Numéro d'ordre : 2009-ISAL-0040 **Année 2009**

INSA de Lyon: "Institut National des Sciences Appliquées de Lyon"

INFOMATHS :" INFORMATIQUE ET MATHEMATIQUES"

THÈSE

The Chameleon : Un Système de Sécurité pour Utilisateurs Nomades en Environnements Pervasifs et Collaboratifs.

pour l'obtention du

Grade de Docteur

(spécialité informatique)

par

Rachid Saadi

Soutenue publiquement le 17 juin 2009 devant la commission d'examen composée de:

Président :	Pr Mohand-Said Hacid	Université Claude Bernard Lyon 1, France
Rapporteurs :	Pr Frédéric Cuppens	ENST de Bretagne, France
	Pr Ernesto Damiani	Université Degli Studi de Milan, Italy
Examinateurs :	Dr Philippe Balbiani	Université Paul Sabatier, Toulouse, France
	Dr Jacques Fayolle	Université Jean Monnet, Saint Etienne, France
	Dr Ahmed Serhrouchni	ENST de Paris, France
Encadrant :	Pr Lionel Brunie	INSA de Lyon, France
Co-encadrant :	Pr Jean-Marc Pierson	Université Paul Sabatier de Toulouse, France

Laboratoire d'InfoRmatique en Image et Systèmes d'information — UMR 5205

Je dédie ma thèse
à mon père
qui m'a transmis ses valeurs
et à mon fils Rayan
que j'espère héritera des miennes.

Remerciements

Un moment émouvant pour le doctorant est le jour où il entreprend d'écrire ses remerciements. Tout d'abord, cela signifie que c'est enfin la concrétisation d'une vie d'étude, ce qui en soi est une très bonne nouvelle. Par ailleurs, cela me donne également l'occasion d'exprimer ma gratitude en remerciant toutes les personnes qui m'ont aidé et supporté au cours de ces dernières années. Comme le veut la tradition, je vais entamer le dernier mais non moins le plus difficile exercice de la page des remerciements en tachant de n'oublier personne. C'est pourquoi, je remercie par avance ceux dont le nom n'apparaît pas dans cette page et qui m'ont aidé d'une manière ou d'une autre.

Le premier grand merci ira pour mes encadrants Lionel Brunie et Jean-Marc Pierson qui ont accepté de me suivre depuis le master jusqu'à l'aboutissement de la thèse, de m'avoir promulguer leurs savoirs mais surtout d'avoir supportés ma ténacité afin d'être plus ouvert, tolérant et surtout de me pousser à continuellement m'améliorer. Je tiens encore à leur exprimer ma profonde gratitude pour leurs précieux conseils, ainsi que pour le climat de confiance et de convivialité dans lequel j'ai eu le plaisir de travailler.

Le second merci ira aux professeurs Philippe Balbiani, Frédéric Cuppens, Ernesto Damiani, Mohand-Said Hacid, Jacques Fayolle et Ahmed Serhrouchni pour avoir témoignés de leur intérêt envers mes travaux en acceptant d'être membres de mon jury de thèse. Et plus particulièrement, merci à Frédéric Cuppens et Ernesto Damiani qui m'ont fait l'honneur d'être les rapporteurs de mon manuscrit. Merci pour leurs conseils et leurs remarques constructives qui ont participé à la pertinence et la clarification du mémoire.

Ma plus grande gratitude ira à toute ma famille, tout particulièrement mon épouse qui a été tout le temps à mes côtés, m'a soutenu et avec qui j'ai partagé les meilleurs moments et surtout les pires ; mon père qui représente (et le représentera toujours) mon modèle et l'exemple que je suit ; ma mère qui m'a permis d'être la personne que je suis aujourd'hui ; mes soeurs qui ont toujours cru en moi et qui j'espère ne jamais décevoir. Je tiens également à remercier ma belle famille pour leur soutien et encouragement.

Je tiens à remercier les amis qui représentent, en France, ma seconde famille (ils se reconnaîtront) Imene, Omar, Rim, Samer et Abdallah, ainsi que Mounir, Hakim, Faiza, Bahae, Hiba et Youssef à qui j'ai une pensée particulière.

Je tiens à remercier toute l'équipe de recherche du LIRIS avec qui j'ai pu lier des liens de sympathie, l'ensemble des doctorants et autres chercheurs du labo que j'ai eu la chance de côtoyer Rami, Chérine, Fred et ses Drôle de dames : Reim, Sana, Claudia, mais aussi Sandro, Yann, Uddam, Ny Haingo, Mariane, Dejene, Girma, Julien..., Je tiens également à remercier tout le personnel de l'IUT de Mont Marsan Réseau et télécom pour leur Pins de Mai, tout spécialement Jean Jacque à qui je resterai toujours reconnaissant, Laurent et Sophie pour avoir relue ma thèse, Manu pour la pomme qu'il hésite à croquer mais aussi, Christophe, Angel, Anthony, Béatrice, Majirus, Samuel... Je ne pourrais pas les citer tous, aussi j'espère qu'ils se reconnaîtront.

Résumé

Le partage et la distribution des ressources et des services constituent un défi important de l'informatique moderne. Ainsi, le développement des systèmes distribués vient comme une réponse à des besoins liés à la propagation et à l'acquisition d'informations et de services sans aucune contrainte ni barrière.

L'informatique mobile connaît de nos jours un essor fulgurant, en raison de l'apparition d'une nouvelle classe d'utilisateurs à caractère nomade liée à la création de dispositifs portables capables d'offrir de plus en plus de ressources et de services, communiquant grâce à des infrastructures légères et sans fil. Ainsi, la convergence entre les infrastructures terrestres et mobiles permet aux usagers d'avoir à disposition un réseau très vaste d'informations et de services accessibles quels que soient le lieu et le moment.

L'avènement de l'informatique mobile a fait évoluer les systèmes distribués, en permettant la mise en place d'environnements intelligents et communicants offrant à l'utilisateur la possibilité de faire interagir ses équipements (téléphone, PDA, etc.) avec son environnement de manière aisée et transparente. Cette évolution a abouti au concept d'Informatique pervasive qui, en plaçant l'utilisateur nomade au cœur de l'informatique moderne, a ouvert de nouvelles perspectives théoriques et applicatives.

La mobilité de l'utilisateur et la multiplication des dispositifs légers et autonomes accentuent et rendent plus complexes les problèmes de sécurité. En effet, la mise en place d'environnements pervasifs permet à l'utilisateur nomade de solliciter ou de communiquer avec d'autres utilisateurs, des ressources ou des services au sein d'environnements ou de domaines qui lui sont inconnus et réciproquement.

Les mécanismes et les solutions existants sont inadéquats pour pallier les nouveaux challenges issus principalement des problèmes d'authentification, de contrôle d'accès et de protection de la vie privée. Dans un système aussi fortement distribué et imprédictible, l'existence d'une autorité centralisatrice n'est en effet pas possible. Il est donc nécessaire de rendre la décision d'accès plus autonome et donc répartie. Dans cette thèse, nous défendons l'idée que l'implémentation d'un système de sécurité basé sur la confiance constitue une solution particulièrement adaptée aux environnements pervasifs.

Notre travail à conduit à l'élaboration d'une architecture générique de sécurité prenant en charge l'utilisateur depuis l'identification jusqu'à l'attribution de droits d'accès dans des milieux inconnus. Nommée « Chameleon », celle-ci prend en charge les problèmes issus de l'authentification et du contrôle d'accès dans les environnements mobiles et ubiquitaires. La mise en place de notre approche a nécessité (i) la définition d'un modèle de confiance inter-organisations (T2D) dont le but est d'étendre le champ d'accès des utilisateurs mobiles aux ressources environnantes ; (ii) l'élaboration d'un nouveau format de certificat « X316 » ; (iii) l'implémentation d'un nouveau processus de signature (FeMoS) permettant l'adaptation des documents signés au contexte de l'utilisateur.

Mots-clés: Environnement Pervasif, Authentification, Contrôle d'accès, Protection de la vie privée, Gestion de la confiance, Disposition à la confiance, Certificat XML et Morph signature

Abstract

Users of Information Technologies are somehow contradictory ! On one hand, they want their life easy, so they prefer to access transparently to the large set of information, appliances and devices being in their environment. On the other hand, they want to have a clear understanding of what they access and they want to ensure that only the minimum information about themselves is delivered to third parties. They want to be trusted, but are reluctant to trust.

In our thesis, we explore this contradiction and try to give way to mechanisms that allow a certain balance reachable between paranoia and naiveness.

This goal is large, and we will certainly not address all kind of issues this statement opens. We focus our work on trust management in collaborative distributed environments. They witness a growing interests with the meet of web-enhanced information technologies and wireless devices. The anywhere anytime access to information and services is nowadays a must. All the approaches rely on a kind of trust to establish the necessary regulations for authentication, authorization and access control.

While the trust is easy to set up between the known participants of a communication, the evaluation of trust becomes a challenge when confronted with unknown environment. It is more likely to happen that the collaboration in the mobile environment will occur between totally unknown parties. An approach to handle this situation has long been to establish some third parties that certify the identities, roles and/or rights of both participants in a collaboration. In a completely decentralized environment, this option is not sufficient. To decide upon accesses one prefer to rely only on what is presented to him by the other party and by the trust it can establish, directly by knowing the other party or indirectly, and vice-versa. Hence a mobile user must for example present a set of certificates known in advance and the visited site may use these certificates to determine the trust he can have in this user and thus potentially allow an adapted access. In this schema the mobile user must know in advance where she wants to go and what she should present as identifications. This is difficult to achieve in a global environment. Moreover, the user likes to be able to have an evaluation of the site she is visiting to allow limited access to her resources. And finally, an user does not want to bother about the management of her security at fine grain while preserving her privacy. Ideally, the process should be automatized.

Our work was lead to define the Chameleon architecture. Thus the nomadic users can behave as chameleons by taking the "colors" of their environments enriching their nomadic accesses. It relies on a new T2D trust model which is characterized by support for the disposition of trust. Each nomadic user is identified by a new morph certification model called X316. The X316 allows to carry out the trust evaluation together with the roles of the participants while allowing to hide some of its elements, preserving the privacy of its users and adapting to the trustfulness of the environment.

Keywords: Pervaive environment, authentication, Access control, Privacy, Trust management, Disposition to trust, XML Certificate and Morph signature

Table des matières

Table des figures

Liste des tableaux

Introduction générale

Le partage et la distribution des ressources et des services constituent un défi important de l'informatique moderne. Ainsi, le développement des systèmes distribués vient comme une réponse à des besoins liés à la propagation et à l'acquisition d'informations et de services sans aucune contrainte ni barrière.

D'une part, Internet, qui est le plus grand système distribué à l'heure actuelle, s'étend sur l'ensemble du globe offrant un accès illimité à une quantité incommensurable d'informations et de services.

D'autre part, l'informatique mobile connaît de nos jours un essor fulgurant, en raison de l'apparition d'une nouvelle classe d'utilisateurs à caractère nomade liée à la création de dispositifs portables capables d'offrir de plus en plus de ressources et de services, communiquant grâce à des infrastructures légères et sans fil. Ainsi, la convergence entre les infrastructures terrestres (LAN, fibre optique, etc) et mobiles (GSM, 3G et WIFI) permet aux usagers d'avoir à disposition un réseau très vaste et illimité d'informations et de services accessibles quels que soient le lieu et le moment.

L'avènement de l'informatique mobile a fait évoluer les systèmes distribués, en permettant la mise en place d'environnements intelligents et communicants offrant à l'utilisateur la possibilité de faire interagir ses équipements (téléphone, PDA, etc.) avec son environnement de manière aisée et transparente aboutissant au concept d'Informatique Pervasive. Le développement de ces environnements coopératifs et collaboratifs engendre de nouvelles perspectives mettant l'utilisateur nomade au cœur de l'informatique moderne.

La mobilité de l'utilisateur et la multiplication des dispositifs légers et autonomes accentuent et rendent plus complexes les problèmes de sécurité. En effet, la mise en place d'environnements pervasifs permet à l'utilisateur nomade de solliciter ou de communiquer avec d'autres utilisateurs, des ressources ou des services au sein d'environnements ou de domaines qui lui sont inconnus et réciproquement.

Dans le cadre de notre thèse, on considère [1] que quel que soit le lieu où évolue l'usager et où il veut communiquer, ce lieu est obligatoirement régi par une administration. Chaque administration est ainsi responsable d'identifier les utilisateurs visiteurs et de leur attribuer des droits. Ces derniers offrent au bénéficiaire la possibilité d'utiliser certaines ressources et d'interagir en toute sécurité avec toutes les entités (utilisateurs, ressources ou services) du domaine. Cela implique de mettre en place des mécanismes permettant l'interconnexion des différents domaines, souvent hétérogènes, impliqués dans le système pervasif. L'étude des ces mécanismes fait l'objet de ce manuscrit.

Dans le paragraphe suivant, nous présentons nos motivations et mettons en évidence les problématiques liées aux environnements collaboratifs à caractère pervasif : l'authentification, le

1. C'est le cas d'un réseau constitué d'organisations contrôlées par une autorité administrative (ex : administration publiques, entreprises, universités etc.) et non pas des réseaux de type ad-hoc ou interpersonnels.

contrôle d'accès et la protection de la vie privée.

1 Motivation

L'informatique pervasive constitue le point de convergence des systèmes distribués et des environnements mobiles. Dans la littérature, ce paradigme revêt différentes significations. Une définition de notre vision de ces environnements émergents s'impose donc afin de situer le cadre de notre travail. Nous introduisons par la suite deux cas d'utilisation pour mettre en évidence les besoins indispensables (en terme de sécurité) à la mise en place d'un environnement collaboratif à caractère pervasif.

1.1 Cadre de notre travail

1.1.1 Qu'est ce qu'est "l'Informatique Pervasive" ?

"Pervasive" est un terme anglais qui signifie omniprésent ou envahissant [2]. Ce mot a été utilisé en informatique pour décrire un monde où l'information et les services sont présents partout et tout le temps. Dans la littérature, ce paradigme a suscité différents axes de recherche plus au moins équivalents : "Informatique Pervasive" (Pervasive computing), "Informatique Ubiquitaire" (Ubiquitous Computing) ou "Intelligence Ambiante" (Ambient Intilligence). Tous ces travaux ont cependant pour principal objectif le partage de l'information, des ressources et des services entre les utilisateurs nomades et leur environnement, où et quand ceux-ci en ont besoin et de manière non intrusive.

Mark Weiser, l'un des pionniers dans ce domaine, a initié le principe de l'informatique pervasive en tant que technologie intégrée dans l'environnement jusqu'à devenir invisible aux utilisateurs. *"We are trying to conceive a new way of thinking about computers in the world, one that takes into account the natural human environment and allows the computers themselves to vanish into the background"* [Weis 01].

Les systèmes pervasifs mettent en place de nouvelles techniques issues de l'évolution de certaines architectures existantes. Ils partagent certains principes des systèmes distribués, des grilles de calcul, des réseaux adhoc, des réseaux pair à pair et des réseaux de capteurs. Ces systèmes constituent le point de convergence de l'évolution de ces différents domaines.

Ainsi, pour mieux comprendre le principe même de l'informatique pervasive, nous allons énumérer, dans ce qui suit, les différents challenges [Camp 02] imposés par la mise en place d'applications ou d'environnements pervasifs.

– L'élargissement des frontières : Traditionnellement, l'utilisateur était figé dans son propre environnement de données, de programmes et de ressources. L'apparition de la notion de système distribué a fortement contribué à élargir le champ d'utilisation des ressources à des utilisateurs de plus en plus distants.
  Plus encore, l'informatique pervasive est un domaine de recherche qui va au-delà des infrastructures, en étendant et en facilitant l'accès des utilisateurs nomades à tout ce qui les entoure, offrant ainsi de nouvelles perspectives de développement d'applications communicantes, personnalisées, riches et évolutives.

2. http ://www.wordreference.com/

- La connectivité : Le point essentiel des systèmes pervasifs est de permettre à l'utilisateur d'être toujours accessible quel que soit le lieu ou le moment, et cela :
  - En lui permettant d'interagir avec l'environnement qui l'entoure de manière aussi transparente que possible.
  - En faisant abstraction de l'infrastructure de communication (Bluetooth, Infrarouge, WiFi, etc.).
  - En instaurant une politique de sécurité extensible accordant un accès personnalisé aux utilisateurs des ressources.

- L'hétérogénéité : Les environnements ubiquitaires intègrent une grande diversité de technologies. Ainsi, la difficulté réside dans le fait de créer un système intermédiaire qui fonctionne tel une passerelle, offrant la possibilité à tous les équipements, hétérogènes sur le plan physique et logiciel, de s'interconnecter, de communiquer et d'interagir.

- L'adaptabilité et la proactivité : L'environnement doit offrir une interface personnalisée en fonction du profil de chaque utilisateur. Ceci nécessite une adaptation des services et une disponibilité des ressources en fonction du contexte de l'usager, à savoir ses dispositifs, sa localisation, etc. Ainsi, l'objectif est de répondre au mieux aux attentes des utilisateurs, voire anticiper leurs besoins de manière proactive.

- La simplicité : Une des caractéristiques importantes des systèmes pervasifs est la simplicité d'utilisation. L'informatique ubiquitaire ayant été conçue dans le but d'être déployée partout et d'être accessible par la majorité des utilisateurs (souvent non spécialistes), les dispositifs utilisés ainsi que les logiciels embarqués doivent être maniables et accessibles pour tous. Ainsi, le déploiement de cette technologie doit permettre une prise en main intuitive et naturelle : ***"if it is not easy to use, it will not be used"***. Cependant, la simplicité attendue peut présenter un défi important surtout quand il s'agit de maintenir un équilibre entre accessibilité et sécurité.

1.1.2 Exemples d'utilisation

Dans cette partie, deux cas d'utilisation sont présentés. Ils permettront, dans le cadre de cette thèse, de mettre en évidence les problèmes et les besoins de sécurité liés aux environnements pervasifs.

Le professeur : Bob est professeur à l'université "A", il doit se rendre le matin à l'université "B" pour participer à une conférence et le lendemain à une réunion qui se tiendra à l'université "C" (voir figure 1). Lors de ses déplacements, Bob veut communiquer avec ses collègues (membres d'autres universités) et souhaite, par là-même, solliciter un certain nombre de ressources (ex : un vidéo-projecteur, une imprimante, etc). Afin de permettre à cet usager nomade d'être reconnu localement et d'obtenir les privilèges nécessaires pour utiliser ces équipements locaux, l'autorité régissant l'environnement (dans laquelle se trouve le professeur à un moment donné) lui attribue un profil reflétant son statut (professeur). Ainsi, Bob peut solliciter les ressources auxquelles il a droit et être identifié à l'instar de n'importe quel utilisateur interne.

Selon la philosophie de l'informatique pervasive, l'utilisateur doit acquérir des privilèges de manière dynamique et transparente sans que l'environnement sollicite continuellement son attention. On peut considérer que chaque université est délimitée géographiquement, et a la res-

ponsabilité de toute entité ou ressource communicante se trouvant à l'intérieur de ses locaux. Donc, dès que le professeur accède au site de l'université, son dispositif communique et "négocie", en utilisant la borne la plus proche, avec la politique de sécurité du domaine. Cela aboutit à l'acquisition, de manière dynamique, d'un document dénotant un profil personnalisé (reconnu localement) qui valide les privilèges auxquels Bob a droit au sein de ce domaine.

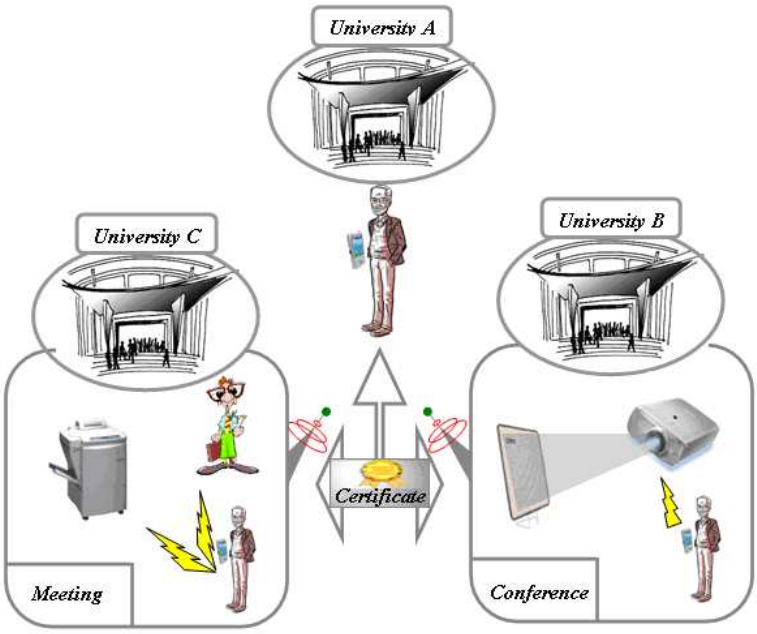

FIGURE 1 – Le scénario du professeur

Le consommateur :

Pour acquérir des droits, l'utilisateur est généralement amené à décliner des informations d'identification (voir section suivante). Supposons qu'Alice possède un certificat d'identité contenant son nom, sa photo, sa date de naissance, son adresse, son numéro de sécurité sociale, son employeur et sa fonction. Si elle veut louer une voiture, Alice doit présenter un document (certificat) l'identifiant et attestant d'un certain nombre d'informations personnelles telles que : son nom, sa photo et son adresse. Cependant, ce même document peut contenir d'autres informations qu'Alice ne souhaite pas divulguer, telles que son âge ou sa fonction. Ce cas de figure n'est pas unique. Comme le montre la figure 2, lors d'une consultation chez un médecin, Alice doit être capable de présenter un document faisant apparaître uniquement son nom, sa date de naissance et son numéro de sécurité sociale. Ceci illustre le besoin de mécanismes permettant la sélection et la personnalisation des informations dans le but d'assurer la protection des données privées et personnelles des usagers.

1.2 La sécurité en environnement pervasif

Nous nous sommes intéressés, dans notre travail, aux problèmes de sécurité dans le contexte d'environnements fortement distribués et proactifs. En effet, dans la littérature, la sécurité dans

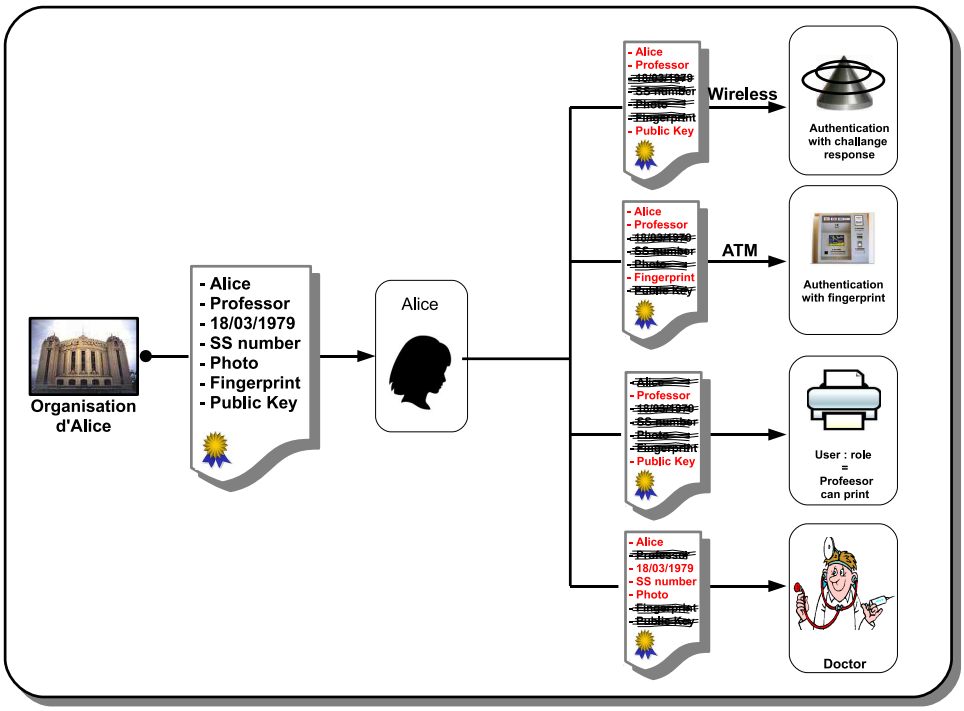

FIGURE 2 – Le scénario du consommateur

les systèmes ubiquitaires a été abordée de manière très simplifiée. Les solutions proposées supposent que l'utilisateur nomade est reconnu par les différents équipements comme étant un membre du domaine ayant un droit d'accès physique à cet espace. Cependant, en pratique, l'usager nomade n'est généralement pas membre du domaine local ; il peut être un professeur qui voudra accéder à une université qui n'est pas la sienne, ou un médecin qui sera amené à consulter dans plusieurs hôpitaux.

Ainsi, la mise en place d'un système de sécurité en environnement collaboratif à caractère pervasif, nécessite de considérer les problèmes suivants :

1.2.1 L'authentification :

Dans un système de sécurité, l'identification sûre d'un utilisateur donné se fait à travers un processus d'authentification. Cependant, l'évolution technologique, dans sa diversité, offre différents niveaux de sécurité et une variété d'outils permettant de procéder à l'authentification. Ainsi, on peut dénombrer trois principales catégories : [Salo 01]

- L'authentification utilisant ce que l'utilisateur connaît (le code PIN du portable, le login, etc).
- L'authentification utilisant un objet possédé par l'utilisateur (carte bancaire, badge, etc).
- L'authentification utilisant l'utilisateur lui-même (authentification biométrique : empreinte digitale, empreinte rétinienne, reconnaissance vocale, etc).

Comme illustré précédemment, l'environnement pervasif est un environnement hétérogène, riche et diversifié, ce qui rend l'authentification très difficile. En effet, celle-ci doit prendre en considération une multitude de contraintes liées à :

- la confiance attribuée à tel ou tel type de terminal.
- la position géographique de l'utilisateur (ex : zone sécurisée ou pas).
- la capacité de calcul des différents dispositifs.
- la multitude des types d'authentification que l'architecture pervasive doit supporter, implémenter voire hybrider.

L'intérêt de mettre en place de nouveaux mécanismes d'authentification est illustré par le premier scénario. Reconnaître le professeur Bob au sein de toutes les universités qu'il pourra visiter n'est pas une tâche facile, mais effectuer l'authentification de manière satisfaisante quel que soit le dispositif utilisé (un PDA, un téléphone, un tag RFID, etc) est encore plus complexe et dépend principalement de la capacité des terminaux employés.

1.2.2 Le Contrôle d'accès :

La mise en place et l'administration d'une politique de sécurité, prenant en compte l'organisation et les exigences internes de chaque structure, peuvent s'avérer très complexes. Ceci a suscité de nombreux travaux sur la conception et la modélisation de politiques de sécurité flexibles et génériques telles que RBAC[Ferr 01], ORBAC[Kala 03], MAC[Losc 98], DAC[Mich 04], etc.(voir chapitre suivant)

Les environnements pervasifs ont été définis dans le but de s'ouvrir sur un monde dans lequel l'information est accessible n'importe où et n'importe quand. De ce fait, l'administration, l'intégration et la collaboration des différentes politiques de sécurité deviennent plus complexes dans la mesure où ces dernières sont généralement hétérogènes d'un point de vue structurel (rôle, niveau d'accréditation, etc) et sémantique (codification, langue, etc).

En résumé, deux contraintes peuvent être identifiées :

- La première réside dans la difficulté à contrôler l'accès de cette diversité d'intervenants à l'information fournie par le réseau pervasif.
- La seconde réside dans la difficulté à utiliser, fusionner et mettre en correspondance, dans un cadre ubiquitaire, les différentes politiques existantes.

Un modèle de contrôle d'accès "pervasif" doit permettre une administration dynamique des ressources partagées par leur propriétaire. En effet, les deux problématiques qui doivent être traitées sont les suivantes :

- Comment un domaine identifie-t-il et contrôle-t-il l'accès des visiteurs nomades à ses ressources ?
- Comment un utilisateur nomade partage-t-il et contrôle-t-il l'accès des utilisateurs de l'environnement à ses ressources ?

Dans l'exemple 1, lorsque Bob visite l'université C, celle-ci doit l'identifier et contrôler à quoi il peut avoir accès (ex : imprimante HP 4041). Par contre, si le professeur dans l'université B doit faire une présentation, son dispositif doit être capable de contrôler qui a le droit de visionner sa présentation.

1.2.3 La confidentialité et la protection de la vie privée :

Le développement de nouvelles technologies contribue à l'évolution de l'informatique ubiquitaire. Cependant, utilisées à mauvais escient, celles-ci peuvent porter atteinte à la vie privée des usagers. En effet, un utilisateur a une perception très limitée des risques potentiels issus des différents équipements qu'il peut embarquer. Prenons l'exemple du passeport biométrique qui contient un tag RFID[Bird 07] et qui peut communiquer l'identité de son propriétaire juste en le passant devant un lecteur. Une personne peut ainsi être reconnue à n'importe quel endroit, en passant par exemple par le centre commercial ou en prenant le train.

Les informations stockées dans les RFID ou toutes autres informations échangées entre l'utilisateur(ou ses dispositifs) et l'environnement sont en général signées (ex : les certificats). Tel qu'illustré dans le second scénario, ces documents numériques sont la propriété de l'usager qui doit avoir le droit de disposer des différentes informations fournies par ces certificats de manière sélective et contextuelle.

1.3 Problématique de la thèse

Dans un environnement pervasif, l'utilisateur nomade constitue l'entité de premier ordre. Il doit pouvoir interagir avec les dispositifs qui l'entourent de manière sécurisée. Dans ce cadre, nous allons dans ce manuscrit principalement traiter les deux problématiques suivantes :

- Comment définir une politique de sécurité capable de procéder à l'authentification et au contrôle d'accès (personnalisé) des visiteurs nomades inconnus par les administrations des sites visités ?
- Comment protéger la vie privée des utilisateurs nomades qui sont dans l'obligation de dévoiler un certain nombre d'informations personnelles via des documents numériques signés (ex : les certificats) ?

2 Principales contributions

- **Mise en place du système Chameleon :** Nous proposons une architecture générique nommée Chameleon qui vient se greffer sur les architectures de sécurité existantes. Celle-ci prend en considération l'hétérogénéité des différentes politiques de sécurité déjà présentes en adressant principalement les problèmes d'authentification et de contrôle d'accès des visiteurs nomades. Cette architecture permet aux utilisateurs nomades de s'authentifier et d'interagir, de manière contrôlée, n'importe où dans l'environnement avec les ressources et les usagers se trouvant dans leur voisinage.

- **Un Modèle de confiance :** Nous proposons un modèle de confiance nommé T2D (Trust to Distrust). Celui-ci met en place un protocole qui permet à chaque entité d'évaluer la confiance à accorder à un usager nomade inconnu en fonction du lien de confiance (construit de manière directe ou transitive) liant le site local au site d'appartenance de l'usager.

- **Un modèle de certificat générique :** Nous proposons un nouveau type de certificat flexible, capable d'être généré dynamiquement et adaptable au contexte applicatif. Nommé X316 (Morph Access Pass), ce certificat est formalisé en XML et utilise un nouveau type de signature, la signature morph.

– **La signature morph :** Dans le but de protéger la vie privée des usagers nomades, nous avons défini un nouveau mécanisme de signature, portant le nom : “Fenrir Morph Signature”. Cette nouvelle signature autorise le propriétaire du document signé à masquer un certain nombre d’informations jugées sensibles avant de l’envoyer sur le réseau.

