentionnons la *Information Cards Foundation* (Information Card Foundation, 2011), le comité technique d'OASIS concernant l'interopérabilité dans le Métasystème d'identité (OASIS, 2011a), le projet Pamela (Dingle, 2009), Higgins (The Eclipse Foundation, 2011a), le projet Bandit (Bandit Team, 2009) et openinfocard (Openinfocard Team, 2011).

2.5.3 Shibboleth

Le projet Shibboleth (Internet2, 2011e) est une implémentation du standard SAML dans laquelle la plupart des développeurs principaux travaillent aussi sur cette spécification au sein de l'organisation OASIS. Pour cette raison, Shibboleth suit le standard SAML de très près et a fait ses preuves en tant que solution pour plusieurs organisations à travers le monde (Internet2, 2011c). Il s'agit d'un projet au code source ouvert où tout le monde peut prendre part, mais son principal contributeur est le consortium de recherche et d'éducation Internet2 (**?**).

En raison de ses racines provenant du monde académique, Shibboleth est fortement utilisé sur les sites Web de campus universitaires afin de réaliser du partage de ressources inter-campus et pour fournir des mécanismes d'authentification unique. Toutefois, comme il s'agit d'une

implémentation complète de SAML, il est possible de protéger toute sorte de ressources, y compris des services Web grâce à l'utilisation du profil ECP (voir Section 2.4.1.1).

Le projet Shibboleth est composé de deux modules principaux, un IdP et un SP, ainsi que d'un service additionnel de découverte d'IdP. Le rôle de l'IdP et du SP a déjà été décrit à la Section 2.2.1. En ce qui concerne le service de découverte d'IdP, il s'agit d'une application permettant à un sujet de choisir son IdP, tel qu'indiqué à l'étape 2 de la Figure 2.3. Étant donné que Shibboleth suit le standard SAML, un système utilisant cette solution pourrait voir ses IdPs et SPs remplacés par tout autre IdP ou SP suivant le standard, et ce, de manière complètement transparente.

2.5.3.1 Le fournisseur d'identité

L'IdP est une application Java pouvant être déployée dans un conteneur de *Servlet* (Oracle, 2011) tel que Tomcat (The Apache Software Foundation, 2011b) ou Jetty (Mort Bay Consulting, 2011). Un sujet peut s'authentifier à l'IdP en interagissant avec un *Login Handler*. Cinq sont fournis par défaut, mais il est possible de développer son propre *Login Handler* et l'intégrer à l'IdP. Les cinq qui sont fournis permettent de (i) déléguer le processus d'authentification au conteneur de *Servlet*, (ii) déléguer le processus d'authentification à un système externe, (iii) présenter une page Web demandant un nom d'utilisateur et un mot de passe, (iv) identifier le sujet en fonction de son adresse IP et (v) réutiliser une session ayant déjà été établie dans un contexte d'authentification unique.

Une fois qu'un sujet s'est authentifié à son IdP, les attributs sont acheminés de l'IdP vers un SP. Ce processus est réalisé en quatre étapes :

a. L'IdP récupère les attributs bruts d'une source de données telle qu'une base de données relationnelle ou un annuaire LDAP.

b. Les attributs sont transformés (combinés, divisés, reformatés, etc.) puis encodés dans le format SAML.

c. L'IdP évalue ses politiques de divulgation d'attributs. Celles-ci permettent de spécifier quels attributs peuvent être communiqués à quels SPs et dans quelles conditions. De telles politiques peuvent être élaborées par l'administrateur responsable de l'IdP, mais aussi par les sujets gérés par un IdP voulant exercer un plus grand contrôle sur leur vie privée.

d. Les attributs sont acheminés au SP dans un jeton SAML transporté par le sujet. Notons que l'identifiant unique utilisé pour faire référence au sujet dont les attributs sont contenus dans ce jeton SAML peut varier en fonction du SP pour qui le jeton est destiné et peut même varier d'une session à l'autre chez un même SP. L'utilisation d'un identifiant variable est un autre mécanisme permettant à un sujet de bénéficier d'un plus grand contrôle sur sa vie privée.

2.5.3.2 Le fournisseur de services

Le SP Shibboleth est composé de deux éléments. Premièrement, un processus ou daemon exécuté en arrière plan et ensuite, un module pour les serveurs Web Apache (The Apache Software Foundation, 2011a), IIS (Microsoft Corporation, 2011) ou NSAPI (Oracle Corporation, 2010). Le module doit être intégré à un serveur Web hébergeant des ressources à protéger. La principale raison pour laquelle la logique du SP est divisée en deux composants a trait à la simplicité d'implémentation. Le SP doit en effet conserver de l'information quant à l'état d'un serveur Web qui est généralement constitué de plusieurs processus. La façon la plus simple de gérer l'état de l'ensemble d'un tel serveur est de le faire dans un processus distinct.

Le SP doit être configuré pour intercepter toutes requêtes effectuées pour une ressource protégée. Ensuite, si le sujet a préalablement été récupéré un jeton SAML de son IdP en utilisant le profil ECP (voir Section 2.4.1.1), le jeton est directement fourni au SP qui peut l'analyser et prendre une décision de contrôle d'accès. Dans le cas contraire, le SP doit rediriger le sujet vers son IdP. Cette redirection peut toutefois poser problèmes. En effet, dans une fédération, il y a généralement plusieurs IdPs. De plus, un SP peut être membre de plusieurs fédérations. Le

SP doit donc être mis au courant de l'IdP du sujet effectuant une requête. Ceci peut être réalisé de plusieurs manières :

- Le SP peut afficher une page Web statique listant les IdPs supportés.
- Le sujet peut indiquer son IdP dans sa requête. L'identité de l'IdP peut notamment être incluse dans les paramètres de l'URL utilisée pour effectuer une requête vers une ressource protégée. Lorsque le SP analyse la requête, il est donc en mesure de déterminer où rediriger le sujet.
- Le sujet peut être redirigé vers un service de découverte d'IdP. Ce service présente un formulaire de sélection d'IdP similaire à ce qu'il est possible d'accomplir avec une page Web statique, mais pouvant entre autres être mis à jour dynamiquement. Un tel service peut aussi être directement intégré à l'interface d'un SP.

Une fois que le SP détient un jeton SAML émis par un IdP, il y extrait les attributs qu'il contient, les décode et les évalue en fonction de ses politiques de contrôle d'accès. Si la requête est acceptée, une session est créée pour le sujet et celui-ci est en mesure d'accéder à la ressource protégée.

2.5.4 Comparaison entre le Métasystème d'identité et Shibboleth

On peut voir que Shibboleth et le Métasystème d'identité sont deux approches abordant plus ou moins le même problème, c'est-à-dire, la création de fédérations (voir Section 2.2.1) et le partage de ressources interdomaines. Cependant, étant donné que le Métasystème d'identité se base sur les lois de l'identité (voir Section 2.5.2.1, il aborde directement des considérations qu'a un sujet humain interagissant avec un système. Le Métasystème d'identité essaie de simplifier les interactions entre un humain et les systèmes gérant les différentes représentations de ses identités. Ceci est réalisé en unifiant les interfaces usager et en concrétisant l'abstraction d'une *identité* dans le concept d'une *Information Card* (voir Section 2.5.2.2. Cet aspect n'est pas du tout abordé par Shibboleth, qui constitue en fait en une implémentation du standard

SAML (voir Section **??**). Ce standard se concentre sur la définition des entités composants une fédération et sur les messages qu'ils s'échangent, et non sur les intérêts d'un utilisateur humain.

Le fait que Shibboleth n'aborde pas de considérations spécifiques concernant les interactions avec un utilisateur humain fait en sorte qu'il peut être déployé de manière totalement transparente pour le sujet. Il peut, par exemple être utilisé pour protéger des ressources qui sont accédées depuis un fureteur Web, en utilisant des fonctionnalités présentes dans ce type de client. Mentionnons notamment la redirection et les témoins HTTP (Internet2, 2011a). Quant à lui, un déploiement du Métasystème d'identité demande toujours que le sujet utilise un sélecteur d'identité dédié (voir Section 2.5.2.2).

Au-delà de ces différences, le Métasystème d'identité et Shibboleth on aussi des caractéristiques communes. Ces deux projets reposent en effet sur le même principe de gestion de l'identité fédérée découplant les responsabilités d'authentification de celles d'autorisation (voir Section 2.2.1). Par conséquent, dans les deux systèmes, les fournisseurs d'identité et fournisseurs de services peuvent se concentrer sur leurs responsabilités respectives. De plus, les sujets disposent d'un plus grand degré d'anonymat, car il est possible de ne fournir qu'un ensemble restreint d'attributs aux fournisseurs de services (voir Section 2.4.1 et 2.5.2.1.2). Mentionnons en terminant que Shibboleth est aussi en mesure de suivre la loi de l'identité *Contrôle et consentement de l'utilisateur*, d'une manière similaire à ce qui est possible avec le sélecteur d'identité du Métasystème d'identité. Rappelons que ce dernier demande qu'une confirmation soit effectuée par le sujet avant que ses attributs ne soient divulgués (voir Section 2.5.2.2). Shibboleth comprend lui aussi le même genre de mécanisme (Internet2, 2011d).

2.5.5 Shintau

Le projet Shintau (Chadwick et Inman, 2010) vient régler un problème qu'a Shibboleth lorsqu'il est utilisé dans des VO dynamiques. Le modèle de Shibboleth repose sur l'hypothèse que l'institution d'où provient un sujet agit en tant qu'IdP, c'est-à-dire qu'elle l'authentifie et fournit aussi les attributs qui seront utilisés par le SP pour réaliser l'autorisation. Cette hypothèse est plausible dans le contexte pour lequel Shibboleth a été conçu à l'origine, celui du

partage de ressources entre institutions scolaires, mais elle est moins valable dans un contexte de VO dynamique. Par exemple, si un étudiant de l'université A désire accéder à une ressource académique d'une autre université, il est normal que l'université A maintienne les attributs à long terme de l'étudiant tels que son nom et son département, et que l'autre université effectue des décisions de contrôle d'accès en fonction de la valeur de ces attributs. Or, si l'on désire permettre à ce même étudiant de rejoindre et quitter fréquemment différentes VOs associées à des clubs étudiants ou autres activités par exemple, il devient compliqué de demander à ce que l'administration de l'université A maintienne tous les attributs relatifs aux différentes VOs dynamiques impliquées. Dans certains cas, il est même possible que l'administration de l'université n'ait pas l'autorité pour émettre des attributs spécifiques à certaines de ces VOs.

L'approche choisie par Shintau se base sur deux approches : la médiation par les IdPs (Alliance, 2005) et le relai de l'identité (Klingenstein, 2007). Dans la médiation par les IdPs, un sujet lie ensemble des identités qu'il possède chez des IdPs distincts. Afin de réaliser ceci, le sujet s'authentifie explicitement aux deux IdPs et leur fournit la même chaine pseudo aléatoire. Les IdPs s'échangent ensuite cette chaine, ce qui leur permet de confirmer qu'ils communiquent avec le même sujet et de lui assigner un alias utilisé lors de l'échange de messages. Lorsqu'un de ces IdPs retourne des attributs à un SP concernant un sujet, il inclut des détails sur le second IdP avec qui le sujet fait affaire, ainsi que l'alias généré aléatoirement pour identifier le sujet. Le SP peut donc faire une requête à ce second IdP afin d'avoir des attributs concernant le sujet identifié par l'alias. Ce modèle est affligé par un problème de protection de la vie privée. En effet, les IdPs qui sont reliés savent que les attributs d'un sujet sont distribués chez plusieurs IdPs et connaissent l'identité de ces IdPs. Ceci pourrait être indésirable dans certains contextes. En ce qui concerne la seconde approche sur laquelle se base Shintau, le relai de l'identité, il s'agit d'un modèle se basant sur le concept d'IdP mandataire. Celui-ci maintient un lien avec les différents IdPs avec qui le sujet fait affaire et retransmet les assertions émises par ceux-ci vers le SP désirant autoriser un sujet.

La solution proposée par Shintau introduit une nouvelle entité, un *Service de liaison*. Celle-ci est responsable de maintenir une table de correspondance entre un sujet et les différents

IdPs avec qui il fait affaire. La solution a été conçue en considérant des aspects concernant les résultats d'un sondage (Chadwick *et al.*, 2007) réalisé auprès d'experts dans le domaine du contrôle d'accès dans un contexte d'organisations virtuelles. L'un de ces aspects concerne l'importance de la protection de la vie privée des sujets. Une des fonctions de ce service de liaison est en fait de permettre au sujet d'exercer un plus grand contrôle sur la protection de sa vie privée. Par exemple, lorsqu'un sujet interagit avec le service de liaison, il est en mesure de spécifier des politiques indiquant quels attributs de quels IdPs peuvent être communiqués à quels SPs.

La Figure 2.7 illustre comment un sujet interagit avec le service de liaison lorsqu'il désire y inscrire des IdPs. Le processus décrit implique que le service de liaison est une application Web accédée par un fureteur Web et se base sur des mécanismes de redirection HTTP. Notons toutefois que le service de liaison est une application totalement distincte des actifs sécurisés fournis par des fournisseurs de services. Il est donc possible pour un sujet de s'inscrire auprès d'un service de liaison en utilisant un fureteur Web, et de subséquemment utiliser un client ECP (voir Section 2.4.1.1) afin de récupérer des ressources d'un fournisseur.

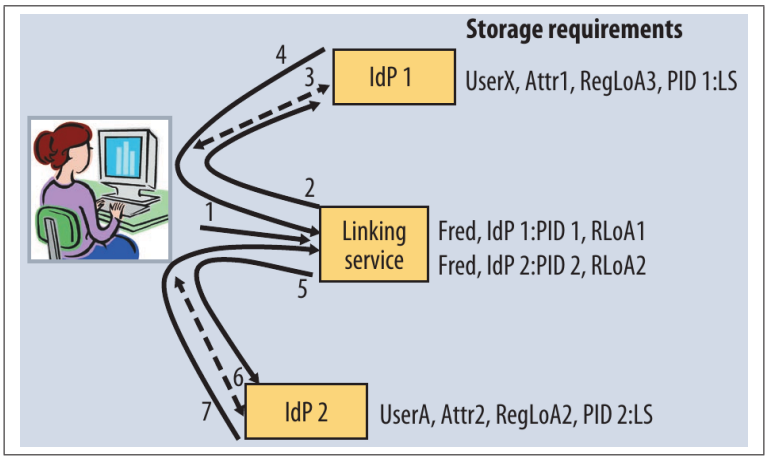

Figure 2.7 Enregistrement d'IdP auprès du service de liaison
Tiré de Chadwick et Inman (2009)

Tel qu'expliqué pat Chadwick et Inman (2009),

1. Le sujet contacte un service de liaison de son choix et choisit un de ses IdPs dans la liste présentée par le service de liaison. Les IdPs présentés sont ceux envers qui le service de liaison a établi une relation de confiance.

2. Le service de liaison redirige le sujet vers l'IdP sélectionné.

3. Le sujet s'authentifie auprès de son IdP en utilisant une des méthodes supportées par celui-ci. De par la nature de la requête qu'il reçoit, l'IdP sait qu'un sujet est en train de s'enregistrer auprès d'un service de liaison et est informé de l'identité de ce service. Si l'authentification réussie, l'IdP génère un identifiant pseudoaléatoire permanent appelé PID, à utiliser avec le service de liaison.

4. Le sujet est redirigé vers le service de liaison et détient une assertion d'authentification contenant le niveau d'assurance [1] *Level of Assurance – LOA* associé à l'authentification ainsi que le PID généré à l'étape précédente. Ces informations sont stockées dans la table de liaison du service.

5. Le sujet choisit un autre IdP dans la liste du service de liaison et recommence le processus.

Une fois qu'un sujet a enregistré ses IdPs auprès d'un service de liaison, il peut effectuer des requêtes vers des SPs nécessitants des attributs provenant de plusieurs sources. La Figure 2.8 démontre comment ceci est réalisé. Encore une fois, les requêtes sont effectuées dans un contexte d'application Web accédée par un fureteur Web.