3 Organisation du manuscrit

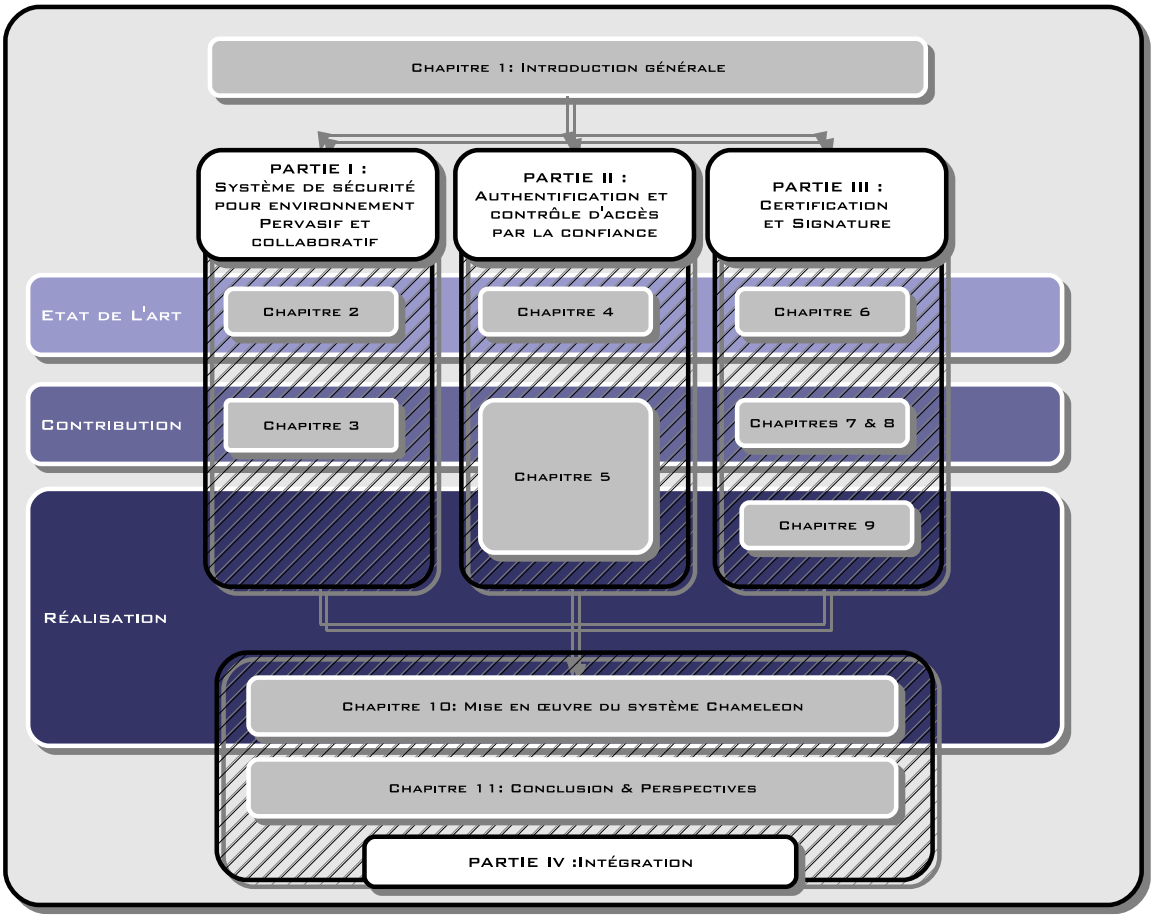

Figure 3 – Organisation du manuscrit

Comme l’illustre la figure 3, la thèse est divisée en quatre parties, comme suit :

– **Première partie “Système de sécurité pour environnement pervasifs et collaboratif” :** Cette partie permet d’introduire et de décrire le cadre de notre travail. Elle est composée des deux chapitres suivants :

  – **Chapitre 2 “Authentification, Contrôle d’accès et protection de la vie privée en environnements distribués et collaboratifs” :** Dans le but de mettre en évidence l’état de l’art sur l’authentification et le contrôle d’accès dans les systèmes pervasifs, nous nous sommes intéressés à la sécurité, plus précisément, aux solutions et aux mécanismes élaborés pour la gestion de l’identité et des privilèges, dans les systèmes distribués au

sens large.

- **Chapitre 3 "Le système Chameleon" :** Ce chapitre décrit les différents modules, composant le système Chameleon, nécessaires pour assurer l'authentification et le contrôle d'accès des utilisateurs nomades.

– **Seconde Partie "Authentification et contrôle d'accès par la confiance" :** Dans cette section nous montrons l'avantage que procure la gestion par la confiance pour la mise en place d'une politique de sécurité pervasive. Cette partie est composée de deux chapitres comme suit :

- **Chapitre 4 "Etat de l'art sur la confiance" :** Dans ce chapitre nous décrivons la notion de confiance, en mettant en évidence l'importance du rapport risques(Faire confiance)/ coûts.

- **Chapitre 5 "Le modèle de confiance T2D" :** ce chapitre présente, étudie, et décrit notre modèle de confiance. Nommé T2D, ce modèle gère le processus d'évaluation de la confiance transitive, en se basant principalement sur la notion de "subjectivité". Il met également en avance une plateforme de test permettant de simuler des réseaux de confiance et de calculer des paramètres nécessaires pour le fonctionnement du modèle de confiance, nous avons mis en place un simulateur de réseaux gérant la personnalité des différents nœuds, pour l'évaluation de la confiance.

– **Troisième partie "Certification et signature" :** Dans cette avant dernière partie de la thèse, nous traitons principalement l'un des plus importants problèmes des systèmes pervasifs, la protection des données numériques considérées comme sensibles et personnelles. Cette partie est composée des quatre chapitres suivants :

- **Chapitre 6 "Etat de l'art sur les certificats et la signature numérique"** : Ce chapitre illustre la diversification des supports et des techniques de signature numérique.

- **Chapitre 7 "Le certificat X316" :** Nous avons mis en place un nouveau modèle de certificat formaté en XML, répondant aux besoins de sécurité des environnements ubiquitaires.

- **Chapitre 8 "Fenrir Morph Signature" :** Le contexte constitue un point très important dans les environnements pervasifs. De là, nous avons défini un nouveau mécanisme de signature taillé sur mesure afin de protéger au mieux les données numériques qui seront amenées à être propagées sur le réseau.

- **Chapitre 9 "Implémentation" :** Dans le but de montrer la modularité de la "Fenrir Morph signature", nous avons l'avons intégrée dans la distribution OpenSource d'XMLDSig de MONO.

– **Quatrième partie "Intégration"**

- **Chapitre 10 : "Mise en œuvre du système Chameleon" :** Dans ce chapitre, nous montrons comment les deux parties précédentes (le modèle de confiance et le certificat

X316) s'intègrent dans l'architecture Chameleon.

– **Chapitre 11 "Conclusion et Perspectives" :** Enfin, nous concluons notre travail de thèse et introduisons les perspectives pouvant découler des différents apports illustrés dans ce manuscrit.

Première partie

Système de sécurité pour environnements pervasifs et collaboratifs

Chapitre 2

Authentification, Contrôle d'accès et protection de la vie privée en environnements distribués et collaboratifs

Sommaire

2.1 Introduction

Durant les dernières décennies, la maturité des technologies réseaux et le développement de nouvelles applications très variées en terme de services et de ressources, ont conduit au développement de nouveaux services et mécanismes de gestion de données nécessitant des environnements de déploiement adaptés. Ces services visent à être accessibles au plus grand nombre d'utilisateurs par l'intermédiaire de réseaux variés tels que : Internet, réseaux adhoc, réseaux pair à pair, etc.

L'émergence de ces nouveaux services fait apparaître de nouveaux problèmes de sécurité pour lesquels les solutions et les mécanismes existants sont inadéquats notamment vis-à-vis des problèmes d'authentification et de contrôle d'accès. Dans un système fortement distribué et ubiquitaire, l'existence d'une politique de sécurité centralisée et homogène n'est en effet pas possible. Il est donc nécessaire de donner plus d'autonomie aux systèmes de sécurité, en les dotant principalement de mécanismes instaurant une coopération et une collaboration dynamiques et flexibles.

Dans le but de comparer l'état de l'art des systèmes de sécurité existants, nous allons définir, dans un premier temps, les exigences en termes de contraintes et de besoins des environnements pervasifs. Puis, nous décrirons la démarche à suivre pour concevoir une architecture de sécurité distribuée et collaborative. Enfin, nous étudierons les différentes solutions élaborées pour la gestion de l'identité et des privilèges et les évaluerons en regard des besoins illustrés au début du chapitre.

2.2 Exigences d'un système de sécurité pervasif

Les environnements pervasifs sont, à la base, des architectures distribuées. Cependant, ils sont sujets à de nouvelles contraintes que nous analysons dans ce chapitre, puis nous identifions les besoins auxquels une architecture de sécurité doit répondre pour sécuriser ces environnements émergents.

2.2.1 Contraintes opérationnelles des environnements pervasifs

2.2.1.1 Mobilité des utilisateurs et des services

La mobilité est l'un des plus importants challenges de l'informatique pervasive. En effet, comme l'illustre le scénario du professeur, en se déplaçant d'une organisation à une autre, Bob voudra interagir avec les ressources et les utilisateurs environnants. Ainsi, dans un réseau pervasif, le professeur doit pouvoir se déplacer, acquérir des privilèges et partager de manière sécurisée les ressources et les services qui sont en sa possession. La mise en place d'un tel scénario nécessite une politique de sécurité flexible prête à offrir aux différents usagers un environnent sécurisé. Elle doit permettre l'authentification et le contrôle d'accès de toute entité qui se trouve dans le domaine, qu'elle en soit membre ou étrangère.

2.2.1.2 Hétérogénéité des environnements

Les utilisateurs nomades sont amenés à échanger des informations et à partager un certain nombre de services et de ressources, en utilisant une très grande diversité de dispositifs dans des environnements totalement différents en termes de domaine d'activité, d'organisation, de membres, etc. Cette contrainte impose aux politiques de sécurité de s'adapter et d'offrir un moyen permettant de faire abstraction de cette hétérogénéité.

2.2.1.3 Equipements à capacité variable

L'utilisation de dispositifs mobiles et légers (ex : PDA, téléphones portables, capteurs, etc.) constitue une des caractéristiques de l'informatique pervasive. Ces dispositifs sont plus ou moins limités en termes de capacité de calcul, de longévité, d'autonomie d'énergie, de fiabilité, etc. Cette diversité matérielle peut générer des problèmes de sécurité ; à l'inverse, elle peut également ouvrir de nouvelles voies favorisant l'évolution des techniques de sécurité classiques. Considérons

l'exemple de la biométrie. De plus en plus d'équipements mobiles intègrent un lecteur d'empreinte digitale comme une alternative aux procédés classiques d'identification.

2.2.1.4 Subjectivité et confiance

Dans un système pervasif, la confiance constitue un des piliers sur lequel se basent toutes interactions, échanges ou partages effectués par les différents acteurs de cet environnement. En effet, la confiance permet d'élargir la couverture des politiques de sécurité. Cependant, l'appréciation de la confiance n'est pas universelle, et dépend de la personnalité de chaque entité. Prenons l'exemple de deux utilisateurs Alice et Bob, qui se différencient par leur caractère confiant pour l'une et méfiant pour l'autre. Ces deux personnes possèdent un ami commun Rayan, qui leur demande de lui prêter une certaine somme d'argent. En fonction de leur caractère respectif, Alice aura tendance à honorer la requête de Rayan plus facilement que Bob. Ceci montre l'importance de la personnalité dans tout scénario impliquant un choix nécessitant une prise de décision par la confiance. De même, en informatique pervasive, le propriétaire d'une ressource (domaine ou utilisateur) est libre d'administrer sa propre politique de sécurité. S'il présente un caractère confiant il aura tendance à avoir une politique de contrôle d'accès ouverte en partageant plus librement ses ressources.

2.2.1.5 Exposition

Dans le but d'interagir avec son voisinage, l'usager nomade est dans l'obligation de "s'exposer", en dévoilant un certain nombre d'informations sensibles pouvant être utilisées à mauvais escient. Ainsi, dans le second cas d'utilisation, les données contenues dans un certificat peuvent communiquer des informations non souhaitées par leur propriétaire.

2.2.2 Besoins de sécurité en informatique pervasif

2.2.2.1 B1 : Décentralisation

Dans un système distribué, la politique de sécurité mise en place se doit d'être la plus décentralisée possible. La centralisation des attributs d'identification et de contrôle d'accès est une limitation du système. En effet, l'utilisateur nomade doit avoir accès à ses attributs et prouver son identité n'importe où dans l'environnement sans tout le temps solliciter le serveur centralisé de son organisme d'appartenance, surtout si ce dernier n'est pas accessible.

2.2.2.2 B2 : Interopérabilité

Avec la généralisation et l'expansion des réseaux informatiques, l'interaction, la fédération et la coalition entre organisations sont considérées comme les plus importantes perspectives à venir.

Considérons le premier cas d'utilisation. Le professeur Bob a besoin d'interagir avec trois politiques de sécurité différentes pour pouvoir accéder aux ressources qui lui sont nécessaires.

L'hétérogénéité est une réalité intrinsèque aux applications pervasives. La conception d'un système décentralisé permettant l'interaction et la collaboration entre organisations hétérogènes est la seule approche réaliste.

2.2.2.3 B3 : Gestion de la confiance

Les politiques de sécurité utilisent largement la notion de confiance (voir section suivante), permettant ainsi une gestion décentralisée des mécanismes d'authentification et de contrôle d'ac-

cès. Dans le premier cas d'utilisation, le professeur Bob arrive à acquérir des droits d'accès au sein des universités B et C. Cela est dû au fait que les trois organismes se font confiance et ont établi des conventions leur permettant de collaborer. Mais, dans le cas où le professeur voudrait se déplacer vers une autre université, qui ne figure pas dans le cercle de connaissance de son organisme d'appartenance, il risque de ne se voir reconnaître aucun droit malgré son statut. Une collaboration plus étendue permettant d'élargir le cercle de confiance en utilisant la propagation de la confiance de départ, que chaque domaine a dans son voisinage, à travers les tiers de confiance, permettrait d'interconnecter les différents organismes de manière pair à pair et dynamique.

En outre, de manière générale, la confiance envers un tiers dépend de manière intrinsèque de la disposition de chaque entité à évaluer la confiance (directe et transitive). Cette caractéristique peut être considérée comme une solution permettant à chaque organisation de mettre en place la politique de sécurité qui lui convient le plus.

2.2.2.4 B4 : Traçabilité et non répudiation

Le déploiement d'une architecture de sécurité pour les systèmes distribués dépend principalement du type d'environnement cible : militaire, public, administratif, etc. En effet, en fonction du degré de confidentialité, le système doit trouver un compromis entre sécurité et accessibilité. Prenons l'exemple des cartes bancaires : elles contiennent un numéro qui est très utile pour effectuer des transactions par téléphone ou via internet. Cependant, ce numéro figure clairement sur la carte et constitue ainsi une faille potentielle, du fait qu'il peut être facilement recopié (ex : par un serveur lors du paiement d'une consommation). Pour combler cette lacune, il existe des institutions (ex : les sociétés d'assurance) et des entités compétentes qui s'occupent de garder une trace à court, moyen ou long terme de toutes les transactions offrant ainsi la possibilité d'identifier et de pister a posteriori tout comportement frauduleux.

En réalité, dans un système distribué ou pervasif, la sécurité absolue est difficilement réalisable voire impossible. Par contre, la mise en œuvre de mécanismes permettant d'identifier rapidement des menaces ou des attaques (ex : non répudiation / traçabilité) fournit un compromis acceptable entre ouverture et sécurité.

2.2.2.5 B5 : Autonomie

Chaque usager doit pouvoir se déplacer et acquérir des droits au sein d'environnements pervasifs de manière autonome sans solliciter son organisme d'appartenance. Dans le premier cas d'utilisation, le professeur Bob doit pouvoir négocier avec les politiques de sécurité des universités B et C, en prouvant qu'il est professeur, afin d'obtenir les droits lui permettant d'utiliser les ressources dont il a besoin. D'un autre coté, chaque usager doit pouvoir également gérer ses propres ressources et services de manière indépendante et sécurisée.

2.2.2.6 B6 : Transparence et proactivité

L'objectif principal de l'informatique ubiquitaire est la simplicité d'utilisation. L'utilisateur doit ainsi pouvoir s'authentifier et acquérir des privilèges de manière aisée et intuitive. De même, les équipements utilisés doivent solliciter leur propriétaire le moins souvent possible, ou mieux encore, anticiper les besoins de l'utilisateur. Un certain nombre de techniques ont été mises en place pour faciliter les procédures d'authentification récurrentes (ex : Single Sign On (voir section suivante), cookies, etc.).

Dans les applications et environnement pervasifs, les problématiques de l'authentification et du contrôle d'accès s'avèrent plus complexes, en raison de la criticité de certaines ressources partagées et de l'absence de mécanisme d'identification unifié. L'adaptation à l'environnement et aux besoins de l'utilisateur passe par la mise en place de techniques d'apprentissage et de négociation.

2.2.2.7 B7 : Flexibilité

Dans les systèmes distribués, les certificats représentent actuellement le moyen le plus utilisé pour identifier une personne, en utilisant la technique du challenge/réponse pour s'authentifier. Néanmoins, l'avancée technologique a permis l'émergence de nouvelles techniques telles que la biométrie, les RFID, etc.

De ce fait, un système de sécurité pour environnement pervasif doit pouvoir intégrer différents moyens d'identification. Mieux encore, il doit adapter les mécanismes d'authentification en fonction du contexte de l'utilisateur, en particulier la capacité et le type du dispositif utilisé. Ainsi, selon les besoins et le degré de confiance de l'environnement de connexion, une politique flexible doit être mise en place pour permettre, par exemple, de sélectionner ou de combiner le ou les identifiants pouvant être gérés par le dispositif de l'usager.

D'un autre côté, la politique de sécurité doit pouvoir octroyer un certain nombre de privilèges adaptés aux besoins de l'utilisateur nomade et aux capacités de son terminal.

2.2.2.8 B8 : Protection de la vie privée

Actuellement les réseaux informatiques véhiculent de grandes quantités d'informations personnelles (publiques et privées). Or, comme le montre le second cas d'utilisation, ce type de données peuvent renseigner sur les habitudes et les préférences des utilisateurs.

Pour limiter ce type d'abus, les équipements de l'usager doivent être capables d'évaluer l'environnement afin d'en sécuriser ses ressources et de n'échanger que les informations strictement nécessaires.

2.2.2.9 B9 : Passage à l'échelle

Dans les environnements distribués et collaboratifs, l'interaction entre les entités membres de différentes organisations constitue l'un des challenges les plus importants.

Plus précisément, un système de sécurité pervasif peut passer à l'échelle s'il arrive à évaluer, à partir de son cercle de connaissances, un maximum d'organisations (de manière transitive) afin de pouvoir mettre en place un système capable d'accueillir un flux croissant d'utilisateurs nomades.

2.3 Conception d'un système de sécurité

Pour comprendre le fonctionnement des différents systèmes de sécurité qui seront introduits par la suite, nous avons choisi de présenter dans cette section les différentes phases nécessaires pour à la mise en place d'une architecture de sécurité gérant l'authentification et le contrôle d'accès.

2.3.1 Définition du mode d'autorisation

Pour construire un système de sécurité, il est important de définir le mode d'autorisation qui correspond le mieux à l'architecture de sécurité cible. La RFC 2904 [Voll 00] a proposé trois modes d'autorisation : le mode agent, le mode *push* et le mode *pull* (voir figure 2.1). Ils définissent le comportement des diverses entités (utilisateur, serveur d'autorisation et ressource) intervenant lors d'une procédure de connexion.

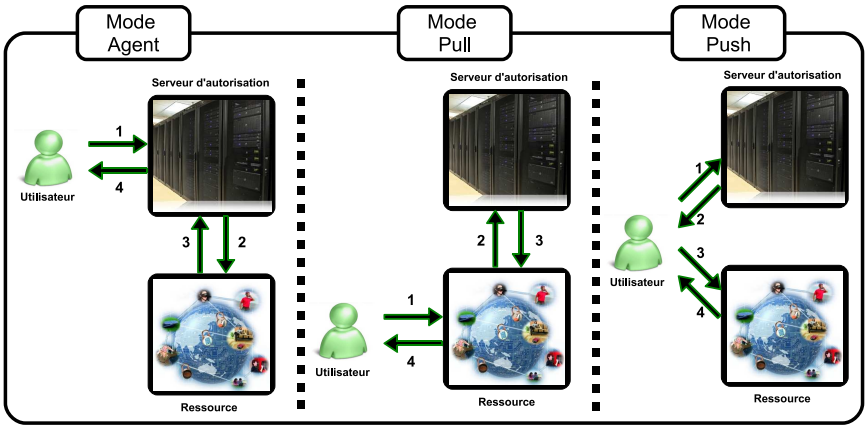

FIGURE 2.1 – Modes d'autorisation

2.3.1.1 Mode agent

Le serveur d'autorisation fonctionne comme un intermédiaire (un agent) entre l'utilisateur et la ressource. Dans ce modèle, la requête de l'utilisateur et la réponse de la ressource passent par le serveur d'autorisation qui autorise ou refuse l'accès en fonction de la politique associant l'utilisateur à la ressource demandée.

2.3.1.2 Mode *pull*

Dans ce cas, ce n'est plus le serveur d'autorisation qui est sollicité en premier, mais la ressource directement. Celle-ci contacte son serveur d'autorisation afin de vérifier si cet utilisateur a les droits nécessaires pour l'utiliser. Ainsi, si la réponse est positive, la ressource donne accès à l'utilisateur en fonction des droits validés par le serveur d'autorisation.

2.3.1.3 Mode *push*

Dans ce mode, c'est le client qui joue le rôle d'intermédiaire. Un premier contact est établi avec le serveur d'autorisation qui autorise l'accès de l'utilisateur en lui délivrant un jeton d'accès (certificat). Ce dernier est ensuite utilisé comme une autorisation prouvant les droits de l'utilisateur auprès de la ressource.

2.3.2 Les composantes d'une politique de sécurité

En faisant une synthèse d'un certain nombre de RFCs (2904 [Voll 00], 3084[Chan 01]), on peut décomposer une politique de sécurité en quatre principaux modules (voir figure 2.2) :

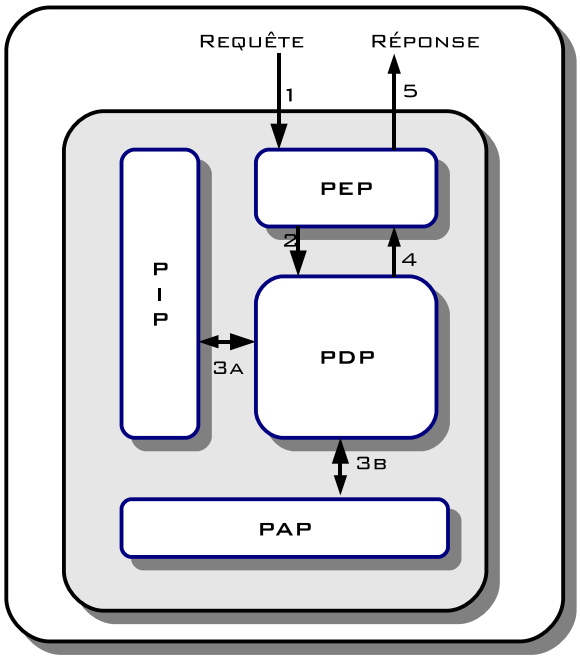

Figure 2.2 – Composition d'une politique de sécurité

PEP (Policy Enforcement Point) : Ce composant représente l'interface de la politique de sécurité. Il réceptionne les requêtes, se charge de l'authentification (étape 1) et formule la réponse prise par le PDP (étape 5).

PDP(Policy Decision Point) : Ce module s'occupe du traitement des requêtes. Il constitue le cœur du système. Le PDP se charge de récolter, à partir du PAP et du PIP, un certain nombre d'informations relatives à la requête (étapes 3a et 3b). Une fois toutes ces informations réunies, le PDP peut alors prendre la décision d'honorer ou de rejeter la requête (étape 4).

PAP (Policy Access Point) Dans cette composante est décrite la politique de contrôle d'accès, c'est-à-dire l'ensemble des règles mises en place afin de filtrer et d'attribuer des privilèges aux différentes entités du domaine (étape 3b).

PIP (Policy Information Point) : Cette partie de la politique de sécurité regroupe toutes les bases de données susceptibles de renseigner et de décrire les utilisateurs, les ressources, l'environnement, etc. (étape 3a)

2.3.3 Gestion de l'identité

2.3.3.1 Définition de l'identité

En tout premier lieu, chaque utilisateur ou entité du domaine doit être identifiable et reconnu de manière unique. Ainsi, en fonction du mode d'autorisation choisi, l'administrateur doit mettre en place les identifiants qui correspondent au mieux aux exigences de la politique de sécurité. De manière générale, il existe trois types d'identifiant :

Les "username/password" : C'est l'identifiant le plus utilisé et le plus simple à mettre en place. Cependant, un mot de passe doit être bien choisi en combinant les lettres minuscules et majuscules, les chiffres et les caractères spéciaux, afin de se prémunir contre l'usurpation d'identité en devinant le mot de passe. Plus encore, la système de sécurité peut obliger les usagers à changer de mot de passe périodiquement. Ce type d'identifiant convient bien le mieux à une architecture de sécurité fonctionnant en mode agent, car dans ce type d'identifiant l'utilisateur ne peut communiquer que via un serveur sécurisé.

Les clés publiques : Une architecture à Clés Publiques (ICP), ou architecture de Gestion de Clefs (IGC), ou encore Public Key architecture (PKI)[Step 08], s'appuie sur la cryptographie asymétrique. Les PKIs permettent de s'affranchir de la nécessité d'avoir recours systématiquement à un canal sécurisé pour échanger des clés. Dans une PKI, chaque utilisateur possède une paire de clés, une publique qui est distribuée à tout le réseau, et une secrète qui est gardée précieusement par son propriétaire. Ainsi, une clé publique peut identifier de manière unique son propriétaire puisque toute information signée avec la clé secrète ne peut être vérifiée qu'avec la clé publique correspondante (connue de tout le monde). En revanche, la diffusion de la clé publique doit se faire via des serveurs centralisés dignes de confiance ou bien entre utilisateurs en utilisant des certificats (voir section 2.3.3.2) pour s'assurer que la clé publique est bien celle de son propriétaire, que le propriétaire de la clé est digne de confiance et que la clé est toujours valide. Les PKIs permettent d'implémenter les trois modes d'autorisation, car n'importe quel utilisateur peut prouver son identité de manière sécurisée sans forcement passer par le serveur central.

La biométrie : Durant tout le XX^{e} siècle, le mot biométrie[Secu 09] a été utilisé dans l'étude quantitative des êtres vivants, notamment à l'aide de méthodes statistiques. La biométrie est une technique globale visant à établir l'identité d'une personne en mesurant une ou plusieurs de ses caractéristiques physiques. Cette technique utilise les caractéristiques de certaines parties du corps humain comme l'empreinte digitale, la main, le visage mais aussi la couleur de l'iris, le timbre de la voix, etc. Grâce à des capteurs, les appareils de biométrie arrivent à comparer une forme, une couleur ou un son par rapport à une représentation de référence. D'une fiabilité de plus en plus efficace, la biométrie apparaît de plus en plus dans notre quotidien (ex : les ordinateurs portables, les PDAs, les passeports, lors de contrôle de frontière, etc.). L'engouement que connaît la biométrie a motivé le groupe de travail OASIS pour la création d'un langage standard nommé XCBF (XML Common Biometric Format)[Larm 03]. Basé sur XML, il permet de faciliter l'échange des données biométriques entre des systèmes d'informations hétérogènes. Comme la clé publique, les identifiants biométriques permettent l'utilisation des trois modes d'autorisation.

2.3.3.2 Support des identifiants

Les identifiants cités précédemment doivent être stockés sur des supports fiables et accessibles. Dans ce qui suit, nous allons introduire les plus utilisés :

Annuaire : Dans le monde de l'informatique, un annuaire est un système de stockage de données, dérivé des bases de données hiérarchisées, permettant en particulier de conserver les données d'une organisation, à savoir les coordonnées des personnes, les courriels , etc.

Lightweight Directory Access Protocol (LDAP)[Harr 06] est l'un des protocoles les plus utilisés dans l'interrogation des services d'annuaire. Il est basé sur la norme X.500[Chad 96] qui est un modèle de service d'annuaire conforme aux recommandations de l'OSI. X500 définit un espace de noms et contient des protocoles pour interroger l'annuaire et le mettre à jour. Cependant, X.500 s'étant révélé trop complexe, LDAP à été défini comme une alternative plus facile à adopter. Comme X.500, il fournit un modèle d'espace de noms et de données pour une gestion d'annuaire. Toutefois, LDAP a été conçu pour s'appuyer sur la couche TCP/IP. Simple à mettre en œuvre, LDAP est très souvent utilisé pour l'authentification. Il sert à stocker les paires (nom d'utilisateur, mot de passe (souvent haché)). De ce fait, n'importe quel serveur autorisé à accéder à la base de données LDAP peut vérifier l'identité des utilisateurs inscrits s'il est autorisé à voir cette information.

Dans la même lignée, Microsoft a mis en place son propre service d'annuaire nommé "Active Directory" (AD). D'un point de vue conceptuel, AD ressemble à un annuaire LDAP. Il permet de stocker toutes les informations de sécurité relatives au domaine dans lequel il est installé.

Certificat d'identité : Il permet d'associer une clé publique à l'identité du propriétaire du certificat (une personne, une machine, etc), afin d'en assurer la validité à l'aide d'un mécanisme d'authentification. Le certificat d'identité est en quelque sorte la carte d'identité de la clé publique qui est délivrée par l'organisme de certification(CA). L'autorité de certification fait office de tiers de confiance et atteste du lien entre l'entité physique et l'identité numérique (clé publique).

Le modèle le plus utilisé pour la création des certificats numériques est X.509[Hous 99]. Ce dernier est le standard défini par l'Union Internationale des Télécommunications. Il a été créé en 1988 dans le cadre de la norme X.500. Dans le modèle X509, une autorité de certification attribue un certificat liant une clé publique à un nom distinctif (Distinguished Name), à une adresse e-mail ou à un enregistrement DNS. Un certificat peut devenir invalide pour d'autres raisons que l'expiration naturelle, telles que la perte ou la compromission de la clé privée, ou un comportement malveillant de son propriétaire. C'est pourquoi des mécanismes de révocation sont mis à la disposition des autorités de certification.

2.3.3.3 Définition du protocole d'authentification

En fonction du mode d'identification choisi et du support utilisé, plusieurs protocoles d'authentification ont été mis en place afin de s'assurer de l'identité des interlocuteurs, mais aussi dans le but de transmettre les paramètres d'identification de manière sécurisée.

Les protocoles les plus utilisés, pour l'identification par mot de passe, sont PAP et CHAP [Simp 96].

PAP (Password Authentication Protocol) a pour avantage son extrême simplicité à implémenter, mais présente le grand défaut de faire circuler le mot de passe en clair. Donc, pour plus de sécurité, il est préférable d'implémenter le protocole CHAP (Challenge Handshake Authentication Protocol). Ce dernier combine le mot de passe avec un challenge avant de l'envoyer, ce qui

permet de ne pas dévoiler le mot de passe en clair. De plus, le challenge est différent à chaque connexion, ce qui empêche la réutilisation.

D'un autre côté, l'utilisation des certificats offre une plus grande sécurité et permet à l'utilisateur d'avoir plus d'autonomie. Dans la littérature, divers protocoles ont été mis en place et on peut principalement citer les trois protocoles suivants : SSL[Dier 99], EAP[Abob 04] et Kerberos[Neum 05].

SSL est un protocole qui a été développé à l'origine par Netscape. Il permet de créer un tunnel entre deux entités authentifiées au-dessus de la couche transport du modèle OSI. L'authentification se fait en général à l'aide de certificats X509. Dans le même contexte, EAP (Extensible Authentication Protocol) est un protocole d'authentification générique qui est surtout utilisé pour les connexions web sans fil (WiFi). Il englobe une multitude de protocoles qui permettent, à l'image de SSL, d'offrir un canal sécurisé pour chaque usager authentifié. Kerberos est un protocole d'authentification qui se situe au niveau de la couche applicative. Il fonctionne en mode *push* et permet de délivrer un ticket de session aux usagers ayant un certificat d'identité. Ainsi, le possesseur de ce ticket a la possibilité d'être authentifié directement par la ressource.

2.3.4 Gestion des privilèges

2.3.4.1 Définition du modèle organisationnel

La gestion de la politique de sécurité nécessite une représentation abstraite de l'environnement réel. Une modélisation formelle et standardisée est nécessaire pour implémenter une politique de contrôle d'accès précise et efficace.

Les modèles formels de sécurité permettent de décrire, sans ambiguïté, les entités du système et les relations qui les lient. Ils se différencient principalement par la définition de l'entité ou l'acteur de de l'organisation (utilisateurs, objets et actions).

Durant les dernières décennies, une multitude d'approches ont vu le jour. Néanmoins, l'hétérogénéité de l'environnement réel fait que certains modèles sont plus adaptés que d'autres en fonction de la structure, du contexte et des besoins de l'organisation.

Dans la littérature, les modèles de contrôle d'accès se regroupent en trois classes : discrétionnaire, mandataire et à base de rôles.

Le contrôle d'accès discrétionnaire : Les politiques de contrôle d'accès discrétionnaire DAC (Discretionary Access Control)[Mich 04] sont basées sur les notions de sujets, d'objets et de droits d'accès. Elles doivent leur nom au fait que l'attribution des permissions d'accès se fait "à la discrétion" du propriétaire d'une ressource. Autrement dit, les droits d'accès sur un fichier sont positionnés par l'utilisateur déclaré comme le propriétaire de ce fichier. C'est le type de politique qui vient le plus naturellement à l'esprit lorsqu'on veut définir des règles d'accès à des objets, car on associe directement à l'objet les politiques de sécurité dont il dépend.

Ce type de politique se retrouve dans la plupart des systèmes d'exploitation actuels (famille des Unix, Windows) dans lesquels pour chaque objet (fichier, répertoire) une liste de contrôle d'accès ACL (Access Control List) associe un utilisateur à une liste de droits (lire, écrire, exécuter).

Le contrôle d'accès mandataire : Le modèle de contrôle d'accès mandataire ou obligatoire MAC (Mandatory Access Control)[Losc 98] a vu le jour pour répondre à la faiblesse du modèle DAC. Il repose sur la délégation du contrôle d'accès à l'autorité de l'organisation. L'existence de

cette entité indépendante garantit que la politique de sécurité ne peut être compromise par les utilisateurs du système.

La politique MAC s'appuie sur la classification des objets et des sujets d'un système. En effet, chaque utilisateur et chaque objet sont associés à un label dénotant leur niveau de sécurité.

Le label d'un objet reflète la sensibilité des informations qu'il contient. De même, le label d'un sujet traduit son niveau d'accréditation dans l'organisation. De cette manière, l'accès à un objet par un sujet n'est accordé que si les deux niveaux de sécurité associés sont compatibles.

Une politique de contrôle d'accès obligatoire n'est applicable que dans la mesure où toutes les parties du système répondent à des relations d'ordre. Or, au sein d'une même entreprise, il est souvent difficile de spécifier si une information venant d'un service est plus secrète ou plus critique que celle provenant d'un autre, ce qui limite l'utilisation du modèle MAC.

Le contrôle d'accès basé sur les rôles : l'article de[Ferr 01] a mis en évidence le fait que, la plupart du temps, les permissions d'accès ne sont pas attribuées en fonction de l'identité des utilisateurs. C'est plutôt le type d'activité qui les détermine, c'est-à-dire le rôle qu'occupe un utilisateur dans une entreprise ou une organisation. C'est ainsi qu'est né le modèle RBAC (Role-Based Access Control).

La notion de rôle simplifie de manière significative la définition des politiques de sécurité. Effectivement, plutôt que d'attribuer directement des droits à chaque utilisateur du système, un certain nombre de rôles sont définis et un ensemble de rôles est assigné à chaque utilisateur. Ensuite les droits d'accès sont associés aux rôles plutôt qu'aux utilisateurs.

Les auteurs de l'article [Sand 99] ont introduit la notion de hiérarchisation de rôles dont l'objectif est d'intégrer la notion d'héritage des permissions entre rôles. Ce modèle apporte les définitions de rôles juniors, avec des degrés de privilèges inférieurs et de rôles seniors possédant des privilèges supérieurs.

Durant la dernière décennie, le modèle à base de rôle est celui qui a connu le plus grand succès. Ainsi, plusieurs variantes ont vu le jour pour améliorer le modèle de base en le rendant plus flexible et plus adapté aux exigences des nouveaux systèmes distribués.

Le modèle RB-RBAC (Rule-Based RBAC)[Al K 02] introduit la notion d'autorisation négative à RBAC pour exprimer qu'un utilisateur n'a pas le droit de jouer un rôle s'il n'a pas certains attributs. Le modèle TRBAC (Temporal RBAC[Bert 01, Josh 03], comme son nom l'indique, utilise des contraintes temporelles dans l'attribution du rôle : un utilisateur peut avoir un rôle le matin et un autre l'après-midi. Le modèle OrBAC (Organisation Based Access Control)[Kala 03] définit un niveau d'abstraction du monde réel décrit principalement par trois classes : Rôle, Activité et Vue. Un rôle est un ensemble de sujets, une activité un ensemble d'actions et une vue est un ensemble d'objets. Chaque classe définit un groupe d'entités sur lesquelles s'appliquent les mêmes règles de sécurité. Ce modèle est récent et se veut extensible. En effet, certains concepts novateurs ont été intégrés lors de sa conception tels que la gestion du contexte et la délégation. Le modèle ORBAC a été étendu aux applications complexes, hétérogènes, interopérables et distribuées sous le nom de Multi-OrBAC[Kala 06] (pour MultiOrganization-Based Access Control). Ce modèle permet de spécifier, dans un cadre homogène, plusieurs politiques de sécurité pour des organisations hétérogènes devant coopérer.

2.3.4.2 Définition du support

Afin de permettre à n'importe quelle entité du réseau de vérifier les droits de chaque utilisateur, ces derniers doivent soit être accessibles à travers un serveur centralisé (annuaire) soit présentés sous forme d'un certificat.

Annuaires : Tout comme pour l'authentification, les droits des utilisateurs peuvent être stockés dans des annuaires (LDAP ou Active Directory). Ainsi, pour vérifier si une entité a le droit d'utiliser une certaine ressource, il suffit d'interroger l'annuaire sur les attributs de l'utilisateur authentifié. Cette organisation est utilisée surtout pour des modèles de contrôle DAC, cependant ces annuaires permettent de définir des groupes et des sous-groupes (ce qui peut être assimilé à des rôles ou à des niveaux) afin de faciliter l'administration des droits des utilisateurs.

Certificat d'attributs : En intégrant les attributs, le rôle d'un certificat devient plus étendu grâce à l'attribution de permissions au possesseur d'une clé. Il contient donc le profil de l'usager définissant ainsi ses privilèges. Les deux propositions les plus répandues sont le certificat d'attributs X.509 et le certificat d'attributs SPKI[Elli 99a]. Chacune d'entre elles offre des services semblables mais avec des mécanismes et des formats différents. Cette alternative nous semble plus intéressante, car elle permet à l'utilisateur d'avoir une certaine autonomie en gérant lui-même ses droits.

2.3.4.3 Définition des règles

La définition des règles de contrôle d'accès dépend de manière intrinsèque du modèle organisationnel. En effet les privilèges sont assignés :

- directement à l'identifiant de l'utilisateur et de l'objet dans le modèle DAC.
- à un niveau de sécurité dans le modèle MAC.
- aux rôles dans les modèles issus de RBAC.

XACML (eXtensible Access Control Markup Language)[OASI 05] est le standard le plus cité. Il a été créé par OASIS et a comme vocation d'offrir un langage commun permettant l'interopérabilité des systèmes d'authentification et d'autorisation. XACML fournit un langage de description des politiques de contrôle d'accès surtout à base de rôles. Celui-ci est fondé sur des règles décrites en XML, qui permettent de définir les droits des utilisateurs (personne ou service applicatif) sur les ressources informatiques (données, services ou composants d'un système). La dernière version 2.0 a été ratifiée par l'organisme de normalisation OASIS en Février 2005. XACML se base principalement sur le modèle RBAC afin de définir les autorisations. Ainsi, la version actuelle de XACML souffre de même limitation de modèle RBAC, ne permettant pas d'exprimer et la contexte et la délégation (voir section suivante). Dans la littérature un certain nombre de travaux [Abi 06] ont étendu XACML afin qu'il puisse représenter d'autres modèles de contrôle d'accès basés sur RBAC (ex : OrBAC, GRBAC). Plus récemment, une nouvelle version 3.0 [SICS 07] est en préparation. Elle permet, de manière native, d'évaluer le contexte et de représenter la délégation.

2.3.5 Extension de la politique de sécurité

Une fois que la politique de sécurité est mise en place, les domaines peuvent souhaiter ouvrir l'accès à leurs ressources à des utilisateurs non membres de l'organisation, sous réserve d'un certain nombre de conditions. Dans cette section, nous allons étudier les différentes méthodes usuellement implémentées dans ce cadre.

2.3.5.1 La confiance

La confiance est le point de départ pour établir et tisser des liens entre les différents acteurs et entités de l'environnement (organisations et utilisateurs). Dans les environnements distribués, la confiance est utilisée aux trois niveaux suivants :

Confiance entre organisation et utilisateurs : Au sein d'une organisation, les différents membres sont identifiés et possèdent des profils locaux, ex : étudiant, professeur, secrétaire, etc. A ce niveau, les utilisateurs membres de l'organisation peuvent se faire confiance à travers les certificats délivrés par leur domaine d'appartenance. Ce dernier est considéré comme étant l'autorité locale (CA : Certification Authority). Ainsi, tous les membres se fient au CA du domaine et peuvent faire confiance à toute autre entité certifiée par celle-ci.

La politique de sécurité peut utiliser le principe de **la délégation**[Band 02]. C'est une notion très répandue dans ce type de relation. Elle a été définie dans le but d'élargir la portée des politiques de sécurité pour les rendre plus flexibles et dynamiques. Cette approche étend l'autorité en utilisant la propagation de la confiance. Ainsi, une source d'autorité peut déléguer à une entité de confiance la gestion de certains droits sur ses ressources. Ce mécanisme est de plus en plus intégré dans les architectures distribuées. Il permet notamment à la politique de sécurité d'impliquer les utilisateurs locaux qui ont la possibilité d'attribuer des droits d'accès sur les ressources du domaine en fonction de leur perception de la confiance. Par exemple : un visiteur extérieur au domaine peut avoir accès à Internet ou utiliser une imprimante, s'il possède un "ami" (membre du domaine) habilité à attribuer ce type de droits. Certains systèmes [Seit 05] vont plus loin, en proposant un mécanisme de délégation à plusieurs niveaux qui permet de déléguer le droit de déléguer.

Confiance entre organisations : Une politique de sécurité peut vouloir étendre son champ d'activité en ouvrant des droits aux membres d'autres domaines. Cette relation est en général le résultat d'accords ou de contrats établis entre les différentes organisations pour mettre au point une collaboration à court, à moyen ou à long terme. En prenant l'exemple d'un projet impliquant plusieurs structures, la mise aux point d'une politique commune va permettre plus facilement aux intervenants du projet appartenant aux différents domaines, de se déplacer et d'utiliser les ressources partagées dans le cadre de cette alliance. En informatique, ce type de collaboration a vu le jour sous le nom d'**Organisation Virtuelle (VO : Virtual Organization)**[Fost 01].

La recherche autour des Organisations Virtuelles a commencé vers les années 90. Une VO est une structure organisationnelle distribuée qui regroupe plusieurs organismes indépendants. La difficulté d'établir ce type d'alliance est liée aux contraintes imposées par les différentes administrations qui sont hétérogènes et potentiellement conflictuelles, ayant l'objectif commun de coopérer, sans mettre en danger leur système d'information.

Ce type de collaboration peut être encore plus étendu. Dans ce cas de figure, la confiance est assimilée à une relation fédératrice, implantée souvent entre des organisations similaires, telles que : les universités (Shibboleth,[Shib 09], eduroam [3]), les états (visa schengen), etc.

Confiance entre utilisateurs : Les différents utilisateurs ont besoin de se faire confiance afin de communiquer et de s'échanger des services et des informations. Ce type de lien peut être créé si les utilisateurs qui veulent communiquer sont membres du même domaine. Ainsi, la confiance

3. Le projet eduroam vise à offrir un accès sans fil sécurisé à l'Internet, aux personnels, et éventuellement aux étudiants, des établissements d'enseignement supérieur et de recherche. (http ://www.eduroam.fr/)

qu'ils ont dans leur organisme d'appartenance leur permet de dialoguer de manière sécurisée. Ce lien peut aussi exister si les usagers appartiennent à des organisations différentes, mais à la condition qu'il y ait une relation directe ou transitive qui lie ces organisations.

D'un autre côté, les utilisateurs eux-mêmes peuvent recommander d'autres entités en les certifiant comme dignes de confiance ou en les annotant d'une évaluation positive ou négative. Ce dernier mécanisme est connu sous le nom de réputation. Cette classe de confiance a été utilisée surtout dans les réseaux pair à pair (ex : [Hasa 08, Bert 04, Xion 03]) et dans les sites marchands (ex : ebay) et les communautés virtuelles (ex :facebook).

Le principe de base des protocoles de réputation est d'évaluer la confiance qui peut être accordée à une entité en fonction des recommandations des autres membres de la communauté.

2.3.5.2 Le roaming

Un des points principaux d'une architecture de sécurité est la capacité des serveurs d'autorisation à pouvoir communiquer et se transmettre des requêtes. Cette opération, dite de roaming[Voll 00], permet le déploiement de relations inter-domaines, en propageant des informations (requête et réponse) d'un serveur à l'autre. Ainsi, si un serveur d'autorisation reçoit une requête d'un utilisateur à laquelle il ne peut répondre, il devient alors client d'un autre serveur en demandant à son PDP de transmettre la requête reçue au PEP du second serveur d'autorisation et ainsi de suite. Cette méthode peut être reproduite le nombre de fois nécessaire pour atteindre le serveur d'autorisation capable de traiter l'information.

2.3.5.3 L'authentification unique (SSO)

L'authentification unique (identification unique ou Single Sign-On (SSO))[Pash 03] est une méthode permettant à un utilisateur de ne procéder qu'à une seule authentification pour accéder à plusieurs applications informatiques. Les objectifs sont multiples :

- Simplifier la gestion des mots de passe. En effet, plus l'utilisateur doit gérer de mots de passe, plus il aura tendance à utiliser des mots de passe similaires ou simples à mémoriser, abaissant par la même occasion le niveau de sécurité.
- Simplifier la gestion des données personnelles détenues par les différents services en ligne, en les coordonnant par des mécanismes de type méta-annuaire.
- Simplifier la définition et la mise en œuvre des politiques de sécurité.

Les approches utilisant des systèmes d'authentification unique peuvent être regroupées en trois grandes classes [Wiki 09a] : les approches centralisées, fédératives et coopératives.

Approche centralisée : Le principe est de disposer d'une base de données globale et centralisée de tous les utilisateurs telle qu'un annuaire. Cela permet également de centraliser la gestion de la politique de sécurité.

Approche fédérative : Dans cette approche, chaque service gère une partie des données d'un utilisateur (l'utilisateur peut donc disposer de plusieurs comptes), mais partage les informations dont il dispose sur l'utilisateur avec les services partenaires. Cette approche est développée pour répondre à un besoin de gestion décentralisée des utilisateurs, où chaque service partenaire désire conserver la maîtrise de sa propre politique de sécurité.

Approche coopérative : Celle-ci découle du principe que chaque utilisateur dépend d'une entité ou d'une organisation. Ainsi, lorsqu'il cherche à accéder à un service du réseau, l'utilisateur est authentifié par l'organisme dont il dépend. Cependant, chaque service du réseau gère indépendamment sa propre politique de sécurité. Cette approche répond notamment aux besoins des structures institutionnelles dans lesquelles les utilisateurs sont dépendants d'une organisation, par exemple les universités, les laboratoires de recherche, les administrations, etc.

L'engouement pour l'authentification unique a fait évoluer le standard SAML d'OASIS [OASI 08] afin d'intégrer ce type de fonctionnalité. Actuellement, SAML est adopté par un grand nombre de solutions traitant de SSO. Concrètement, ce langage définit des formats de messages XML et une terminologie pour véhiculer les informations utilisées par une procédure d'authentification (codes d'accès, privilèges, méthodes d'authentification, autorité de certification, etc). Il a été conçu avec suffisamment d'abstraction pour rendre interopérables des systèmes hétérogènes et s'articule au mieux avec d'autres mécanismes de gestion d'identités.

2.3.5.4 Mise en correspondance (mapping)

La délégation est un mécanisme certes très intéressant mais pas assez puissant pour permettre d'étendre totalement le champ d'accès des politiques de sécurité. Le mapping, ou la mise en correspondance, entre politiques de contrôle d'accès offre une solution intéressante pour étendre les possibilités du contrôle d'accès.

Il existe actuellement deux types de mapping : le mapping par accord et le mapping sémantique.

- **Le mapping par accord** définit une mise en correspondance issue d'un accord préalable entre organisations (VO). Par exemple : dans le cadre d'un projet, un rôle commun "Proj" est établi par les organisations pour définir les droits d'accès aux ressources nécessaires pour cette collaboration au sein de chaque domaine. Ainsi, une mise en correspondance entre les membres du projet (ex : par leur rôle) et le rôle "Proj" permet aux utilisateurs membres du projet d'avoir accès aux ressources nécessaires pour travailler dans les meilleurs conditions.

- **Le mapping sémantique** est la mise en correspondance la plus forte et la plus difficile à réaliser [Lisc 06]. Elle s'applique principalement aux modèles de contrôle d'accès à base de rôles, en utilisant des dictionnaires et des ontologies [Coma 08]. Elle vise à faire correspondre les rôles d'une organisation aux rôles de n'importe quelle autre organisation du même domaine d'activité (ex : médecin correspond à docteur).

2.4 Systèmes de gestion de l'identité et des privilèges

Cette section présente l'état de l'art des principaux systèmes de sécurité assurant l'authentification et le contrôle d'accès. Nous avons répertorié les différentes solutions en deux sous-sections. La première regroupe les systèmes axés sur la gestion de l'identité. La seconde, quant à elle, englobe les systèmes qui gèrent le mécanisme d'authentification mais implémentent aussi une politique de contrôle d'accès permettant la gestion des privilèges de manière plus complexe et plus poussée.

2.4.1 Systèmes axés sur la gestion de l'identité

2.4.1.1 RADIUS

Le protocole RADIUS[Rign 97b, Rign 97a] permet de faire la liaison entre des besoins d'identification et une base d'utilisateurs en assurant le transport des données d'authentification de manière normalisée. Il offre nativement deux protocoles d'authentification, par mot de passe : PAP (échange en clair du nom et du mot de passe) et CHAP (échange basé sur un hachage de part et d'autre avec uniquement l'échange du 'challenge'(voir section précédente).

L'opération d'authentification est initiée par un client RADIUS, qui peut être un boîtier d'accès distant (NAS : Network Access Server), un point d'accès réseau sans fil, un pare-feu (firewall), un commutateur ou un autre serveur.

Le client RADIUS génère une requête Access-Request contenant les informations d'authentification et l'envoie au serveur. Ce dernier traite lui-même cette requête s'il reconnaît l'utilisateur ; sinon il joue le rôle d'un Proxy RADIUS en la transmettant à un autre serveur. Les échanges sont retransmis par la chaîne de serveurs Radius Proxy intermédiaires dans un sens et ensuite dans l'autre. Quand un serveur Radius dispose de suffisamment d'éléments pour l'identification, il valide la requête ou la refuse en renvoyant un paquet de type Access-Accept ou Access-Reject.

RADIUS fonctionne en mode agent. La gestion des autorisations existe mais reste basique. Il peut par exemple filtrer les requêtes par rapport aux adresses IPs en fixant le temps de connexion.

L'un des principaux avantages de RADIUS est qu'il permet de faire du "roaming". Cependant, la décision reste binaire (Accept, Reject) et ne prend pas en charge la confiance qui peut être attribuée au chemin qui a servi à reconnaître l'utilisateur.

2.4.1.2 LemonLDAP

Lemonldap[Lemo 07] est une approche centralisée de l'authentification unique. C'est un service réseau qui est un point d'entrée unique de toutes les requêtes voulant utiliser une ressource protégée. LemonLDAP propose deux modes de connexion, le mode direct (mode *pull*) et le mode sélection (mode agent).

Dans le premier mode, si un utilisateur veut accéder à une application protégée, le système lui réclame son nom d'utilisateur et un mot de passe. Ainsi, après vérification positive, l'utilisateur est redirigé directement vers la ressource protégée. Dans le second mode, l'utilisateur accède après authentification à un menu listant l'ensemble des applications sur lesquelles il possède les autorisations d'accès.

LemonLDAP est une solution qui a été conçue pour de petites structures mais pas pour faire de l'interopérabilité entre les politiques de sécurité. De plus, elle ne peut pas passer à l'échelle étant donné qu'elle présente l'inconvénient d'une gestion SSO centralisée.

2.4.1.3 OpenID

OpenID[Reco 06]est un mécanisme fédératif d'authentification unique et de partage d'attributs. Il permet à un utilisateur de s'authentifier auprès de plusieurs sites sans avoir à retenir un identifiant différent pour chacun d'entre eux mais en utilisant un identifiant unique OpenID. Ce modèle fonctionne en mode *pull* ; il se base sur des liens de confiance préalablement établis entre les fournisseurs de services (sites web, forum, email utilisant OpenID) et les fournisseurs d'identité (OpenID providers).

Le fournisseur d'identité OpenID est responsable de protéger et de surveiller l'identité des utilisateurs. Il permet à chaque membre de visualiser la liste des fournisseurs de services auxquels

il est abonné, ainsi que les informations qui ont été partagées avec ces derniers. Les données personnelles ne sont jamais diffusées sans l'accord du propriétaire et sont stockées sur un serveur dédié et sécurisé.

Avec ce système, un seul identifiant de type URI (ex : "openidfrance.fr/votrenom") suffit à se connecter rapidement aux différents services que peut offrir le web, à condition que ces derniers délèguent l'authentification à un fournisseur d'identité OpenID.

OpenID est encore en phase d'adoption et devient de plus en plus populaire. Les grandes organisations telles que : AOL, Microsoft, Sun et Novell commencent à l'adopter. Aujourd'hui, on estime qu'il y a plus de 160 millions d'URIs OpenID avec près de dix mille sites soutenant ce mécanisme d'authentification.

L'un des avantages d'OpenID est la protection des attributs des utilisateurs et le mécanisme de traçabilité qui permet au propriétaire du compte d'avoir un historique de ses abonnements. Néanmoins, l'adoption d'OpenID impose à l'usager d'abandonner tous ses comptes existants en créant un nouveau compte géré par son nouveau fournisseur. Enfin, OpenID dépend d'une connexion web (http) et ne fonctionne pas en mode *push* (déconnecté).

2.4.1.4 Liberty Alliance

Les spécifications initiales du projet Liberty concernaient l'authentification unique fédérative. Fondée en 2001 par Sun, Liberty Alliance[Alsa 06] permet de coupler les exigences d'une authentification forte avec le respect de la vie privée des usagers.

Les spécifications Liberty Alliance définissent trois types d'acteurs :

- **L'utilisateur**, personne physique ou morale qui peut acquérir une identité.
- **Le fournisseur d'identité** qui crée et gère l'identité des utilisateurs.
- **Le fournisseur de services** qui fournis les services aux utilisateurs une fois qu'ils sont authentifiés par un fournisseur d'identité.

Pour le bon fonctionnement de ce système, des cercles de confiance doivent être définis. Ils sont un groupement de fournisseurs d'identité et de services qui se sont mis d'accord pour mettre en commun (fédérer) l'identité de leurs utilisateurs.

En résumé, l'authentification unique dans Liberty Alliance, donne la possibilité à l'utilisateur d'accéder, après s'être identifié à l'aide d'un compte unique, à des services proposés par différents fournisseurs appartenant à un même "cercle de confiance". Ces spécifications techniques et fonctionnelles décrivent donc les mécanismes qui ont pour objectifs, d'une part simplifier l'authentification des usagers et leur permettre, d'autre part, de maîtriser la diffusion de leurs données personnelles (nom, prénom, adresse électronique, etc) en choisissant celles qu'ils désirent partager.

La mise en place de cette architecture s'articule autour de trois modules. ID-FF (Liberty Identity Federation Framework) s'occupe de l'identification et de l'authentification (utilisant SAML) selon le principe de l'authentification unique fédérée. ID-WSF (Liberty Identity Web Services Framework) constitue la brique de base pour la construction du cercle de confiance. Il gère principalement la découverte et la description de services ainsi que la politique de partage des attributs des utilisateurs. ID-SIS (Liberty Identity Services Interface Specifications), correspond à l'interface permettant l'interaction entre fournisseurs d'identité.

Liberty Aliance est une approche intéressante, car séparer le fournisseur d'identité et le fournisseur de services procure à ce dernier plus d'autonomie pour administrer ses ressources. Néanmoins, la gestion de l'identification avec Liberty Aliance est une approche fédérée qui présente le même problème qu'OpenID, étant donné qu'un nouveau compte doit être mis en place afin de regrouper tous les comptes existants. Cependant, à l'inverse d'OpenID, les comptes orignaux de l'utilisateur existent toujours.

2.4.1.5 Shibboleth

Shibboleth[Shib 09](développé à partir de 2001) est un système d'authentification unique coopératif qui fonctionne en mode *pull*. Il utilise SAML pour la fédération d'identités, et met en place principalement deux fonctionnalités importantes : la délégation d'authentification et le partage d'attributs. Shibboleth a été conçu pour répondre aux besoins des communautés de l'enseignement supérieur et est déjà utilisé dans plusieurs pays : Etats-Unis, Angleterre, Suisse, Finlande, etc.

La brique logicielle Shibboleth a été conçue pour s'intégrer de la façon la plus transparente possible dans le système d'information d'un fournisseur d'identités ou de services. Elle peut s'appuyer sur n'importe quel système d'authentification et peut s'interfacer avec un ou plusieurs référentiels (annuaire LDAP, base SQL, etc).

Shibboleth simplifie considérablement la manière dont on accède aux ressources. L'utilisateurs se connecte au fournisseur de services qui le redirige vers son institution afin qu'il s'authentifie. Un fois l'utilisateur authentifié celui-ci est renvoyé vers le fournisseur muni des informations (attributs) dont il a besoin.

L'ensemble des organismes qui coopèrent en utilisant Shibboleth peut être considéré comme une organisation virtuelle (VO). Etant donné que les grilles de données sont composées principalement de ce type de structure, une version de Shibboleth nommée "GridShib"[Grid 09] a été intégrée dans la dernière version de l'intergiciel de grille 'Globus Toolkit'[Glob 09] afin d'offrir aux utilisateurs de la grille la possibilité de faire du SSO coopératif.

L'avantage de Shibboleth est son aspect coopératif. En effet, un utilisateur n'a plus besoin de créer un nouveau compte, puisqu'il peut utiliser le compte qu'il possède déjà pour s'authentifier dans n'importe quelle organisation qui fait confiance à son organisme d'appartenance. Cette particularité confère à Shibboleth une place de choix au sein des universités. Cependant, l'extension de la politique de sécurité en utilisant le roaming n'a pas été considérée lors de la conception de cette solution, ce qui limite les collaborations aux cercles de confiance (VO).

2.4.1.6 WS-Security

WS-Security[OASI 06] vise à établir des protocoles de sécurité point à point, dédiés à l'échange de messages SOAP[W3C 07] entre services web. Il s'appuie sur un mécanisme de jetons de sécurité combiné à des signatures numériques pour protéger et authentifier les messages. Les jetons de sécurité affirment l'identité de l'émetteur du message, qui est prouvée par un mécanisme d'authentification. Principalement, trois profils de jetons ont été définis :

- Le Username Token Profile, qui comme son nom l'indique, est un profil minimal pour transporter un nom d'utilisateur et un mot de passe.
- Le X509 Token Profile qui permet de transporter des certificats X509.
- Le SAML Token Profile qui intègre des jetons SAML à la spécification WS-Security.

Cependant, il existe des profils pour d'autres mécanismes d'authentification, tels que : Kerberos [Neum 05], XML Common Biometric Format (XCBF), etc.

WS-Trust [OASI 07] fournit des extensions à WS-Security définissant un protocole d'obtention et d'échange de jetons de sécurité. Cette spécification permet d'émettre une demande de jeton de sécurité auprès d'un service spécifique nommé STS (Security Token Service), pour l'utiliser auprès d'un autre Web Service. Ainsi, le STS joue le rôle d'un tiers de confiance, permettant d'échanger un jeton provenant d'un domaine inconnu contre un jeton d'un domaine connu, sans l'existence d'une relation de confiance entre ces deux domaines.

Afin d'étendre la portée de WS-Trust, WS-Federation a été défini comme un mécanisme permettant la communication entre les différents STS.

L'un des avantages de WS-Security est de permettre un mécanisme d'authentification flexible et transparent de l'utilisateur, étant donné que le STS joue le rôle d'intermédiaire permettant de dialoguer avec divers modes d'authentification. De plus, à l'aide de WS-TRUST les différentes politiques de sécurité peuvent s'interconnecter en utilisant du roaming.

Cependant, un usager ne peut utiliser un service que s'il existe une règle qui lui en donne explicitement le droit. Or, l'administrateur du service ne peut connaître toutes les entités de l'environnement, ce qui restreint l'accès à une sous-catégorie d'utilisateurs connus directement.

2.4.2 Systèmes axés sur la gestion des privilèges

2.4.2.1 Akenti

Akenti[Thom 99] est une architecture destinée à fournir des services de sécurité dans un environnement totalement distribué. Il est développé depuis 1998 par "le Département des Systèmes distribués" du laboratoire Lawrence Berkeley.

L'architecture Akenti implémente principalement deux types de certificats :

- Les certificats d'identité X.509 qui permettent d'identifier un utilisateur.
- Les certificats d'attributs qui se présentent en deux catégories :
  - Les certificats d'utilisation qui contiennent les conditions d'accès à une ressource.
  - Les certificats d'autorisations qui sont associés à chaque ressource et spécifient les différents administrateurs autorisés à créer des certificats d'utilisation pour la ressource en question.

Akenti fonctionne en mode *pull* et *push*. Dans le mode *push*, l'utilisateur obtient un certificat d'autorisation dénotant son droit d'accès à la ressource. En présentant cette attestation avec le certificat d"attribut, il permet à la ressource de lui accorder l'accès.

Akenti est une approche intéressante, car en utilisant les certificats d'autorisation l'utilisateur peut, de manière autonome, accéder à une ressource à laquelle il a droit. Cependant le problème qui se pose est la relation qui lie un utilisateur à une ressource qui impose à l'utilisateur de solliciter un certificat par ressource s'il veut faire du *push*. De plus, la propagation de la confiance n'est pas implémentée par Akenti.

2.4.2.2 PERMIS

A l'instar d'Akenti, PERMIS (**P**rivil**E**ge and **R**ole **M**anagement **I**nfrastructure **S**tandards Validation)[Chad 03] est un système de contrôle d'accès basé sur des certificats. Il est développé depuis 2001 par le groupe "Systems Security Research" de l'université de Salford (UK).

PERMIS implémente un mécanisme de contrôle d'accès qui s'appuie sur le modèle RBAC en utilisant les certificats d'attributs X.509. Les autorisations sont représentées en XML et stockées dans un annuaire LDAP. PERMIS peut fonctionner en mode *push* ou *pull* pour récupérer les attributs d'autorisation.

Ce système intègre la notion de délégation statique d'autorité. Ainsi, chaque entité (utilisateur ou administration) définit, en dur dans les autorisations, les entités de confiance ayant le droit d'attribuer des rôles.

Cependant, "DyVOSE"[Watt 06] et "e-Science centre" de l'université de Glasgow ont intégré en 2006 la délégation dynamique à PERMIS. Elle permet de créer une chaîne de délégation qui peut être limitée en taille.

La propagation de la confiance par chaîne de délégation représente une grande évolution du projet PERMIS. Elle étend le champ d'accès des politiques de sécurité mais reste limitée, car chaque autorité déléguée doit spécifier manuellement les entités (qu'elle connaît) qui peuvent prétendre aux privilèges autorisés par la délégation.

2.4.2.3 CAS

Community Authorization Service (CAS)[Pear 02] développé depuis 2002, est une approche proposée par Globus[Glob 09] pour la gestion du contrôle d'accès dans une organisation virtuelle pour les grille. Il est l'un des plus importants systèmes de contrôle d'accès déployé au sein des grilles. CAS constitue l'autorité centrale d'une organisation virtuelle. Il s'occupe de la gestion des ressources et des utilisateurs entre les organisations qui collaborent dans un projet commun.

L'idée derrière CAS est de considérer que les organisations membres de la VO délèguent la gestion d'un ensemble de leurs ressources au serveur CAS qui administre cette organisation virtuelle.

CAS fonctionne en mode *push*. Ainsi, un utilisateur membre de la VO récupère un certificat d'attribut, nommé Proxy Certificate, signé par le serveur CAS correspondant à cette communauté. De cette manière, n'importe quelle entité peut vérifier la véracité du certificat et autoriser ainsi le porteur du certificat à utiliser la ressource dénotée par ce même certificat.

Plus récemment, une extension de CAS nommée CAS++[Arda 06], a été implémentée. CAS++ est orientée vers l'authentification unique fédérative (SSO) en enrichissant l'authentification standard de CAS par l'intégration de l'identification biométique ou en utilisant les cartes à puce.

D'autres travaux[Sale 04] d'extension et d'intégration ont été menés pour enrichir le système CAS, en offrant la possibilité aux serveurs CAS de s'interconnecter pour gérer les droits d'accès de manière décentralisée et coopérative.

La collaboration entre serveurs CAS permet de mettre en correspondance les différents profils implémentés par chaque VO. Ceci constitue une réelle évolution et se rapproche de notre vision d'un système de sécurité collaboratif. Cependant le mapping entre VOs se fait de manière statique et ne peut se créer et évoluer dynamiquement. De plus, chaque VO possède un serveur centralisé qui doit gérer toutes les correspondances qu'il peut y avoir entre les domaines membres.

2.4.2.4 VOMS

Virtual Organization Management Service (VOMS)[Alfi 04] a été développé entre 2001 et 2004 dans le cadre du projet européen "DataGrid"[The 04].

VOMS est très similaire à CAS. La principale différence entre ces deux systèmes réside dans le mécanisme d'autorisation. VOMS peut fonctionner en mode *push* mais également en mode *pull*.

Pour se faire et à l'instar de CAS, les attributs concernant la liste des rôles et des groupes membres de la VO sont stockés chez le serveur VOMS. Par contre, à la différence de CAS, les règles d'autorisation sont présentes chez la ressource. Ainsi, cette dernière est la seule à décider à quoi l'utilisateur a droit. En résumé, ce système considère que le propriétaire de la ressource

(non pas l'administrateur de la VO) est l'entité responsable de la mise en place de sa politique de sécurité.