Tel qu'expliqué par Chadwick et Inman (2009),

1. Le sujet effectue une requête à un SP pour une ressource.

1. Un niveau d'assurance et une force associée à la technique utilisée pour authentifier un sujet. Par exemple, un nom d'utilisateur et un mot de passe sont plus faciles à deviner ou à obtenir de manière frauduleuse qu'une caractéristique biométrique comme la forme de l'iris d'un sujet. Par conséquent, un sujet s'authentifiant avec un nom d'utilisateur et un mot de passe verra sa session associée à un plus petit niveau d'assurance qu'un sujet s'authentifiant en utilisant une technique biométrique (Chadwick, 2008; Zhang *et al.*, 2007)

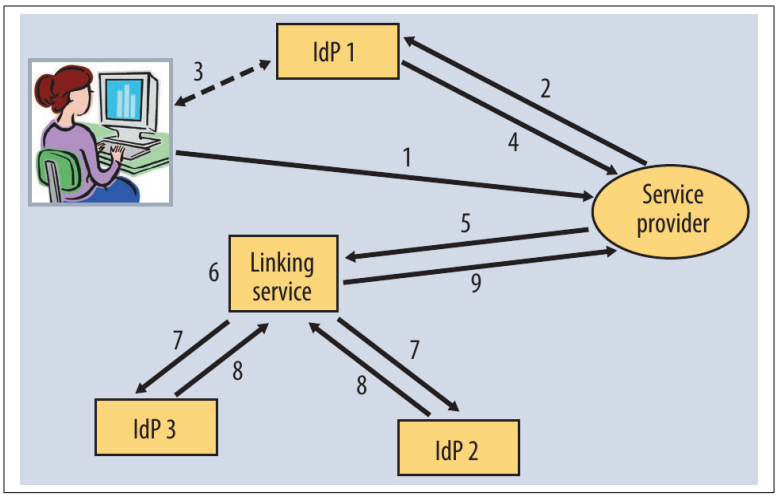

Figure 2.8 Fourniture d'un service par un SP nécessitant des attributs provenant de plusieurs IdPs et impliquant un service de liaison
Tiré de Chadwick et Inman (2009)

2. Le SP redirige le sujet vers son IdP. Afin de déterminer vers quel IdP rediriger le sujet, le SP peut utiliser une des méthodes introduites à la section 2.5.3.2.

3. Le sujet s'authentifie à son IdP. À cette étape, l'interface d'authentification présente un choix au sujet lui permettant d'activer ou non l'agrégation d'attributs. L'IdP génère aussi un identifiant pseudoaléatoire à utiliser pour faire référence au sujet auprès du SP lors de l'étape 9.

4. L'IdP retourne une assertion d'authentification au SP. Cette assertion contient les attributs du sujet qui est identifié grâce à l'identifiant pseudoaléatoire généré à l'étape précédente et, si l'agrégation des attributs a été activée une référence vers le service de liaison du sujet et le PID du sujet, chiffrée à l'intention du service de liaison.

5. Lorsque le SP reçoit l'assertion, il tente de prendre une décision de contrôle d'accès en analysant les attributs contenus dans l'assertion. S'il lui manque des attributs pour prendre une décision, il suit la référence vers le service de liaison et lui transmet l'assertion d'authentification, prouvant que le sujet a été authentifié et lui indiquant son PID.

6. Le service de liaison effectue une recherche dans sa table de liaison afin de savoir avec quels autres IdPs le sujet fait affaire. Optionnellement, le service de liaison peut être configuré pour ne prendre en compte que les IdPs avec qui l'enregistrement s'est préalablement effectué avec un LoA supérieur ou égal à celui de l'authentification du sujet à l'étape 3. Les différents LoAs associées aux enregistrements avec les IdPs sont inscrit dans la table du service à l'étape 4 de la Figure 2.7.

7. Le service de liaison contacte les IdPs identifiés à l'étape 6 et effectue une requête pour les attributs du sujet identifié par le PID qu'il a reçus à l'étape 5, au nom du SP.

8. Les IdPs contactés répondent à la requête en émettant chacun une assertion d'attributs concernant le sujet qu'ils signent et chiffrent à l'intention du SP.

9. Le service de liaison retourne les assertions d'attributs au SP en indiquant qu'elles font référence à un sujet possédant l'identifiant pseudoaléatoire généré à l'étape 3. L'ensemble des attributs provenant des différents IdPs pour un seul sujet est donc retourné au SP en faisant référence à un nom commun au lieu de faire référence à un ensemble d'identifiants. À l'interne, les différents IdPs utilisent en effet chacun un identifiant distinct pour faire référence à un sujet donné.

Mentionnons aussi qu'il est possible que le SP effectue lui même l'agrégation des attributs, sans passer par le service de liaison. Dans ce cas, les étapes suivantes sont modifiées :

7a. Le service de liaison retourne des références aux IdPs identifiés à l'étape 6, vers le SP. Le SP effectue alors une requête pour les attributs du sujet.

9a. Le SP est maintenant en possession des assertions émisses à l'étape 8 et considère qu'elles font référence à un sujet possédant l'identifiant pseudoaléatoire généré à l'étape 3. L'ensemble des attributs provenant des différents IdPs pour un seul sujet est donc retourné au SP en faisant référence à un nom commun au lieu de faire référence à un en-

semble d'identifiants. Les différents IdPs utilisent en effet chacun un identifiant distinct pour faire référence à un sujet donné.

Une fois que le SP est en possession de tous les attributs nécessaires, il est en mesure d'effectuer une décision de contrôle d'accès.

CHAPITRE 3

PLATEFORME TECHNOLOGIQUE

3.1 Le IaaS Framework

Le IaaS Framework (IaaS Framework Team, 2010) est une solution qui permet la création et la gestion de ressources IaaS virtuelles à partir de ressources physiques ou virtuelles. Les ressources ainsi créées peuvent par conséquent avoir une interface et un modèle uniforme. Le IaaS Framework permet aussi de créer des réseaux virtuels grâce à la virtualisation de routeurs et de commutateurs. Ceci permet de relier plusieurs fournisseurs désirant partager des ressources, créant du même coup un marché électronique de ressources virtuelles. Un tel marché de ressources virtuelles est présentement en cours de conception chez la compagnie Inocybe.

On peut donc voir que la vocation du IaaS Framework est quelque peu différente de celle des gestionnaires de nuages disponibles sur le marché tels que OpenStack (OpenStack, 2011) ou OpenNebula (OpenNebula, 2010) par exemple. Alors que ceux-ci sont utilisés pour gérer un seul nuage privé, public ou hybride appartenant à une organisation, le IaaS Framework a comme but de gérer les ressources composant la fédération d'un ensemble de nuages informatiques. Une fédération peut même utiliser conjointement un gestionnaire de nuage tel qu'OpenStack et le gestionnaire de ressources IaaS Framework. Dans un tel scénario, le gestionnaire de nuages fournit des ressources virtuelles (c.-à-d., des machines virtuelles) au IaaS Framework et ce dernier se charge de les rendre disponibles aux autres entités participant à la fédération.

La Figure 3.1 expose les différents composants du IaaS Framework ainsi que les relations entre eux. La Figure 3.2 présente quant à elle un déploiement plus complet et réaliste.

Voici maintenant une courte description des concepts importants. Il est à noter qu'une description détaillée des acteurs et de leurs besoins est présentée au Chapitre 4.

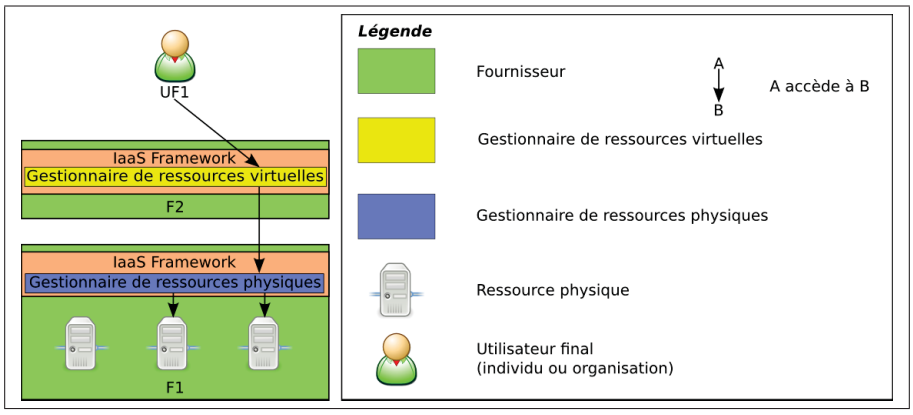

Figure 3.1 Les composants du IaaS Framework

Ressource physique Une pièce d'équipement physique telle qu'un disque dur ou un serveur.

Fournisseur Un fournisseur possède des ressources physiques et/ou virtuelles qu'il rend disponible à ses clients grâce à son instance du IaaS Framework.

Gestionnaire de ressources physiques Une composante logicielle qui interagit directement avec des ressources physiques pouvant être (i) un hôte exécutant une solution de virtualisation, (ii) une unité de stockage, (iii) de l'équipement réseau (commutateur ou routeur), (iv) une source d'énergie et une unité d'alimentation et (v) un senseur de conditions climatiques.

Gestionnaire de ressources virtuelles Une composante logicielle qui fournit des ressources virtuelles à partir des ressources physiques disponibles pouvant être accédées depuis le gestionnaire de ressources physiques. Plusieurs types de gestionnaires de ressources virtuelles sont présents : un *Facility Manageur* pour les sources d'énergie, les unités d'alimentation et les senseurs de conditions climatiques, un *Cloud Manager* pour les hôtes virtuels et les unités de stockage virtuelles et un *Network Manager* pour les commutateurs et routeurs virtuels

Les deux derniers items sont les principaux composants pouvant être déployés dans une instance donnée du IaaS Framework.

Si la Figure 3.1 représentait un déploiement réel et non une simplification visant à illustrer les constituants du IaaS Framework, il aurait été plus judicieux de déployer les deux gestionnaires de ressources dans l'instance présente au site de F1. En observant la Figure 3.2, qui illustre un scénario de déploiement plus réaliste, les avantages derrières la possibilité de déployer les gestionnaires de ressources dans des sites différents deviennent plus clairs. Ce scénario est utilisé au Chapitre 4 afin de présenter les acteurs et de définir leurs besoins en matière de gestion de l'identité et de contrôle d'accès.

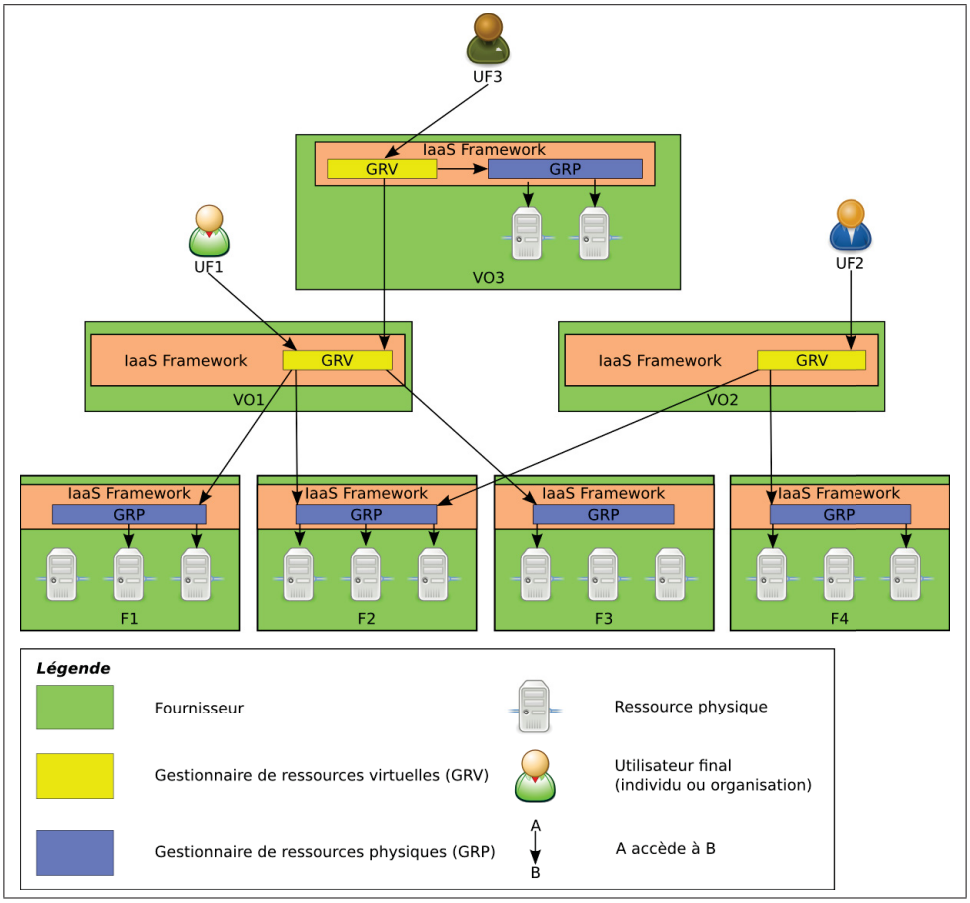

Figure 3.2 Un exemple de déploiement du IaaS Framework

La Figure 3.2 exprime aussi un parallèle intéressant qui peut être fait avec l'Internet en général. Alors que l'Internet est constitué d'un réseau de réseaux interconnectés, le IaaS Framework permet l'interconnexion de nuages informatiques. Par exemple, VO1 et VO3 sont interconnec-

tés et sont tout les deux des nuages informatiques. En effet, ils partitionnent des ressources existantes en ressources virtuelles et y fournissent un accès grâce à des services de nuages.

3.2 Le réseau GreenStar

Le réseau GreenStar (Synchromedia, 2010) est un projet pilote financé par l'organisme CANARIE (CANARIE, 2009) dont la première phase s'échelonne sur deux ans, de janvier 2010 à décembre 2011. Ce projet réfère à deux concepts. Premièrement, un volet technique qui consiste en une extension au IaaS Framework qui (i) permet d'abstraire des hôtes physiques en machines virtuelles et (ii) fournit de la logique d'affaire utilisée pour distribuer ces machines virtuelles d'une manière écologique à travers différents noeuds distribués géographiquement dans un réseau. Ensuite, il s'agit d'un protocole de réduction des émissions de gaz à effet de serre pour les projets en technologies de l'information et des télécommunications.

3.2.1 Le volet technique

L'objectif de la partie technique du projet est de permettre à des machines virtuelles d'être alimentées autant que possible par des sources d'énergies vertes. Ceci est réalisé grâce au principe *Follow the Wind/Follow the Sun* (suivre le vent/suivre le soleil). Ce principe nécessite que différents noeuds alimentés par des sources d'énergies renouvelables (p. ex. solaire ou éolien) soient géographiquement distribués. Le *Contrôleur* du Réseau GreenStar surveille ces noeuds, leur charge, leurs capacités, et y fait migrer des machines virtuelles en temps réel. Par exemple, à un instant *t*, le *Fournisseur 1* est en mesure d'alimenter son infrastructure physique grâce à de l'énergie solaire. Le *Contrôleur* fait donc migrer un maximum de machines virtuelles vers ce fournisseur. Par contre, à l'instant *t+x*, le soleil se couche, faisant en sorte que le *Fournisseur 1* doive s'approvisionner en électricité à partir du réseau public, alimenté par une centrale au charbon. À ce moment, le *Contrôleur* récupère les machines virtuelles hébergées par le *Fournisseur 1* et les migre vers le *Fournisseur 2*, dont l'infrastructure physique est alimentée grâce à de l'énergie éolienne. Notons que la migration d'une machine virtuelle est transparente pour un usager. En effet, des expérimentations ont démontré qu'il est possible de diffuser une vidéo

à partir d'une machine virtuelle qui migre d'un hôte à un autre, avec peu ou pas d'impact sur les clients distants la visionnant Synchromedia (2010).

À terme, la première phase du projet aboutira à une preuve de concept incluant des noeuds canadiens, européens et américains fournie par les organisations suivantes :

RackForce Networks Un centre de données à Kelowna en Colombie-Britannique alimenté par de l'hydroélectricité et refroidi à la géothermie.

École de technologie supérieure Une université hébergeant un centre de données à Montréal alimenté par l'hydroélectricité. Au sein de cette université, le laboratoire Synchromedia (Synchromedia, 2011) est aussi responsable du développement de la solution logicielle rendant le réseau GreenStar possible.

Communication Research Centre Un centre de recherche à Ottawa fournissant un noeud alimenté à l'énergie solaire. Étant donné que ce centre dispose d'une grande expertise en réseautique, on y conçoit et on y teste aussi toute l'infrastructure du réseau GreenStar.

Cybera Inc. Centre de recherche en informatique distribuée en Alberta qui met à la disposition du réseau GreenStar, un noeud alimenté à l'énergie solaire.

HEAnet Le réseau de recherche national de l'Irlande fourni un noeud alimenté à l'énergie solaire et à l'énergie éolienne.

i2Cat Un centre de recherche à Barcelone qui dispose d'une branche reliée à la virtualisation des réseaux et qui fournit un noeud alimenté à l'énergie solaire.

NORDUnet Une collaboration de cinq pays nordiques (Danemark, Finlande, Islande, Norvège et Suède), qui opère un réseau dédié à la recherche, contribue un noeud alimenté par géothermie.

Trois autres noeuds ont manifesté un intérêt à participer au réseau GreenStar dans le futur. Premièrement, IBBT, un institut de recherche en Belgique. Ensuite, Calit2, l'institut des té-

lécommunications et des technologies de l'information en Californie. Finalement, le réseau Starlight.

La relation entre l'extension GreenStar du IaaS Framework et le IaaS Framework lui-même est la suivante. L'extension GreenStar virtualise et partitionne des hôtes physiques en machines virtuelles et les rend disponibles au gestionnaire de ressources virtuelles du IaaS Framework. Ce dernier s'occupe ensuite des différentes opérations sur ces ressources, telles que la gestion de leur cycle de vie et l'exécution des commandes sur celles-ci. De plus, avec la logique et les algorithmes de son *Contrôleur*, l'extension GreenStar permet aussi au gestionnaire de ressources virtuelles du IaaS Framework de distribuer ses machines virtuelles de manière à ce qu'elles soient alimentées autant que possible par des sources d'énergie vertes.

3.2.2 Le protocole de réduction des émissions de gaz à effet de serre

Le protocole de réduction des émissions de gaz à effets de serre du réseau GreenStar vient du constat que la tendance mondiale va dans le sens d'une augmentation de la dépendance aux technologies de l'information et des télécommunications. Si l'on considère qu'une grande proportion de l'électricité servant à alimenter le matériel fournissant ces technologies est produite par des combustibles fossiles, on peut affirmer que les technologies de l'information et des télécommunications produisent de plus en plus de gaz à effet de serre. Or, dans ce domaine, il n'y a pas de protocole en place permettant de déterminer si un projet donné permettra de réduire considérablement les émissions de gaz à effet de serre. Le protocole GreenStar vient pallier cette lacune en guidant les instigateurs de projets sur les valeurs à mesurer, comment les mesurer et comment établir des conclusions (Synchromedia, 2010).

CHAPITRE 4

BESOINS DU IAAS FRAMEWORK EN TERMES DE GESTION DE L'IDENTITÉ DE DU CONTRÔLE D'ACCÈS

Rappelons qu'un des problèmes avec les plateformes de nuages informatiques actuelles est le fait qu'elles ne permettent pas l'interopérabilité entre les fournisseurs. Ceci les empêche de se regrouper, ce qui pourrait être particulièrement utile aux petits fournisseurs afin de les aider à atteindre des objectifs communs et à conquérir de nouveaux marchés. Quant à eux, les clients de ces fournisseurs pourraient bénéficier de cette interopérabilité, car elle contribuerait à réduire le verrouillage envers les fournisseurs.

La plupart des plateformes permettant la création de nuages informatiques ne prennent en compte que les ressources appartenant à un seul fournisseur. Par conséquent, elles ne permettent pas de partager des ressources et de créer des fédérations. Au contraire, les différents types de ressources pouvant être virtualisées par le IaaS Framework peuvent être échangés entre les différents membres d'une fédération. Ce chapitre traite des différents besoins du IaaS Framework en ce qui a trait à la gestion de l'identité au contrôle d'accès afin de permettre la création de telles fédérations.

4.1 Acteurs

Cette section présente les différents acteurs du IaaS Framework, leurs interactions et leurs rôles

Fournisseur de ressources physiques Une entité qui détient le contrôle sur des ressources physiques et qui fournit un accès à celle-ci. Dans la Figure 3.2, F1 à F4 et VO3 sont des fournisseurs de ressource physique. VO3 est dans cette catégorie étant donné qu'il fournit des ressources physiques à lui-même.

Fournisseur de ressources virtuelles Une entité qui partitionne des ressources physiques et/ou virtuelles existantes en ressources virtuelles afin de les rendre disponible à ses clients.

Dans la Figure 3.2, VO1 à VO3 sont des fournisseurs de ressources virtuelles. Notez que VO3 s'appuie sur les ressources virtuelles de VO1.

Organisation virtuelle Dans le contexte du IaaS Framework, une organisation virtuelle (VO) est un fournisseur de ressources virtuelles qui récupère des ressources de plusieurs fournisseurs.

Utilisateur final Cet acteur accède à des ressources virtuelles grâce à des applications s'appuyant sur l'API de ces ressources virtuelles. Par exemple, un utilisateur final peut stocker des fichiers sur un service de stockage, configurer un service de routeur virtuel, etc. De plus, en utilisant une instance du IaaS Framework, un utilisateur peut partager ses ressources et devenir lui-même un fournisseur.

Propriétaire de ressource Cet acteur reçoit des ressources d'un fournisseur de ressources virtuelles en fonction d'une entente de service. Lorsqu'il a accès à une ressource, le propriétaire obtient un contrôle total sur celle-ci et peut y définir des droits d'accès pour n'importe quel autre sujet. Dans la Figure 3.2, VO1 peut être le propriétaire d'une ressource spécifique hébergée chez F1. En outre, UF1 peut être le propriétaire d'une ressource virtuelle fournie par VO1.

4.2 Besoins

L'objectif général du IaaS Framework est de permettre à un fournisseur d'offrir un accès à ses ressources pour ses clients. Ces clients peuvent être des utilisateurs finaux ou bien d'autres fournisseurs. Cette seconde possibilité est illustrée dans la Figure 3.2 avec la relation entre VO1 et VO3. VO3 pourrait ne pas disposer d'assez de ressources pour tous ses clients lors des périodes de haute demande. Par conséquent, il pourrait incorporer un sous-ensemble des ressources de VO1 dans sa propre réserve de ressource. L'allocation de ressources à la demande est en fait un cas d'utilisation typique des nuages informatiques (Foster *et al.*, 2008; Zhang *et al.*, 2010). Dans ce scénario, VO1 doit autoriser VO3 avant que quelconque accès ait lieu. Le processus de partage de ressource doit aussi être réalisé de manière à respecter les politiques de contrôle d'accès définies par VO1.

Dans les sous-sections suivantes, les besoins spécifiques nécessaires pour accomplir notre but général de gestion de l'identité et de contrôle d'accès sont présentés.

4.2.1 Nuages informatiques fédérés

Tel que mentionné aux Sections 1.1 et 2.1, les solutions de nuages informatiques existantes ne sont pas conçues pour permettre l'interopérabilité interfournisseurs. Ceci empêche les fournisseurs de petite et moyenne taille de se regrouper, ce qui pourrait faciliter la réalisation de projets communs, ou bien simplement permettre le partage de ressources afin de bâtir des infrastructures virtuelles plus importantes. *Pour cette raison, les fournisseurs de ressources devraient être en mesure de collaborer et de partager des ressources dans une réserve commune (Besoin I).*

Par définition, nous désirons donc créer des nuages informatiques fédérés (Rochwerger *et al.*, 2009). Dans la Figure 3.2, nous considérons VO1, VO2 et VO3 comme des nuages fédérés étant donné qu'ils sont constitués de ressources de F1, F2, F3, F4 et VO1.

4.2.2 Politiques de contrôle d'accès flexibles

Il devrait être possible de définir des politiques de manière flexible. Par défaut, le IaaS Framework devrait être configuré avec des politiques de contrôle d'accès simples et efficaces. Or, certains fournisseurs pourraient être insatisfaits avec les paramètres par défaut et pourraient vouloir les peaufiner selon leurs besoins. Puisque le IaaS Framework n'est pas une solution en soi, mais plutôt une base pour d'autres projets, il se doit d'être assez souple pour accommoder plusieurs cas d'utilisations.

Pour favoriser la flexibilité, il devrait être possible de prendre des décisions de contrôle d'accès sur la base de n'importe quel attribut de (i) le sujet effectuant une requête, (ii) la ressource ciblée, (iii) l'action à accomplir par le sujet sur la ressource, (iv) les conditions environnementales dans lesquelles a lieu l'action. Par exemple, des énoncés de politiques similaires à l'énoncé suivant devraient être permis : *Lorsque la charge de travail de l'infrastructure atteint*

le seuil X, seuls les utilisateurs de type Y devraient être en mesure de modifier les configurations des routeurs de catégorie Z. Par conséquent, *les politiques de contrôle d'accès doivent pouvoir être le plus flexible possible (Besoin II).*

4.2.3 Organisations virtuelles dynamiques

L'illustration de la relation entre VO1 et VO3 dans l'introduction de la Section 4.2 souligne le fait qu'*il devrait être possible de créer des VOs dynamiques et de courtes durées où les relations de confiance seraient définis de manière* ah hoc *et pair-à-pair (Besoin III).*

Dans l'exemple décrit plus tôt, comme seules deux entités étaient impliquées, le besoin d'établir des relations de confiance complexes n'est pas vraiment présent. Il est par contre facile d'extrapoler et d'imaginer un scénario où plusieurs fournisseurs voudraient se regrouper afin de former une VO de courte durée. Par exemple, une telle VO pourrait exister uniquement pour les besoins d'un projet scientifique donné concernant des entités collaborant entre elles. L'architecture de gestion de l'identité et de contrôle d'accès doit par conséquent être en mesure de traiter l'établissement des relations de confiance nécessaires pour ce type de regroupement.

Dans un même ordre d'idée, *le système de gestion des politiques de sécurité doit être en mesure de traiter les aspects dynamiques de ces environnements composés de domaines où des ressources et des sujets peuvent être ajoutés, supprimés et modifiés à tout moment (Besoin IV).* Cette caractéristique est en fait une propriété typique des environnements orientés service, tel que décris par Yuan et Tong (2005).

Dans la Section 2.2.1, il est expliqué qu'un domaine est un ensemble de ressources et de sujets gérés par une entité administrative donnée. Dans la Figure 3.2, toutes les ressources de F1 font partie du domaine F1. De même, VO1 utilise des ressources de F1, F2 et F3 afin de créer des ressources virtuelles qu'elle rend disponibles à ses clients, UF1 et VO3. Ces nouvelles ressources font partie du domaine de VO1.

4.2.4 Centralisation vs. décentralisation

4.2.4.1 Identité des sujets

L'environnement illustré à la Figure 3.2 possède une structure hiérarchique composée d'arbres indépendants. Par exemple, l'identité des sujets accédant à VO1 et VO3 n'a pas de lien avec VO2. Par conséquent, l'identité de tous les clients de tous les fournisseurs n'a pas à être centralisée à un seul endroit. Pour des raisons d'évolutivité, de facilité de gestion et de performance, un environnement centralisé est souhaitable dans les VOs statiques ou de petites envergures et non souhaitable dans les plus grosses VOs dynamiques (Sinnott *et al.*, 2008). Comme le IaaS Framework vise à supporter ce second type de VO, une solution de gestion de l'identité entièrement centralisée n'est pas appropriée.

Au contraire, des fournisseurs peuvent avoir un grand nombre de clients et ces clients peuvent avoir accès à plusieurs fournisseurs. Sans mécanismes de gestion adéquats, cette situation pourrait causer de multiples répétitions de comptes et rapidement mener à un système incohérent et ingérable. Par exemple, VO1 utilise les ressources de 3 fournisseurs. Si ceux-ci maintenaient individuellement les informations relatives à VO1, toutes les modifications apportées à ses attributs d'identité ou ses droits d'accès en plus des révocations de justificatif pouvant résulter d'une brèche de sécurité chez VO1 devraient être propagés aussi rapidement que possible afin de prévenir les divergences et les vulnérabilités de sécurité (Baier *et al.*, 2010; Chakrabarti, 2007). En outre, *un équilibre doit être atteint entre un système de gestion de l'identité centralisé et distribué (Besoin V).*

4.2.4.2 Politiques de contrôle d'accès

Les mécanismes de contrôle d'accès devraient eux aussi permettre un certain degré de centralisation. En effet, tel que présenté à la Section 2.2.1, les ressources et les sujets à l'intérieur d'un domaine devraient être gouvernés par un seul ensemble de politiques. Il est toutefois possible que les ressources d'un domaine soient gérées par des hôtes distribués à travers plusieurs sites composant le domaine d'un fournisseurs. *Il devrait être possible pour tous ces hôtes d'avoir ac-*

cès au même ensemble de politique afin d'être en mesure de prendre des décisions de contrôle d'accès de manière unifiée (Besoin VI).

4.2.4.3 Correspondance des métabesoins

Le Tableau 4.1 illustre comment les six besoins qui sont maintenant formellement définis correspondent au métabesoins généraux introduits à la Section 1.3.

Tableau 4.1 Matrice de correspondance des métabesoins

Besoin	**Metabesoin**
I - Nuages fédérés	1. Coopération entre des fournisseurs
II – Politiques de contrôle d'accès flexibles	4. Flexibilité des politiques
III – Possibilité de former des VO dynamiques et de courte durée, de manière *ad hoc* IV – Capacité de gérer les politiques efficacement dans des environnements dynamiques	2. Création de VO dynamiques avec minimum d'adaptations aux politiques
V – Équilibre entre un système de gestion de l'identité centralisé et distribué	
VI – Possibilité pour tous les hôtes d'un domaine de prendre des décisions de contrôle d'accès de manière unifiée	3. Variabilité et cohérence dans les droit d'accès

On peut voir qu'après l'analyse formelle du contexte et des besoins du IaaS Framework faite tout au long du Chapitre 4, presque chaque besoin concret correspond à un métabesoin général. L'exception est le *Besoin V* qui demande un équilibre entre un système de gestion de l'identité centralisé et distribué. Ce besoin n'aurait pas pu être élaboré avant d'analyser concrètement le contexte et le fonctionnement du IaaS Framework. Il n'a donc pas été prévu lors de l'établissement des métabesoins et n'a aucune correspondance dans le Tableau 4.1.

CHAPITRE 5

SOLUTION

5.1 Architecture

À la lumière des besoins et objectifs décrits au Chapitre 4 en termes de gestion de l'identité et de contrôle d'accès, nous avons conçu une solution qui est à la fois fédérée et hiérarchique. Cette solution aborde les besoins indiqués au Chapitre 4 et illustrés dans la Figure 3.2.

5.1.1 Déroulement du contrôle d'accès

Premièrement, il est important de noter que toutes les VOs accèdent à des ressources qui proviennent de plusieurs fournisseurs. Ceci est vrai même pour VO3, car il obtient des ressources virtuelles de VO1 et des ressources physiques de lui-même. Pour les raisons énoncées à la Section 4.2.4, les identités des VOs ne doivent pas être dupliquées auprès de tous les fournisseurs de ressources avec qui ils font affaire. Dans ces conditions, *leurs identités doivent être gérées par un IdP externe contrôlé par un tiers de confiance (Élément de solution A)*.

Les fournisseurs de ressources physiques doivent avoir confiance en ce que ce tiers authentifie correctement l'identité des VOs effectuant des requêtes pour des ressources. De leur côté, les VOs doivent avoir confiance en ce que le tiers gère leurs identités de manière juste et sécuritaire.

La Figure 5.1 démontre la séquence d'évènements préalable à ce que VO1 puisse accéder à des ressources de F1 et de F2. Ces évènements sont représentés par des lignes pointillées :

1. VO1 s'authentifie avec l'IdP de confiance

2. Après une authentification réussie, l'IdP retourne un justificatif à VO1

3. VO1 effectue une requête afin d'accéder à une ressource de F1. La requête contient le justificatif émis par l'IdP. En fonction de ce justificatif, F1 donne accès ou non à la

ressource demandée. La décision de F1 est basée sur ses politiques de contrôle d'accès locales.

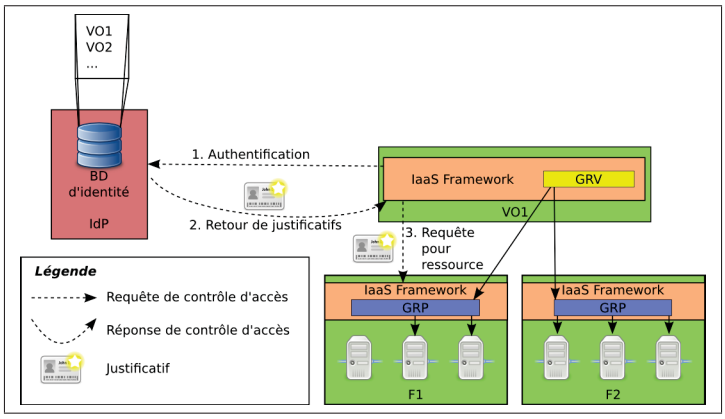

Figure 5.1 Un tiers de confiance gère l'identité des VOs

Cet aspect de la solution consiste en un système de gestion de l'identité fédérée typique tel que décrit à la Section 2.2.1. Il règle le problème de la duplication des comptes pour les clients qui accèdent à des ressources de plusieurs fournisseurs.

Ensuite, afin de permettre la création de VO plus dynamique et de courte durée (voir *Besoin III*), *il devrait être possible pour un fournisseur de ressource d'agir à la fois comme un SP et au besoin, comme un IdP (Élément de solution B)*. Ce second aspect de la solution suit le modèle décris dans Suess et Morroney (2009) dans lequel un fournisseur authentifie ses utilisateurs finaux et leur fourni aussi un service.