Comme pour CAS, la mise en place de l'organisation virtuelle passe par un serveur centralisé. De plus, l'interconnexion des VOs n'est pas considérée par VOMS, ce qui réduit la portée du mécanisme d'interconnexion.

2.4.2.5 O2O

O2O[Cupp 06] (for Organization to Organization) est un système de sécurité qui permet de construire une VO (Virtual Organisation) à partir de plusieurs VPOs (Virtual Private Organisation). A l'image d'un VPN, une VPO permet de construire une passerelle entre deux organisations. La politique de contrôle d'accès utilise le modèle OrBAC et s'est inspirée de l'approche fédérative de Liberty Aliance afin que chaque membre d'une organisation utilise un seul profil (le profil initialement attribué par l'organisation d'appartenance) et puisse avoir des privilèges chez les organisations connectées à l'autre bout des passerelles VPO. A l'instar du système CAS, la construction d'une VO se fait en regroupant les différentes politiques de chaque VPO dans un serveur centralisé afin de gérer au mieux les conflits qui peuvent apparaître suite à l'alliance.

L'approche O2O est assez intéressante puisqu'elle permet de manière décentralisée de construire les liens (passerelles VPO) entres les organisations. Cependant, dés lors où le système s'étend en explorant une pseudo-transitivité, mettant en place des organisations virtuelles, celui-ci repasse par une centralisation qui est limitatif en terme de collaboration et d'extension.

2.4.2.6 Sygn

Sygn[Seit 05] [4] est un système de contrôle d'accès qui permet une gestion décentralisée et dynamique des permissions. Toutes les permissions en Sygn sont encodées sous forme de certificats stockés chez le propriétaire. Des permissions peuvent êtres créées à tout moment par les possesseurs des ressources ou par des administrateurs auxquels ce droit a été délégué. Pour la création de ce type de permissions, aucune interaction avec un système centralisé n'est nécessaire.

La délégation à plusieurs niveaux fait la particularité de Sygn. Elle est définie par un mécanisme de chaînage de certificats propriétaires. Sygn permet également de définir une permission sur un ensemble de ressources.

Sygn évite l'utilisation de services centralisés et minimise l'utilisation de tiers de confiance. Cependant, comme PERMIS, la propagation de la confiance est d'un côté, ne se fait qu'à travers la délégation et aucun mécanisme n'a été défini pour calculer la profondeur maximum pour une chaîne de certificats.

2.4.2.7 GAIA OS

GAIA[Roma 02] est une architecture générique permettant de construire des environnements pervasifs fournissant les principaux services nécessaires pour supporter et contrôler des espaces d'applications pervasives. Ces espaces peuvent contenir tous types d'équipements (systèmes audio surround, écrans plasmas tactiles, systèmes infrarouges, WiFi, Bluetooth, les appareils photos numériques, etc). GAIA a la particularité d'intégrer la sécurité comme une composante de l'architecture connue sous le nom de GAIA OS Security. Cette composante permet de faire de

4. Sygn est un système qui a été développé en local au sein du laboratoire LIRIS (Laboratoire d'InfoRmatique en Image et Systèmes d'information).

l'authentification et du contrôle d'accès. Nous avons fait le choix de la citer dans cette section, plutôt que de l'introduire deux fois en la découpant en deux parties distinctes.

GAIA OS Security permet d'authentifier l'utilisateur à partir d'une variété de dispositifs utilisant des protocoles différents, par exemples des PDA, de la reconnaissance biométrique, des badges à puce, etc. Chaque authentification est évaluée par un nombre compris entre 0 et 1. Cette valeur reflète la confiance qui peut être accordée aux dispositifs et aux protocoles utilisés. Ainsi, pour qu'un usager gagne en confiance, il peut cumuler ou combiner plusieurs protocoles pour accroître la valeur de confiance qui lui est associée.

Outres ces mécanismes de confiance, le temps et l'espace sont intégrés dans la politique de contrôle d'accès de cette architecture. En effet, les règles sont définies sur des rôles qui peuvent activer un certain nombre de privilèges dans un espace bien précis et pour une durée de temps déterminée.

Dans le but d'étendre la politique de sécurité, toute entité (service, programme, ou usager) peut déléguer à une autre entité le droit d'utiliser ses ressources. Pour ce faire, deux types de délégation ont été considérés, la délégation pour une entité de confiance et la délégation pour une entité inconnue. La première permet d'accorder à l'entité de confiance tous les droits (que possède celui qui délègue) permettant de manipuler la ressource en question. Alors que la seconde permet de définir un accès plus restreint dans le temps et dans l'espace.

GAIA OS offre un module de sécurité très complet, capable d'intégrer plusieurs technologies avec divers niveaux de sécurité. De plus, à l'inverse des solutions précédentes, le mécanisme d'authentification se rapproche de notre vision de la confiance, car l'utilisateur est évalué numériquement et non pas de manière binaire. Cependant, l'architecture qui est décrite n'évalue pas l'utilisateur, elle se contente d'évaluer son contexte (médium de communication et dispositif). Enfin, la collaboration entre les différentes organisations n'est pas prise en compte, seule la délégation est implémentée, ce qui restreint la politique de sécurité au domaine local uniquement.

2.4.2.8 TrustAC

Trust-Based Access Control (TrustAC)[Mend 05] est un système pour environnements pervasifs assurant le contrôle d'accès en utilisant un mécanisme d'évaluation de la confiance nommé PTM[Alme 04, Mend 06]. Dans TrustAC, c'est le dispositif de l'utilisateur qui administre sa propre politique de sécurité en mode Ad-hoc. Chaque personne gère une liste évaluée d'entités dignes de confiance ou malicieuses, en leur attribuant un degré de confiance relatif à leur réputation et aux recommandations des autres utilisateurs. A la différence du modèle RBAC et à l'instar du modèle MAC, dans ce système chaque utilisateur n'est pas assimilé à un rôle mais à une valeur numérique comprise entre 0 et 1, qui correspond à son degré de confiance au sein de la communauté.

Nous considérons cette approche comme étant très intéressante et la plus proche de nos travaux sur la confiance, étant donnée que l'accès et la mise en correspondance sont évalués et non pas binaires. Cependant TrustAC n'évalue pas l'identité de l'organisation d'appartenance ni le contexte de l'utilisateur, par exemple : le dispositif utilisé (PDA, ordinateur portable, etc.).

2.4.2.9 TACP

Comme pour TrustAC, TACP[Gian 07] (Trust-Based Access Control Policy) est un système de contrôle d'accès basé sur la confiance, utilisant le concept de la réputation et de l'évaluation de l'historique. La solution proposée débute par l'estimation de la valeur de confiance qui pourra

être accordée à la requête émise par l'utilisateur. Ensuite et en fonction de cette évaluation, la politique de sécurité va décider d'honorer ou de rejeter la demande de cet usager. De manière pratique, chaque utilisateur est évalué par une valeur de confiance comprise entre 0 et 1. De même, à chaque ressource, correspond un seuil de confiance compris lui aussi dans l'intervalle [0,1]. Ainsi, si un usager possède un degré de confiance supérieur au seuil de confiance de la ressource sollicitée, il sera autorisé ; dans le cas contraire sa demande sera bloquée.

Comme pour TrustAC, TACP est une solution proche de notre conception du contrôle d'accès par la confiance. Mais, à l'instar de la solution précédente, elle souffre d'inconvénients importants. TrustAC ne prend en effet en compte ni la confiance initiale de l'usager nomade (i.e. la confiance accordée à son domaine d'appartenance), ni l'évaluation du contexte. De même, dans TACP, à chaque ressource correspond un seuil de confiance qui n'est pas calculé mais imposé comme contrainte forte.

2.5 Synthèse

Dans cette partie nous allons comparer les différentes approches présentées dans ce chapitre en se basant sur les besoins identifiés par le premier chapitre. Le résultat de la comparaison est illustré dans les deux tableaux 2.1 2.2. Chaque case du tableau peut avoir l'une des deux mentions suivantes :

- Oui : Le besoin est pris en charge par la solution.
- Non : La solution n'est pas adaptée à ce besoin.

Ainsi, dans ce qui suit nous allons résumer les deux tableaux, en exposant les limites présentées par les solutions existantes vis-à-vis des besoins identifiés :

B1 -Décentralisation- : Les systèmes de sécurité illustrés par l'état de l'art se font le plus décentralisés possible. La plupart des solutions implémentent un système collaboratif tel que Shibboleth, WS -Sécurité, un système hybride alliant centralisation et décentralisation tel que CAS , ou bien un système totalement décentralisé tel que : Akenti, PERMIS, Sygn, GAIA OS, TrustAC et TACP, ou chaque utilisateur gère sa propre politique de sécurité.

B2 -Interopérabilité- : Afin de permettre aux différents usagers nomades de faire valoir leurs droits, dans leur site d'appartenance et au sein de sites distants, l'interopérabilité des différentes politiques de sécurité s'avère indispensable. Cependant, les systèmes de sécurité élaborés spécifiquement pour les environnements pervasifs ne gèrent pas la collaboration entre organisations. GAIA OS, TrustAC et TACP ne permettent pas d'évaluer l'utilisateur nomade en tant que personne ayant un statut bien défini, spécifié et certifié par son organisme d'appartenance.

B3 -Propagation de la confiance- : Ce besoin est difficilement adoptée par les politiques de sécurité (voir Partie II). En effet, dans le domaine informatique la confiance transitive n'est généralement pas toujours tolérée. Comme illustré précédemment, pour les solutions LemonLDAP, OpenID, Liberty Aliance, Shibboleth, VOMS et O2O, la confiance ne s'installe qu'entre des entités se connaissant directement. Dans ce cas de figure, la relation liant les différents acteurs de l'environnement est soit fédérée dans un serveur centralisé, soit représentée par une sorte d'organisation virtuelle regroupant toutes les entités se faisant confiance. En revanche, dans d'autres systèmes (tels que : Sygn, Permis) la réponse à ce besoin reste partielle, car ils utilisent la délégation, qui certes est une approche intéressante mais pas assez puissante pour répondre aux

exigences d'environnements pervasifs. De plus, aucune des solutions n'adresse la disposition de chaque entité à évaluer la confiance.

B4 -Traçabilité- : Dans un système basé sur la confiance, la traçabilité est très importante. Généralement, elle se base sur des mécanismes assurant la non-répudiation afin de pister toute personne ayant commis une action frauduleuse. TrustAC et TACP sont deux solutions qui s'appuient principalement sur la confiance pour gérer leur politique de sécurité. Cependant, en l'absence de traçabilité (mécanisme de signature), la politique de sécurité n'est pas capable de réagir ni de détecter si la propagation de la confiance a permis à un usager malveillant de contourner les limitations instaurées par le système de sécurité.

B5 -Autonomie- : Les environnements émergents ont pour objectif de permettre à l'utilisateur et à toutes ses ressources d'être accessibles n'importe où et n'importe quand. Liberty Alliance et Shibboleth n'autorisent que le déplacement des utilisateurs et ne prennent pas en compte la mobilité des ressources et des services. En effet, la gestion des ressources est centralisée au niveau des domaines d'appartenance. Cette restriction dans ces systèmes, qui est aussi valable pour CAS et VOMS où certains attributs sont gérés exclusivement par la VO, empêche l'utilisateur nomade de disposer de ses droits dans des environnements inconnus du domaine ou de la VO à laquelle il appartient.

B6 -Transparence et proactivité- : Ce besoin constitue actuellement un des axes de recherche de l'informatique ubiquitaire. Des cookies à la SSO, LemonLDAP, OpenID, Liberty Aliance, WS-Security et Shibboleth, offrent un moyen de réduire l'interaction de l'usager pendant le processus d'authentification. Cependant, l'authentification n'est pas l'unique tâche qui risque de solliciter continuellement le dispositif de l'usager, puisqu'on peut citer pour exemple le choix du médium de communication, la découverte des services disponibles, etc. GAIO OS offre ainsi une architecture capable de réagir de manière personnalisée au profil et au contexte de chaque utilisateur.

B7 -Flexibilité- : A l'heure actuelle, la diversification des dispositifs et des équipements utilisés en l'informatique pervasive fait en sorte que l'utilisateur nomade a la possibilité de se connecter et d'interagir avec son environnement en utilisant différents moyens d'identification. GAIA OS et WS-Security offrent la possibilité à l'usager de s'authentifier quelle que soit la capacité des dispositifs utilisés.

B8 -Protection de la vie privée- : L'identité et les attributs d'une personne constituent des informations confidentielles qu'il faut impérativement protéger. La plupart des solutions présentées dans ce chapitre utilisent des certificats numériques. Ceci peut porter atteinte à la vie privée de leur propriétaire (voir scénario "nomade"). Néanmoins, un certain nombre de solutions (ex : OpenID, Liberty Alliance et Shibboleth) donnent à l'usager le choix, à la première connexion, de partager seulement les attributs nécessaires pour l'authentification et le contrôle d'accès. Cette approche est intéressante, mais présente l'inconvénient de stocker les différents privilèges des utilisateurs sur un serveur sécurisé qui doit être accessible tout le temps. Une autre approche qui est également intéressante est celle de O2O qui compte en perspective intégrer un mécanisme de négociation au moment de la connexion de l'utilisateur.

B9 -Passage à l'échelle- : Le passage à l'échelle est un besoin lié fortement au mécanisme utilisé pour interconnecter les différentes politiques de sécurité. L'interconnexion se fait, en général, via des mécanismes d'VO ou de délégation. Cependant, comme il a été souligné précédemment, ces approches restent limitées car elles imposent principalement une mise en place manuelle des règles d'autorisations que ce soit par l'administrateur de la VO, ou par l'entité déléguée. D'un autre côté, les systèmes TrustAC et TACP, qui utilisent l'évaluation de la confiance sur une architecture pair à pair, permettent de déduire et d'évaluer la confiance à partir de la connaissance de chaque entité.

	RADIUS	LemonLDAP	OpenID	Lib Aliance	Shibboleth	WS-Sec
B1 : Décentralisation	oui	non	oui	oui	oui	oui
B2 : Interopérabilité	oui	non	oui	oui	oui	oui
B3 : Propagation de la confiance	oui	non	non	non	non	oui
B4 : Traçabilité	oui	oui	oui	oui	oui	oui
B5 : Autonomie	oui	non	oui	oui	oui	oui
B6 : Transparence et proactivité	oui	oui	oui	oui	oui	oui
B7 : Flexibilité	non	non	non	non	non	oui
B8 : Protection de la vie privée	non	non	oui	oui	oui	non
B9 : Passage à l'échelle	oui	non	oui	non	non	oui

TABLE 2.1 – Comparatif des systèmes axés sur la gestion de l'identité.

	Akenti	PERMIS	CAS	VOMS	O2O	Sygn	GAIA OS	TrustAC	TACP
B1 : Décentralisation	oui	non	oui	non	oui	oui	oui	oui	oui
B2 : Interopérabilité	oui	oui	oui	oui	oui	oui	non	non	non
B3 : Propagation de la confiance	non	oui	oui	non	non	oui	oui	oui	oui
B4 : Traçabilité	non	oui	non	non	non	oui	oui	non	non
B5 : Autonomie	oui	oui	oui	oui	oui	oui	non	oui	oui
B6 : Transparence et proactivité	non	non	non	non	non	non	oui	non	non
B7 : Flexibilité	non	non	non	non	non	non	oui	non	non
B8 : Protection de la vie privée	non	non	non	non	non	non	non	non	non
B9 : Passage à l'échelle	non	oui	oui	non	oui	oui	non	oui	oui

TABLE 2.2 – Comparatif des systèmes axés sur la gestion de privilèges.

2.6 Conclusion

L'étude de l'état de l'art montre qu'aucune des solutions ne permet l'interconnexion des politiques de sécurité en exploitant pleinement la confiance. Les organisations virtuelles restent limitées aux membres de chaque organisation, sauf pour CAS qui permet d'interconnecter les VOs mais de manière manuelle. Les mécanismes de délégation offrent une approche plus flexible que la mise en place de VO, mais n'exploitent que partiellement la notion de chaîne de délégation.

Enfin, la plupart des solutions citées, qu'elles utilisent la délégation ou l'organisation en VO, implémentent une mise en correspondance manuelle des politiques de sécurité. Or, la construc-

tion d'un environnement pervasif facilement extensible impose un mapping dynamique.

Pour ces raisons, nous avons fait le choix de définir un nouveau système nommé Chameleon qui s'installe sur chaque domaine sans modifier l'organisation. Le Chameleon permet d'interconnecter les différents domaines, non pas en VOs, mais de manière pair à pair, ce qui permet d'exploiter pleinement la confiance. Ainsi, n'importe quel utilisateur nomade peut être évalué en fonction de son appartenance, de son statut (et non pas à partir de zéro) et de son contexte (ex : dispositif utilisé, niveau de sécurité supporté, ressources à partagées, etc.) et obtenir un certificat lui donnant accès aux ressources disponibles dans son environnement en mode *push*. La disposition de chaque entité à percevoir la confiance (subjectivité) est, elle aussi, intégrée de manière forte dans la structure du modèle de confiance.

La protection des attributs de chaque utilisateur constitue un important challenge de l'informatique pervasive. Cependant les systèmes qui permettent d'assurer cette confidentialité utilisent un serveur centralisé (ex : OpenID, Shibboleth, etc), En effet, dès lors que les certificats d'attributs sont utilisés, ces derniers doivent, dans les solution existantes, être présentés en clair sans pouvoir sélectionner seuls les attributs qui seront nécessaires pour effectuer la transaction en cours. Ainsi, la personnalisation du contenu d'un certificat ou d'un document signé en général, a été notre second centre d'intérêt.

Le chapitre suivant décrit la structure du système Chameleon et permet d'introduire notre modèle de confiance qui sera détaillé dans la seconde partie de la thèse ainsi que la signature personnalisable qui sera présentée dans le troisième volet.

Chapitre 3

Le système “Chameleon”

Sommaire

3.1 Introduction

L'avènement de l'informatique pervasive a suscité notre intérêt pour la définition d'un système de sécurité qui prend en charge les besoins cités dans le chapitre précédent.

Le système Chameleon, proposé dans ce chapitre, décrit les différents modules nécessaires pour assurer l'authentification et le contrôle d'accès des utilisateurs nomades. La définition de ce système constitue le point de départ qui nous a permis de développer et d'élaborer, tout au long de la thèse, les différentes contributions qui seront présentées par la suite.

Cependant, avant de détailler notre architecture, nous allons présenter les motivations qui nous ont poussés à la choisir.

Pourquoi la dénomination Chameleon ?

Ce nom est inspiré des caractéristiques physiologiques du caméléon, petit reptile de l'ordre des sauriens, qui a la capacité de changer de couleur et ainsi de se fondre dans son environnement. Considérons le premier cas d'utilisation (voir chapitre 1), L'utilisateur nomade se déplace et veut acquérir des droits d'accès dans les différents organismes visités. Pour que ce scénario se réalise, deux approches sont possibles : soit ce sont les politiques de sécurité qui s'adaptent pour

reconnaître de plus en plus d'utilisateurs, soit c'est au profil de l'usager nomade de pouvoir s'adapter à la politique locale.

La première solution est très limitée et ne peut être envisagée à large échelle. En effet, elle nécessite le plus souvent l'intervention des administrateurs ce qui impose de fait que la politique locale évolue lentement au cas par cas. La deuxième approche consiste à définir des mécanismes permettant à l'utilisateur externe, qui n'est pas reconnu localement, de "s'adapter" c'est-à-dire d'obtenir un profil local (comme un caméléon) qui correspond à son statut ou aux droits qu'il a préalablement acquis.

Actuellement, la première approche est malheureusement la plus répandue, se qui ne permet pas d'offrir aux usagers un accès réellement "pervasif" et donc dynamique aux ressources et services déployés par l'environnement. Cependant, comme l'illustre l'état de l'art, de plus en plus de solutions [Shib 09, Chad 03, Seit 05] tendent à définir des mécanismes permettant la collaboration entre environnements et politiques hétérogènes. De la délégation à l'authentification unique, les modèles illustrés précédemment proposent des solutions qui traitent l'hétérogénéité des politiques de sécurité en modifiant au minimum l'existant.

En s'inspirant du caméléon, notre système propose la mise en place d'une architecture offrant à l'utilisateur la capacité de "s'acclimater" au système environnant. Prenant un exemple. Si l'utilisateur visite un organisme allemand et communique en français, il a peu de chances de dialoguer et d'échanger des informations. Pour cela, il doit soit parler une langue comprise localement (ex : anglais ou allemand), soit avoir un dispositif qui sait faire la traduction, car obliger l'organisme et tous ses acteurs à utiliser la langue du visiteur nomade (ici le français) est irréaliste.

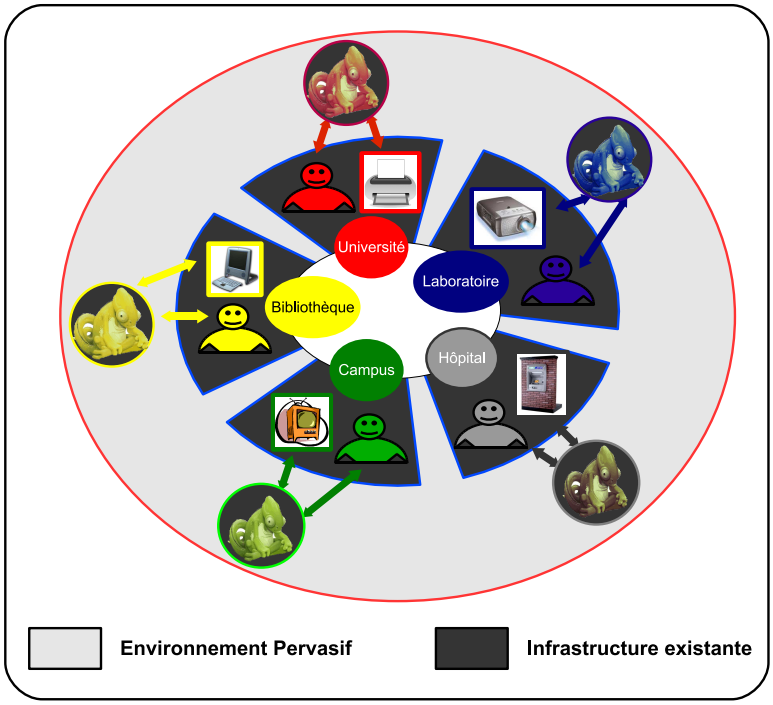

FIGURE 3.1 – L'approche Chameleon

Comme le montre la figure 3.1, l'idée à la base du système Chameleon est de proposer à l'usager nomade une identité reconnue localement, par exemple : étudiant à l'université, cher-

cheur dans un laboratoire, lecteur dans une bibliothèque, etc. En d'autres termes, l'organisme accueillant fournit aux utilisateurs étrangers des identités locales correspondant à leur statut d'origine ce qui est doublement avantageux. D'un coté, la politique locale reste inchangée, car le nouveau profil accordé est un profil interne et par là-même reconnu par les ressources locales. D'un autre coté, les différents membres et services du domaine peuvent évaluer et communiquer avec ce visiteur comme avec un membre à part entière.

3.2 L'approche Chameleon

Le système défini dans ce chapitre a pour principal objectif l'interconnexion de manière pair à pair les différentes politiques de sécurité afin de permettre aux usagers nomades de disposer d'un accès, de manière autonome, à n'importe quel domaine adoptant le système Chameleon. Afin de mettre en place cette architecture, chaque organisation définit son cercle de confiance qui regroupe toutes les organisation auxquelles elle fait confiance. Ensuite elle met en place une politique de mapping qui permet de faire correspondre les statuts des utilisateurs nomades avec des statuts analogues gérés par la politique locale de contrôle d'accès. Ainsi, n'importe quel utilisateur nomade peut se voir attribuer des privilèges dans plusieurs organisations, dès lors que ces dernières font confiance à son domaine d'appartenance. Comme le montre la figure 3.2, du fait que l'université B fait confiance à l'université A, le professeur Bob, membre de l'université A, peut obtenir le rôle *Enseignant* qui représente l'équivalent de son statut (Professeur) au sein de l'université B.

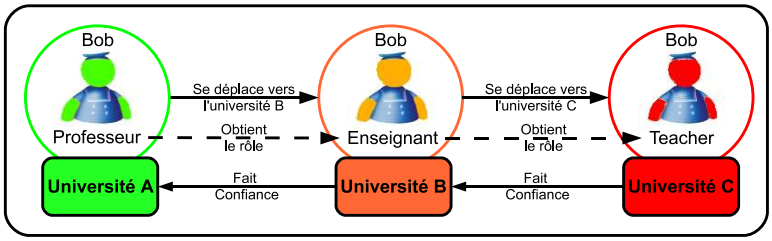

FIGURE 3.2 – Propagation de la confiance

Plus encore, afin d'étendre le champ d'accès des usagers nomades, l'utilisation de la transitivité permet d'interconnecter les différents domaines de manière accrue. Ainsi (figure 3.2), ce même professeur est reconnu par l'université C, grâce au lien de confiance qui est établi entre l'université B et l'université C. Ceci permet à Bob d'acquérir le rôle *Teacher* dans l'université C, qui correspond à son rôle d'origine défini par son organisme d'appartenance.

L'état de l'art nous montre que l'utilisation de certificats d'attributs s'impose pour permettre au professeur d'être autonome, tout en ayant la possibilité de prouver ses droits et son identité n'importe où dans l'environnement. Une attestation peut être délivrée par une autorité de domaine à tout utilisateur dès lors que celle-ci lui fait confiance, c'est-à-dire que l'utilisateur est membre du domaine, ou que son organisme d'appartenance est considéré comme un tiers de confiance. Ainsi, chaque utilisateur nomade possède au moins un certificat (délivré par son domaine d'appartenance) et peut en acquérir d'autres, en fonction des liens de confiance liant son organisation avec les autres organisations de l'environnement.

3.3 L'architecture Chameleon

3.3.1 Positionnement du système

Afin de mettre en place l'approche Chameleon décrite ci-dessus, nous avons défini une architecture modulaire (voir figure 3.3) respectant une contrainte forte, qui est la sauvegarde de l'existant.

Comme le montre la figure 3.3, notre système se positionne entre l'utilisateur nomade et le site local visité ; mais aussi entre le site local et l'ensemble des sites de confiance. Ces derniers sont, en effet, amenés à s'évaluer, les uns les autres, afin de reconnaître les fournisseurs des certificats des usagers nomades.

Le système s'installe sur chaque domaine comme une interface permettant aux visiteurs de s'authentifier et d'acquérir un profil (identité) local(e). La reconnaissance des utilisateurs nomades est alors prise en charge par le module de découverte qui s'occupe de rechercher dans son espace de confiance quels droits d'accès peuvent leur être attribués. Ainsi, l'usager nomade acquiert un profil local en mode *push* dès que son dispositif détecte le serveur Chameleon du domaine. Notre système fonctionne aussi en mode *pull*. Dans ce cas de figure, c'est au moment de solliciter une entité (utilisateur, ressource ou service) du domaine local, que celle-ci contacte son serveur Chameleon afin de fournir à l'usager (de manière transparente) un certificat local valide et lisible par la ressource en question.

3.3.2 Composition de l'architecture

Dans cette section, le système proposé est décrit d'un point de vue architectural et structurel mettant en évidence l'installation, les modules et le comportement de celui-ci.

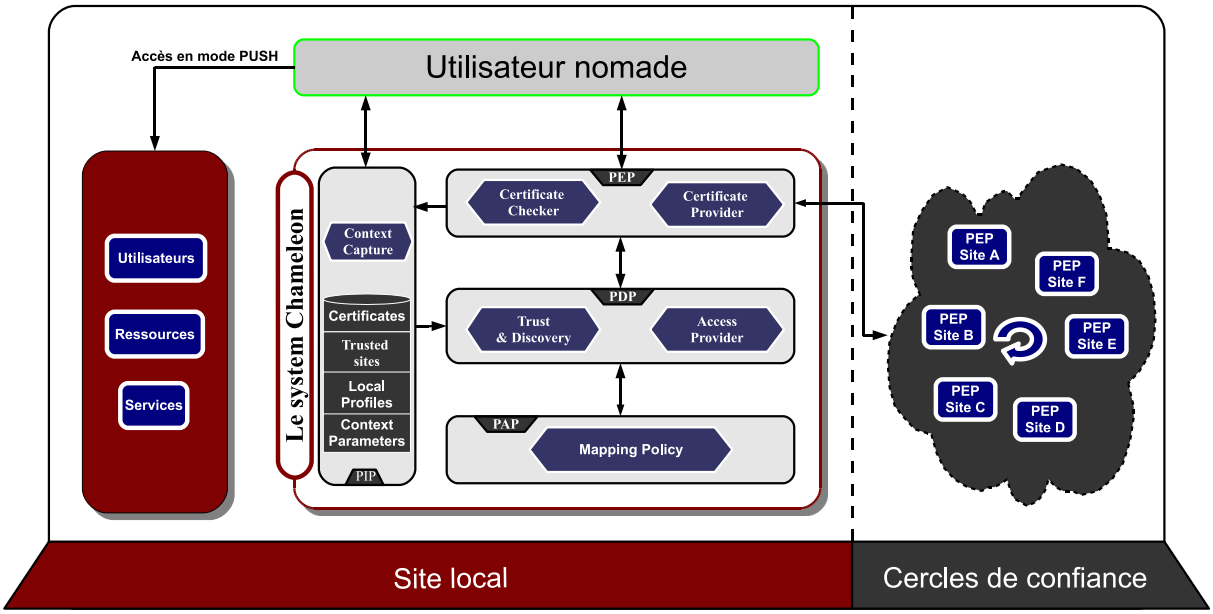

FIGURE 3.3 – Le système Chamleon

Comme nous l'avons illustré dans le chapitre précédent, la figure 3.3 représente la composition

de notre système de sécurité, qui comporte les quatre couches, PEP, PDP, PAP et PIP[Voll 00] :

- PEP "Policy Enforcement Point" : Cette couche constitue le point d'accès de l'architecture. Dans les systèmes Chameleon, toutes les informations entrantes ou sortantes du système sont représentées sous forme de certificats ayant une durée de vie variable selon leur type. Ces informations peuvent représenter des attributs de l'utilisateur nomade, des requêtes, ou des réponses qui permettent aux différents serveurs Chameleon de communiquer. Le PEP est composé principalement des deux modules de certification suivants :

  - Certificate Checker : Ce module n'est accessible de l'extérieur qu'en entrée. Il s'occupe de vérifier la véracité de tous les certificats échangés par le système et d'authentifier leurs propriétaires. Dans le cas où le certificat est valide, ce module le transmet au PDP.

  - Certificate Provider : ce module, commandé par le PDP, n'est accessible qu'en interne. Il s'occupe de produire les certificats commandés par le PDP et de les envoyer aux entités concernées, à savoir les utilisateurs s'il s'agit de certificats d'attributs, ou bien des serveurs Chameleon distants s'il s'agit de certificats présentant une requête ou une réponse.

- PDP "Policy Decision Point" : cette couche est la plaque tournante de notre architecture. Elle n'est accessible que via le PEP qui s'occupe de filtrer l'accès au système. Elle s'occupe principalement d'identifier l'utilisateur nomade et de lui fournir des privilèges valables dans l'environnement administré. Le PDP est composé de deux modules, principalement responsables de la gestion de la confiance et de l'attribution des profils d'accès.

  - Trust & Discovery : ce module a la charge d'évaluer le signataire du certificat fourni par l'utilisateur. Ceci permet d'appliquer la politique de sécurité sur un groupe d'usagers (identifiés par leur site d'appartenance) et non pas sur chaque utilisateur de manière individuelle. Dans le cas où l'organisme d'appartenance du visiteur nomade n'est pas reconnu, une requête est générée et propagée via le Certificate Provider au réseau de confiance. Ce module se charge aussi de répondre ou de diffuser les requêtes venant des systèmes Chameleon voisins.
    Une fois l'organisation du visiteur nomade identifiée, ce module transmet le certificat de l'utilisateur à l'Access Provider.

  - Access Provider : ce module s'occupe de fournir un profil local permettant à l'utilisateur nomade d'être reconnu dans l'environnement. Il commande ensuite au module Mapping Policy de sélectionner un profil local qui correspond le mieux à la confiance qui peut être accordée à l'identité du visiteur. Ensuite, l'Access Provider commande au "Certificate Provider" de produire pour l'usager nomade un nouveau certificat d'attributs à durée limitée dénotant sa nouvelle identité et comportant les mêmes identifiants (e.x : clé publique) que ceux embarqués dans le certificat d'origine.

    L'Access Provider a également la charge de révoquer les certificats des utilisateurs. Il s'occupe de commander au "Certificate Provider" de propager un certificat de révocation à tous les sites susceptibles de fournir un accès à partir du certificat révoqué. Chaque fois qu'un certificat de révocation est reçu, le Certificate Provider destinataire rajoute une entrée dans la base de données des certificats portant la mention révoquée.

- PAP "Policy Access Point" : cette composante constitue le lien permettant d'interfacer la politique locale avec les politiques de sécurité des tiers de confiance. Elle est composée du module "Mapping Policy". Ce module récupère les attributs de l'utilisateur fournis par l'Access Provider : rôle, statut, dispositif utilisé, etc. Il se charge alors de rechercher dans la base des profils locaux, le profil reconnu localement par les acteurs du domaine qui correspond le mieux à l'utilisateur nomade.

- PIP "Policy Information Point" : cette composante représente la base de connaissances de l'architecture Chameleon. Elle est composée d'un module qui s'occupe de recueillir le contexte des utilisateurs nomades, et d'une base de données partagée par la politique locale et par le système Chameleon :

  - Context Capture : ce module peut être sollicité par l'Access Provider. Il a été défini dans le but de récolter le contexte pervasif de l'utilisateur afin de fournir à la politique de contrôle d'accès des paramètres renseignant le contexte de l'usager, par exemple, le dispositif et le médium de communication utilisés par le visiteur pour se connecter. En incluant ces paramètres, le système de sécurité a la possibilité d'ajuster le profil d'accès en fonction du contexte du visiteur nomade en lui attribuant ainsi un profil sur mesure.

  - DB : cette base de données contient principalement les informations suivantes :

    - Certificates : cette base contient les informations relatives à tous les certificats qui ont été générés par le système Chameleon, à savoir :

      - Les certificats d'attributs des utilisateurs.
      - Les certificats contenant une requête ou une réponse.
      - Les certificats spécifiant une révocation.

    - Sites de confiance : cette rubrique comporte les tables correspondant aux sites suivants :

      - les sites de confiance qui seront sollicités pour rechercher un site inconnu localement.
      - les sites qui seront contactés afin de révoquer un certificat délivré par le système local.

    - Profils locaux : dans cette partie, sont stockées les tables de mapping, permettant de faire correspondre les profils locaux aux profils des membres nomades des sites auxquels la politique locale fait confiance.

    - Contexte : dans cette rubrique se trouve les variables prises en compte dans l'évaluation du contexte pervasif.

3.4 Comportement du système

Le système Chameleon permet d'effecteur principalement deux processus :
- Donner des droits d'accès aux utilisateurs nomades et, si besoin, explorer le réseau afin

d'évaluer une organisation inconnue.

– Procéder à la révocation des utilisateurs malveillants.

Afin d'illustrer le fonctionnement des différentes composantes de notre système, ces deux processus sont illustrés par un diagramme de séquences.

3.4.1 Processus d'accès et de découverte

Le diagramme illustré par la figure 3.4 définit le rôle des différentes composantes du modèle Chameleon ainsi que la manière dont elles communiquent ensemble. Dans ce qui suit, nous allons présenter pas à pas le processus qui permet au professeur Bob d'obtenir le rôle 'Teacher' au sein de l'université C (voir figure 3.2).

1. **Le professeur Bob se connecte au portail de l'université C**. En fonction du secteur d'activité, ici "Education", Bob sélectionne, dans son trousseau, un certificat qui atteste de son statut ainsi que de son appartenance à un site de même activité en l'occurrence l'université A. Il envoie ce certificat au "Certificate Checker module".

2. Le 'Certificate Checker module' vérifie la validité de la signature du certificat et si ce dernier n'a pas été révoqué.

3. Une fois le certificat validé, le 'Certificate Checker module' demande au "Trust & Discovery Module" d'évaluer l'identité du signataire de ce certificat (en l'occurrence l'université A).

4. Le "Trust Evaluation Module" interroge sa base de données "Sites de confiance".

5. L'université A n'est pas reconnue. En effet, comme le montre la figure 3.2, l'université C ne connaît pas directement l'université A. Donc le "Trust & Discovery Module" va questionner ses tiers de confiance à propos du site d'appartenance de Bob (Université A). Cela se traduit par la formulation d'une requête de découverte qui doit être envoyée à l'université B. Cette requête est transférée au "Certificate Provider".

6. le "Certificate Provider" crée un certificat signé formulant les attributs de la requête et envoie ce dernier aux sites de confiance listées dans la requête, en l'occurrence l'Université B.

7. **Le "Certificat Checker" de l'université B reçoit la requête.**

8. Comme pour tout certificat, ce dernier vérifie la validité de la signature. Si le certificat est valide celui-ci est transféré au "Trust & Discovery module".

9. Le "Trust & Dicovery module" recherche dans sa base de données s'il fait confiance à l'université A. Dans notre exemple, l'université A est connue localement. La requête est alors transférée à l" "Access Provider".

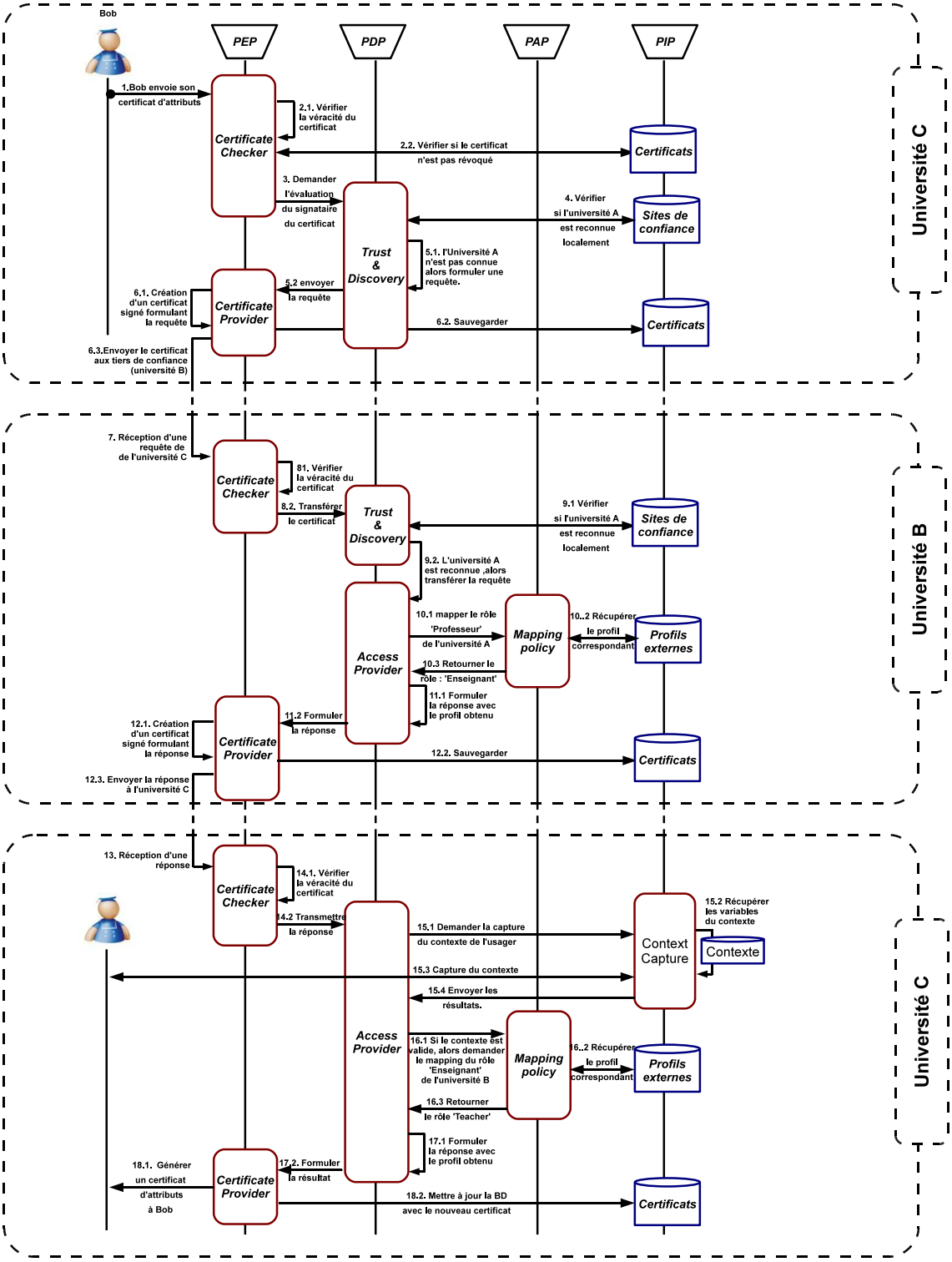

FIGURE 3.4 – Le processus d'accès et de découverte.

10. Puisque l'organisme d'appartenance de Bob est considéré comme un tiers de confiance, la politique locale est capable de mapper le rôle 'Professeur' de Bob. Pour ce faire, le "Mapping Policy module" est sollicité afin d'effectuer la mise en correspondance. Il interroge la base de données contenant les profils externes, et applique les règles de mise en correspondance. Comme illustré dans l'exemple, le rôle 'Enseignant' correspond à la désignation locale du statut de Bob.

11. Une fois que le nouveau rôle de Bob est défini, l'Access Provider formule une réponse à la requête reçue et l'envoie au "Certificate Provider".

12. Ce dernier module génère un certificat exprimant la réponse et l'envoie à l'université C.

13. **Le "Certificate Checker" de l'université C reçoit la réponse.**

14. Il vérifie la véracité du certificat reçu et transmet la réponse, si la signature est valide, directement à l'Access Provider.

15. Maintenant que l'université A est reconnue, le "Context Capture Module" est interrogé par l'Access Provider afin de récupérer les données relatives au contexte de Bob, par exemple : le médium de communication supporté par les deux parties, le type du dispositif utilisé, etc.

16. Si les valeurs du contexte sont satisfaisantes, l'Access Provider sollicite le "Mapping Policy Module" afin qu'il précise la désignation locale qui correspond au rôle 'Enseignant' défini par l'université B. Après interrogation de la base de profils, la politique de mise en correspondance de l'université C retourne le rôle 'Teacher'.

17. L'Access Provider transfère une requête au "Certificate Provider" afin de produire pour Bob un certificat contenant un statut local (Teacher).

18. Enfin, le "Certificate Provider" fournit au professeur Bob un certificat d'attributs dénotant les nouveaux privilèges accordés par le rôle qui lui a été attribué (Teacher), et met à jour la base de données des certificats.

3.4.2 Processus de révocation

Comme pour tout système utilisant des mécanismes de certification, l'instauration d'une politique de révocation est nécessaire. Un utilisateur peut normalement utiliser son certificat jusqu'à une date de validité fixée lors de la création de celui-ci. Cependant, la confiance accordée en lui délivrant un certificat peut diminuer voire disparaître avant la péremption de son certificat (ex : l'utilisateur pourrait un jour être renvoyé, où bien dépossédé d'un certain nombre de ses droits).

Prenons l'exemple du professeur Bob qui change d'université. Son organisme d'appartenance (université A) doit alors révoquer ses droits puisqu'il ne fait plus partie des membres permanents. Comme le montre la figure 3.5, le système Chameleon procède comme suit :

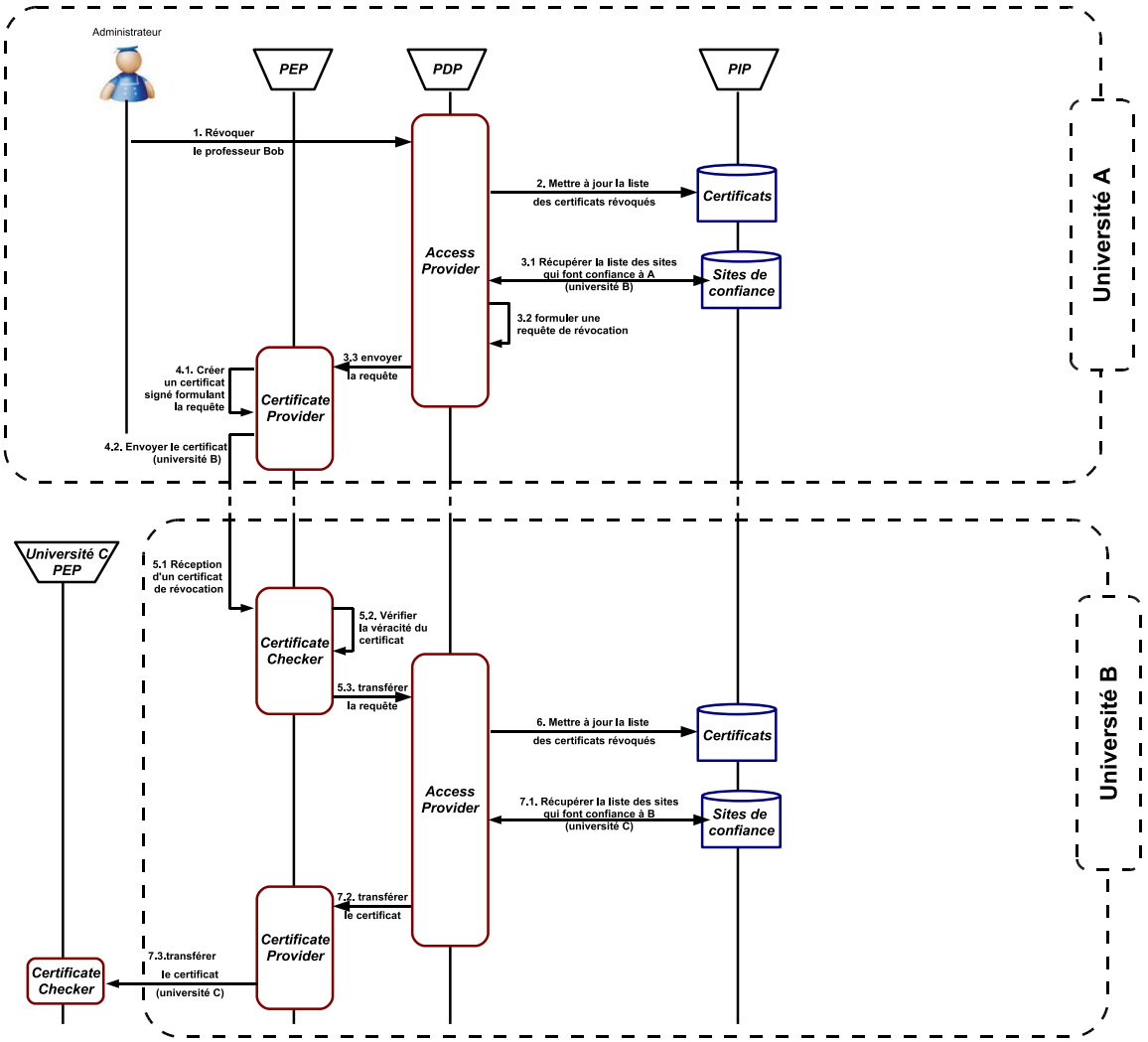

FIGURE 3.5 – Le processus de révocation.

– Chez l'université A :

1. L'administrateur commande à l'Access Provider de révoquer le professeur Bob.
2. L'Access Provider met à jour la table des certificats révoqués.
3. Ensuite, celui-ci doit informer les sites qui font confiance à l'université A afin qu'ils ne considèrent plus ce certificat d'appartenance de Bob. L'acte de révocation est formulé sous forme de requête qui est transmise au "Certificate Provider".
4. Le "Certificate Provider" créé un certificat exprimant la révocation du professeur et transmet la requête au sites de confiance de A (en l'occurrence B)

– Chez l'université B :

5. Le "Certificate Checker" reçoit le certificat dénotant la révocation du professeur par l'université A. Ce module transfère la requête de révocation à l'Access Provider si la signature du certificat est valide.
6. L'Access Provider met à jour sa table des certificats révoqués en rajoutant les entrées

qui correspondent au certificat révoqué par l'université A ainsi que tous ceux délivré localement à partir de ce certificat.

7. Ensuite comme l'université A, l'Access Provider peut demander au "Certificate Provider" de retransmettre le certificat de révocation à l'université C afin qu'elle applique ce même processus de révocation.

3.5 Apports de l'architecture

Dans ce chapitre nous avons mis en place un nouveau système capable de mieux s'intégrer dans un environnement pervasif. Ainsi, le système Chameleon est totalement distribué. Chaque organisation est responsable de sa propre politique de sécurité et l'interconnexion entre elles est possible grâce à la propagation de la confiance. A la différence des systèmes existants, où le mapping est mis en place par un groupe d'organisations (VO). Le système Chameleon est plus décentralisé. Il permet à une administration de gérer sa propre politique de mise en correspondance. De plus, la politique de contrôle d'accès gère, dans le processus de mapping, l'évaluation du contexte pervasif.

Ainsi, en délivrant à l'utilisateur un certificat d'attributs dénotant son identité et tous ses droits, ce dernier acquiert une large autonomie. Celle-ci lui permet de faire valoir ses droits et de récupérer dynamiquement des certificats valides de manière transparente. Ceci procure une certaine proactivité, étant donné qu'avant même de demander l'accès à une ressource appartenant à un domaine inconnu, le dispositif nomade de l'utilisateur peut préalablement récupérer l'attestation nécessaire. Ainsi, les utilisateurs nomades peuvent se déplacer d'un domaine à l'autre, (même inconnu de leur organisation de rattachement) être reconnus et par là-même acquérir un certain nombre de privilèges. De plus, malgré la multiplication des certificats, l'identifiant qui servira à l'authentification (ex : la clé publique) reste toujours le même puisqu'il est réutilisé à chaque fois qu'un nouveau certificat d'attributs est créé.

En outre, la traçabilité, la non-répudiation et la révocation constituent des fonctionnalités très importants dans l'architecture Chameleon. En effet, il est impossible qu'une information soit traitée ou échangée par le système sans être signée et présentée sous forme de certificat avec une durée de validité. De plus, tous les certificats échangés ou produits sont stockés dans une base de données. Cet échange ne se fait que via le PEP, ce qui lui procure le statut d'un pseudo pare-feu, fonctionnant comme une barrière ne laissant filtrer que des informations signées dont la provenance est connue.

Nous reviendrons en détail au chapitre 11 sur l'adéquation de Chameleon aux besoins et aux objectifs exprimés dans l'état de l'art (voir chapitre 2), en précisant les fonctionnalités des différents modules et leur implémentation.

3.6 Conclusion

L'architecture Chameleon permet de mettre en œuvre une politique de sécurité capable de gérer le déplacement des utilisateurs nomades en interconnectant les différentes organisations à l'image d'un réseau pair à pair et en utilisant la confiance.

Dans le but d'offrir une politique de sécurité permettant de mieux gérer la confiance, que se soit pour l'identification ou le contrôle d'accès, nous allons définir dans la deuxième partie du manuscrit un nouveau modèle de confiance intégré dans le module "Trust & Discovery". Il permet de borner et d'évaluer la transitivité de manière plus précise, et cela en intégrant la subjectivité des administrateurs dans la politique de confiance.

Dans le troisième volet de la thèse, nous introduisons un nouveau modèle de certificat capable de donner plus de flexibilité à l'utilisateur en gérant plusieurs types d'identification (clé publique, biométrie, etc). De plus, un nouveau mécanisme de signature sera mis en place dans le but de permettre à l'utilisateur nomade de protéger les attributs qu'il peut avoir à fournir via des certificats et d'en disposer de manière sélective. Ces contributions ont pour but de développer les fonctionnalités des modules Certificate Checker et Provider.

Enfin, dans la dernière partie de la thèse, nous montrerons comment les différents apports qui seront présentés dans la partie II et la partie III s'intègrent dans l'implémentation du système Chameleon.

Deuxième partie

Authentification et contrôle d'accès par la confiance

Chapitre 4

Etat de l'art sur la confiance

Sommaire

4.1 Introduction

La notion de confiance est un concept émergent en informatique. Cependant, ce dernier joue depuis longtemps un rôle important dans de nombreuses autres disciplines notamment : la sociologie, la psychologie, l'économie, les sciences politiques, l'histoire et la philosophie.

La diversité des arrangements contractuels (alliances, accords de coopération) et l'émergence d'une réflexion sur les modes de coordination au sein des réseaux ont créé les conditions favorables à un retour théorique sur la notion de confiance. Pourtant, ce concept n'est pas nouveau et il est déjà intensément utilisé en économie. V.Mangematin[Mang 98]

À ce titre, chaque discipline a tenté de définir ce concept. Le problème de la définition de la confiance est qu'il existe différents types de confiance. Dans certains cas, la confiance se limite à

des relations interpersonnelles ; dans d'autres cas, la confiance est étendue à des institutions. Elle dépend également de la perception de chaque personne, et potentiellement du contexte applicatif.

Dans ce chapitre, nous allons illustrer comment la confiance s'est développée en informatique, avant d'introduire les différentes solutions et axes de recherche qui se fondent sur la confiance afin d'interconnecter et d'évaluer les différentes entités présentes dans un environnement applicatif donné.

4.2 Des définitions de la confiance

La confiance est un concept qui a été largement discuté d'un point de vu économique et social. Dans ce que suit, nous présentons un panorama de définitions issues de divers domaines [Mang 98, Neve 04, Char 05].

- *Rotter 1967* [Rott 67] : La confiance interpersonnelle est l'attente, par un individu (ou groupe d'individus), que la promesse (verbale ou écrite) d'un autre individu (ou groupe d'individus) sera respectée.

- *Zand 1972* [Zand 72] : Décision individuelle s'appuyant sur des attentes optimistes concernant le résultat d'un événement incertain, étant donné une vulnérabilité personnelle et un manque de contrôle personnel sur les actions des autres.

- *Golembie 1975* [Gole 75] : Croyance optimiste subjective, fondée sur les perceptions et les expériences.

- *Butler et Cantrell 1984* [Butl 84] : Attente du comportement de l'autre concernant cinq points : intégrité, compétence, cohérence, loyauté et ouverture.

- *Coleman 1984* [Cole 90] : Relation entre deux acteurs ; la confiance placée par l'un des acteurs dans le deuxième peut dépendre de l'intervention d'un troisième acteur.

- *Granovett 1985* [Gran 85] : La confiance dans le passé mène à la confiance dans l'avenir.

- *Lewis et 1985* [Lewi 85] : La confiance est motivée soit par un fort sentiment affectif envers l'objet (confiance Weigert émotionnelle) soit par des raisons rationnelles (confiance cognitive), soit le plus souvent par une combinaison des deux.

- *Zucker 1986* [Zuck 86] : Ensemble d'attentes sociales partagées par chaque personne impliquée dans un échange économique.

- *Luhmann 1988* [Luhm 88] : Choix de s'exposer à une situation où le préjudice éventuel peut être plus important que les bienfaits attendus.

- *Williamson 1993* [Will 93] : Faire confiance, c'est accepter de s'exposer au risque d'opportunisme.

- *Robinson 1994* [Robi 94] : Attentes, suppositions et croyances concernant la probabilité que les actions futures d'un autre seront favorables ou au moins non préjudiciables à ses

propres intérêts.

- *Mayer 1995* [Maye 95] : Volonté de l'une des parties d'être vulnérable aux actions d'une autre partie, fondée sur l'espoir (expectation) que l'autre réalisera une action importante pour celle qui accorde sa confiance, sans tenir compte de la capacité de contrôler ou surveiller l'autre partie.

- *McKnight 1996* [McKn 98] : Les individus font le choix de faire confiance sur la base d'arguments rationnels issus de la comparaison entre les coûts et les bénéfices de la confiance.

- *Zaheer 1998* [Zahe 98] : Attente qu'on peut se fier à un acteur pour qu'il remplisse ses obligations, que l'acteur se comportera de manière prévisible et que l'acteur agira et négociera de manière équitable en cas d'opportunisme.

- *Shockley-Zalabak* [Shoc 00] : Volonté d'être vulnérable par rapport à une autre partie, s'appuyant sur l'identification (aux buts, valeurs, normes et croyances de l'autre partie) ainsi que sur la croyance que l'autre partie est compétente, ouverte, attentive et fiable.

D'après les définitions citées précédemment, confiance "rime" avec opportunisme, risques, promesses, optimisme, expériences, compétences, dépendance, partage, vulnérabilité etc. Tous ces mots décrivent le paradoxe qu'est la confiance. En effet, ce mécanisme présente des avantages et des inconvénients, le plus important étant de choisir entre :

- faire confiance, autrement dit, s'ouvrir sur le monde extérieur en assumant les risques qui en découlent,
- ou bien refuser la confiance pour avoir une sécurité renforcée, étanche à tout apport extérieur qui est le plus souvent bénéfique.

Ce choix dépend d'un côté, du contexte institutionnel (ex : militaire, médical, universitaire, etc.), et d'un autre côté, du rapport avantages/risques. En effet, une personne accordera plus facilement sa confiance en dépit de la présence d'incertitude, si les avantages attendus sont supérieurs à la prise du risque.

4.3 Garde-fous de la confiance

Beaucoup de chercheurs, d'économistes et de philosophes s'accordent à dire que l'utilisation de la confiance fragilise la sécurité, mais qu'à l'inverse, la confiance peut être vue comme un "lubrifiant" du système. Cette mécanique doit alors s'accompagner de mécanismes sociaux et juridiques (garde-fous) [Mang 98] permettant à celui qui accorde sa confiance de se prémunir contre les risques d'abus.

Les contrats sont le premier garde-fou. Ils constituent une incitation pour les parties à se comporter conformément aux clauses, faute de quoi l'auteur d'un comportement abusif s'expose aux pénalités ou aux sanctions prévues dans le contrat.

Les liens relationnels sont le second garde-fou de la confiance. Ces liens sont corrélés par une appartenance à une entité, ex : famille, organisation, religion, équipe, etc. Celle-ci crée

implicitement un sentiment de confiance où tous les membres ont tout intérêt à adopter une bonne conduite, étant donné que la réussite du groupe dépend du comportement de toutes les entités membres.

La présence de structures physiques représente le troisième garde-fou de la confiance. Cette structure peut être un magasin, un bureau ou une institution. Ceci permet de fournir des informations utiles pour construire une relation de confiance. Ainsi, malgré l'expansion d'Internet, beaucoup d'usagers préfèrent, s'ils ont le choix, acheter au centre commercial dans le magasin le plus proche que de se faire livrer en faisant quelques clics sur Internet. La présence physique procure au client un sentiment d'assurance sur la pérennité et la fiabilité de la relation qu'il pourra établir avec son interlocuteur.

L'assurance présente une protection juridique sous forme de contrat, permettant de se prémunir contre différents types de risques accidentels ou malveillants. Ainsi, les garanties octroyées par les contractants permettent davantage de prise de risque. En effet, conformément au contrat signé, l'assureur s'engage à couvrir, partiellement ou totalement, le contractant.

Le partage d'expériences est le dernier garde-fou. La communication et l'échange d'informations entre les différents acteurs du réseau permettent de partager les expériences passées. Et ainsi de recommander les entités qui se sont bien comportées et inversement. Chaque entité a donc intérêt à se conduire correctement et à honorer ses engagements pour qu'elle soit conseillée aux membres de la communauté.

4.4 La confiance en Informatique

L'évolution de l'informatique pendant les dernières décennies a fait apparaître une nouvelle vision de la sécurité, inspirée fortement du comportement humain. L'expansion des réseaux informatiques a favorisé la prolifération de réseaux sociaux virtuels (ex : les jeux vidéo, les sites de vente, les sites de rencontre, etc.) dans lesquels la notion de confiance est prépondérante et largement adoptée et utilisée. Dans ce sens, un certain nombre d'outils et de mécanismes ont été définis pour intégrer la confiance dans des environnements sensibles, en tirant partie de tous les avantages qu'elle procure.

4.4.1 Confiance vs Sécurité

Dans le domaine de l'informatique, la notion de confiance n'est pas tout le temps tolérée. Elle est considérée, souvent, comme une relation pouvant dégénérer et produire des résultats non prévisibles et donc non contrôlés.

Une partie de la communauté considère que la confiance et la sécurité ne se situent pas dans le même registre. La confiance est du ressort de la perception, avec une forte dimension psychologique, alors que la sécurité renvoie à un univers plus scientifique, et plus formel. Cependant, le paiement par Internet est un parfait exemple de la complémentarité de la confiance et de la sécurité. Bien que la présence de la séquence de chiffres sur la carte bancaire est incontournable pour le e-commerce. Celle-ci présente, d'un point de vue sécuritaire, une faille très importante. En effet, le simple fait de perdre sa carte ou de se faire prélever ce numéro permet d'effectuer des transaction distante sans l'aval du propriétaire. Ainsi, le fait de communiquer le numéro de carte

à un marchand sur Internet dépend principalement de sa réputation ainsi que de la confiance qu'on peut lui accorder.

Malgré tous ces inconvénients ainsi que la dimension incertaine de la confiance, la carte bancaire est actuellement le système de paiement le plus répandu. Dans cet exemple, l'incertitude a fait place à un mécanisme très avantageux où les risques sont tout à fait acceptable au regard des profits.

La relation de la confiance et de la sécurité est très complexe et complémentaire [Bell 04]. D'un coté, la sécurité peut être considérée comme une composante, une méthode ou un préalable indispensable pour construire la confiance. Et d'un autre coté, la confiance peut être considérée comme un moyen d'étendre les politiques de sécurité afin de réduire de manière significative les coûts étant donné que le fait de faire confiance permet de partager la gestion du système de sécurité avec les tiers de confiance.

4.4.2 les garde-fous de la confiance informatique

Les outils définis dans cette section représentent l'équivalent des garde-fous cités précédemment (section 4.3).

La signature numérique : Ce mécanisme permet de générer des documents équivalents aux contrats mais sur support numérique. Ainsi, une fois un document signé, le signataire ne peut ni contester ni répudier son contenu.

Le certificat : Le certificat permet d'exprimer la relation de confiance. C'est une attestation qui est délivrée par une entité à une personne, à une ressource ou à un service, reconnu comme digne de confiance. Ainsi, en utilisant son certificat, une entité peut se voir accorder la confiance d'une autre, si cette dernière reconnaît le signataire de cette attestation.

Les autorités de certification : Une autorité de certification représente une structure de référence (ex : société, service administratif, etc.) qui est habilitée à délivrer des certificats afin que ces derniers soient reconnu par les différents acteurs de l'environnement.

Outils de sécurité : Ces outils sont les firewalls, les antivirus, les anti-spywares etc. Ils se présentent soit sous forme logicielle, soit sous forme matérielle intégrée directement dans des équipements réseaux tels que les routeurs. Ce sont des solutions qui permettent de protéger les machines sur lesquelles ils sont connectées ou installées. Ainsi, le fait d'avoir ces applications procure à l'utilisateur une certaine assurance qui lui permet de se connecter au réseau, de naviguer sur Internet et d'échanger des données de manière sécurisée.

La réputation : Cette notion est actuellement très répandue à travers le web (ex : les sites marchands, les réseaux sociaux etc.). Le principe consiste à impliquer toute la communauté dans l'évaluation des différents acteurs de l'environnement en utilisant des mécanismes d'annotation basés sur le partage d'expériences. Ainsi, le cumul des évaluations permet d'identifier les entités malveillantes dès lors que celles-ci adoptent un comportement hostile. Un tel raisonnement ne se conçoit que lors d'interactions répétées en enrichissant un historique.

4.4.3 La confiance dans la recherche

La confiance constitue un paradigme qui est largement étudié dans le domaine de la recherche informatique. En effet, celle-ci constitue un aspect fondamental dans la mise en place de relations entre domaines [Abdu 97, Mars 94, Beth 94]. Ainsi, nous présentons, dans ce qui suit, les pistes, les plus proche de nos travaux, développées par la communauté scientifique.

4.4.3.1 La transitivité dans la confiance

La transitivité dans la confiance joue un rôle important au sein de réseaux ayant une topologie pair à pair ou adhoc. Elle permet à toute entité d'étendre sa confiance à partir de son ensemble de connaissances. Dans le domaine de l'informatique, cette transitivité est évaluée soit de manière sémantique soit de manière arithmétique.

- **Evaluation sémantique :** Cette évaluation est, en général, effectuée par l'utilisation de labels (ex : bon, moyen, mauvais) ou de chaînes de certificats dans lesquelles chaque certificat représente, en lui-même, une relation de confiance liant le signataire au propriétaire du certificat. Ce dernier peut exprimer toute sorte de relation, par exemple : une confiance relationnelle (ami), une relation d'appartenance, etc.

- **Evaluation arithmétique :** Dans ce type d'évaluation, chaque relation pair à pair est quantifiée par un ou plusieurs paramètres numériques. Une fonction mathématique est alors définie afin de procéder à la composition ou à l'agrégation des différentes évaluations des relations intermédiaires.

Etant donné que la propagation de la confiance constitue un besoin très important pour l'élaboration de notre système Chameleon, la section 4.5 présente en détail un certain nombre de solutions traitant ce type de relation.

4.4.3.2 La réputation :

Dans un réseau décentralisé (ex : pair à pair), chaque utilisateur à la possibilité de communiquer ses opinions sur les informations qu'il a acquises ou les transactions qu'il a effectuées avec les autres pairs. Ces opinions personnelles peuvent être collectées, échangées et évaluées, afin d'être utilisées comme indicateurs pour la recherche d'information pertinente (ex : PagesRanking de Google[Page 99]), ou bien pour la recommandation d'entités fiables (ex : PeerTrust[Xion 03], Poblano[Chen 03], Advogato[Levi 09], Appleseed[Zieg 04], P2PRep[Arin 06]etc.). Ces solutions s'intègrent dans des environnements sociaux, appelés communément "Web-of-Trust"[Wiki 09b]. Ainsi, la confiance prend une dimension sociale, ce qui implique que chaque participant, de cette configuration décentralisée est tenu de se comporter correctement afin d'avoir bonne réputation. D'autres systèmes tel que [Hasa 08] définissent des mécanismes permettant de déceler les recommandations mensongères.