La Figure 5.2 présente cet aspect de la solution de manière plus concrète. Dans le scénario présenté, VO2 peut authentifier ses utilisateurs finaux (i.e., UF2), sans avoir à se fier sur un tiers. De plus, si VO2 et VO3 en venaient à vouloir créer une VO de courte durée pour partager des ressources entre les frontières de leurs domaines, VO3 pourrait configurer son SP pour qu'il soit en mesure d'autoriser des requêtes provenant de sujets authentifiés par l'IdP de VO2, et vice-versa. Ceci permettrait à UF2 d'accéder à des ressources chez VO3, même si son identité est gérée par VO2. De la même manière, ceci permettrait à UF3 d'accéder à des ressources chez VO2. Ceci serait possible grâce à la relation établie dans laquelle VO2 ferait confiance à

ce que VO3 authentifie correctement ses sujet et gère leurs attributs de manière appropriée. Il en va de même pour VO3 qui établirait une relation de confiance analogue envers VO2.

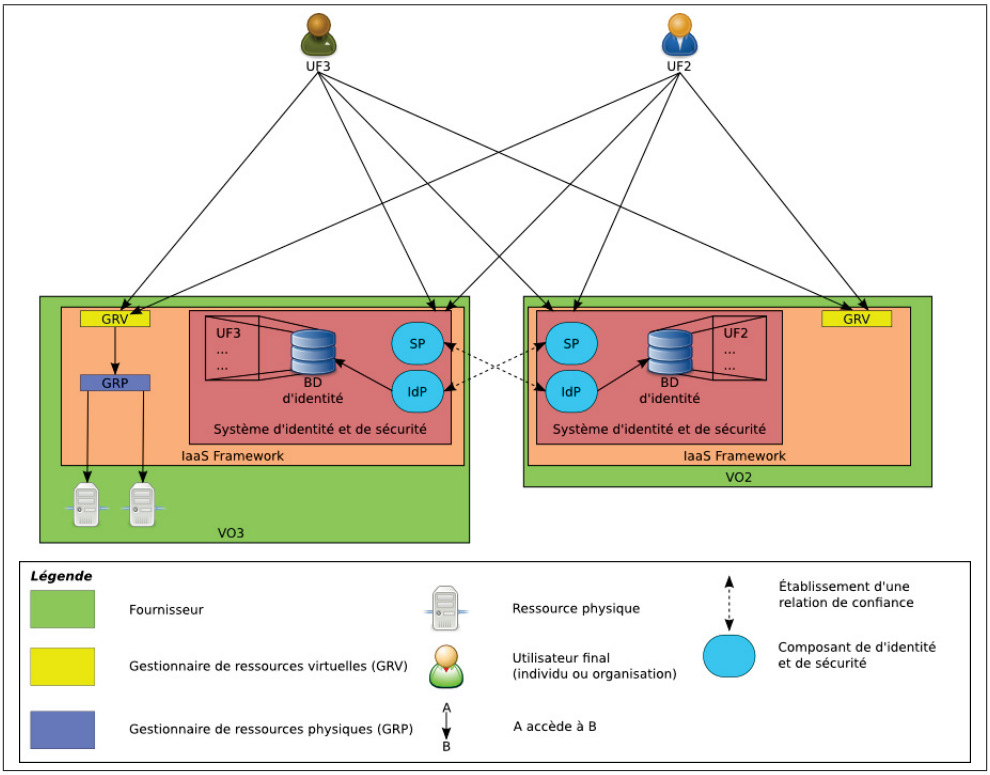

Figure 5.2 Établissement de la confiance pour les VOs dynamiques

Il est important de noter que si un fournisseur n'a pas à gérer directement ses clients, il n'a pas à agir en tant qu'IdP. Par exemple, dans la Figure 3.2, F1, F2, F3 et F4 n'agissent pas en tant qu'IdP étant donné que l'identité de leurs clients, VO1 et VO2, est prise en charge par le tiers de confiance mentionné précédemment.

De plus, l'*Élément de solution B* permet un plus grand contrôle pour à la fois les sujets (qu'ils soient utilisateurs finaux ou fournisseurs de services) et les fournisseurs de services avec qui ils font affaire, en ce qui concerne les entités envers lesquelles ils font confiance. Tout d'abord, selon leurs préférences, les sujets n'ont pas nécessairement besoin de faire affaire avec l'IdP central s'ils ne croient pas que cet IdP soit en mesure d'authentifier correctement ses sujets et de gérer leurs attributs de manière adéquate. Ils peuvent choisir n'importe quel IdP qui leur semble approprié et qui est accepté par le fournisseur de services avec qui ils désirent interagir.

Quant à eux, les fournisseurs de services peuvent réduire au maximum le nombre d'entités impliquées dans des transactions d'identité lorsque le contexte le demande. Par exemple, une VO pourrait créer un nuage privé (Zhang *et al.*, 2010) en ne fournissant des ressources qu'à son parc privé d'utilisateurs, qui pourrait demeurer secret et inconnu du public.

Ensemble, les *Éléments de solution A* et *B* satisfont le principal besoin de la solution de gestion de l'identité du contrôle d'accès (voir *Besoin I*). Ils permettent la collaboration et le partage de ressources entre des fournisseurs indépendants.

5.1.2 Politiques de sécurité

En ce qui concerne la gestion des politiques de sécurité, le IaaS Framework inclus les rôles suivants par défaut :

Comptable Un sujet qui a accès à des données statistiques concernant l'instance du IaaS Framework déployée. Ces données sont utilisées à des fins de comptabilité, pour générer des rapports et pour la facturation.

Administrateur Un sujet qui détient tous les droits sur les ressources gérées par l'instance donnée.

Gestionnaire Un sujet qui est responsable de la gestion de l'identité et des droits d'accès pris en charge par le SP l'IdP (lorsque présent) de l'instance donnée pour les ressources d'un fournisseur.

Utilisateur final Un sujet qui désire accéder des ressources d'une instance données.

Ces rôles peuvent être utilisés par un fournisseur désirant utiliser un contrôle d'accès basé sur les rôles (RBAC) (Bishop, 2004). Toutefois, afin de satisfaire le *Besoin II*, un fournisseur ne peut pas être forcé d'utiliser cette configuration. Un fournisseur pourrait vouloir baser son contrôle d'accès sur des rôles différents ou bien sur un ensemble d'attributs, ce qui n'est pas possible avec le modèle RBAC. En outre, un fournisseur pourrait même vouloir effectuer du

contrôle d'accès sur un paradigme différent tel que LBAC ou IBAC. Comme il a été expliqué dans la Section 2.3, le modèle ABAC est assez flexible pour satisfaire tous ces scénarios. Par conséquent, il a été déterminé que *le modèle de contrôle d'accès devrait être basé sur ABAC Yuan et Tong (2005) (Élément de solution C)*, car il est en mesure de fournir toute la flexibilité nécessaire.

Une autre caractéristique de la gestion de politique présente dans la solution concerne *l'extraction des politiques des ressources et des services qui en dépendent de manière à les centraliser dans le domaine d'un fournisseur (Élément de solution D)*. En centralisant les politiques dans un seul point d'accès, le *Besoin IV* et le *Besoin VI* sont abordés. Le *Besoin IV* demande que le système de gestion des politiques soit en mesure de traiter les environnements dynamiques alors que le *Besoin VI* requiert que toutes les ressources d'un domaine soient gouvernées par un seul ensemble de politiques.

L'*Élément de solution D* aborde le *Besoin IV*, car lorsque des actifs sont ajoutés, supprimés ou modifiés à l'intérieur d'un domaine et lorsque des fournisseurs et des clients sont ajoutés à une VO, les changements reliés aux politiques sont effectués à un seul endroit. L'administration des politiques se voit ainsi simplifié. De plus, comme les politiques sont découplés des composantes logicielles qui les utilisent, les composantes n'ont pas besoin d'être modifiées ou redémarrées lorsque les besoins d'un environnement changent (Laborde *et al.*, 2008).

D'un autre côté, le *Besoin VI* est réalisé, car toutes les décisions de contrôle d'accès qui sont prises à l'intérieur d'un domaine sont accomplies en évaluant un seul ensemble de politiques. Ceci est effectué en mettant en place un système de gestion des politiques modulaire qui suit les principes du RFC 2904 (Vollbrecht *et al.*, 2000) de l'IETF et du standard XACML (OASIS, 2005b) avec un PEP *(Policy Enforcement Point)*, un PIP *(Policy Information Point)*, un PDP *(Policy Decision Point)* et un PAP *(Policy Administration Point)* (voir Section 2.4.3.3). Chaque ressource du IaaS Framework est protégée par un PEP qui communique avec un PIP afin d'avoir tous les attributs nécessaires afin de bâtir une requête qu'il envoie à un PDP. Ce PDP retourne la décision d'autorisation qui est appliquée par le PEP. Quant à lui, le PDP prend une décision en

évaluant des politiques qui sont centralisées dans un référentiel unique pour chaque domaine. Ces politiques sont gérées par un administrateur grâce à un PAP.

La Figure 5.3 illustre les interactions entre ces composants qui sont nécessaires à l'application des politiques de sécurité dans un domaine administratif donné :

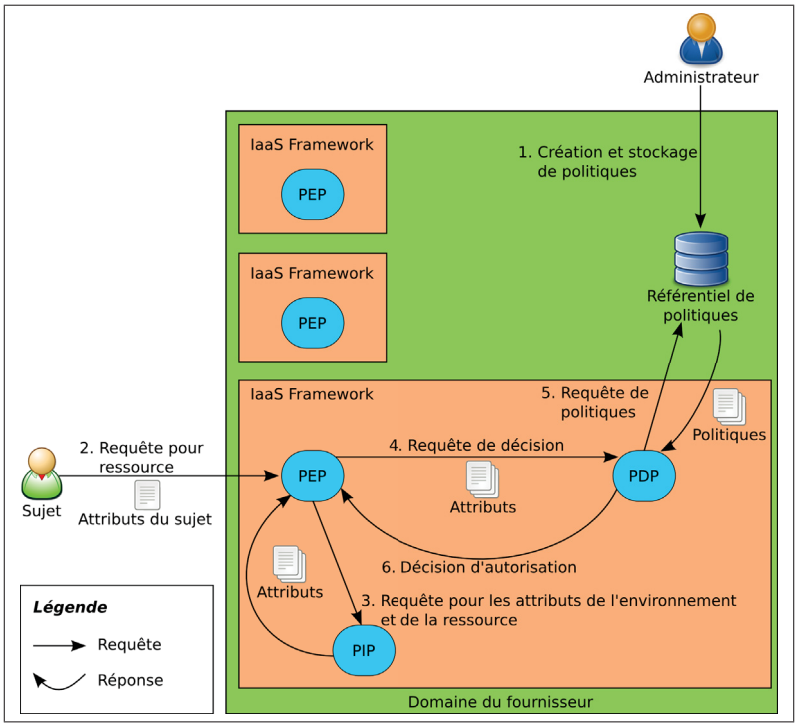

Figure 5.3 Flot des opérations d'autorisation dans le IaaS Framework

1. Un administrateur ou un gestionnaire stocke les politiques de contrôle d'accès concernant les ressources de son domaine.

2. Un sujet ayant été authentifié par son IdP est en possession d'un jeton SAML qui contient ses attributs. Il fournit ses attributs au PEP avec sa requête pour accéder à une ressource.

3. Le PEP récupère les attributs de l'environnement et de la ressource auprès du PIP.

4. Le PEP demande une décision d'autorisation au PDP. Le PEP informe le PDP à propos des détails de la requête ainsi que des attributs du sujet, de la ressource et de l'environnement.

5. Le PDP récupère les politiques appropriés du référentiel de politiques.

6. Le PDP effectue une décision d'autorisation basée sur l'information qu'il possède et la retourne au PEP

Un point mérite d'être abordé en ce qui concerne l'administration des politiques. Plusieurs administrateurs peuvent interagir simultanément avec le référentiel de politiques afin d'en créer de nouvelles ou de les modifier. Dans de telles conditions, il est possible que des contradictions et des ambigüités surviennent. Nous n'avons pas prévu de mécanisme pouvant régler automatiquement ce genre de problème. Une telle fonctionnalité pourrait faire l'objet de recherches futures.

Finalement, un autre point venant aborder le *Besoin IV* est le fait que *les politiques ne concernent que les interactions directes entre un fournisseur et ses clients (Élément de solution E)*. Ceci vient grandement simplifier leur gestion. Par exemple, dans la Figure 3.2, on peut voir qu'une seule entité accède directement aux ressources de F1. Il s'agit de VO1. Il est vrai que UF1, VO3 et UF3 peuvent aussi accéder aux ressources de F1, mais ceci se fait de manière indirecte. En effet, VO1 partitionne les ressources de F1 comme bon lui semble et les rend disponible à UF1 et VO3 (VO3 rend ensuite ces ressources disponibles à UF3), mais du point de vue de F1, la seule entité qui accède ses ressources est VO1. F1 ne tient pas à contrôler les requêtes d'accès effectuées par les clients de ses clients, les clients de ceux-ci, etc. Tout ce qui l'importe est que ses clients directs accèdent à ses ressources en respectant ses politiques de contrôle d'accès. Par conséquent, les politiques de F1 n'ont qu'à prévoir des accès réalisés par VO1. Elles n'ont pas à prévoir VO3 et UF3 qui pourraient éventuellement avoir eux même d'autres clients. Les politiques sont donc simplifiées et plus faciles à gérer dans le cas de l'ajout et du retrait de sujets dans le système. Dans un même ordre d'idée, VO1 n'a pas à contrôler les accès entre UF1 et F1, F2 et F3. Elle ne fait que restreindre les accès entre UF1 et ses propres ressources qu'elle récupère de F1, F2 et F3. Bien entendu, VO1 doit aussi s'assurer de respecter les politiques de ses fournisseurs afin d'être en mesure de récupérer leurs ressources, de les virtualiser et de les offrir à UF1 selon les ententes de services qu'il détient avec celui-ci. En outre, il en va de

la responsabilité de F1, F2 et F3 de respecter les ententes de services qu'ils ont avec VO1. Le problème de la mise en place d'ententes de services et des moyens pouvant être employés pour les respecter sort du cadre de ce mémoire.

En plus de simplifier la gestion des politiques dans un environnement dynamique, ce choix de conception a aussi l'avantage de ne pas nécessiter la mise en place de mécanisme de délégation des droits d'accès. Si ce principe avait été mis en place, il aurait permis, par exemple, à VO1 d'accéder à une ressource de F1 au nom de UF1. UF1 aurait donc dû permettre à VO1 d'agir en son nom et de bénéficier des privilèges qui lui ont été accordés par F1. Plusieurs techniques (Sotomayor, 2005; Internet2, 2009) peuvent être utilisés afin de réaliser de la délégation de droits d'accès, mais elles ont l'inconvénient de complexifier le système. Il peut aussi être problématique d'expliquer à un utilisateur humain qu'en effectuant certaines opérations, il permet à un tiers d'agir en son nom.

En ce qui à trait au *Besoin IV*, l'*Élément de solution E* vient donc simplifier la conception et la gestion des politiques, distribuer les responsabilités au travers des différentes entités composant une VO ainsi que faciliter le processus de contrôle d'accès en général. Ce sont des caractéristiques qui viennent toutes faciliter la gestion d'environnements dynamiques (Sinnott *et al.*, 2008).

5.1.3 Remarques sur la solution

Le Tableau 5.1 résume la manière dont les besoins identifiés à la Section 4.2 sont résolus par la solution proposée dans cette section.