4.4.3.3 La négociation

La négociation est une approche prometteuse pour l'établissement de la confiance dans les systèmes ouverts, où des entités, qui ne se connaissent pas au préalable, peuvent communiquer des informations sensibles. Aujourd'hui, la vie privée est une des préoccupations majeures des usagers, surtout lors d'échanges d'informations sur le Web. Par conséquent, nous partageons

l'avis de [Seam 04], selon lequel les systèmes de négociation doivent aborder efficacement les questions de confidentialité des données personnelles afin d'être largement utilisés.

Les solutions existantes (ex : TrustBuilder[Seam 04], Trust-X[Bert 04], [Haid 07] etc.) mettent en place un protocole et une politique de négociation ne permettant que la divulgation de l'information pertinente qui est généralement présentée dans des certificats. En effet, le certificat constitue un moyen fiable pour véhiculer tout type d'information comme, par exemple, le statut, les privilèges, l'affiliation, les règles de sécurité, etc. De plus, une fois cette attestation signée, celle-ci présente l'avantage d'être vérifiable et infalsifiable. Ainsi, un certificat doit être considéré comme une ressource sensible qui doit être protégée.

Lors d'une négociation, trois types d'approches peuvent être effectuées [Seam 04] :

- Approche naïve : Celle-ci consiste à présenter tous les certificats qu'on possède.
- Approche par rejeu : Dans cette approche, seuls les certificats non sensibles sont présentés en premier ; ensuite, si besoin est, on envoie le reste des certificats.
- Approche par découverte : Afficher, en premier, sa politique de négociation, puis ne divulguer que les certificats nécessaires pour établir la relation de confiance, selon les exigences communes définies par les politiques des deux parties [Bert 04, Haid 07].

Dans le cadre de la thèse, nous nous sommes intéressés à élaborer un modèle de confiance qui permet la collaboration (pair à pair) entre les différentes organisations. Pour cela, nous présentons dans la section suivante une étude plus approfondie des modèles de gestion de la confiance intégrant la transitivité.

4.5 Modèles d'évaluation de la confiance transitive

4.5.1 PGP

PGP (Pretty Good Privacy)[Zimm 95] est un crypto-système mis en place par Philip Zimmermann entre 1984 et 1991. Dans PGP, tout utilisateur est identifié par une clé publique, et peut agir comme une autorité de certification. Un certificat d'identité peut être délivré et signé par un ou plusieurs usagers. Toutefois, une clé publique certifiée n'est valide que si le ou les signataires du certificat sont reconnus comme dignes de confiance.

Dans PGP, chaque utilisateur possède un trousseau de clés publiques, où chacune des clé est annotée par l'utilisateur comme suit :

- Confiance totale : la clé est considérée comme très sûre.
- Confiance partielle : la clé est moyennement sûre.
- Pas de confiance : la clé n'est pas sûre.

De même, chaque clé publique certifiée est considérée par celui qui évalue le certificat la contenant comme étant :

- Complètement valide, s'il fait complètement confiance au certificat.
- Partiellement valide, s'il a partiellement confiance dans le certificat.
- Invalide : s'il n'a aucune confiance dans le certificat.

Chaque usager défini alors localement deux seuils :

- COMPLETS_NEEDED : le nombre minimum de signatures produites par des clés ayant une confiance totale, afin de valider la clé publique certifiée.
- MARGINALS_NEEDED : le nombre minimum de signatures produites par des clés ayant une confiance partielle, afin de valider la clé publique certifiée.

Sachant que NBT et NBM représentent le nombre des signatures, contenues dans le certificat, annotant respectivement une confiance totale (NBT) et une confiance partielle (NBM), le processus de validation est le suivant :

- La *clé publique* est considérée comme étant *complètement valide* si :

  $(NBT \geq COMPLETS\_NEEDED)$ ou $(NBM \geq MARGINALS\_NEEDED)$,

- sinon, la *clé publique* est considérée comme étant *partiellement valide* si :

  $(0 < NBT < COMPLETS\_NEEDED)$ ou $(0 < NBM < MARGINALS\_NEEDED)$,

- sinon, la *clé publique* est considérée comme étant *invalide* si :

  $NB = 0$ et $NBM = 0$.

Dans la version Windows de PGP, les deux variables suivantes sont initialisées par défaut comme suit :

- $COMPLETS\_NEEDED = 1$
- $MARGINALS\_NEEDED = 2$

Ce qui signifie qu'il suffit d'**une** signature complètement valide pour que la clé publique soit considérée comme complètement valide, et de **deux** signatures marginales valides pour en faire une clé complètement valide.

PGP permet de propager la confiance en validant des chaînes de signature. Cependant, les labels de confiance (complet, partiel) offrent une évaluation limitée. De plus, le système permet de créer des chaînes extrêmement longues sans pouvoir contrôler ni connaître le nombre de tiers de confiance.

4.5.2 DTM

A Distrust Trust Model [Abdu 97], est l'un des premiers modèles à évaluer numériquement la confiance dans les environnements distribués. Dans ce système, la confiance est évaluée par deux paramètres :

- Direct Trust 'dTrust' : celle-ci évalue la confiance qui est attribuée à une entité afin d'exécuter une tâche bien précise.
- Recommander Trust 'rTrsut' : celle-ci évalue la confiance qu'a une entité dans les recommandations d'une autre.

Ces deux paramètres de confiance sont évalués sur une échelle allant de -1 à 4. Ainsi, une entité malveillante se voit accorder la valeur -1, une entité inconnue, la valeur 0 et enfin une entité de confiance, une valeur entière comprise entre 1 et 4 (1 : confiance minimum, 2 : confiance moyenne, 3 : bonne confiance et 4 : confiance maximale).

Soit les entités $e_0, e_1, ..., e_n$, qui se font confiance de manière pair à pair, i.e. e_0 fait confiance à e_1, e_1 fait confiance à $e_2, ..., e_{n-1}$ fait confiance à e_n. Cet enchaînement permet à l'entité e_0 de faire confiance de manière transitive à e_n afin d'exécuter une tâche particulière, l'évaluation de la confiance transitive $vTrust(s_0, s_n)$ est calculée par la fonction suivante :

$$vTrust(e_0, e_n) = \prod_{i=0}^{n-2} \frac{rTrust(e_i, e_{i+1})}{4} * dTrust(e_{n-1}, e_n).$$

Comme le montre l'équation ci-dessus, l'évaluation de la confiance transitive consiste à pondérer la confiance du nœud (dTrust) qui connaît directement l'entité capable de répondre à la requête, avec un coefficient compris dans l'intervalle [0,1] (rTrsut). Ce dernier représente une valeur qui permet d'évaluer la confiance qu'on peut avoir dans les recommandations des différents nœuds de la chaîne.

Le modèle DTM permet d'évaluer numériquement la confiance transitive entre deux entités. Cette valeur est liée à la longueur de la chaîne. Cependant, dans DTM la disposition à la confiance n'est pas prise en compte puisque les seuils utilisés sont les mêmes pour toutes les entités du réseau. Par exemple : le valeur '3' représentera toujours une bonne confiance, alors qu'une entité très confiante peut considérer cette valeur comme une confiance maximale.

4.5.3 PTM

PTM [Alme 04] (Pervasive Trust Management Model) est un système décentralisé de la gestion de la confiance destiné aux environnements pervasifs. Il a été implémenté dans le cadre du modèle de contrôle d'accès TrustAC (voir chapitre 2). PTM permet aux différentes entités du réseau de gérer de manière autonome leurs relations de confiance. Celui-ci est fortement inspiré de DTM. Dans PTM, la confiance est évaluée dans l'intervalle [0,1]. Cependant, la confiance évolue dans le temps, ce qui a motivé les concepteurs de ce modèle à intégrer le comportement passé de l'entité évaluée, comme suit :

$$Conf_{new}(A, C) = \begin{cases} P * \beta + Conf_{act}(A, C)(1 - \beta) & P > 0 \\ 0 & sinon \end{cases}$$

Tel que P représente le comportement passé de l'entité C, et β est un paramètre variable défini par C, qui permet de fixer comment le comportement passé influe le présent.

L'avantage de PTM est que la valeur de confiance n'est pas figée et évolue dans le temps. Cependant, à l'instar de DTM, la disposition à la confiance n'est pas implémentée.

4.5.4 Beth et al.

Beth et al. [Beth 94] proposent un modèle de confiance probabiliste, qui ressemble à PTM pour évaluer la relation de confiance directe. La confiance est évaluée dans l'intervalle [0,1] et est calculée en fonction du nombre de transactions positives et négatives effectuées dans le passé. Ainsi, la confiance que A peut avoir en B est calculée comme suit :

$$Trust(A,B) = \begin{cases} 1-\alpha^{p-n} & si\ p > n \\ 0 & sinon \end{cases}$$

où p et n représentent respectivement le nombre des activités positives et négatives observées par l'entité A sur B, et α représente le seuil de confiance minimum.

Dans le même esprit, en prenant trois entités A, B et C tel que A fait confiance à B et B fait confiance à C, le calcul de la transitivité est effectué comme suit :

$$Trust(A,C) = Trust(A,B) \odot Trust(B,C) = 1 - (1 - Trust(B,C))^{Trust(A,B)}$$

Par rapport aux approches précédentes, ce modèle permet d'initialiser la confiance de départ, de chaque entité, à zéro, ensuite faire évaluer cette valeur en fonction du comportement de celle-ci. Cependant, nous considérons que la confiance de départ peut être évaluée grâce aux attributs de chaque entité, à savoir le domaine de compétence, les ressources mises à dispositions, etc. De plus, comme pour les approches précédentes, la subjectivité dans la confiance n'est toujours pas considérée.

4.5.5 Josang et al.

Josang et al[Josa 05], présentent une approche qui permet d'évaluer la confiance en utilisant la logique subjective. Cette logique intègre explicitement l'incertitude dans le processus d'évaluation de la croyance. Josang et al. définissant une fonction nommée : 'Belief fonction' qui représente la croyance (la confiance) d'une entité A en une entité B sous forme de tûple $\omega_B^A(b_B^A, d_B^A, u_B^A)$ tel que :

- b_B^A, d_B^A, u_B^A représentent respectivement la croyance (belief), la défiance (disbelief) et l'incertitude (uncertainty).
- $b_B^A, d_B^A, u_B^A \in [0,1]$ et $b_B^A + d_B^A + u_B^A = 1$

Dans ce modèle, la confiance initiale est calculée en fonction du comportement de chaque nœud. Sachant que r et s représentent respectivement l'ensemble des actions positives et négatives de B observées par A, l'évaluation $\omega_B^A(b_B^A, d_B^A, u_B^A)$ est calculée comme suit :

$$\omega_B^A = \begin{cases} b_B^A = \frac{r}{r+s+2} \\ d_B^A = \frac{s}{r+s+2} \\ u_B^A = \frac{2}{r+s+2} \end{cases}$$

Ainsi, une entité B sans historique est évaluée par toute entité X du réseau avec le tuple $\omega_B^X = (0,0,1)$ qui correspond à l'incertitude maximale. Ensuite, en fonction des actions effectuées, celle-ci peut gagner en confiance ou en méfiance.

Jasong et al. ont défini principalement deux approches pour évaluer une confiance transitive. En prenant l'exemple des sites A, B et C, et sachant que A fait confiance à B et que B fait confiance à C, la confiance transitive que A a en C est calculée comme suit :

- Transitivité favorisant l'incertitude :

$$\omega_C^A = \omega_B^A \otimes \omega_C^B = \begin{cases} b_C^A = b_B^A * b_C^B \\ d_C^A = b_B^A * d_C^B \\ u_C^A = d_B^A + u_B^A + b_B^A * u_C^B \end{cases}$$

– Transitivité favorisant la confiance opposée :

$$\omega_C^A = \omega_B^A \bar{\otimes} \omega_C^B = \begin{cases} b_C^A = b_B^A * b_C^B + d_B^A * d_C^B \\ d_C^A = b_B^A * d_C^B + d_B^A * b_C^B \\ u_C^A = u_B^A + (b_B^A + d_B^A) * u_C^B \end{cases}$$

Ainsi, en choisissant $\omega_B^A = (0.1, 0.9, 0)$ et $\omega_C^B = (0.2, 0.8, 0)$, la première approche tend vers l'incertitude ($\omega_C^A = (0.02, 0.08, 0.9)$) puisque les recommandations viennent d'entités auxquelles la valeur d est proche de 1. Cependant, la seconde approche consiste à tirer partie des opposés qui s'attirent (l'ennemi de mon ennemi est mon ami) ($\omega_C^A = (0.74, 0.26, 0)$).

A l'instar de Josang et al., Theodorakopoulos et al. [Theo 04], définissent un modèle d'évaluation de la confiance basé sur deux valeurs t et c représentant respectivement Trust (confiance) et Confidence (incertitude, confiance en soi). Ainsi, dans ce modèle la confiance $\omega_B^A = (t_B^A, c_B^A)$.

La confiance transitive est alors calculée comme suit :

$$\omega_C^A = \omega_B^A \otimes \omega_C^B = \begin{cases} t_C^A = t_B^A * t_C^B \\ c_C^A = c_B^A * c_C^B \end{cases}$$

Dans ces modèles la subjectivité dans la confiance est définie par l'incertitude dans l'évaluation de la confiance. Cependant, ces deux approches ne considèrent pas la disposition à la confiance qui représente un des plus important aspects de la confiance subjective.

4.6 Synthèse

Les modèles de confiance illustrés précédemment présentent principalement les limites suivants :

1. Evaluation de la confiance initiale : en effet, autant il est simple d'attribuer des étiquettes sémantiques (ex : bon, moyen, mauvais) (DTM, PGP) autant il est plus complexe de donner une évaluation numérique. Ainsi, la plupart des solutions qui traitent de la confiance font l'une des suppositions suivantes :

   – La procédure d'évaluation de la confiance directe n'est pas un problème du modèle, c'est du ressort de l'acteur du système (e.x. PTM). La délégation de cette tâche n'est cependant pas forcément évidente surtout quand il s'agit d'évaluer numériquement un tiers de confiance (e.x : 0.1, 0.2, ou bien 0.15 ?).

   – La confiance de départ est initialisée par la valeur nulle (Beth et al. et Josang et al.). Cette solution est, dans certains cas, intéressante mais ne reflète pas la réalité, car un usager donné possède une identité, un statut, une appartenance etc. ; ce sont ces attributs qui doivent constituer le point de départ de tout modèle de confiance.

2. Non prise en charge de la subjectivité dans l'évaluation de la confiance. En effet, l'évaluation de la confiance diffère d'une personne à une autre, elle dépend principalement de la subjectivité (la personnalité) de chaque entité. Dans l'état de l'art illustré précédemment seules les deux dernières approches [Josa 05] [Theo 04] traitent un aspect de la subjectivité (incertitude) dans l'évaluation de la confiance (directe et transitive). Cependant, cette

valeur ne permet pas d'évaluer la disposition à la confiance de chaque entité.

4.7 Conclusion

Dans les environnements distribués, la confiance constitue l'un des plus importants moyens d'élargir le champ d'accès des différentes politiques de sécurité. Que ce soit entre les utilisateurs, ou entre les organisations, les différentes solutions illustrées par l'état de l'art œuvrent à créer des mécanismes de sécurité flexibles permettant le partage et l'échange de l'information, des services et des ressources de la manière la plus sûre et intuitive possible. Cependant, nous considérons que dans l'état de l'art la subjectivité n'est pas suffisamment prise en compte dans l'évaluation de la confiance de départ ni dans le calcul de la confiance transitive. C'est pour cela que dans le chapitre suivant, nous allons introduire notre propre modèle de confiance qui intègre dans sa conception les paramètres suivants :

1. Evaluation de la disposition à la confiance (la subjectivité),
2. Evaluation de l'entité d'appartenance,
3. Evaluation de la transitivité de manière personnalisée.

Chapitre 5

Le modèle de confiance "T2D" : Trust to Distrust

Sommaire

5.1 Introduction

La confiance est un élément important pour tisser des relations et établir des collaborations et des échanges entre les différents acteurs de l'environnement. Comme on a pu le voir dans le chapitre 2, ce mécanisme permet de construire et d'élargir, à partir d'un ensemble de connaissances,

le cercle d'activités de tout un chacun. Ainsi, l'utilisation de la confiance permet de partager plus de ressources, de propager de l'information et d'interconnecter les différentes communautés.

Dans le chapitre précèdent, nous avons remarqué que les modèles d'évaluation de la confiance présentent certaines lacunes, surtout concernant la disposition de chaque entité à évaluer son environnement. En effet, la personnalité (confiante ou méfiante) de la personne qui a la charge de la politique de sécurité (propriétaire d'une ressource ou bien administrateur) influe fortement sur l'évaluation de la confiance directe ou transitive des entités de l'environnement.

Dans ce qui suit, nous allons proposer un modèle d'évaluation de la confiance (directe et transitive), fondé sur la disposition à la confiance (la personnalité) propre à chaque site du réseau. Cependant, avant d'introduire notre contribution nous allons définir et modéliser la notion de confiance.

5.2 La confiance

5.2.1 Définition de la confiance

5.2.1.1 Définition de la relation $Trust$

Soit $\mathbb{S}$ un ensemble de sites. Considérons deux entités A et B de l'ensemble $\mathbb{S}$, (i.e. : $A \in \mathbb{S}, B \in \mathbb{S}$).

Si A fait confiance à B, une relation de confiance lie ces deux entités. Elle est notée : $A\ Trust\ B$

5.2.1.2 Propriétés de la relation $Trust$

Réflexive : La relation Trust est réflexive, car toute entité a une confiance totale en elle-même.
$\forall A \in \mathbb{S}, A\ Trust\ A$

non-Symétrique : La relation $Trust$ peut n'est pas symétrique, puisque chaque site gère indépendamment sa politique et n'est pas dans l'obligation d'établir une relation de confiance réciproque.

Soit $A, B \in \mathbb{S}$, $A\ Trust\ B \nRightarrow B\ Trust\ A$

Transitive : Pour l'expansion du modèle de confiance, la relation "Trust" est transitive :

$\forall A, B\ et\ C \in \mathbb{S}, A\ Trust\ B \wedge B\ Trust\ C \Rightarrow A\ Trust\ C$

Cette propriété est fondamentale pour le fonctionnement de notre proposition. Car, grâce à la transitivité récursive, une entité peut évaluer une autre sans la connaître directement.

5.2.2 Evaluation de la confiance

5.2.2.1 Types de confiance

La relation de confiance n'est pas booléenne (faire confiance ou pas), mais doit être évaluée et graduée, principalement pour deux raisons :

– Donner plus de flexibilité et de précision au processus d'évaluation.

– Permettre une transitivité évaluée, bornée et contrôlée.

Ainsi, deux types de confiance sont définis par des fonctions d'évaluation, comme suit :

– **La confiance directe :** Elle permet à chaque entité d'évaluer son entourage proche. Cette évaluation peut être effectuée sur une échelle graduée (0, 1, 2, 3, 4, etc), sur un intervalle défini entre 0 et 1, etc.

– **La confiance transitive :** Elle permet d'établir une relation entre deux entités qui ne se connaissent pas directement. Dans ce cas de figure, des intermédiaires contribuent à la construction d'une chaîne permettant de lier, de manière transitive, les deux entités (voir section 5.2.3.3). Cette transitivité peut être évaluée en utilisant les valeurs de confiance des liens intermédiaires.

5.2.2.2 Choix de la fonction d'évaluation de la confiance

Comme illustré dans le chapitre précédent, il existe dans la littérature un certain nombre de fonctions qui permettent d'évaluer la confiance entre deux entités se connaissant directement ou indirectement. Dans le but de sélectionner, modifier ou définir un système d'évaluation de ce type de relation nous avons choisi trois critères de sélection :

– L'atténuation : l'évaluation doit être calculée en fonction de la longueur de la chaîne de confiance. La valeur de confiance doit alors diminuer si la longueur de la chaîne augmente.

– La flexibilité : Chaque fonction de confiance est définie à l'aide d'un certain nombre de paramètres. Ils doivent pouvoir être initialisés ou modifiés de manière simple et logique et non pas imposés par le système.

– Le comportement : le système d'évaluation doit prendre en considération la disposition de chaque entité à faire confiance (entité confiante ou méfiante).

Ainsi, fondé sur ces trois critères, nous allons présenter dans la section 5.3 notre proposition d'une fonction de confiance.

5.2.3 Modélisation de la relation de confiance

Avant d'introduire notre fonction d'évaluation de la confiance, nous allons introduire quelques définitions nécessaires afin d'illustrer le fonctionnement de la confiance au sein d'un réseau.

5.2.3.1 Graphe de confiance

Un réseau de confiance peut être représenté sous forme de graphe orienté et valué, noté $G(\mathbb{S}, \mathbb{E})$ comme suit :

– $\mathbb{S}$ représente l'ensemble des sites (entités) de l'environnement.

– $\mathbb{E}$ représente l'ensemble des arcs orientés (A, B) liant deux entités du graphe. L'existence d'un arc (A, B) signifie que $A\ Trust\ B$.
$\mathbb{E} = \{(A, B)/A\ Trust\ B\}$

– Enfin, chaque arc (A,B) est valué par une valeur unique représentant le degré de confiance liant le site A au site B.

5.2.3.2 Cercle de Confiance

Afin de construire le graphe de confiance, chaque site doit définir ses deux cercles de confiance entrant et sortant (voir figure 5.1), comme suit :

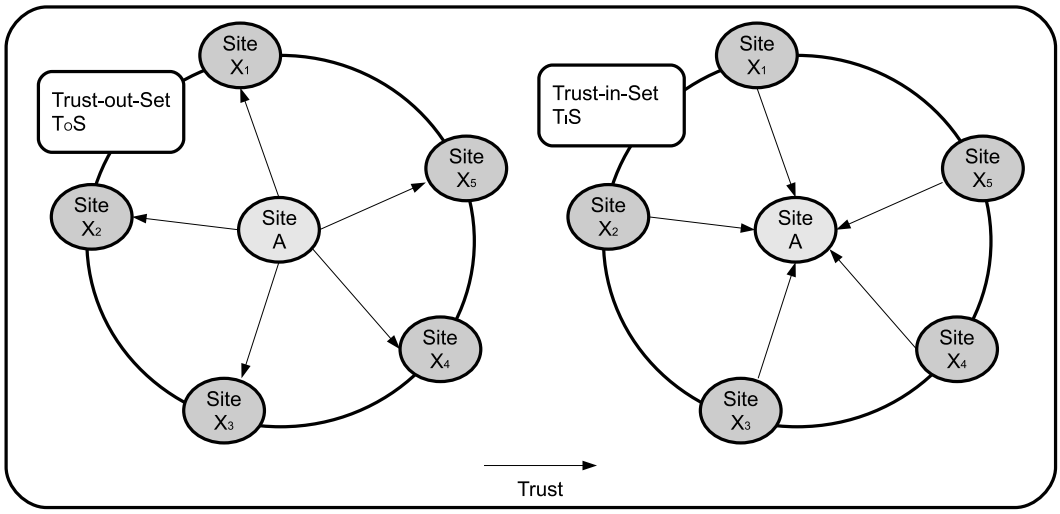

FIGURE 5.1 – Cercles de confiance

Soit $A \in \mathbb{S}$,

$T_oS(A) = \{X \in \mathbb{S}/(A\ Trust\ X)\}$

$T_oS(A)$ représente le cercle de confiance sortant (Trust-out-Set) du site A. Il regroupe tous les sites (X) auxquels A fait confiance. Les membres de ce cercle sont nommés **les sites de confiance** de A.

$T_iS(A) = \{X \in S/(X\ Trust\ A)\}$

$T_iS(A)$ représente le cercle de confiance entrant (Trust-in-Set) du site. Il englobe tous les sites (X) faisant confiance à A, i.e. tous les sites qui considèrent A comme un **site de confiance**.

5.2.3.3 Chaîne de confiance

Une chaîne de confiance permet de lier par transitivité deux sites distincts même s'ils ne se connaissent pas. Cette chaîne est composée d'un ensemble d'intervenants intermédiaires liés deux à deux par la relation "Trust".

– Soit $n \in [0, |\mathbb{S}|]$
– Soit $s_0, s_1, ..., s_n \in \mathbb{S} \mid \forall i = 0, ..., n-1\ (s_i\ Trust\ s_{i+1})$.
– Alors $Pt_n = (s_0, s_1, ..., s_n)$ est appelée chaîne de confiance, avec :

- s_n qui représente le nœud à évaluer et il est nommé **site source** car il représente le site d'appartenance de l'utilisateur nomade.
- s_0 qui représente le nœud qui cherche à évaluer la source s_n. Il est noté **site cible**, car il représente le site dans lequel l'usager nomade sollicite un accès.

5.3 T2D "Trust To Distrust" : De la confiance à la méfiance

Le nom T2D (de la confiance à la méfiance) est inspiré du critère d'atténuation (plus il y a d'intervenants pour construire la chaîne, plus la confiance va diminuer et par là-même la méfiance va augmenter). Notre modèle intègre les deux notions de confiance directe et transitive qui sont respectivement initialisées (critère de faisabilité) et évaluées en fonction d'un mécanisme qui prend en charge la personnalité (critère du comportement) de chaque nœud du graphe de confiance.

Ainsi, dans ce qui suit, deux types de fonctions vont être introduites : ceux pour l'évaluation de la confiance directe et ceux pour l'évaluation de la confiance transitive.

Evaluation de la confiance directe Le calcul de la confiance directe se fait en deux étapes comme suit :

- Définir la fonction d'initialisation "t^0" pour évaluer la confiance initiale (directe).

- Ensuite, définir les fonctions de comportement (**D**isposition to **T**rust) afin d'automatiser le processus d'initialisation à l'aide de la personnalité de chaque entité.

Evaluation de la confiance transitive Comme pour la confiance directe, le processus d'évaluation de la confiance transitive se fait également en deux étapes :

- Définir la fonction de propagation P^0 pour évaluer la confiance transitive. P^0 est le résultat d'une concaténation de plusieurs fonctions t^0.

- Ensuite, définir les fonctions qui permettent d'adapter l'évaluation de la chaîne de confiance à l'aide de la personnalité du site cible. Cette procédure porte le nom de **D**isposition to **T**ransitivity.

5.4 La Confiance directe

Il existe deux visions distinctes pour initier le réseau de confiance. La première est basée sur la réputation. Elle suppose qu'une entité donnée intègre le réseau avec une confiance nulle, qui est amenée à augmenter ou à diminuer en fonction des actions ou des transactions effectuées. La seconde est connue sous le nom de confiance initiale. Au départ, elle est non nulle et peut être initiée par des conventions, des contrats ou des relations.

Après étude de l'existant, nous considérons que les deux approches sont complémentaires. Effectivement, l'initialisation de la confiance ne commence généralement pas à partir de zéro, elle peut être initialisée, car intégrer un réseau social suppose souvent un minimum de relations

comme point de départ. Par contre, cette confiance est amenée à évoluer dans le temps en utilisant, par exemple, des mécanismes basés sur la réputation.

Dans ce qui suit, nous allons définir une fonction qui permet de quantifier et d'évaluer cette confiance initiale.

5.4.1 La fonction d'initialisation t^0

La fonction t^0 permet à chaque site d'évaluer les membres de son cercle de confiance sortant. Elle attribue une valeur comprise entre 0 et un seuil de méfiance local noté (T_A^0), qui représente la borne maximum de méfiance fixée par la politique locale du site A. Le seuil de méfiance local est une valeur entière variable d'un site à un autre, elle peut être égale à 1, à un nombre d'états de confiance, à un nombre de sites de confiance, etc. La liberté du choix du seuil est voulue. Cette contrainte a été intégrée dans notre modèle dès le départ, dans le but de donner plus d'autonomie aux administrateurs du réseau.

La fonction t^0 est définie comme suit :

$t^0 : \mathbb{E} \rightarrow \mathbb{R}^+$ $\quad \mathbb{E}$: l'ensemble des arcs du graphe $G(\mathbb{S}, \mathbb{E})$.
$(A, B) \rightarrow d$ $\quad \mathbb{R}^+$: l'ensemble des réels positifs.

$t^0(A, B) = d \ / \ 0 \leq d \leq T_A^0$

Cette fonction quantifie le degré de confiance entre deux entités. Comme le montre la figure 5.2, quand d augmente, la méfiance augmente et la confiance diminue. Ceci se traduit par :

– $t^0(A, B) < t^0(A, C)$: A fait plus confiance à B qu'à C (i.e. A se méfie plus de C que de B).

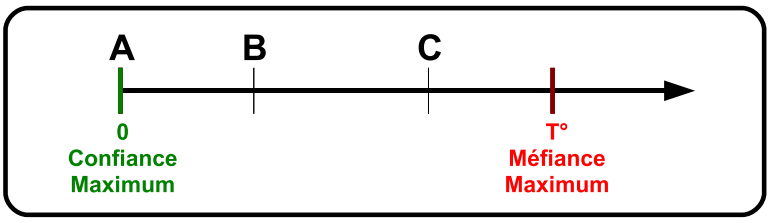

FIGURE 5.2 – L'échelle T2D de l'entité A

Propriété de la fonction t^0 : Conformément à la relation Trust, la fonction t^0 présente les propriétés suivantes :

– ***Self trust :*** $\forall A \in \mathbb{S}, t^0(A, A) = 0$

– ***Non-commutativité :***
Soit $A, B \in \mathbb{S}$, $t^0(A, B) = d_1 \wedge t^0(B, A) = d_2 \nRightarrow d_1 = d_2$

i.e. $t^0(A, B)$ est indépendant de $t^0(B, A)$.

5.4.2 Fonction de comportement : Dispostion to Trust

En appliquant la fonction d'initialisation t^0, chaque site X va procéder à l'évaluation de son cercle de confiance sortant $T_oS(X)$ en attribuant à chaque site de confiance une valeur comprise entre 0 et T_X^0. Comme il a été précisé dans le chapitre précédent, l'attribution de cette valeur numérique à un tiers de confiance dépend de la personnalité de la personne qui évalue. Ainsi, notre approche consiste à définir une nouvelle méthode permettant d'automatiser le calcul de la confiance directe en se basant principalement sur la disposition de chacun à accorder sa confiance.

La méthode proposée consiste à mettre en place un système d'évaluation, comme suit :

1. Mise en place d'un tri du cercle de confiance sortant (Trust Sort).
2. Modélisation mathématique du comportement afin d'exprimer la disposition à la confiance (la fonction Behavior).

L'application de cette méthode permet la création d'une confiance unique qui n'est pas biaisée par la personnalité de l'évaluateur humain(administrateur) mais basée sur une disposition uniforme de la confiance qui est propre à la politique du site.

5.4.3 Trust Sort

Pour faciliter l'uniformisation dans l'évaluation de la confiance directe, nous avons défini le processus $TrustSort$. Le $TrustSort$ est un mécanisme de tri qui permet non pas d'attribuer des valeurs mais de procéder à un ordonnancement, ce qui est largement moins contraignant qu'une évaluation numérique.

Pour le "Trust Sort" deux systèmes de tri ont été définis :

- Le tri unitaire.
- Le tri groupé.

5.4.3.1 Le tri unitaire

Le principe de ce tri est de prendre tous les éléments du cercle de confiance sortant un par un, et de les classer dans l'ordre du site auquel on fait le plus confiance au site auquel on fait le moins confiance. Cet ordonnancement se fait ainsi sur une échelle allant de la confiance à la méfiance. Les membres du T_oS sont alors placés de manière **uniforme** dans l'intervalle $]0, |T_oS|[$.

On note, "$TrustSort_A$" la liste triée définie par l'entité 'A'. Les indices de cette liste représentent les membres du cercle de confiance sortant. La valeur contenue dans chaque case correspond à leur classement dans la liste.

Exemple : Prenons l'exemple du site 'A' qui a un cercle de confiance sortant composé de 6 sites comme suit :

$T_oS = \{Site1, Site2, Site3, Site4, Site5, Site6\}$

Le site A trie son T_oS dans l'ordre suivant :

(Confiance Max)(+). **Site 4. Site 3. Site 5. Site 1. Site 2. Site 6** (-)(Confiance Min) .

Ainsi, comme le montre la figure 5.3 la disposition uniforme des 6 sites dans l'intervalle]0,6[, attribue à chaque membre du T_oS la valeur suivante :

$TrustSort_A[Site1] = 3,5$, $TrustSort_A[Site2] = 4,5$, $TrustSort_A[Site3] = 1,5$
$TrustSort_A[Site4] = 0,5$, $TrustSort_A[Site5] = 2,5$, $TrustSort_A[Site6] = 5,5$

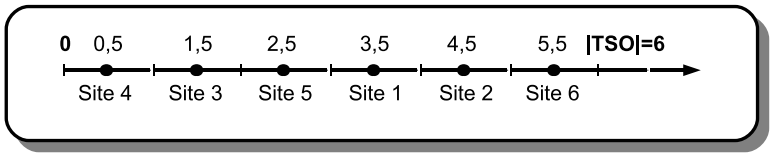

FIGURE 5.3 – Tri unitaire

5.4.3.2 Le tri groupé

Le tri unitaire est une approche assez intuitive mais qui peut être limitée quand il s'agit :
- de trier un grand nombre d'éléments.
- d'ordonner de manière plus spécifique et plus précise.