La solution proposée suit aussi quelques-unes des lois de l'identité de Kim Cameron (voir Section 2.5.2.1)

Divulgation minimale pour une utilisation contrainte Cette loi demande que les entités d'un système conservent le moins d'information d'identité possible. Dans l'architecture proposée, le SP n'a pas à conserver d'information d'identité relative au sujet, car elles lui sont systématique ment transmises par l'IdP. Toutefois, dans ce modèle, on pourrait faire

Tableau 5.1 Matrice de satisfaction des besoins

Besoin	**Solutionné par l'Élément de solution**
I - Nuages fédérés	A – Possibilité d'avoir un IdP externe qui gère l'identité B – Un fournisseur de ressource peut agir en tant qu'IdP
II – Politiques de contrôle d'accès flexibles	C – Système ABAC paramétrable
III – Possibilité de former des VO dynamiques et de courte durée, de manière *ad hoc*	B – Un fournisseur de ressource peut agir en tant qu'IdP
IV – Capacité de gérer les politiques efficacement dans des environnements dynamiques	D – Extraction des politiques des ressources et des services de manière à les centraliser à l'intérieur d'un domaine E – Les politiques ne concernent que les interactions directes
V – Équilibre entre un système de gestion de l'identité centralisé et distribué	A – Possibilité d'avoir un IdP externe qui gère l'identité
VI – Possibilité pour tous les hôtes d'un domaine de prendre des décisions de contrôle d'accès de manière unifiée	D – Extraction des politiques des ressources et des services de manière à les centraliser à l'intérieur d'un domaine

valoir qu'un IdP est une cible séduisante pour un attaquant désirant recueillir des données personnelles. En outre, rien ne peut empêcher un IdP malicieux de se présenter chez un SP en se faisant passer pour un des sujets qu'il gère (Alpár *et al.*, 2011), et rien n'empêche un attaquant d'imiter un IdP grâce à une attaque d'hameçonnage classique (Webopedia, 2011). Ces problèmes peuvent être en partie abordés par des mécanismes d'agrégation d'attributs tels que ceux implémentés par Shintau (voir Section 2.5.5). En effet, même s'il est vrai qu'un IdP sera toujours une cible intéressante pour un attaquant, si un sujet distribue ses attributs à travers plusieurs IdPs, une brèche dans l'un d'eux est potentiellement moins désastreuse, car elle ne permet pas nécessairement à un attaquant d'avoir accès à tous les attributs d'un sujet. Toutefois, l'agrégation d'attributs, du moins, l'implémentation de ce concept par Shintau, ne peut pas régler la vulnérabilité où un IdP malveillant se fait passer pour un sujet ou celle où un attaquant à obtenus les justificatifs d'un sujet par hameçonnage. Un IdP malveillant pourra effectivement toujours émettre

une assertion d'authentification valide pour un sujet et acceptée par un SP, permettant de déclencher le processus d'agrégation des attributs et ultimement d'accéder à un service. En ce qui concerne les justificatifs obtenus par hameçonnage, ils permettent également de déclencher le processus d'agrégation. Ce type de problème pourrait faire l'objet de travaux de recherche futur.

Parties justifiables Chaque entité participant à une transaction d'identité doit avoir sa raison d'être. Grâce à l'*Élément de solution B*, aucune entité n'est forcée d'utiliser les services d'un IdP envers qui elle ne fait pas confiance pour correctement gérer ses informations d'identité. Des organisations désirant partager des ressources peuvent mutuellement consentir à utiliser n'importe quel IdP ou même agir en tant que tel si nécessaire.

Identité dirigée L'hypothèse derrière cette loi est qu'un sujet ne désire pas être uniquement et globalement identifié et ne souhaite pas que les fournisseurs de services soient en mesure que corréler les informations qu'ils ont sur lui. Étant donné que l'IdP que nous utilisons, celui du projet Shibboleth (voir Section 5.2.1), permet d'assigner un identifiant unique à un sujet pour chaque SP qu'il accède et même pour chaque transaction chez un même SP, un sujet peut être complètement anonyme au besoin. Non seulement il serait très difficile pour plusieurs SPs de comploter afin de réunir les informations partielles qu'ils ont sur un sujet, mais il serait aussi difficile pour un SP de reconnaitre un unique sujet de visite en visite.

Mentionnons aussi que les choix technologiques effectués et décrits à la Section 5.2 permettent de suivre la loi *pluralisme des opérateurs et des technologies*. En effet, l'utilisation de standards reconnus rend possible l'interopérabilité entre différentes implémentations de notre solution. En ce qui concerne les deux autres lois, *l'intégration humaine* et *l'expérience cohérente entre les contextes*, elles ne sont pas applicables. Ces lois traitent d'un aspect qui n'est pas abordé par notre solution, les interactions entre un utilisateur humain et un système d'identité.

5.2 Choix technologiques et implémentation

Cette section décrit l'implémentation concrète de la solution à l'intérieur du IaaS Framework en faisant un compte rendu des différents standards et technologies utilisés.

Les deux standards permettant la création de fédération qui répondent à nos besoins sont SAML et WS-Federation (voir Section 2.4.2). Un système basé sur le Métasystème d'identité n'aurait pas été approprié, car cette solution est beaucoup trop centrée sur les interactions avec un utilisateur humain. Dans le IaaS Framework, il est fréquent qu'une instance accède à une autre sans aucune intervention humaine. Par exemple, le scénario décrit dans l'introduction de la Section 4.2 pourrait se produire de manière totalement automatique. Des mesures prises chez VO3 pourraient détecter que des ressources devraient être récupérées chez VO1, le processus d'authentification et d'autorisation pourrait avoir lieu et les ressources échangées, le tout, sans aucune intervention humaine. Le Métasystème d'identité n'a pas été conçu pour ce type de cas d'utilisation.

SAML possède des avantages par rapport à WS-Federatio. Par exemple, WS-Federation dépend de plusieurs autres standards de la série WS-* pour l'ensemble des ses opérations. Ces standards sont WS-Trust, WS-Policy, WS-SecurityPolicy, WS-Eventing, WS-Transfer et WS-ResourceTransfer. Certains de ceux-ci sont régis par des organismes de standardisation, mais à ce jour, WS-Transfert et WS-Eventing n'ont toujours pas été approuvé (Gong, 2007; W3C, 2011).

Somme toute, notre critère de sélection principal a été fait sur la base des implémentations disponibles de ces standards. Notre choix s'est arrêté sur SAML, car nous avons trouvé un IdP (voir Section 5.2.1) et un SP (voir Section 5.2.2 pouvant être facilement intégrés dans le IaaS Framework.

Un autre point important à noter est le fait que la plupart des accès au IaaS Framework sont faits par des clients intelligents utilisant des services Web et non des fureteurs Web utilisant des sites Web. Par conséquent, le profil SAML ECP, introduit à la section 2.4.1.1, doit être utilisé.

Le système de gestion de métadonnées (OASIS, 2005a) de SAML vient aussi simplifier l'administration de fédérations dynamiques. Yan *et al.* (2009) explique qu'un des problèmes avec les services d'authentification et les interactions interdomaines traditionnelles vient du fait que préalablement à ce que deux domaines puissent communiquer ensemble, ils doivent être explicitement enregistrés l'un à l'autre. Par conséquent, lorsqu'une fédération grossit, le nombre d'enregistrements nécessaire augmente de manière quadratique. D'un autre côté, avec le système de gestion des métadonnées de SAML, toute l'information reliée aux IdPs et aux SPs d'une fédération peut être centralisée à un seul endroit accessible à toutes les entités. Ainsi donc, lorsqu'une nouvelle organisation rejoint la fédération, ses SPs et IdPs peuvent être configurés pour se référer à cette source centrale de métadonnées qui peut être mise à jour dynamiquement. Ce référentiel de métadonnées contient de l'information sur les différentes entités dans une fédération, telle que leurs noms, leurs clés publiques, l'adresse où elles peuvent être rejointes, etc.

Les sous-sections suivantes décrivent les implémentations choisies pour l'IdP, le SP, et pour les mécanismes d'application des politiques.

5.2.1 Le fournisseur d'identité

La première interaction dans le flot de contrôle d'accès de la solution proposée a lieu lorsqu'un sujet désire s'authentifier à l'IdP qui gère son identité afin de récupérer un justificatif contenu dans un jeton SAML. Ce processus est représenté par la première étape de la Figure 2.2.

Pour ce premier composant, le fournisseur d'identité de Shibboleth (voir Section 2.5.3) a été utilisé. Pour les raisons énoncées dans l'introduction de la section 5.2, un IdP SAML supportant le profil SAML ECP était nécessaire. L'IdP Shibboleth satisfait ce besoin (Internet2, 2011b). Par conséquent, aucune modification au code ne fut nécessaire afin d'adapter l'IdP à nos besoins.

Toutefois, mentionnons que le IaaS Framework stocke toutes ses informations persistantes dans une base de données OrientDB (OrientDB Team, 2011). Parmi ces informations, on retrouve les

attributs sur les sujets étant gérés par l'IdP. Or, cette source de données n'a pas été prévue par les concepteurs de l'IdP Shibboleth. Pour cette raison, une extension à l'IdP a dû être développée afin de lui permettre d'accéder à des attributs provenant de cette nouvelle source de données. Cette extension a été rendue disponible à la communauté des utilisateurs de Shibboleth (Tellier, 2011).

En outre, l'IdP a dû être réorganisé afin de suivre la philosophie du IaaS Framework. Le IaaS Framework est une application OSGi (OSGi Alliance, 2011) écrite dans le langage Scala et exécutée dans le conteneur OSGi Virgo (The Eclipse Foundation, 2011b) hébergeant plusieurs paquets applicatifs ou *bundles*. Ces paquets peuvent être activés et désactivés à partir d'une interface standard, en fonction des besoins de l'environnement. Tel qu'il a été expliqué à la Section 5.1.1, la fonctionnalité *IdP* est optionnelle. Or, il doit être possible de l'activer et de la désactiver à volonté en utilisant l'interface OSGi. Comme cette possibilité n'a pas été prévue par les développeurs de Shibboleth, nous avons dû l'encapsuler dans un paquet applicatif OSGi. Voici les étapes nécessaires à l'encapsulation :

a. Extraction de tout le code et de toutes les librairies de Shibboleth. Ces éléments ont ensuite été réorganisés et compressés dans un paquet applicatif OSGi.

b. Création d'un nouveau paquet applicatif OSGi contenant les fichiers de configuration de l'IdP. Lorsque ce paquet est installé, il va lire certaines variables d'environnement du conteneur Virgo relatives à la configuration de l'IdP, extrait les fichiers de configuration de l'IdP à l'endroit approprié et les modifie en fonction des variables d'environnement précédemment lues.

c. Création d'un plan Virgo. Celui-ci permet de lier les deux paquets applicatifs dans une seule unité.

5.2.2 Le fournisseur de services

Une fois qu'un sujet est authentifié par son IdP et qu'il est en possession d'un jeton SAML, il le présente à un SP afin d'avoir accès à des ressources protégées. Il s'agit de la troisième étape à la Figure 2.2.

Tout comme c'était le cas pour l'IdP, l'implémentation du SP devait supporter le profil SAML ECP et devait pouvoir être intégrée dans le conteneur OSGi et dans le IaaS Framework. Avec son addition *Security Extension* (SpringSource, 2011), le projet *Spring Security* était adéquat en ce qui concerne les contraintes d'intégration. En effet, ce projet est implémenté en tant que *Servlet Filter* (Oracle, 2010) pouvant être utilisé par un conteneur de *Servlets* afin d'intercepter et de traiter toutes les requêtes qui lui sont envoyées. Étant donné que Virgo peut agir en tant que conteneur de *Servlets*, il est simple d'y intégrer le projet *Spring Security Extensions*. Ce projet avait toutefois une lacune, celle de ne pas supporter le profil SAML ECP. Or, étant donné qu'il s'agit d'un projet au code source ouvert, nous avons pu l'améliorer afin de lui faire supporter ce profil. Notre contribution au projet a été intégrée à la principale arborescence du code source.

Dans le IaaS Framework, le SP a la responsabilité d'extraire les informations provenant du jeton SAML émis par l'IdP. Cette information est ensuite récupéré par les modules responsables des décisions et applications du contrôle d'accès. Le SP n'applique pas immédiatement la décision de contrôle d'accès afin de permettre une plus grande flexibilité et une plus grande exactitude dans les décisions. Si la décision de contrôle d'accès était prise et appliquée au moment où une requête est reçue, la décision ne pourrait être prise qu'avec les informations disponibles au moment de la réception de la requête. Or, l'exécution d'une requête peut prendre du temps et impliquer plusieurs actions sur différentes ressources. Il est donc possible que l'état du système change entre le moment où une requête est reçue par le SP et le moment où une certaine action est effectuée sur un objet quelconque. Cette modification dans l'état du système pourrait très bien avoir un impact sur une décision de contrôle d'accès. Pour cette raison, nous avons

décidé de prendre les décisions de contrôle d'accès et de les appliquer le plus tard possible, tel que décrit à la Section 5.2.3.

5.2.3 Application et décision de contrôle d'accès

Les dernières opérations dans le flot de contrôle d'accès de la solution proposée ont lieu lorsque les modules reliés aux politiques disposent de toute l'information nécessaire pour permettre ou refuser l'accès à une ressource. Ce sont ces modules qui décident ultimement si une requête est permise ou non. Leurs interactions sont représentées à la Figure 5.3.

Les mécanismes reliés à la gestion des politiques à l'intérieur du IaaS Framework ont été implémentés en suivant le standard XACML.

PEP Le PEP a été implémenté en tant que *Trait* Scala (Odersky *et al.*, 2008). Dans le langage de programmation Scala, il est possible de créer une classe en combinant des champs et des méthodes provenant de plusieurs *Traits*. Ceci est similaire au concept de l'héritage multiple, avec quelques différences concernant les détails. Lorsqu'un développeur utilise le IaaS Framework pour créer sa solution et qu'il désire protéger une opération sur une classe quelconque, il doit incorporer le *Trait* PEP dans sa classe. Ensuite, avant tout appel de code effectuant des actions délicates, il peut s'assurer qu'un appel à la méthode *checkSecurity* du PEP est réalisé. Cette méthode prend deux paramètres : le justificatif extrait du SP et l'action à effectuer. Le PEP crée ensuite une requête XACML avec ces données et toute autre information qu'il peut récupérer du PIP. La requête XACML est ensuite acheminée au PDP afin qu'il prenne une décision d'autorisation. Si l'autorisation échoue, le PEP arrête le processus d'exécution.

PDP Le PDP se devait d'offrir des performances raisonnables et de suivre la spécification XACML de très près. Enterprise-java-xacml (Wang, 2011) satisfait ces besoins.

Le PDP enterprise-java-xacml a été encapsulé dans un *Acteur* Scala (Odersky *et al.*, 2008). Un *Acteur* Scala permet à des entités distinctes de s'envoyer des messages. Notre *Acteur* PDP possède sa propre boite aux lettres dans laquelle le PEP envoie des messages

concernant des requêtes d'autorisation. L'*Acteur* PDP est ensuite capable de répondre à ces requêtes en les dirigeant vers le PDP enterprise-java-xacml qu'il encapsule. En fin de compte, c'est le PDP enterprise-java-xacml qui prend la décision d'autorisation en évaluant la requête avec ses politiques.

Référentiel de politiques Tel qu'indiqué dans la Section 5.2.1, le IaaS Framework utilise une base de données OrientDB afin de conserver les différentes informations dont il a besoin. Parmi ces informations, on retrouve les politiques de sécurité.

Le PDP enterprise-java-xacml est seulement en mesure de charger des politiques à partir de fichiers XML. Il ne peut donc pas communiquer avec une base de données OrientDB. Pour régler ce problème, nous avons conçu un service intermédiaire qui charge les politiques de la base de données et les écrits dans des fichiers XML prêts à être lus par enterprise-java-xacml.

PIP Le PIP est l'entité interrogée par le PEP afin d'avoir l'information nécessaire pour construire une requête XACML complète qui peut être analysée par le PDP. Le PIP a été implémenté en tant qu'*Acteur* Scala qui reçoit des messages du PEP et qui est en mesure de fournir de l'information sur l'environnement des instances du IaaS Framework et sur les ressources qu'elles hébergent.

CHAPITRE 6

INTÉGRATION DE L'AGRÉGATION DES ATTRIBUTS

Tel qu'il est expliqué à la Section 2.5.5, l'agrégation des attributs est un concept simplifiant grandement la gestion des VOs dynamique. Cette technique permet à un sujet de faire affaire avec plusieurs IdPs pour la gestion de ses attributs. Ceci peut grandement simplifier à gestion des attributs lorsqu'un sujet est membres de plusieurs VOs ou bien lorsqu'il en rejoint et en quitte fréquemment. Dans ce cas, il peut être non seulement compliqué pour un seul IdP de gérer tous les attributs qu'un sujet utilise dans le contexte de ces VOs, mais il est aussi possible que l'IdP ne soit même pas une source d'autorité pour ces attributs. Comme notre solution vise à permettre la création de telles VOs dynamiques, l'agrégation des attributs pourrait être très utile dans le IaaS Framework. Toutefois, l'organisation des priorités et la gestion du projet on fait en sorte qu'elle n'y ait pas été implémentée. Voici une courte analyse expliquant les étapes à suivre pour ajouter des fonctionnalités d'agrégation d'attributs au IaaS Framework grâce à Shintau.

6.1 IdP

Les modules fournissant les fonctionnalités de fournisseur d'identité dans le projet Shintau consistent en une version modifiée de l'IdP Shibboleth lié à une version modifiée du service de découverte IDWSF de la Liberty Alliance (Liberty Alliance, 2006). La spécification du service de découverte IDWSF défini un EPR *(WSA Endpoint Reference)* (W3C, 2006b), une référence pouvant être utilisée pour contacter des instances de services d'identité. Quant à un service d'identité, il est décrit par la spécification comme un service Web maintenant des données concernant un sujet ou effectuant des actions au nom d'un sujet.

Le service de découverte IDWSF de l'IdP peut être questionné pour savoir quel EPR contacter afin d'effectuer une requête à l'IdP pour une assertion d'attributs SAML concernant un sujet (voir Section 2.4.1). Le service de découverte est interrogé par le service de liaison lors de son processus visant à récupérer les attributs d'un sujet auprès de ses IdPs. Le service de découverte

IDWSF effectue aussi la correspondance entre un PID et l'identifiant pseudoaléatoire utilisé pour identifier, de manière unique, l'ensemble des assertions d'attributs pour un sujet, émises par plusieurs IdPs (voir Section 2.5.5). Il est possible de demander à l'IdP ayant émis cet identifiant pseudoaléatoire de récupérer des attributs supplémentaires sur un sujet à n'importe quel moment. Cet IdP doit donc être en mesure de faire correspondre cet identifiant avec le PID qu'il utilise à l'interne afin de récupérer les attributs du sujet. Toutes les informations nécessaires à l'accomplissement de ces responsabilités n'ayant pas été initialement prévues par Shibboleth sont conservées dans une base de données PostgreSQL. Ces données comprennent aussi les références vers les services de liaisons des sujets gérés par l'IdP. Rappelons qu'à la section 2.5.5 il est expliqué que lors du processus d'enregistrement à un service de liaison, la requête transmise aux IdPs contient une référence vers le service liaison du sujet.

La Figure 6.1 illustre l'architecte de l'IdP de Shintau.

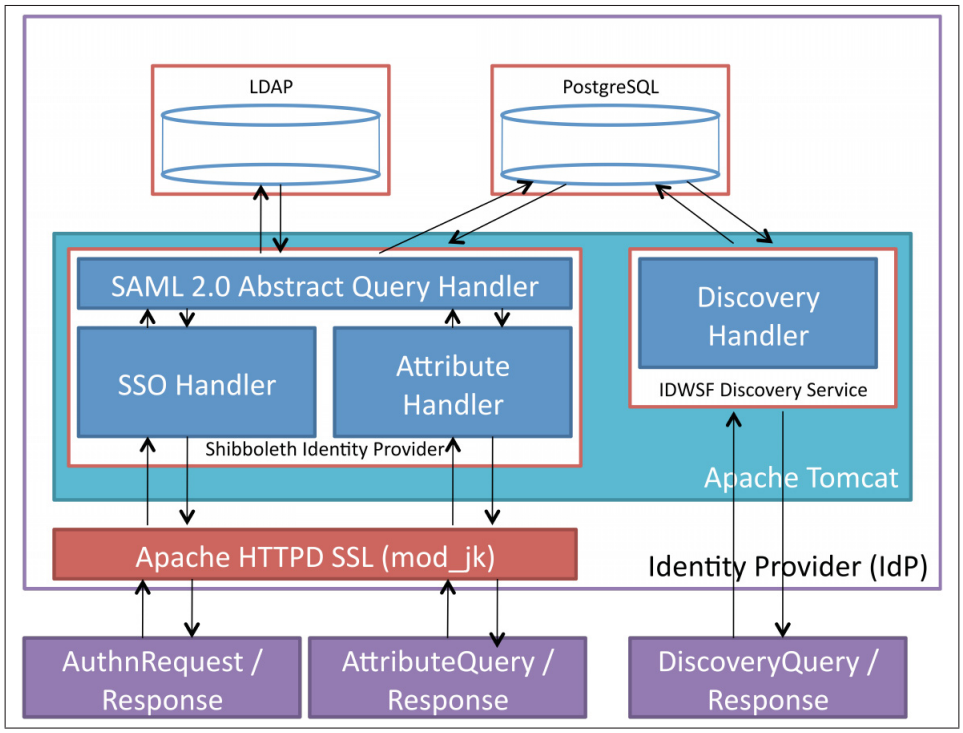

Figure 6.1 Composants de l'IdP Shintau
Tiré de Inman (2009)

La première modification nécessaire à cet IdP afin qu'il puisse être intégré au IaaS Framework consiste en le remplacement de la base de données PostgreSQL par des technologies déjà présentes dans le IaaS Framework. En effet, comme il est qu'indiqué à la Section 5.2.1, le IaaS Framework utilise une base de données OrientDB afin de stocker toutes ses informations persistantes. Les correspondances entre les PIDs et les identifiants locaux devraient donc elles aussi y être entreposées. Ceci implique premièrement que la structure de la base de données OrientDB soit adaptée afin qu'elle puisse remplacer la base de données PostgreSQL utilisée dans Shintau. L'architecture de cette base de données est décrite dans Inman (2009) et pourrait aisément être reproduite dans la base de données OrientDB utilisée par le IaaS Framework. Ensuite, des classes ont été ajoutées à l'IdP et au service de découverte IDWSF afin de communiquer avec la base de données PostgreSQL. Ce code devrait être adapté afin de faire des appels à une base de données OrientDB. Les développeurs de Shintau ont prévu que la base de données PostgreSQL pourrait être remplacée par une autre source de données. Pour cette raison, ils ont isolé tout le code communiquant avec celle-ci dans deux interfaces Java : `PersistantIDStorage` et `PersistentIDNameMapping`. La substitution de la base de données PostgreSQL par la base de données OrientDB utilisée dans le IaaS Framework implique donc l'implémentation de ces interfaces avec des appels vers OrientDB.

Par ailleurs afin que l'IdP Shintau suive la philosophie du IaaS Framework, il devrait être encapsulé dans un paquet OSGi. Le processus permettant de réaliser ceci est décrit à la Section 5.2.1. Le service de découverte IDWSF devrait lui aussi être encapsulé dans un paquet OSGi et tous les paquets devraient être liés dans un unique plan Virgo.

Notons finalement que dans Shintau, l'interface d'authentification Web de l'IdP contient un mécanisme permettant au sujet de spécifier s'il désire activer l'agrégation d'attributs. Comme il est expliqué à la Section 2.5.5, ceci indique à l'IdP qu'il doit inclure une référence vers le service de liaison du sujet ainsi que son PID, dans l'assertion qu'il émet à l'intention du SP. Or, comme nous utilisons le profil SAML ECP (voir Section 2.4.1.1), le sujet interagit avec l'IdP par l'entremise d'un client intelligent et non en utilisant son interface Web. Le mécanisme

utilisé par le client ECP pour s'authentifier auprès de l'IdP devrait donc être augmenté d'un paramètre permettant d'activer ou non l'agrégation des attributs.

6.2 Service de liaison

Le service de liaison a deux responsabilités qui sont réalisées par deux composants distincts. La première responsabilité consiste en la liaison des identités d'un sujet à travers plusieurs IdPs. Cette étape doit être réalisée manuellement par le sujet préalablement à ce que l'agrégation des attributs ait lieu. La seconde responsabilité est la liaison des entités chez différents IdPs, afin qu'un SP puisse autoriser un sujet sur la base d'attributs provenant de plusieurs sources.

La liaison des différentes entités est réalisée par un service de découverte IDWSF. Il s'agit en fait du même composant logiciel qui est utilisé par l'IdP Shintau. Cette fois-ci par contre, il détermine quels sont les EPRs associés à des IdPs qui devraient être questionnés afin d'obtenir l'ensemble des attributs sur un sujet.

L'association entre ces ERPs et le sujet est manuellement réalisé par le sujet lui-même grâce à une interface Web. Cette interface est une application PHP communicant avec un SP Shibboleth afin de récupérer de l'information sur l'identité du sujet et les IdPs les gérant. Cette information est ensuite enregistrée dans une base de données PostgreSQL et est subséquemment accédée par le service de découverte IDWSF afin qu'il puisse être en mesure de créer sa table de liaison (voir Section 2.5.5). Le SP Shibboleth est aussi le composant qui permet au sujet d'être redirigé vers ses IdPs pour s'y authentifier. La Figure 6.2 démontre les différents modules du service de liaison Shintau.

Étant donné que le service de découverte IDWSF est le même que celui utilisé par l'IdP Shintau, les modifications décrites à la Section 6.1 sont toujours valables. Le paquet OSGi mentionné à la Section 6.1 et contenant le service de découverte IDWSF pourrait être réutilisé pour le service de liaison. De la même manière, la base de données PostgreSQL devrait être remplacée par la base de données OrientDB utilisée par le IaaS Framework, tel qu'exposé à la Section 6.1

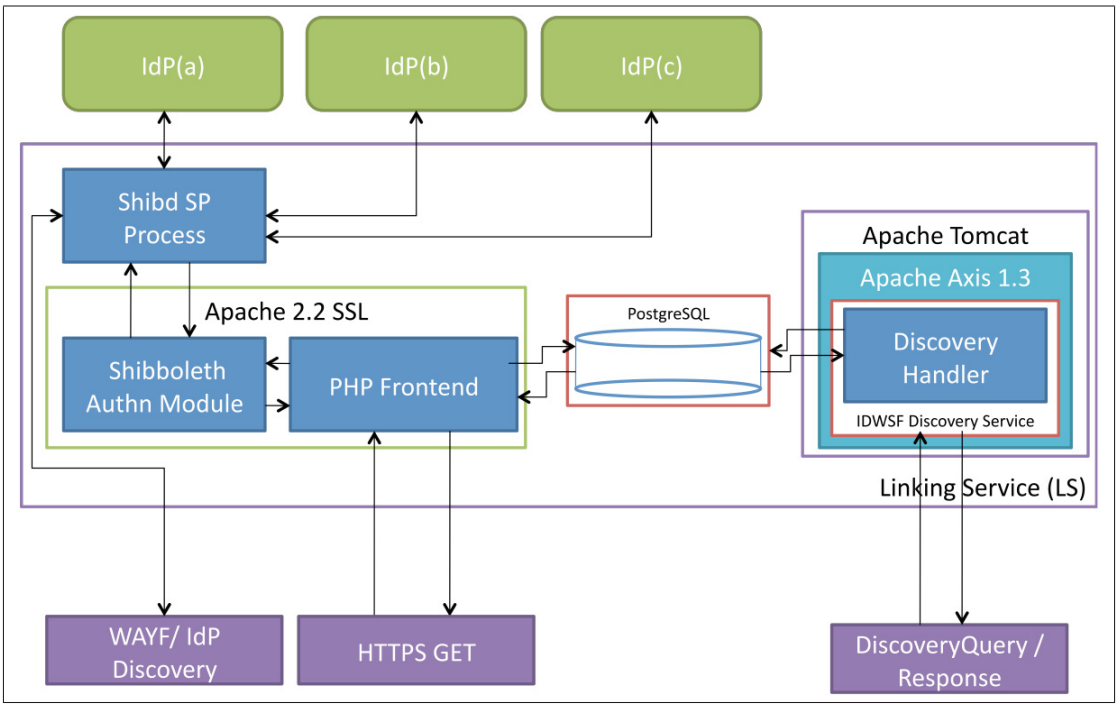

Figure 6.2 Composants du service de liaison Shintau
Tiré de Inman (2009)

En ce qui concerne l'application PHP utilisée par le sujet pour lier ses différentes identités, elle devrait être complètement réécrite. En effet, le IaaS Framework n'est pas en mesure d'héberger d'applications PHP. Il serait donc nécessaire de concevoir une application Web Java ou Scala reproduisant les fonctionnalités de l'application PHP. En outre, le SP Shibboleth, implémenté en tant que module du serveur Web Apache ne peut pas être utilisé. Pour le remplacer, le SP du projet *Spring Security Extensions* (voir Section 5.2.2) serait adéquat, car il constitue une implémentation complète d'un SP SAML.

6.3 SP

Les composants utilisés par le fournisseur de services et leurs interactions dans un déploiement de Shintau suivent le même modèle que ce qui est implémenté dans le IaaS Framework. Premièrement, le SP Shibboleth est utilisé pour protéger des ressources hébergées à l'intérieur d'un serveur Apache en interceptant toutes les requêtes faites vers celles-ci. De son côté, le

IaaS Framework utilise le projet *Spring Security Extensions* pour intercepter les requêtes faites vers le conteneur de *Servlets* Virgo (voir Section 5.2.2).

Ensuite, dans un déploiement de Shintau, le processus d'autorisation est délégué au système PERMIS (Chadwick *et al.*, 2008), un système d'autorisation complexe comprenant plusieurs modules pouvant réaliser les responsabilités d'un PEP, PIP, PDP et PAP (voir Section 2.4.3.3). En ce qui concerne le IaaS Framework, l'application du contrôle d'accès est déléguée au PEP Scala qui récupère des attributs du PIP afin de bâtir une requête d'autorisation à envoyer au PDP.

Finalement, dans un déploiement Shintau, entre le SP Shibboleth et PERMIS, se retrouve le module Apache SAAM. Celui-ci récupère les données traitées par le SP Shibboleth et les achemine à PERMIS afin d'avoir une décision d'autorisation. En ce qui concerne le IaaS Framework, les données traitées par le SP sont directement récupérées par le PEP.

La Figure 6.3 illustre les différents composants d'un SP Shintau.

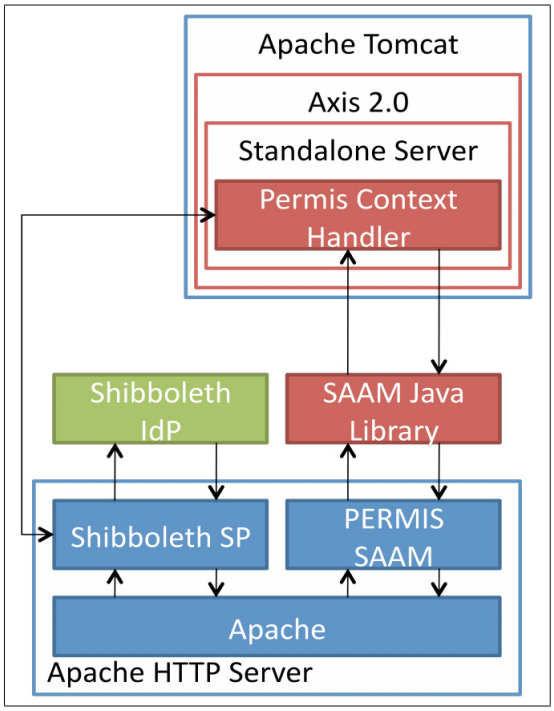

Figure 6.3 Composants du SP Shintau
Tiré de Inman (2009)

Les deux architectures suivent le même principe, mais les technologies utilisées sont différentes et difficilement adaptables. De plus, étant donné que PERMIS utilise son propre format de politique, l'ensemble des politiques XACML déjà en place devrait être réécrit si l'on venait à utiliser ce système. L'approche à privilégier serait donc de reproduire les fonctionnalités d'agrégation présentes dans PERMIS, vers le PEP et le PIP Scala utilisé par le IaaS Framework. C'est-à-dire que le PEP pourrait initialement suivre le processus d'autorisation standard. Ensuite, dans le cas d'une autorisation échouant, le PEP pourrait vérifier si le SP lui a transmis une référence vers un EPR pointant vers le service de liaison où le sujet fait affaire, ainsi que le PID du sujet. Rappelons qu'étant donné que l'IdP connait le service de liaison avec qui un sujet fait affaire, il est en mesure d'inclure la référence vers son EPR et son PID dans les assertions qu'il fournit au SP. Si une telle référence est présente, le PEP pourrait la transmettre au PIP en lui demandant de récupérer les attributs additionnels concernant le sujet. Le PIP pourrait alors faire une requête d'attributs SAML au service de liaison identifié par le EPR. Le service de liaison ferait alors son travail de récupérer les attributs du sujet auprès de ses différents IdPs. Ces attributs pourraient ensuite être retournés au PIP, qui les transmettrait au PEP. Ce dernier, bénéficiant maintenant d'information additionnelle, pourrait transmettre une nouvelle requête d'autorisation au PDP en y incluant ces données.

Les modifications nécessaires au PEP se résument donc à :

a. Après une première requête d'autorisation refusée auprès du PDP, vérifier si une référence vers un EPR se trouve parmi les attributs transmis par le SP vers le PEP.

b. Si une telle référence est présente, la transmettre au PIP en lui demandant de la questionner afin de récupérer les attributs additionnels du sujet.

c. Effectuer une seconde requête d'autorisation auprès du PDP, en y incluant les attributs additionnels reçus du PIP.

d. Appliquer la réponse de cette seconde requête d'autorisation

En ce qui concerne le PIP, les modifications se résument à :

a. Lorsqu'on lui demande de récupérer des attributs concernant un sujet donné en spécifiant une référence vers un EPR, il devrait être en mesure d'effectuer une requête d'attributs SAML auprès de ce EPR, pour le sujet.