Pour combler ces limites, nous avons défini une variante du tri unitaire, le Tri groupé. Elle permet d'insérer une couche d'abstraction au-dessus du tri unitaire contenant des groupes d'éléments, comme suit :

- Définir les différents groupes (ex : bon, moyen, très bon, très mauvais, etc).
- Classer les groupes, par exemple :
  (Confiance Max)(+), Très bon, Bon, Moyen, Très mauvais, (-)(Confiance Min).
- Affecter les membres du T_oS au groupe correspondant.
- Trier chaque groupe en utilisant le tri unitaire.

Cette méthode permet d'un côté de réduire la complexité du tri, et d'un autre de représenter des schémas de tri plus complexes et plus précis.

Exemple : Afin d'illustrer les différentes possibilités, prenons l'exemple de trois sites A, B et C (voir figure 5.4) qui classent les 6 sites numérotés de S1 à S6, comme suit :

- les Sites A, B et C définissent trois groupes représentant une confiance forte, moyenne et faible.

- la confiance est évaluée sur trois plages :
  - Groupe à confiance forte de 0 à 2.
  - Groupe à confiance moyenne de 2 à 4.
  - Groupe à confiance faible de 4 à 6.

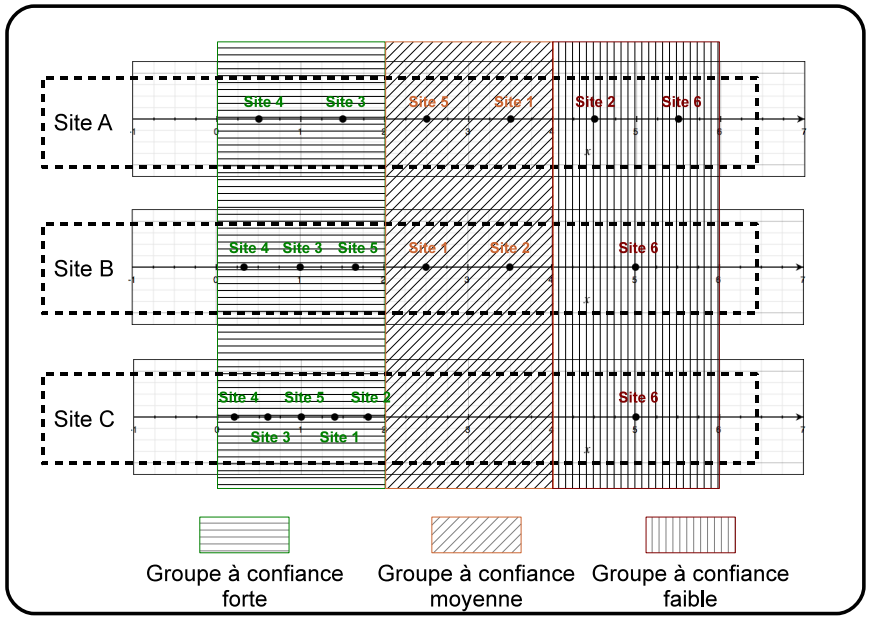

FIGURE 5.4 – Exemple de tri groupé

- Les sites (S1 à S6) sont rangés/classés dans leur groupe respectif comme suit :
  - Site A :
    - Groupe 1 : Site 4, Site 3.
    - Groupe 2 : Site 5, Site 1.
    - Groupe 3 : Site 2, Site 6.
  - Site B :
    - Groupe 1 : Site 4, Site 3, Site 5.
    - Groupe 2 : Site 1, Site 2.
    - Groupe 3 : Site 6.
  - Site C :
    - Groupe 1 : Site 4, Site 3, Site 5, Site 1,Site 2.
    - Groupe 3 : Site 6.

Noter que, de manière globale, A,B et C classent leur sites de confiance (1,2, ...,6) dans le même ordre [5].

5. Le classement dans le même ordre permet de mettre en évidence le découpage en groupe

	Site A	Site B	Site C
Site 4	0.5	0.33	0.2
Site 3	1.5	1	0.6
Site 5	2.5	1.66	1
Site 1	3.5	2.5	1.4
Site 2	4.5	3.5	1.8
Site 6	5.5	5	5

TABLE 5.1 – Exemple comparatif du trustsort groupé

L'exemple illustré par le tableau ci-dessus, montre que ce tri permet de donner plus de flexibilité dans l'évaluation. Comme on peut le constater, bien que que le Site 2 se trouve dans la 5éme position dans l'ordre général de tous les sites, il se voit attribuer les valeurs 4.5 , 3.5 et 1.8 en fonction du groupe auquel il est affecté ainsi que de sa position dans ce groupe.

5.4.4 Modélisation fonctionnelle du comportement

La disposition à la confiance (personnalité) est la propension inhérente d'un individu dans un certain contexte à faire confiance ou à se méfier. Dans le but de modéliser cette caractéristique du comportement, nous nous sommes intéressés à définir un modèle de fonctions de comportement. Ce dernier permet d'attribuer à chaque comportement une fonction mathématique qui prend en entrée une valeur de **confiance neutre** dépourvue de toute influence liée à la personnalité (TrustSort), et fournit en sortie une évaluation de la **confiance pondérée** par la personnalité qui est modélisée par la fonction utilisée.

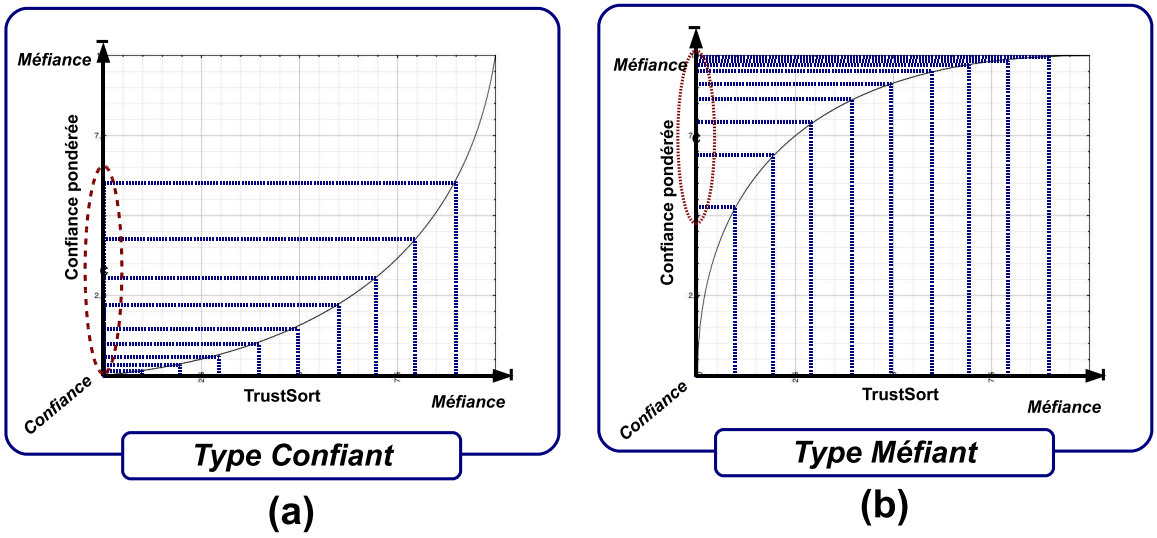

FIGURE 5.5 – Types de comportement

Dans la figure 5.5, en se plaçant sur une échelle allant de la confiance (valeur=0) à la méfiance (valeur=max), deux catégories de comportement correspondant à deux types de courbes ont été identifiées, comme suit :

1. **Type confiant :** Il représente les entités ayant une personnalité ouverte à caractère confiant. Nous définissons cette caractéristique par une fonction de forme convexe. Comme illustré dans la figure 5.5(a), ces entités ont tendance à faire confiance et à attribuer des valeurs plus proches du zéro (la valeur de confiance maximum) que de la valeur maximum de méfiance.

2. **Type méfiant :** Ce comportement est traduit par une courbe concave. Cette forme de courbe modélise le comportement d'entités à caractère méfiant, puisqu'elle attribue des valeurs de confiance plus proches de la valeur maximum de méfiance que de la valeur maximum de confiance (voir figure 5.5(b)).

Ainsi, le comportement d'une entité peut être modélisé à l'aide de courbes qui vont d'une forme convexe (type confiant) à une forme concave (type méfiant).

Nota : *Dans le but de corroborer notre modélisation du comportement, nous avons réalisé un petit questionnaire qui a été diffusé à travers la toile universitaire. Les résultats obtenus ont été très satisfaisant et ont étayé notre approche (voir Annexe B pour les détails de ce test).*

En prenant trois points (P_0, P_1, P_2), il est possible de dessiner une concavité ou une convexité (voir figure 5.6), comme suit :

– Le point d'origine : le point $P_0(0, 0)$.

– Le point repère : Il permet de fixer les bornes supérieures de la courbe. Il a pour coordonnées $P_2(h_x, h_y)$, où : $h_x = |T_o S(s_0)|$ correspond au nombre de sites de confiance de s_0, et $h_y = T^0_{s_0}$ le seuil de méfiance local de s_0.

– Le point de comportement : C'est le point le plus important, qui permet d'ajuster l'amplitude des courbes en les faisant passer d'une forme convexe à une forme concave. Ses coordonnées sont $P_1(b_x, b_y)$ (b pour Behavior). La figure 5.6 montre que quand le point P_1 se trouve dans l'espace méfiant (point $P_1^{\text{‘}}$) la courbe est concave, et quand il se situe dans l'espace confiant (point $P_1^{\text{“}}$) la courbe résultant a une forme convexe.

Les courbes de Bézier

Pour définir un schéma de fonctions qui modélise nos courbes concaves et convexes, nous nous sommes intéressés aux courbes paramétrées et plus précisément aux courbes de Bézier et aux B-Splines. Notre choix s'est porté sur les courbes de Bézier pour les raisons suivantes :

– De manière générale, les courbes paramétrées offrent une grande flexibilité dans le maniement et la définition de formes qui peuvent être obtenues à partir d'un certain nombre de points.
– Contrairement à une courbe de Bézier qui passe par le premier point (P_0) et le troisième point (P_2) en se rapprochant du second (P_1), une B-spline tend mais ne passe par aucun de ces points de contrôle [Cart 09]. Ce qui fait que les B-Splines ne répondent pas à nos besoins étant donné que les courbes qu'on veut obtenir doivent au moins passer par le point

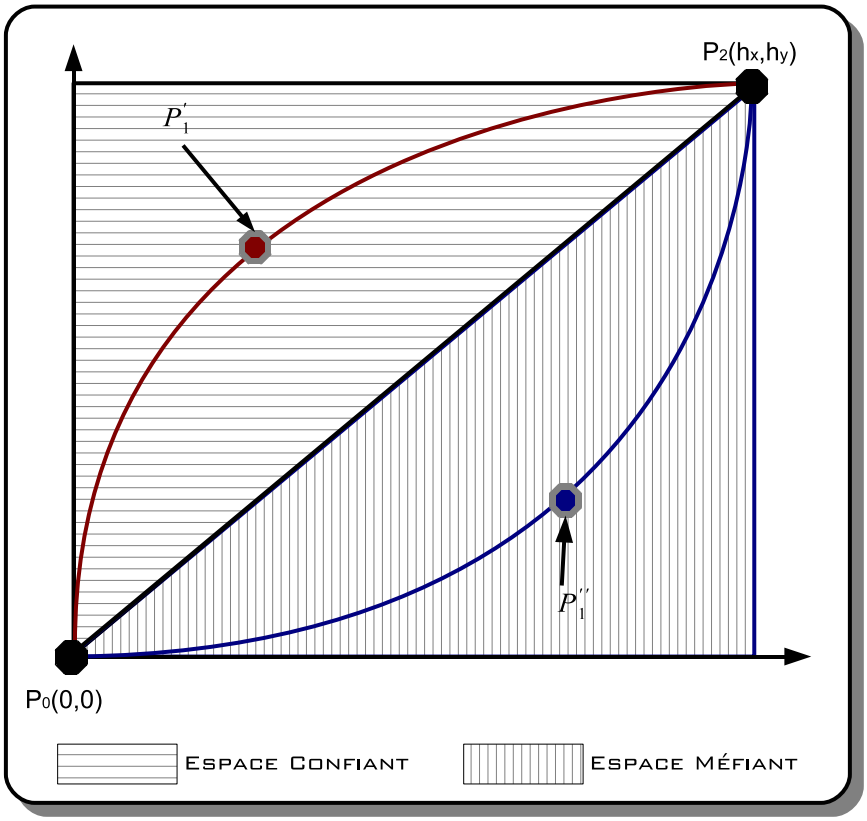

FIGURE 5.6 – La fonction Behavior

d'origine et le point repère.

Présentation d'une courbe de Bézier : Les courbes de Bézier [Poug 98] sont des courbes polynomiales paramétriques décrites pour la première fois en 1962 par l'ingénieur français Pierre Bézier, qui les utilisa pour concevoir sur ordinateur des pièces d'automobiles. Elles sont le plus souvent utilisées dans la synthèse et le traitement d'images. Les courbes de Bézier sont des fonctions qui permettent d'établir des courbes lisses définies grâce à un ensemble de points $P_0, P_1 ... P_N$. Elles se dessinent à partir du premier point P_0 et se rapprochent de manière asymptotique des points intermédiaires $P_1, P_2 ... P_{n-1}$ dans l'ordre jusqu'à finir sur le dernier point P_n.

5.4.4.1 La fonction BehaVior BV

Pour les besoins de notre modèle de confiance, nous avons utilisé l'équation d'une courbe de Bézier quadratique, qui est définie comme suit [Poug 98] :

Une courbe de Bézier quadratique est un chemin tracé par l'équation B(t), qui relie les points P_0, P_1 *et* P_2 comme suit :

$B(t) = (1-t)^2 * P_0 + 2t(1-t) * P_1 + t^2 * P_2$. avec $t \in [0, 1]$

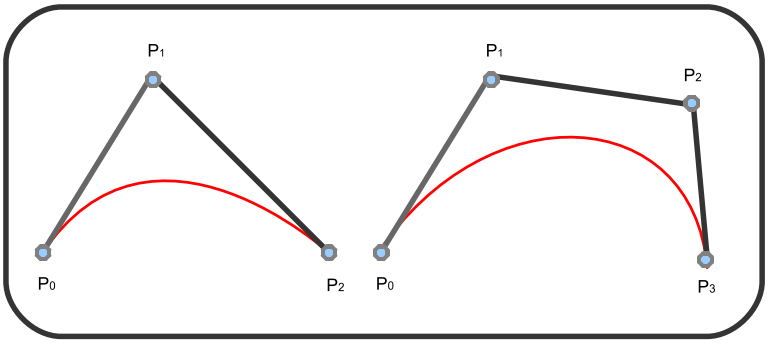

FIGURE 5.7 – Les courbes de Bézier

En appliquant cette équation sur les points définis précédemment à savoir :

- le point d'origine $P_0(0,0)$,
- le point repère $P_2(h_x, h_y)$, tel que : $h_x > 0$ et $h_y > 0$
- le point de comportement $P_1(b_x, b_y)$ tel que : $0 \leq b_x \leq h_x$ et $0 \leq b_y \leq h_y$

On obtient l'équation paramétrée suivante :

$$B(t) = \begin{cases} B_x(t) & = 2t(1-t) * b_x + t^2 * h_x \\ B_y(t) & = 2t(1-t) * b_y + t^2 * h_y \end{cases}$$

L'équation telle qu'elle est définie ne convient pas à notre modèle, car le but de notre fonction est de calculer la confiance pondérée $(B_y(t))$ en fonction de la confiance neutre $(B_x(t))$.

Ainsi nous allons transformer cette équation paramétrée pour obtenir une équation cartésienne du type $f(X) = Y$, comme suit :

1. Sachant que $B_x(t) = X$, on doit calculer la fonction inverse $B_x^{-1}(X) = t$.
2. Remplacer t dans $B_y(t)$ pour obtenir $B_y(B_x^{-1}(X)) = Y$.
3. Et enfin obtenir la fonction **B**eha**V**ior suivante : $BV_{P_1,P_2}(X) = Y$.

Etape 1 : Résoudre l'équation $t = B_x^{-1}(X)$

$$\begin{aligned} & B_x(t) = X \\ \Leftrightarrow \quad & 2t * (1-t) * b_x + t^2 * h_x = X \\ \Leftrightarrow \quad & 2t * (1-t) * b_x + t^2 * h_x - X = 0 \\ \Leftrightarrow \quad & (h_x - 2b_x)t^2 + 2b_x t - X = 0 \end{aligned} \tag{5.1}$$

Discuter l'équation 5.1 Sachant que $h_x > 0$ et que $0 \leq b_x \leq h_x$:

- Si $(h_x = 2b_x)$ alors $t = \frac{X}{2b_x}$.
- Sinon, rechercher les racines de l'équation 5.1 en calculant son discriminant $\Delta'_{5.1}$.

Calcul du discriminant réduit [6] de l'équation 5.1

$$\Delta'_{5.1} = b_x^2 - 2Xb_x + Xh_x \tag{5.2}$$

Discuter l'équation 5.2
– Si $b_x = 0$ alors l'équation 5.2 est positive ou nulle car :

$$\begin{aligned} & X \geq 0 \wedge h_x > 0 \ (hyp) \\ \Rightarrow \quad & X * h_x \geq 0 \end{aligned} \tag{5.3}$$

– Sinon calculer le discriminant $\Delta'_{5.2}$ de l'équation 5.2.

Calcul du discriminant réduit de l'équation 5.2

$$\Delta'_{5.2} = X^2 - h_x X = X(X - h_x) \tag{5.4}$$

Discuter l'équation 5.4 le discriminant de l'équation 5.2 est négatif ou nul ($\Delta'_{5.2} \leq 0$) car :

$$X > 0 \wedge X \leq h_x \ (hyp)$$

Conclusion sur le discriminant de l'équation 5.1
– Si $(b_x = 0) alors (\Delta'_{5.1} \geq 0)$ (voir 5.3)
– Sinon, si le discriminant $(\Delta'_{5.2} = 0) alors (\Delta'_{5.1} \geq 0)$ car le $\Delta'_{5.1}$ possède une racine double.
– Sinon, si le discriminant $(\Delta'_{5.2} < 0)$ alors l'équation $\Delta'_{5.1}$ n'admet pas de racine et donc $\Delta'_{5.1} > 0$ car dans ce cas de figure le discriminant $(\Delta'_{5.1})$ est du même signe que le coefficient de l'élément du second ordre (b_x^2>0) de l'équation qui a permis de le calculer (5.4).

$\Delta'_{5.1}$ est toujours positif ou nul, donc admet l'équation 5.2 admet deux solutions (éventuellement double).

Etape 2 : Calcul de t en fonction de X
En analysant l'équation 5.1 :

– Si $(h_x - 2b_x = 0)$ alors $t = \frac{X}{2b_x}$
– Sinon en fonction du $\Delta'_{5.1}$, l'équation 5.1 possède deux racines comme suit :
  ∘ $t_1 = \frac{-b_x + \sqrt{{b_x}^2 - 2b_x * X + h_x * X}}{h_x - 2b_x}$.
  ∘ $t_2 = \frac{-b_x - \sqrt{{b_x}^2 - 2b_x * X + h_x * X}}{h_x - 2b_x}$.

Sachant que $0 \leq b_x \leq h_x$ et que $0 \leq X \leq h_x$, il faut trouver quelle racine (t_1 ou t_2) se situe dans l'intervalle [0,1] (rappel $t \in [0, 1]$) pour toutes les valeurs que peuvent prendre b_x, h_X *et* X.

6. Car le second coefficient de l'équation $(2Xb_x)$ est pair.

Racine t_1 : Il faut démontrer que :

$$\forall X, b_x \in [0,1], (0 \leq \frac{-b_x + \sqrt{{b_x}^2 - 2b_x * X + h_x * X}}{h_x - 2b_x} \leq 1 \; sachant \; que \; (h_x - 2b_x) \neq 0) \quad (5.5)$$

Raisonnement par l'absurde :

Supposons $t_1 < 0$, alors

$$\frac{-b_x + \sqrt{{b_x}^2 - 2b_x * X + h_x * X}}{h_x - 2b_x} < 0 \quad (5.6)$$

– Si le dénominateur de 5.6 est positif $(h_x - 2b_x > 0)$, alors :

$$\begin{aligned} & -b_x + \sqrt{{b_x}^2 - 2b_x * X + h_x * X} < 0 \\ \Leftrightarrow \quad & \sqrt{{b_x}^2 - 2b_x * X + h_x * X} < b_x \\ \Leftrightarrow \quad & {b_x}^2 - 2b_x * X + h_x * X < b_x^2 \; car \; (b_x \geq 0) \\ \Leftrightarrow \quad & -2b_x * X + h_x * X < 0 \\ \Leftrightarrow \quad & X * (h_x - 2b_x) < 0 \end{aligned}$$

Or $(X \geq 0)$ et $(h_x - 2b_x > 0)$ (hyp)

Donc **impossible**. Ainsi, si $(h_x - 2b_x > 0)$ alors $t_1 \geq 0$.

– Si le dénominateur de 5.6 est négatif $(h_x - 2b_x < 0)$, alors :

$$\begin{aligned} & -b_x + \sqrt{{b_x}^2 - 2b_x * X + h_x * X} > 0 \\ \Leftrightarrow \quad & \sqrt{{b_x}^2 - 2b_x * X + h_x * X} > b_x \\ \Leftrightarrow \quad & {b_x}^2 - 2b_x * X + h_x * X > b_x^2 \; car \; (b_x \geq 0) \\ \Leftrightarrow \quad & -2b_x * X + h_x * X > 0 \\ \Leftrightarrow \quad & X * (h_x - 2b_x) > 0 \end{aligned}$$

Or $X \geq 0$ et $(h_x - 2b_x < 0)$ (hyp).

Donc impossible. Ainsi, si $(h_x - 2b_x < 0)$ alors $t_1 \geq 0$.

Finalement, si $(h_x - 2b_x \neq 0)$ alors $t_1 \geq 0$.

Supposons $t_1 > 1$, alors :

$$\frac{-b_x + \sqrt{{b_x}^2 - 2b_x * X + h_x * X}}{h_x - 2b_x} > 1 \quad (5.7)$$

– Si le dénominateur de 5.7 est positif $(h_x - 2b_x > 0)$, on a :

$$\begin{aligned}
& -b_x + \sqrt{{b_x}^2 - 2b_x * X + h_x * X} > h_x - 2b_x \\
\Leftrightarrow \quad & \sqrt{{b_x}^2 - 2b_x * X + h_x * X} > h_x - b_x \\
\Leftrightarrow \quad & {b_x}^2 - 2b_x X + h_x X > h_x^2 + b_x^2 - 2h_x b_x \ car\ (h_x \geq b_x) \\
\Leftrightarrow \quad & -2b_x X + h_x X > h_x^2 - 2h_x b_x \\
\Leftrightarrow \quad & X(h_x - 2b_x) > h_x(h_x - 2b_x) \\
\Leftrightarrow \quad & X > h_x \ car\ (h_x - 2b_x > 0)(hyp)
\end{aligned}$$

Or $(0 \leq X \leq h_x)$ (hyp)

Donc **impossible**. Ainsi, si $(h_x - 2b_x > 0)$ alors $t_1 \leq 1$.

– Si le dénominateur de 5.7 est négatif $(h_x - 2b_x < 0)$, on a :

$$\begin{aligned}
& -b_x + \sqrt{{b_x}^2 - 2b_x * X + h_x * X} < h_x - 2b_x \\
\Leftrightarrow \quad & \sqrt{{b_x}^2 - 2b_x * X + h_x * X} < h_x - b_x \\
\Leftrightarrow \quad & {b_x}^2 - 2b_x X + h_x X < h_x^2 + b_x^2 - 2h_x b_x \ car\ (h_x \geq b_x) \\
\Leftrightarrow \quad & -2b_x X + h_x X < h_x^2 - 2h_x b_x \\
\Leftrightarrow \quad & X(h_x - 2b_x) < h_x(h_x - 2b_x) \\
\Leftrightarrow \quad & X > h_x \ car\ (h_x - 2b_x < 0)(hyp)
\end{aligned}$$

Or $(0 \leq X \leq h_x)$ (hyp)

Donc **impossible**. Ainsi, si $(h_x - 2b_x < 0)$ alors $t_1 \leq 1$.

Finalement, si $(h_x - 2b_x \neq 0)$ alors $0 \leq t_1 \leq 1$.

Racine t_2 : Il faut démontrer que :

$$\forall X, b_x \in [0, h_x], (0 \leq \frac{-b_x - \sqrt{{b_x}^2 - 2b_x * X + h_x * X}}{h_x - 2b_x} \leq 1 \ sachant\ que\ (h_x - 2b_x) \neq 0) \quad (5.8)$$

t_2 : Vérifions si t_2 est compris entre 0 et 1

$$0 \leq \frac{-b_x - \sqrt{{b_x}^2 - 2b_x * X + h_x * X}}{h_x - 2b_x} \leq 1 \quad (5.9)$$

Nous savons que le numérateur de t_2 est négatif ou nul car $(b_x \geq 0) \wedge (\sqrt{\Delta'_{5.1}} \geq 0)$

– Si le dénominateur de t_2 est strictement positif $(h_x - 2b_x > 0)$ alors $t_2 \leq 0$
– Si le dénominateur de t_2 est strictement négative $(h_x - 2b_x < 0)$ alors $t_2 \geq 0$

Vérifions maintenant si $(h_x - 2b_x < 0)$ **alors** t_2 **est inférieur ou égale à 1**

$$\begin{aligned} & \frac{-b_x - \sqrt{{b_x}^2 - 2b_x * X + h_x * X}}{h_x - 2b_x} \leq 1 \\ \Leftrightarrow \quad & -b_x - \sqrt{{b_x}^2 - 2b_x * X + h_x * X} \geq h_x - 2b_x \; car \; (h_x - 2b_x < 0) \\ \Leftrightarrow \quad & -\sqrt{{b_x}^2 - 2b_x * X + h_x * X} \geq h_x - b_x \end{aligned}$$

Etant donné que $(h_x - b_x > 0)$ car $b_x \in [0, h_x]$ (hyp)
et que $-\sqrt{{b_x}^2 - 2b_x * X + h_x * X} < 0$

t_2 ne peut être strictement inférieur à 1.
Cela implique que $t_2 \geq 1$ si $(h_x - 2b_x < 0)$.

Finalement, si $(h_x - 2b_x \neq 0)$ alors $t_2 \in \mathbb{R} -]0, 1[$.

Conclusion : Compte-tenu des contraintes imposées par la fonction BV ($(h_x - 2b_x \neq 0), (0 \leq b_x \leq h_x)$ *et* $(0 \leq X \leq h_x)$**, seul** t_1 **prend toujours des valeurs dans l'intervalle [0,1]**

Ainsi, en remplaçant le t par t_1 dans la fonction $B_y(t)$ on obtient la fonction cartésienne "BV" qui peut dessiner n'importe quelle courbe approchant le point $P_1(b_x, b_y)$ qui se trouve dans le rectangle délimité par $P_0(0,0)$ et $P_2(h_x, h_y)$, comme suit :

$$\begin{aligned} BV : [0, h_x] & \longrightarrow [0, h_y] \\ X & \longrightarrow Y \end{aligned}$$

$$BV_{P_1,P_2}(X) = \begin{cases} \frac{(h_y - 2b_y)}{4b_x^2} X^2 + \frac{b_y}{b_x} X & si \; (h_x - 2b_x = 0) \\ (h_y - 2b_y)(t_1)^2 + 2b_y t_1, & si \; (h_x - 2b_x \neq 0) \end{cases}$$

Où $t_1 = \frac{-b_x + \sqrt{{b_x}^2 - 2b_x * X + h_x * X}}{h_x - 2b_x} \; \wedge (0 \leq b_x \leq h_x) \wedge (0 \leq X \leq h_x) \wedge (h_x > 0)$

5.4.4.2 Disposition level l

Cette valeur conditionne la position du point $P_1(b_x, b_y)$. Selon la position de celui-ci, la courbe définie par la fonction BV peut passer d'une forme concave à une forme convexe (voir figure 5.6). Cependant, faire glisser le point de comportement sur l'antidiagonale, qui est représenté par l'équation $Y = \frac{-h_y}{h_x} * X + h_y$, est suffisant pour modéliser une variété de comportements pour les deux types de courbes décrites précédemment.

Nous définissons la fonction D (D pour Disposition) qui permet de fixer le point P_1 sur l'antidiagonale, à partir d'une la valeur de l, qui est comprise dans l'intervalle [0,n] (où '0' représente une personnalité très confiante et 'n' correspond à la personnalité la plus méfiante) comme suit :

$$\begin{aligned} D : [0, n] & \longrightarrow [0, h_x] * [0, h_y] \\ l & \longrightarrow (b_x, b_y) \end{aligned}$$

$$D(l) = \begin{cases} b_x = \frac{-h_x}{n} l + h_x \\ b_y = \frac{h_y}{n} l \end{cases}$$

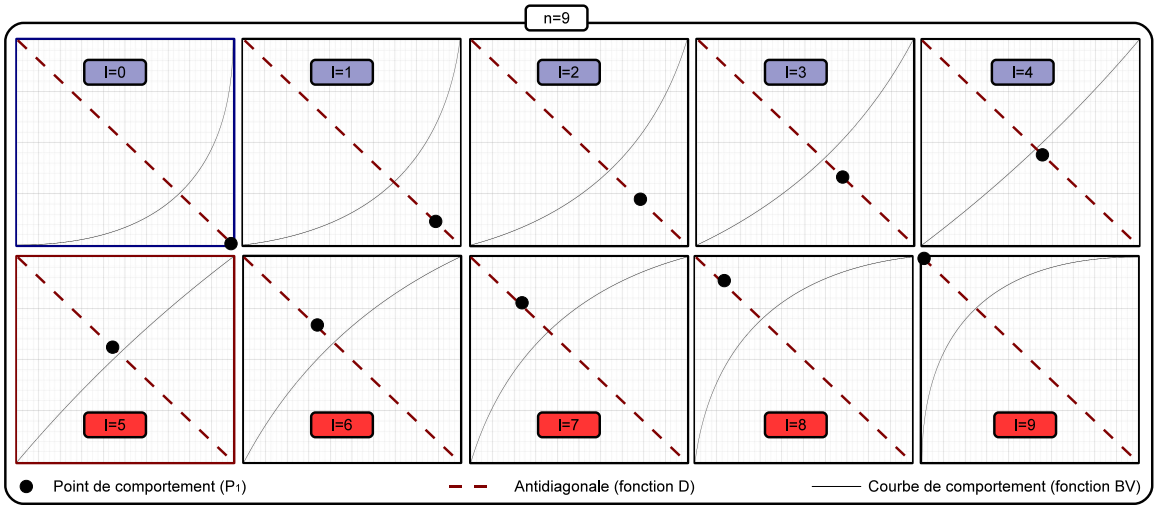

FIGURE 5.8 – Exemples de courbes dessinées avec la fonction BV pour un l allant de 0(confiance) à 9(méfiance).

Comme le montre la figure 5.8, il suffit à une entité de fixer son "disposition level" pour générer une courbe correspondant au degré de confiance ou de méfiance de l'entité en question.

Nota : Pour le reste du chapitre, nous utiliserons $l^0 = \frac{l}{n}$ dans toutes les équations afin de définir le disposition level.

5.4.4.3 Redéfinition de la fonction t^0

En utilisant la fonction BV et le TrustSort, le calcul de la confiance directe est obtenu comme suit :

$$t^0 : \mathbb{S} * \mathbb{S} \rightarrow \mathbb{R}^+$$
$$(A, B) \rightarrow t^0(A, B)$$

$$t^0(A, B) = BV_{P_1,P_2}(TrustSort_A[B]).$$

Sachant que :
- pour le point P_2
  - $h_x = |T_oS(A)|$ cette valeur correspond au nombre d'éléments dans le $T_oS(A)$.
  - $h_y = T^0_A$.

- pour le point P_1, les valeurs b_x et b_y dépendent de la valeur l^0 qui est définie par le site A.

Exemple : Prenons l'exemple de deux sites : 'A' et 'B', qui partagent le même cercle de confiance sortant T_oS. Supposons que les sites de confiance sont classés dans le même ordre que dans l'exemple précédent (voir section 5.4.3.1.0). Cependant, dans cet exemple-ci on suppose en outre que le site A est confiant et le site B est méfiant.