CONCLUSION

Ce mémoire aborde le problème qu'ont la plupart des plateformes actuelles de nuages informatiques qui empêche les fournisseurs de ressource de se regrouper en fédérations. Cette possibilité peut être spécialement utile aux petits fournisseurs disposant d'une infrastructure physique limitée. Grâce aux fédérations, un ensemble d'organisations peuvent réunir leurs ressources afin de collaborer à l'atteinte d'un objectif commun ou bien afin de bâtir un nuage informatique de taille assez importante pour rivaliser avec de plus gros joueurs. De plus, de par la nature des fédérations, les ressources ne sont pas contraintes à un seul fournisseur, mais plutôt partagées entre ceux-ci. Cela réduit les possibilités qu'un client soit verrouillé à un fournisseur. Le IaaS Framework est une plateforme permettant de telles possibilités. En outre, contrairement aux autres solutions de nuages informatiques disponibles telles que OpenStack, OpenNebula ou Eucalyptus, cette plateforme IaaS ne se limite pas à la gestion de machines virtuelles. Elle peut en effet être utilisée pour gérer toute sorte de ressources virtuelles telles que des routeurs, commutateurs, des matrices de disque, etc. Ce mémoire décrit l'architecture de gestion de l'identité du contrôle d'accès du IaaS Framework.

Le IaaS Framework possède des fonctionnalités uniques en termes de gestion de l'identité et de contrôle d'accès. Premièrement, il gère l'identité d'une manière fédérée et hiérarchique, ce qui permet la création de VOs dynamiques et de nuages informatiques constitués d'autres nuages informatiques. La solution comporte un IdP externe, contrôlé par un tiers, ce qui permet de centraliser l'identité des sujets d'une fédération à un seul endroit. Par contre, étant donné que n'importe quel fournisseur de service peut aussi agir en tant qu'IdP, il est également possible de créer des structures hiérarchiques et des relations de confiances dynamiques.

Ensuite, grâce à son modèle de contrôle d'accès ABAC, tout fournisseur utilisant le IaaS Framework peut rédiger des politiques permettant d'autoriser des sujets en fonction de n'importe quel critère désiré. Le modèle ABAC a été développé afin de répondre aux contraintes d'environnements de plus en plus complexes. Il généralise les fonctionnalités des autres modèles de contrôle d'accès tels que ceux basés sur l'identité ou sur les rôles et permet d'appliquer un

contrôle d'accès sur la base de n'importe quel attribut des sujets, des ressources et de l'environnement. De plus, comme les politiques sont centralisés chez un fournisseur, celui-ci peut être assuré que ses politiques sont suivies dans l'ensemble de son domaine. Toujours en ce qui concerne les politiques, étant donné qu'elles ne concernent que les interactions directes entre un fournisseur et ses clients, elles n'ont pas besoin d'être révisées lorsque de nouvelles entités rejoignent ou quittent un environnement dynamique. Toutefois, comme plusieurs administrateurs sont en mesure de gérer les politiques de manière concurrente, des conflits peuvent surgir. En ce moment, il en est de la responsabilité des administrateurs de déceler ce genre de problème, mais dans des travaux futurs, il pourrait être possible de développer un système faisant ce travail automatiquement.

La solution développée aborde aussi trois des lois de Kim Cameron aidant un sujet à conserver le contrôle sur son identité. Une quatrième loi est aussi suivie, celle demandant à ce qu'un système d'identité utilise des protocoles de communication standardisés et supporte de multiples technologies. Étant donné que l'implémentation concrète de la solution s'appuie sur des standards reconnus tels que SAML et XACML elle est de nature ouverte et interopérable.

Les composants techniques utilisés pour bâtir la solution sont multiples. Premièrement, l'IdP Shibboleth a été choisi en tant que fournisseur d'identité, car il répondait à la plupart des exigences. Il a néanmoins été nécessaire d'étendre ses fonctionnalités afin de lui permettre d'accéder aux informations persistantes contenues dans la base de données OrientDB du IaaS Framework. De plus, un nouveau paquet applicatif contenant l'IdP a dû être créé afin de lui permettre d'être déployé dans le IaaS Framework.

Ensuite, en ce qui concerne le SP, le projet Spring Security Extension a été modifié afin de lui faire supporter le profil SAML ECP nécessaire dans le IaaS Framework. Ce profil est utilisé dans un déploiement SAML lorsque les clients désirant s'authentifier sont des clients intelligents et non des fureteurs Web. Rappelons que dans le IaaS Framework, la seule responsabilité du SP est l'analyse des jetons SAML transmis par les sujets, et la transmission des informations contenues dans ces jetons vers les modules responsables du contrôle d'accès.

Finalement, trois modules sont nécessaires à l'établissement des décisions de contrôle d'accès. Premièrement, un PEP qui reçoit les requêtes d'accès. Ce composant logiciel peut être intégré à chaque ressource que l'on désire sécuriser et a été développé dans le langage de programmation du IaaS Framework, Scala. Ce PEP communique avec un PDP enterprise-java-xacml qui a été modifié pour être en mesure de récupérer ses politiques de la base de données OrientDB du IaaS Framework. Il analyse les requêtes qu'il reçoit du PEP et les approuve ou les refuse. Un PIP a aussi été développé. Celui-ci récupère des informations additionnelles pouvant aider le PDP à prendre une décision de contrôle d'accès.

Dans le futur, plusieurs autres pistes de recherche et de travaux pourraient être empruntées. Premièrement, une technique d'agrégation d'attributs telle que celle introduite à la Section 2.5.5 pourrait être intégrée au IaaS Framework. Ce concept pourrait simplifier la gestion de l'identité dans les VO dynamiques en distribuant la responsabilité de la gestion des attributs à plusieurs autorités. À cette fin, une analyse sommaire expliquant comment le projet Shintau pourrait être intégré au IaaS Framework est présentée au Chapitre 6. Une autre matière à recherche est introduite à la Section 5.1.3 et concerne le fait qu'un IdP malveillant peut facilement se faire passer pour un des sujets dont il gère l'identité. Au meilleur de nos connaissances, aucune solution à ce problème n'est présentée dans la littérature.

Finalement, plusieurs des modules et technologies composants notre solution comprennent des mécanismes permettant au sujet d'exercer un contrôle sur sa vie privée. Rappelons par exemple le fait qu'avec le standard SAML, il est possible pour un fournisseur de services de n'être au courant que de certains attributs sur un sujet, mais de quand même réaliser un contrôle d'accès efficace. Ensuite, l'IdP Shibboleth permet à un sujet de décider lesquels de ses attributs peuvent être communiqués à quels fournisseurs de services, et il est même possible pour l'IdP de faire référence à un sujet auprès d'un fournisseur de services en utilisant des pseudonymes variables. La protection de la vie privée n'a toutefois pas été un sujet jugée importante par les gestionnaires du IaaS Framework. Pour cette raison, aucune recherche ou expérimentation poussée n'a été faite dans ce domaine. Nous croyons toutefois qu'il serait judicieux pour le IaaS Framework de considérer plus sérieusement cette facette. Dans le futur,

il serait intéressant d'analyser les besoins en protection de la vie privée afin de déterminer si des mécanismes additionnels devraient être développés ou bien si les composants déjà présents dans les modules de notre solution sont adéquats.

Rappelons qu'une analyse préliminaire des travaux décrits dans ce mémoire a été publiée dans Tellier *et al.* (2010) et qu'un article décrivant les travaux eux-mêmes a été soumis pour publication au *International Journal of Grid Computing and eScience*.

BIBLIOGRAPHIE

Liberty Alliance. 2005. « Liberty Alliance ID-FF 1.2 Specifications ». In *Liberty Alliance Project*. En ligne. <http://projectliberty.org/resource_center/specifications/liberty_alliance_id_ff_1_2_specifications/?f=resource_center/specifications/liberty_alliance_id_ff_1_2_specifications>. Consulté le 11 novembre 2011.

Gergely Alpár, Jaap-Henk Hoepman, et Johanneke Siljee. 2011. « The Identity Crisis. Security, Privacy and Usability Issues in Identity Management ». *Computing Research Repositoty (CoRR)*, vol. abs/1101.0427, p. 1–15.

Amazon. 2010. « Amazon Elastic Compute Cloud (Amazon EC2) ». In *Amazon Web Services*. En ligne. <http://aws.amazon.com/ec2/>. Consulté le 3 octobre 2010.

Amazon. 2011. « Amazon Elastic Compute Cloud API Reference ». In *Amazon Web Services*. En ligne. <http://aws.amazon.com/documentation/ec2/>. Consulté le 11 novembre 2011.

ArticSoft Technologies Limited. 2011. « An Introduction to PKI (Public Key Infrastructure) ». In *ArticSoft*. En ligne. <http://www.articsoft.com/public_key_infrastructure.htm>. Consulté le 11 novembre 2011.

Dominick Baier, Vittorio Bertocci, Keith Brown, Matias Woloski, et Eugenio Pace, 2010. *A Guide to Claims-Based Identity and Access Control : Patterns & Practices*. Microsoft Press, 148 p.

Bandit Team. 2009. « Bandit - Trac ». In *Bandit Project's Code pages*. En ligne. <http://code.bandit-project.org/trac>. Consulté le 11 novembre 2011.

Elisa Bertino et Kenji Takahashi, 2010. *Identity Management : Concepts, Technologies, and Systems*. Artech House Publishers, 196 p.

Karthikeyan Bhargavan, Cedric Fournet, Andrew D. Gordon, et Nikhil Swamy. 2008. « Verified implementations of the information card federated identity-management protocol ». In *Proc. of the third ACM Symposium on Information, Computer and Communications Security (ASIACCS)*. (Tokyo, Japan, March 18-20 2008), p. 123–135. ACM.

Matt Bishop, 2004. *Introduction to Computer Security*. Addison-Wesley Professional, 784 p.

Board of Trustees of the University of Illinois. 2009. « GridShib ». In *GridShib*. En ligne. <http://gridshib.globus.org/>. Consulté le 11 novembre 2011.

Businessweek. 2006. « Jeff Bezos' Risky Bet ». In *Bloomberg Businessweek*. En ligne. <http://www.businessweek.com/magazine/content/06_46/b4009001.htm>. Consulté le 11 novembre 2011.

Kim Cameron. « Introduction to the Laws of Identity ». In *Kim Cameron's Identity Weblog*. En ligne. <http://www.identityblog.com/?p=354>. Consulté le 9 novembre 2011.

Kim Cameron. May 2005. « The laws of identity ». *Microsoft Corp*, p. 12.

Kim Cameron. 2011a. « A "change in user behavior" ». In *Kim Cameron's Identity Weblog*. En ligne. <http://www.identityblog.com/?p=1166>. Consulté le 11 novembre 2011.

Kim Cameron. 2011b. « The Clay Feet of Giants ? ». In *Kim Cameron's Identity Weblog*. En ligne. <http://www.identityblog.com/?p=1167>. Consulté le 11 novembre 2011.

Kim Cameron et Michael B. Jones. 2006. « Design rationale behind the identity metasystem architecture ». p. 11.

CANARIE. 2009. « CANARIE ». In *CANARIE*. En ligne. <http://www.canarie.ca/>. Consulté le 11 novembre 2011.

David. Chadwick et George Inman. may. 2009. « Attribute Aggregation in Federated Identity Management ». *Computer*, vol. 42, n° 5, p. 33 -40.

David Chadwick et George Inman. 2010. « Shintau ». In *Shib-Grid Integrated Authorization (Shintau)*. En ligne. <http://sec.cs.kent.ac.uk/shintau/>. Consulté le 11 novembre 2011.

David Chadwick, George Inman, et Nate Klingenstein. 2007. « Authorisation using Attributes from Multiple Authorities–A Study of Requirements ». In *Proc. of the First Human Capital and Social Innovation Technology Summit - ePortfolio International Conference (HCSIT)*. (Maastricht, Netherlands, Oct. 18-19 2007), p. 16–19. European Institute for E-Learning (EIfEL).

David Chadwick, Gansen Zhao, Sassa Otenko, Romain Laborde, Linying Su, et Tuan Anh Nguyen. 2008. « PERMIS : a modular authorization infrastructure ». *Concurr. Comput. : Pract. Exper.*, vol. 20, n° 11, p. 1341–1357.

David W. Chadwick. 2008. « Federated Id : Foundations of Security Analysis and Design V, LNCS ». (Bertinoro, Italy, Aug. 30-31 2008), p. 96–120. Springer.

Anirban Chakrabarti, 2007. *Grid Computing Security*. Springer, 331 p.

Identity Commons. 2011. « IdCommons ». In *Identity Commons*. En ligne. <http://wiki.idcommons.net>. Consulté le 11 novembre 2011.

Rackspace Cloud Computing. 2011. « OpenStack Manuals ». In *OpenStack Cloud Software*. En ligne. <http://docs.openstack.org/>. Consulté le 11 novembre 2011.

Yuri Demchenko, Mihai Cristea, et Cees de Laat. 2009. « XACML Policy Profile for Multi-domain Network Resource Provisioning and Supporting Authorisation Infrastructure ». In *Proc. of the 10th IEEE International Symposium on Policies for Distributed Systems and Networks (POLICY)*. (London, UK, July 20-22 2009), p. 98 -101. IEEE Computer Society.

Pamela Dingle. 2009. « The Pamela Project ». In *The Pamela Project*. En ligne. <http://pamelaproject.com/>. Consulté le 11 novembre 2011.

Eucalyptus Systems. 2010. « Eucalyptus | Your environment. Our industry leading cloud computing software. ». In *Eucalyptus Cloud Computing Software*. En ligne. <http://www.eucalyptus.com>. Consulté le 11 novembre 2011.

David Ferraiolo et Richard Kuhn. 1992. « Role-Based Access Control ». In *Proc. of the 15th NIST-NCSC National Computer Security Conference*. p. 554–563.

Ian Foster, Carl Kesselman, et Steven Tuecke. 2001. « The Anatomy of the Grid : Enabling Scalable Virtual Organizations ». *International Journal of High Performance Computing Applications*, vol. 15, p. 200–222.

Ian Foster, Yong Zhao, Ioan Raicu, et Shiyong Lu. 2008. « Cloud Computing and Grid Computing 360-Degree Compared ». In *Proc. of the Grid Computing Environments Workshop (GCE)*. (Austin, Texas, Nov. 12-16 2008), p. 1 -10. IEEE Computer Society.

Ian T. Foster, Carl Kesselman, Gene Tsudik, et Steven Tuecke. 1998. « A Security Architecture for Computational Grids ». In *Proc. of the fifth ACM Conference on Computer and Communications Security (CCCS)*. (San Francisco, California, Nov. 2-5 1998), p. 83–92. ACM.

Hubert A. Le Van Gong. 2007. « Deep-dive on SAML 2.0 vs. WS-Federation ». In *C'est la vie !*. En ligne. <http://blogs.oracle.com/hubertsblog/entry/deep_dive_on_saml_2>. Consulté le 11 novembre 2011.

Marc Goodner, Maryann Hondo, Anthony Nadalin, Michael McIntosh, et Don Schmidt. 2007. *Understanding ws-federation*. Technical report. IBM Corporation and Microsoft Corporation, 49 p.

Google. 2011. « Google App Engine - Google Code ». In *Google Code*. En ligne. <http://code.google.com/appengine/>. Consulté le 11 novembre 2011.

IaaS Framework Team. 2010. « IaaS Framework | Infrastructure as a Service made Easy ». In *IaaS Framework*. En ligne. <http://www.iaasframework.com>. Consulté le 11 novembre 2011.

Identity and Access Team. 2011. « Beyond Windows CardSpace ». In *Claims-Based Identity Blog*. En ligne. <http://blogs.msdn.com/b/card/archive/2011/02/15/beyond-windows-cardspace.aspx>. Consulté le 11 novembre 2011.

Information Card Foundation. 2011. « The Information Card Ecosystem ». In *Information Cards*. En ligne. <http://informationcard.net/>. Consulté le 11 novembre 2011.

George Inman. 2009. « Shintau Architecture Design ». En ligne. 48 p. <http://sec.cs.kent.ac.uk/shintau/ShintauArchitectureDesign.pdf>. Consulté le 1 novembre 2011.

Inocybe Technologies. 2007. « Welcome to Inocybe Technologies inc. ». In *Inocybe Technologies*. En ligne. <http://www.inocybe.ca/>. Consulté le 11 novembre 2011.

Internet2. 2009. « Home - Shib-uPortal ». In *Internet2 Wiki*. En ligne. <https://spaces.internet2.edu/display/ShibuPortal/Home>. Consulté le 11 novembre 2011.

Internet2. 2011a. « FlowsAndConfig ». In *Shibboleth 2 Wiki*. En ligne. <https://wiki.shibboleth.net/confluence/display/SHIB2/FlowsAndConfig>. Consulté le 20 décembre 2011.

Internet2. 2011b. « ECP ». In *Shibboleth 2 Wiki*. En ligne. <https://wiki.shibboleth.net/confluence/display/SHIB2/ECP>. Consulté le 11 novembre 2011.

Internet2. 2011c. « Shibboleth in Use ». In *Shibboleth*. En ligne. <http://shibboleth.internet2.edu/shib-in-use.html>. Consulté le 11 novembre 2011.

Internet2. 2011d. « UserConsent ». In *Shibboleth 2 Wiki*. En ligne. <https://wiki.shibboleth.net/confluence/display/SHIB2/UserConsent>. Consulté le 20 décembre 2011.

Internet2. 2011e. « Shibboleth ». In *Shibboleth*. En ligne. <http://shibboleth.internet2.edu/>. Consulté le 11 novembre 2011.

ITU-T. January 2009a. « NGN identity management framework ». En ligne. 34 p. <http://www.itu.int/rec/dologin_pub.asp?lang=e&id=T-REC-Y.2720-200901-I!!PDF-E&type=items>. Consulté le 11 novembre 2011.

ITU-T. 2009b. « X.509 : Information technology - Open systems interconnection - The Directory : Public-key and attribute certificate frameworks ». In *International Telecommunication Union*. En ligne. <http://www.itu.int/rec/T-REC-X.509>. Consulté le 11 novembre 2011.

Mike Jones. 2011. « Personal Reflections on the CardSpace Journey ». In *Mike Jones : Self-Issued*. En ligne. <http://self-issued.info/?p=458>. Consulté le 11 novembre 2011.

Farhang Kassaei. 2011. « CardSpace II : "Change of Authentication Behavior" ». In *Software For All Seasons*. En ligne. <http://softwareforallseasons.blogspot.com/2011/03/cardspace-ii-change-of-authentication.html>. Consulté le 11 novembre 2011.

Carl Kesselman et Ian Foster, 1998. *The Grid : Blueprint for a New Computing Infrastructure*. Morgan Kaufmann Publishers, 675 p.

Nate Klingenstein. 2007. « Attribute Aggregation and Federated Identity ». In *Proc. of the seventh International Symposium on Applications and the Internet Workshops (SAINT)*. (Hiroshima, Japan, Jan. 15-19 2007), p. 26–27. IEEE Computer Society.

Romain Laborde, Michel Kamel, François Barrere, et Abdelmalek Benzekri. 2008. « PEP = Point to Enhance Particularly ». In *Proc. of the 9th IEEE International Workshop on Policies for Distributed Systems and Networks (POLICY)*. (Palisades, New York, June 2-4 2008), p. 93–96. IEEE Computer Society.

Butler W. Lampson. 1969. « Dynamic Protection Structures ». *Proc. of the International Workshop on Managing Requirements Knowledge*, p. 1–27.

Leonard J. LaPadula et D. Elliot Bell. 1973. *Secure computer systems : A mathematical model*. Technical report. MITRE, 31 p.

Liberty Alliance. 2006. « Liberty Alliance ID-WSF 2.0 Specifications including Errata v1.0 Updates ». In *Liberty Alliance Project*. En ligne. <http://projectliberty.org/resource_center/specifications/liberty_alliance_id_wsf_2_0_specifications_including_errata_v1_0_updates>. Consulté le 11 novembre 2011.

Eve Maler et Drummond Reed. 2008. « The Venn of Identity : Options and Issues in Federated Identity Management ». *IEEE Security and Privacy*, vol. 6, n° 2, p. 16–23.

Microsoft. 2011. « Windows Azure Platform | Microsoft Cloud Services ». In *Windows Azure*. En ligne. <http://www.microsoft.com/windowsazure/>. Consulté le 11 novembre 2011.

Microsoft Corporation. 2011. « The Official Microsoft IIS Site ». In *The Official Microsoft IIS Site*. En ligne. <http://www.iis.net/>. Consulté le 11 novembre 2011.

Ruben S. Montero. 2009. « [one-users] OpenNebula Vs Eculayptus ». In *OpenNebula Users Mailing List*. En ligne. <http://lists.opennebula.org/pipermail/users-opennebula.org/2009-July/000551.html>. Consulté le 11 novembre 2011.

Timothy Prickett Morgan. 2010. « NASA and Rackspace open source cloud fluffer ». In *The Register*. En ligne. <http://www.theregister.co.uk/2010/07/19/nasa_rackspace_openstack/>. Consulté le 11 novembre 2011.

Mort Bay Consulting. 2011. « jetty - Jetty WebServer ». In *Codehaus Foundation*. En ligne. <http://jetty.codehaus.org/jetty/>. Consulté le 11 novembre 2011.

Vineela Muppavarapu. 2009. « Semantic and Role-Based Access Control for Data Grid Systems ». Thèse en computer science and engineering, Inde, Wright State University. 125 p.

OASIS. March 2005. « Assertions and Protocols for the OASIS Security Assertion Markup Language (SAML) V2.0 ». En ligne. 86 p. <http://docs.oasis-open.org/security/saml/v2.0/saml-core-2.0-os.pdf>. Consulté le 11 novembre 2011.

OASIS. 2005. « SAML V2.0 Executive Overview ». En ligne. 7 p. <http://www.oasis-open.org/committees/download.php/13525/sstc-saml-exec-overview-2.0-cd-01-2col.pdf>. Consulté le 11 novembre 2011.

OASIS. March 2005a. « Metadata for the OASIS Security Assertion Markup Language (SAML) V2.0 ». En ligne. 43 p. <http://docs.oasis-open.org/security/saml/v2.0/saml-metadata-2.0-os.pdf>. Consulté le 11 novembre 2011.

OASIS. 2005b. « eXtensible Access Control Markup Language (XACML) Version 2.0 ». En ligne. 141 p. <http://docs.oasis-open.org/xacml/2.0/access_control-xacml-2.0-core-spec-os.pdf>. Consulté le 11 novembre 2011.

OASIS. 2006. « Web Services Security : SOAP Message Security 1.1 (WS-Security 2004) ». In *OASIS*. En ligne. <http://docs.oasis-open.org/wss/v1.1/>. Consulté le 11 novembre 2011.

OASIS. 2007. « OASIS WS-Trust Specification ». In *OASIS*. En ligne. <http://docs.oasis-open.org/ws-sx/ws-trust/200512>. Consulté le 11 novembre 2011.

OASIS. 2009. « WS-SecurityPolicy 1.3 ». In *OASIS*. En ligne. <http://docs.oasis-open.org/ws-sx/ws-securitypolicy/v1.3/os/ws-securitypolicy-1.3-spec-os.html>. Consulté le 11 novembre 2011.

OASIS. 2011a. « OASIS Identity Metasystem Interoperability (IMI) TC ». In *OASIS*. En ligne. <http://www.oasis-open.org/committees/tc_home.php?wg_abbrev=imi>. Consulté le 11 novembre 2011.

OASIS. 2011b. « Standards ». In *OASIS*. En ligne. <http://www.oasis-open.org/standards>. Consulté le 11 novembre 2011.

OASIS. 2011c. « SAML V2.0 Enhanced Client or Proxy Profile Version 2.0 Working Draft 04 ». En ligne. 20 p. <http://www.oasis-open.org/committees/download.php/43310/sstc-saml-ecp-v2.0-wd04.pdf>. Consulté le 11 novembre 2011.

Graziano Obertelli. 2010. « Is the live migration surported in Eucalyptus 1.6 ? ». In *Eucalyptus Forums*. En ligne. <http://open.eucalyptus.com/forum/live-migration-surported-eucalyptus-16>. Consulté le 11 novembre 2011.

Martin Odersky, Lex Spoon, et Bill Venners, 2008. *Programming in Scala*. éd. 1re. Artima Inc, 776 p.

Openinfocard Team. 2011. « openinfocard - Open Source Information Card Selector and Relyingparty and Security Token Server Java Library ». In *Google Project Hosting*. En ligne. <http://code.google.com/p/openinfocard/>. Consulté le 11 novembre 2011.

OpenNebula. 2010. « OpenNebula : The Open Source Toolkit for Cloud Computing ». In *OpenNebula.org*. En ligne. <http://www.opennebula.org/>. Consulté le 11 novembre 2010.

OpenNebula Project. 2010. « OpenNebula Position on OpenStack Announcement ». In *blog.OpenNebula.org*. En ligne. <http://blog.opennebula.org/?p=683>. Consulté le 11 novembre 2011.

OpenNebula Project Leads. 2011. « OpenNebula EC2 User Guide 2.2 ». In *OpenNebula.org*. En ligne. <http://opennebula.org/documentation:rel2.2:ec2qug>. Consulté le 11 novembre 2011.

OpenStack. 2011. « OpenStack Open Source Cloud Computing Software ». In *OpenStack Open Source Cloud Computing Software*. En ligne. <http://www.openstack.org/>. Consulté le 11 novembre 2011.

Oracle. 2010. « The Essentials of Filters ». In *Oracle Technology Network*. En ligne. <http://www.oracle.com/technetwork/java/filters-137243.html>. Consulté le 11 novembre 2011.

Oracle. 2011. « Java Servlet Technology ». In *Oracle Technology Network*. En ligne. <http://www.oracle.com/technetwork/java/javaee/servlet/index.html>. Consulté le 11 novembre 2011.

Oracle Corporation. 2010. « Sun Java System Web Server 7.0 NSAPI Developer's Guide ». In *Oracle Wikis*. En ligne. <http://wikis.sun.com/display/WebServerdocs/NSAPI+Developer%27s+Guide>. Consulté le 11 novembre 2011.

OrientDB Team. 2011. « OrientDB ». In *Google Project Hosting*. En ligne. <http://code.google.com/p/orient/>. Consulté le 11 novembre 2011.

OSGi Alliance. 2011. « OSGi Alliance | Main / OSGi Alliance ». In *OSGi Alliance*. En ligne. <http://www.osgi.org/Main/HomePage>. Consulté le 11 novembre 2010.

Gunnar Peterson. 2011. « Cardspace Thoughts ». In *1 Raindrop*. En ligne. <http://1raindrop.typepad.com/1_raindrop/2011/02/cardspace-thoughts.html>. Consulté le 11 novembre 2011.

Bart Priem, Ronald Leenes, Eleni Kosta, et Aleksandra Kuczerawy. 2011. « The Identity Landscape ». In *Digital Privacy*, sous la dir. de Camenisch, Jan, Leenes, Ronald et Sommer, Dieter, p. 33–51. Berlin (Allemagne) : Springer.

Benny Rochwerger, David Breitgand, Eliezer Levy, Alex Galis, Kenneth Nagin, Ignacio Martín Llorente, Rubén Montero, Yaron Wolfsthal, Erik Elmroth, Juan Caceres, Muli Ben-Yehuda, Wolfgang Emmerich, et Fermín Galan. july 2009. « The Reservoir model and architecture for open federated cloud computing ». *IBM Journal of Research and Development*, vol. 53, n° 4, p. 1–11.

Salesforce.com. 2010. « CRM - salesforce.com ». In *Salesforce.com*. En ligne. <http ://www.salesforce.com/>. Consulté le 5 novembre 2010.

Roger R. Schell, Peter J. Downey, et Gerald J. Popek. 1973. *Preliminary notes on the design of secure military computer systems*. Technical report. Electronic Systems Division Air Force Systems Command, 82 p.

Richard O. Sinnott, David W. Chadwick, Thomas Doherty, David Martin, Anthony Stell, Gordon Stewart, Linying Su, et J. Watt. 2008. « Advanced Security for Virtual Organizations : The Pros and Cons of Centralized vs Decentralized Security Models ». In *Proc. of the Eighth IEEE International Symposium on Cluster Computing and the Grid (CCGRID)*. (Lyon, France, May 19-22 2008), p. 106–113. IEEE Computer Society.

Borja Sotomayor. 2005. « Delegation and single sign-on (proxy certificates) ». In *GSI : Grid Security Infrastructure*. En ligne. <http://gdp.globus.org/gt4-tutorial/multiplehtml/ch10s05.html>. Consulté le 11 novembre 2011.

SpringSource. 2011. « Spring Security - The Extensions Project ». In *SpringSource*. En ligne. <http://static.springsource.org/spring-security/site/extensions.html>. Consulté le 11 novembre 2011.

Jack Suess et Kevin Morroney. 2009. « Identity Management and Trust Services : Foundations for Cloud Computing ». *EDUCAUSE Review*, vol. 44, n° 5, p. 24-43.

Sun Microsystems Inc. 2003. « A Brief Introduction to XACML ». In *OASIS*. En ligne. <http://www.oasis-open.org/committees/download.php/2713/Brief_Introduction_to_XACML.html>. Consulté le 11 novembre 2011.

Synchromedia. 2010. « GreenStar Network | Building a Zero Carbon Network ». In *GreenStar Network*. En ligne. <http://www.greenstarnetwork.com/>. Consulté le 11 novembre 2011.

Synchromedia. 2011. « Presentation | Synchromedia ». In *Synchromedia*. En ligne. <http://www.synchromedia.ca/>. Consulté le 11 novembre 2011.

SYS-CON Media. 2008. « Twenty-One Experts Define Cloud Computing ». In *Cloud Computing Journal*. En ligne. <http ://cloudcomputing.sys-con.com/node/612375>. Consulté le 9 juin 2010.

Jonathan Tellier. 2011. « OrientDB Connector ». In *Shibboleth2 Wiki*. En ligne. <https://wiki.shibboleth.net/confluence/display/SHIB2/OrientDB+Connector>. Consulté le 11 novembre 2011.

Jonathan Tellier, Jean-Marc Robert, et Mohamed Cheriet. 2010. « Le contrôle d'accès dans les environnements fédérés ». In *Proc. of the ninth colloque francophone sur la Gestion de Réseaux Et de Services (GRES)*. p. 1-8.

The Apache Software Foundation. 2011a. « Welcome ! - The Apache HTTP Server Project ». In *Apache*. En ligne. <http://httpd.apache.org/>. Consulté le 11 novembre 2011.

The Apache Software Foundation. 2011b. « Apache Tomcat - Welcome ! ». In *Apache*. En ligne. <http://tomcat.apache.org/>. Consulté le 11 novembre 2011.

The Eclipse Foundation. 2011a. « Higgins Home ». In *Higgins Personal Data Service*. En ligne. <http://www.eclipse.org/higgins>. Consulté le 11 novembre 2011.

The Eclipse Foundation. 2011b. « Virgo ». In *Eclipse*. En ligne. <http://www.eclipse.org/virgo/>. Consulté le 11 novembre 2011.

University of Chicago. 2010. « The Globus Alliance ». In *The Globus Alliance*. En ligne. <http://www.globus.org/>. Consulté le 24 août 2010.

J. Vollbrecht, P. Calhoun, S. Farrell, L. Gommans, G. Gross, B. Bruijn, Cees de Laat, et M. Holdrege. Août 2000. *AAA authorization framework*. RFC 2904. Internet Engineering Task Force, 35 p.

W3C. 2001. « Web Services Description Language (WSDL) 1.1 ». In *W3C*. En ligne. <http://www.w3.org/TR/wsdl>. Consulté le 11 novembre 2011.

W3C. 2006a. « Web Services Policy 1.2 - Framework (WS-Policy) ». In *W3C Member Submission*. En ligne. <http://www.w3.org/Submission/WS-Policy/>. Consulté le 11 novembre 2011.

W3C. 2006b. « Web Services Addressing 1.0 - Core ». In *W3C*. En ligne. <http://www.w3.org/TR/2006/REC-ws-addr-core-20060509/>. Consulté le 11 novembre 2011.

W3C. 2011. « All Standards and Drafts - W3C ». In *W3C*. En ligne. <http://www.w3.org/TR/>. Consulté le 11 novembre 2011.

Ppzian Wang. 2011. « enterprise-java-xacml ». In *Google Project Hosting*. En ligne. <http://code.google.com/p/enterprise-java-xacml/>. Consulté le 11 novembre 2011.

Webopedia. 2011. « phishing ». In *Webopedia*. En ligne. <http://www.webopedia.com/TERM/P/phishing.html>. Consulté le 20 décembre 2011.

Rich Wolski. 2009. « OpenNebula Vs Ecalyptus ». In *Eucalyptus Forums*. En ligne. <http://open.eucalyptus.com/forum/opennebula-vs-ecalyptus>. Consulté le 11 novembre 2011.

Liang Yan, Chunming Rong, et Gansen Zhao. 2009. « Strengthen Cloud Computing Security with Federal Identity Management Using Hierarchical Identity-Based Cryptography ». In *Cloud Computing, LNCS*. (Beijing, China, December 1–4 2009), p. 167-177. Springer Berlin / Heidelberg.

Eric Yuan et Jin Tong. 2005. « Attributed Based Access Control (ABAC) for Web Services ». In *Proc. of the third IEEE International Conference on Web Services (ICWS)*. (Orlando, Florida, July 11-15 2005), p. 561-569. IEEE Computer Society.

Ning Zhang, Li Yao, Aleksandra Nenadic, Jay Chin, Carole Goble, Alan Rector, David Chadwick, Sassa Otenko, et Qi Shi. 2007. « Achieving fine-grained access control in virtual organizations : Research Articles ». *Concurrency and Computation : Practice and Experience*, vol. 19, n° 9, p. 1333–1352.

Qi Zhang, Lu Cheng, et Raouf Boutaba. 2010. « Cloud computing : state-of-the-art and research challenges ». *Journal of Internet Services and Applications*, vol. 1, n° 1, p. 7–18.