Dans le but d'évaluer les membres du cercle de confiance sortant en fonction de la personnalité de A et de B nous fixons les paramètres suivants :

- Le cercle de confiance sortant est composé des 6 membres suivants :
  $T_oS(A)$=$T_oS(B)$={Site 1, Site 2, Site 3, Site 4, Site 5, Site 6}

- Le $T_oS(A)$ et le $T_oS(B)$ sont triés comme suit :
  (Confiance Max)(+). **Site 4. Site 3. Site 5. Site 1. Site 2. Site 6** (-)(Confiance Min)

- Le 'Disposition level' l^0 :
  - le Disposition level du site A est : $l^0 = 1/9$.
  - le Disposition level du site B est : $l^0 = 8/9$.

- Le point repère $P_2(h_x, h_y)$ est défini comme suit :
  - $h_x = |T_oS(A)| = |T_oS(B)| = 6$.
  - $h_y = T^0_A = T^0_B = 50$

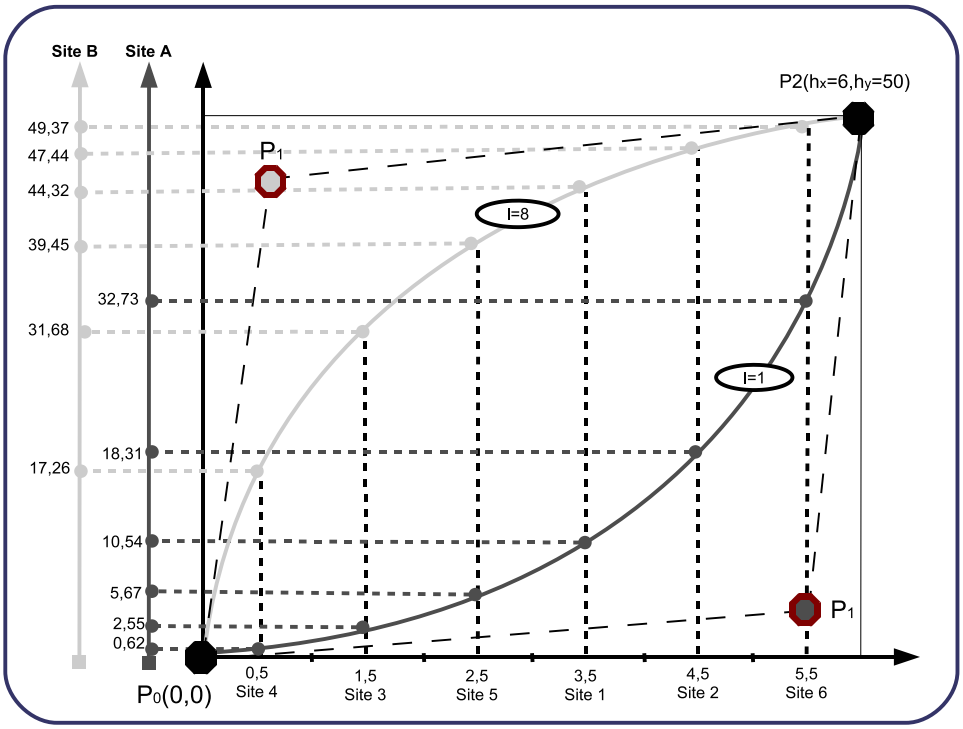

FIGURE 5.9 – Exemple d'évaluation du cercle de confiance sortant.

En appliquant la fonction 'BV' sur la liste $TrustSort$ des sites A et B (figure 5.9), on obtient

les résultats suivants :

	Site 1	Site 2	Site 3	Site 4	Site 5	Site 6
Site A	10.54	18.31	2.55	0.62	5.67	32.73
Site B	44.32	47.44	31.68	17.26	39.45	49.37

Dans cet exemple, en prenant la valeur $t^0(A, S1) = 10.54$ et $t^0(B, S1) = 44.32$, on remarque que $t^0(A, S1)$ est inférieur à $t^0(B, S2)$ (de même pour tous les autres sites). Ce résultat reflète la personnalité du site A qui est plus confiant que le site B. Ceci se traduit par des valeurs de confiance plus proches du zéro pour le site A et plus proche de 50 pour le site B.

5.5 La confiance transitive

La section précédente illustre les fonctions qui initialisent le réseau de confiance en permettant de créer et d'évaluer le cercle de confiance sortant de n'importe quelle entité du réseau. Dans ce qui suit, nous allons traiter l'aspect transitif de la fonction $Trust$ en définissant une nouvelle dimension pour le modèle T2D portant le nom de : "Disposition to Transitivity".

5.5.1 La fonction de propagation P^0

Nous avons fait le choix de définir une évaluation qui va de la confiance à la méfiance, car ce mécanisme constitue une approche naturelle pour calculer des chaînes de confiance. En effet, plus il y a d'intervenants pour établir une connexion entre deux entités, plus le lien est fragilisé, ce qui se traduit par une forte méfiance. Et inversement, moins il y a d'intervenants, plus le lien est fiable et ainsi la confiance plus forte.

Dans cette section, nous allons introduire pas à pas la fonction de propagation P^0. Elle est définie sur une chaîne de confiance (Pt_n) et retourne une valeur qui représente la confiance transitive d'une entité envers une autre.

La confiance transitive $P^0(Pt_n)$ liant le sommet s_0 au sommet s_n est calculée en additionnant les différentes valeurs comme suit :

$$P^0(Pt_n) = \begin{cases} t^0(s_0, s_n) & si\ 0 \leq n \leq 1 \\ P^0(Pt_{n-1}) + t^0(s_{n-1}, s_n) & si\ n > 1 \end{cases}$$

On peut constater que l'addition répond à nos besoins, car c'est une fonction croissante qui augmente au fur et à mesure que la chaîne s'allonge. Cependant la fonction P^0, telle qu'elle a été définie, peut retourner un résultat biaisé. En effet, comme l'illustre la figure 5.10, le simple fait d'additionner les poids des différents arcs peut induire en erreur, car chaque poids a été défini par rapport au seuil de méfiance local 'T^0'.

Ainsi, avant de procéder à l'addition, il faut replacer le deuxième opérande de manière à ce qu'il soit évalué dans le contexte local de la cible $(T^0_{s_n})$, comme suit :

Redéfinition de la fonction P^0 :

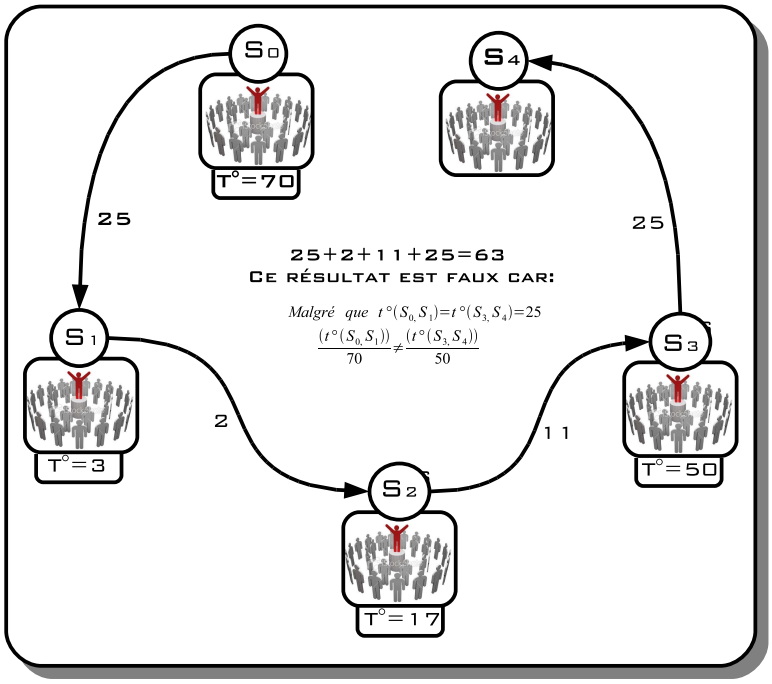

FIGURE 5.10 – Propagation de la confiance

$$P^0(Pt_n) = \begin{cases} t^0(s_0, s_n) & si\ 0 \leq n \leq 1 \\ P^0(Pt_{n-1}) + \frac{T^0_{s_0}}{T^0_{s_{n-1}}} * t^0(s_{n-1}, s_n) & si\ n > 1 \end{cases}$$

Exemple :

L'exemple représenté dans la figure 5.11, montre que l'évaluation se fait à partir de l'avant dernier maillon de la chaîne (le site s_3) de manière récursive jusqu'au site s_0. Arrivé à s_0, l'évaluation de la fonction P^0 permet de quantifier la confiance qui peut être accordée au site s_4.

5.5.2 Les seuils de méfiance

Pour la fonction t^0, un seuil de méfiance local (T^0) a été défini. Il représente la valeur maximum pour l'évaluation des membres du T_oS. Cependant, l'intervalle [0,T^0] ne permet de borner la fonction P_0, étant donné que le résultat obtenu n'est pas une moyenne (qui permet de se replacer dans cet intervalle) mais une fonction d'addition croissante. Donc, un seuil différent doit être mis en place afin de fixer une valeur maximum pour le calcul de la confiance transitive.

Nous définissons un **seuil de méfiance transitive**. Noté θT^0, celui-ci représente l'évaluation de la plus grande chaîne de confiance tolérée où tous les poids des arcs composant cette chaîne sont égaux au seuil de méfiance local (T^0). Ainsi, θT^0_S est proportionnel à T^0 et à une longueur de chaîne maximum 'P^m_S' spécifique à chaque site, comme suit :

$\theta T^0_S = T^0_S * P^m_S$ (sachant que $P^m_S \geq 1$)

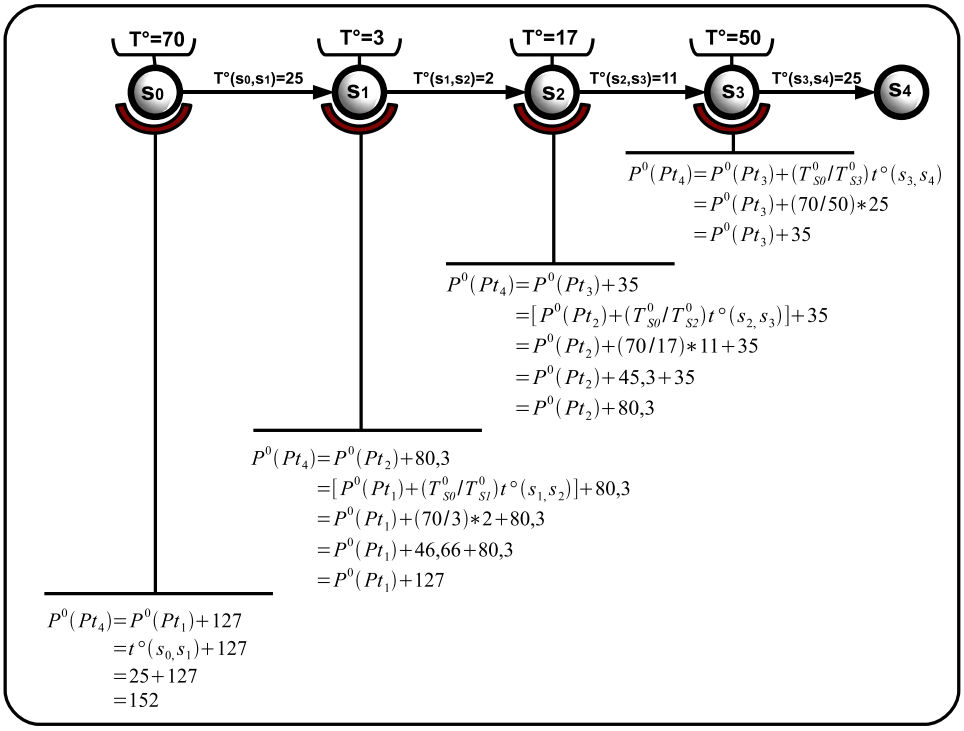

FIGURE 5.11 – Propagation de la confiance par rapport au seuil de méfiance de s_0.

En résumé, pour notre modèle T2D, chaque entité du graphe de confiance doit définir deux seuils de méfiance :

- T_S^0 représente le seuil de méfiance local du site S.
- $\theta T_S^0 = T_S^0 * P_S^m$ représente le seuil de méfiance transitive du site S.

Ainsi, pour une chaîne Pt_n le site source s_n est considéré comme digne de confiance :

ssi $0 \leq P^0(Pt_n) \leq \theta T_{s_0}^0$.

En reprenant l'exemple représenté par la figure 5.11, en fonction du $P_{S_0}^m$ que le site S_0 a défini, les utilisateurs du site S_4 se voit accorder ou refuser l'accès comme suit :

- Si $P_{S_0}^m = 2$ alors le site s_4 n'est pas considéré comme un site de confiance :
  car $[P^0(Pt_4) = 152] > [\theta T_{s_0}^0 = 70 * 2 = 140]$).

- Si $P_{S_0}^m = 3$ alors le site s_4 est considéré comme un site de confiance car :
  $0 \leq [P^0(Pt_4) = 152] \leq [\theta T_{s_0}^0 = 70 * 3 = 210]$).

5.5.3 Disposition to Transitivity

La fonction P^0 définie précédemment, permet de calculer la confiance transitive de manière continue tout le long de la chaîne de confiance.

Comme pour le calcul de la confiance directe, la confiance transitive doit être évaluée diffé-

remment en fonction de la personnalité de la cible (confiante ou méfiante). Ainsi, la fonction P^0 doit être pondérée en fonction du disposition level l^0 afin de retourner un résultat qui reflète le comportement du site cible.

Afin d'ajuster l'évaluation de la confiance transitive en fonction de la personnalité de chaque entité, deux solutions sont envisageables :

1. Ajuster le θT^0 : Cette première option permet d'adapter le seuil de méfiance transitive en le diminuant ou en l'augmentant selon que le site est méfiant ou confiant.
2. Ajuster le calcul de P^0 : Cette seconde possibilité consiste à pénaliser le poids de chaque arc afin d'augmenter ou de diminuer le calcul de la fonction de propagation en fonction du disposition level l^0.

5.5.3.1 Ajustement du seuil de méfiance transitif

L'ajustement de θT^0 se fait à travers la variable P^m. Cette longueur maximum peut être calculée à partir d'une valeur dite : *langueur du chemin maximum de référence* et notée θP^m.

Définition : La longueur du chemin maximum de référence θP_S^m du nœud "S" peut être déduite à partir de l'évaluation minimum (calculer grâce à la fonction de propagation) du chemin reliant "S" au nœud le plus éloigné du réseau dans un scénario où tous les nœuds du réseau exhibent une personnalité totalement confiante (i.e, $l^0 = 0$) (voir annexe C.2).

Ainsi, à partir de θP_S^m l'entité la plus méfiante a un $P_S^m = 1$ et la plus confiante un $P_S^m = \theta P_S^m$ comme suit :

$P_S^m = l^0(1 - \theta P_S^m) + \theta P_S^m$ (où l^0 dénote le Disposition Level).

En prenant l'exemple de la figure 5.11 et en prenant $\theta P_S^m = 3$, les utilisateurs du site s_4 peuvent se voir accorder ou refuser l'accès :

- Si s_0 est un site confiant affichant un Disposition level : $l^0 = \frac{1}{9}$ alors :
  $P_S^m = \frac{1}{9} * (1 - 3) + 3 = 2,78$.
  Dans ce cas de figure, le site s_4 est considéré comme un site de confiance :
  car $0 \leq [P^0(Pt_4) = 152] \leq [\theta T_{s_0}^0 = 70 * 2,78 = 194,6]$).

- Si s_0 est un site affichant un Disposition level : $l = \frac{8}{9}$ alors :
  $P_S^m = \frac{8}{9} * (1 - 3) + 3 = 1,22$.
  Dans ce cas de figure, le site s_4 n'est pas considéré comme un site de confiance :
  car $[P^0(Pt_4) = 152] > [\theta T_{s_0}^0 = 70 * 1,22 = 85,4]$).

5.5.3.2 Ajustement de la fonction de propagation

La solution proposée consiste à pondérer chaque arc en fonction de sa position dans la chaîne de confiance. Un coefficient est alors attribué à chaque lien, ce qui permet d'évaluer différemment les nœuds proches des nœuds distants.

La fonction P^0 est alors redéfinie comme suit :

Redéfinition de la fonction P^0

$$P^0_\Phi(Pt_n) = \begin{cases} \Phi(0) * t^0(s_0, s_n) & si\ 0 \leq n \leq 1 \\ P^0(Pt_{n-1}) + \Phi(n-1) * \frac{T^0_{s0}}{T^0_{s_{n-1}}} * t^0(s_{n-1}, s_n) & si\ n > 1 \end{cases}$$

Où Φ est une fonction croissante. Elle fournit un coefficient supérieur ou égal à 1 à chaque élément de la chaîne en fonction de sa position.

A partir de cette redéfinition, une chaîne Pt_n est évaluée en utilisant les valeurs retournées par la fonction Φ. Ainsi chaque maillon de la chaîne (s_n) multiplie la valeur de son évaluation $t^0(s_n, s_{n+1})$ par le coefficient $\Phi(n)$ afin d'obtenir une valeur pondérée par rapport à la position du tiers de confiance dans la chaîne.

Exemple :

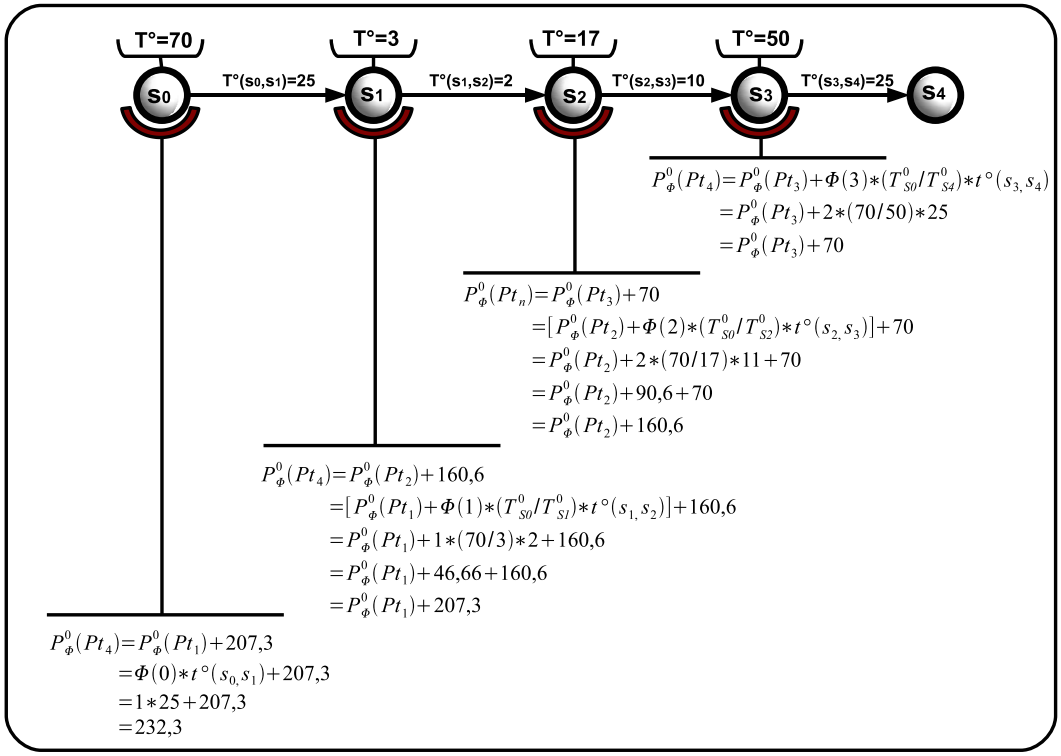

FIGURE 5.12 – Ajustement de la fonction de propagation

Reprenons le dernier exemple (figure 5.11) et choisissant la fonction Φ suivante :

$$\Phi(x) = \begin{cases} 1 & si\ 0 \leq x < 2 \\ 2 & si\ x \geq 2 \end{cases}$$

Comme le montre la figure 5.12, P^0_Φ, en encapsulant toutes les équations de la figure 5.12 on obtient le résultat suivant :

$$P^0_\Phi(Pt_4) = [\Phi(0) * t^0(s_0, s_1] + [\Phi(1) * \frac{T^0_{s_0}}{T^0_{s_1}} t^0(s_1, s_2)] + [\Phi(2) * \frac{T^0_{s_0}}{T^0_{s_2}} t^0(s_2, s_3)] + [\Phi(3) * \frac{T^0_{s_0}}{T^0_{s_3}} t^0(s_3, s_4)]$$
$$= [1 * 25] + [1 * \frac{70}{3} * 2] + [2 * \frac{70}{17} * 11] + [2 * \frac{70}{50} * 25].$$
$$= 25 + 46,7 + 90,6 + 70 = 232,3.$$

Dans ce cas de figure, si $P^m_{s_0} = 3$, le site s_4 n'est pas considéré comme un site de confiance : car $[P^0_\Phi(Pt_4) = 232,3] > [\theta T^0_{s_0} = 70 * 3 = 210]$).

Définition de la fonction Φ

La fonction Φ retourne des éléments strictement positifs. Nous proposons de les calculer qui peuvent être calculés à l'aide d'une fonction croissante ajustable grâce au Disposition Level l^0. L'idée est la suivante : Plus une personnalité est confiante plus elle a tendance à faire confiance à des chaînes longues. A l'inverse, plus la personnalité est méfiante, plus elle a tendance à se méfier des chaîne longues (i.e. En pratique, attribuer des poids forts aux arcs éloignés de la chaîne). Notée Φ_{l^0}, cette fonction permet d'attribuer à chaque arc d'une chaîne de confiance, un coefficient qui correspond à sa position dans la chaîne. Ainsi, plus on avance dans la chaîne de confiance plus le coefficient va augmenter et se rapprocher de la borne supérieure.

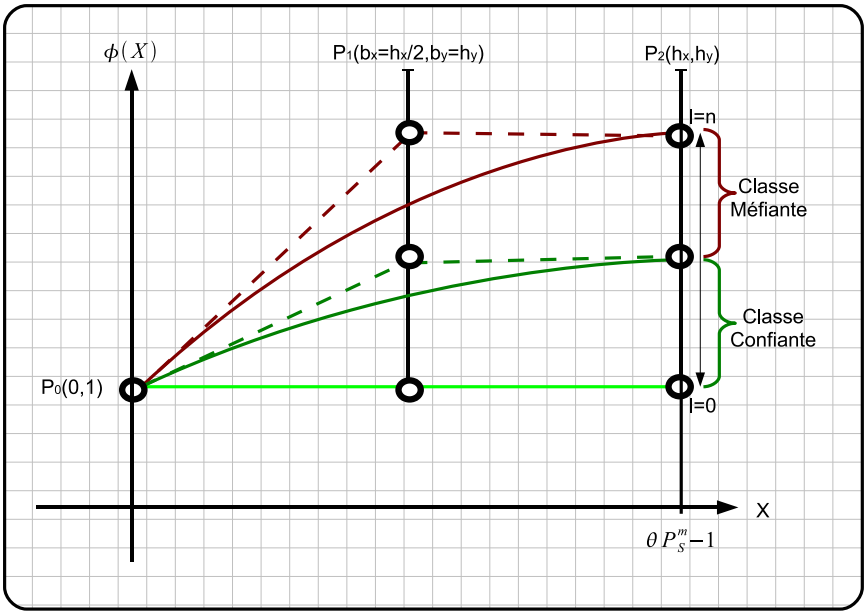

FIGURE 5.13 – La fonction BV pour l'ensemble Φ

Avec peu de modifications, la fonction BV, définie précédemment, peut parfaitement convenir pour décrire notre schéma de fonctions. La figure 5.13 montre qu'afin de définir la fonction Φ_{l^0} il suffit de prendre la fonction BV en initialisant ses trois points comme suit :

– $P_0(0,1)$ est le point d'origine. Il a pour coordonnées (0,1), car la valeur minimale que peut prendre $\Phi_{l^0}(X)$ est 1.

- Les points P_1 et P_2 possèdent la même ordonnée et augmentent ainsi, en parallèle en fonction du h_y comme suit :
  - $P_2(h_x, h_y)$: le point repère a les coordonnées suivantes :
    - $h_x = \theta P_S^m - 1$. Notre objectif est de définir des coefficients différents pour tout arc se trouvant en position inférieure à θP_S^m. Ces coefficients seront compris entre 1 et h_y. A l'inverse, tous les arcs de la chaîne se trouvant à une distance supérieure ou égale à θP_S^m auront comme coefficient $h_y = \Phi_{l^0}(\theta P_S^m - 1)$.
    - h_y est calculé en fonction de 'Disposition level l^0' et permet de définir la valeur maximale que peut prendre un coefficient. Nous proposons :
      - le h_y correspondant à **un type confiant**, est compris dans l'intervalle $[1, \lceil \frac{Coef^0}{2} \rceil]$.
      - le h_y correspondant à **un type méfiant**, est compris dans $[\lceil \frac{Coef^0}{2} \rceil, Coef^0]$..
      - le $Coef^0$ correspond à la valeur maximum que peut prendre la fonction Φ.
  - $P_1(b_x, b_y)$: le point de comportement est fonction du point repère à savoir $b_x = h_x/2$ et $b_y = h_y$

Afin d'adapter la fonction BV, nous allons procéder ainsi :

1. Définir la fonction BV à partir du point $P_0 = (0, 0)$ en remplaçant les paramètres comme suit :
   - $h_x = \theta P^m - 1$.
   - $h_y = (Coef^0 - 1) * l^0$. pour avoir une valeur comprise entre 0 et $(Coef^0 - 1)$.
   - $b_x = \frac{h_x}{2} = \lceil \frac{(\theta P^m - 1)}{2} \rceil$.
   - $b_y = h_y$.
2. procéder à une translation du point d'origine $P_0(0, 0)$ aux nouvelles coordonnées (0,1).

Remplacement des paramètres

Pour $b_x = \frac{h_x}{2}$ la fonction $BV_{P_1,P_2}(X) = \frac{(h_y - 2b_y)}{4b_x^2} X^2 + \frac{b_y}{b_x} X$.

En remplaçant les paramètres par leurs valeurs correspondantes, on obtient la fonction cartésienne suivante :

$$Y = \frac{-(Coef^0 - 1) * l^0}{(\theta P_S^m - 1)^2} X^2 + \frac{2(Coef^0 - 1) * l^0}{\theta P_S^m - 1} X$$

Translation

Pour déplacer la courbe au point d'origine (0,1) il suffit que $\Phi_{l^0}(X) = Y + 1$, ainsi :

$$\Phi_{l^0}(X) = \begin{cases} \frac{-(Coef^0-1)*l^0}{(\theta P_S^m-1)^2}X^2 + \frac{2(Coef^0-1)*l^0}{\theta P_S^m-1}X + 1 & si\ X < \theta P_S^m \\ \Phi_{l^0}(\theta P_S^m - 1) = (Coef^0 - 1) * l^0 + 1 & si\ X \geq \theta P_S^m \end{cases}$$

Exemple
Reprenons l'exemple précédent de la figure 5.12, fixant le $Coef^0$ à la valeur 3 et faisant varier le disposition Level l^0 pour les valeurs 0, 2/9, 4/9, 6/9, *et* 8/9. On obtient le tableau ci-dessous qui affichent les résultats retournés par la fonction de propagation $P^0_{\Phi_{l^0}}$ pour les chaînes Pt_1, Pt_2, Pt_3 et Pt_4.

Level	$\Phi_{l^0}(0)$	$P^0_{\Phi_{l^0}}(Pt_1)$	$\Phi_{l^0}(1)$	$P^0_{\Phi_{l^0}}(Pt_2)$	$\Phi_{l^0}(2)$	$P^0_{\Phi_{l^0}}(Pt_3)$	$\Phi_{l^0}(3)$	$P^0_{\Phi_{l^0}}(Pt_4)$
0	1	25	1	71,7	1	117	1	152
2/9	1	25	1,33	87,1	1,44	152,3	1,44	202,7
4/9	1	25	1,66	102,5	1,88	187,6	1,88	253,4
6/9	1	25	2	118,3	2,33	223,9	2,33	305,4
8/9	1	25	2,33	133,7	2,77	259,2	2,77	356,1

TABLE 5.2 – Exemple d'ajustement de la fonction de propagation

On remarque que plus la chaîne Pt_n est longue, plus la valeur retournée par la fonction $P^0_{\Phi_{l^0}}$ augmente. De même, plus l^0 augmente plus la valeur retournée par $P^0_{\Phi_{l^0}}$ augmente.

En fixant $\theta T^0_{s_0} = T^0_{s_0} * \theta P^m_{s_0} = 70 * 3 = 210$, on peut remarquer que plus la valeur l^0 augmente plus la taille des chaînes valides diminue (voir les cases grises du tableau) comme suit :

- $l^0 = 0$ et $l^0 = 2/9$: tous les sites sont considérés comme des sites de confiance car la valeur retournée par la fonction de propagation est inférieure à 210.
- $l^0 = 4/9$: seul s_4 n'est pas considéré comme un site de confiance car $P^0_{\Phi_{4/9}}(Pt_4) = 245,7$ qui est supérieure à 210.
- $l^0 = 6/9$ et $l^0 = 8/9$: les sites s_3 et s_4 ne sont pas des sites de confiance car les valeurs retournées par la fonction de propagation pour Pt_3 et Pt_4 sont supérieures à 210.

Comparatif Comme défini précédemment, dans la section 'Disposition to Transitivity', deux méthodes d'ajustement ont été mises en place :

1. Ajustement du seuil de méfiance transitif.
2. Ajustement de la fonction de propagation.

Cependant, la première solution est la moins coûteuse à implémenter étant donné que la seconde impose à celui qui initie la requête (la cible) de fournir à toutes les entités de confiance, son Disposition Level afin qu'elles puissent calculer la fonction Φ_{l^0}. D'un autre côté, le fait d'ajuster les valeurs des arcs d'une chaîne de confiance en fonction de leur position permet de différencier un certain nombre de chaînes considérées équivalentes par la première méthode utilisant l'addition simple. Dans le but d'illustrer cette différence considérons l'exemple suivant :

Exemple : Deux chaînes de confiance vont être comparées et évaluées par les deux méthodes d'ajustement définies précédemment, comme suit :

1. La première chaîne à évaluer correspond à celle utilisée par les exemples précédents, la chaîne : Pt_4 :
   - $T^0_{s_0} = 70$, $T^0_{s_1} = 3$, $T^0_{s_3} = 17$ et $T^0_{s_4} = 50$.
   - $t^0(s_0, s_1) = 25$, $t^0(s_1, s_2) = 2$, $t^0(s_2, s_3) = 11$ et $t^0(s_3, s_4) = 25$.
2. La seconde chaîne ressemble à la première sauf pour les valeurs de confiance $t^0(s_0, s_1)$ et $t^0(s_3, s_4)$ comme suit :
   - $T^0_{s_0} = 70$, $T^0_{s_1} = 3$, $T^0_{s_3} = 17$ et $T^0_{s_4} = 50$.
   - $t^0(s_0, s_1) = 4$, $t^0(s_1, s_2) = 2$, $t^0(s_2, s_3) = 11$ et $t^0(s_3, s_4) = 40$.

En fixant le 'Disposition level' du site s_0 à $l^0 = 2/9$ et $\theta P^m_{s_0}$ à 3, on obtient, pour les deux méthodes d'ajustement, les paramètres suivants :

Méthode d'ajustement du seuil θT^0 :
- $P^m_{s_0} = 1 + (1 - \frac{2}{9}) * (3 - 1) = 2.55$
- $\theta T^0_{s_0} = 70 * 2,55 = 178,5$.

	$P^0(Pt_1)$	$P^0(Pt_2)$	$P^0(Pt_3)$	$P^0(Pt_4)$
Chaîne 1	25	71,7	117	152
Chaîne 2	4	50,7	96	152

TABLE 5.3 – Comparatif utilisant l'ajustement du seuil

Méthode d'ajustement de la fonction P^0 :
- $P^m_{s_0} = \theta P^m_{s_0} = 3$
- $\theta T^0_{s_0} = 70 * 3 = 210$
- $\Phi_{2/9}(0) = 1$, $\Phi_{2/9}(1) = 1.33$, $\Phi_{2/9}(2) = 1.44$ et $\Phi_{2/9}(3) = 1.44$.

	$P^0_{\Phi_{2/9}}(Pt_1)$	$P^0_{\Phi_{2/9}}(Pt_2)$	$P^0_{\Phi_{2/9}}(Pt_3)$	$P^0_{\Phi_{2/9}}(Pt_4)$
Chaîne 1	25	87,1	152,3	202,7
Chaîne 2	4	66,1	131,3	211,9

TABLE 5.4 – Comparatif utilisant l'ajustement de la fonction de propagation

On peut remarquer qu'en utilisant l'ajustement du seuil (tableau 5.5.3.2.0), les deux chaînes sont évaluées de manière équivalentes ce qui signifie que dans les deux cas le site s_4 est un site de confiance.

Par contre, la seconde méthode (tableau 5.5.3.2.0) retourne des valeurs distinctes. Ainsi, dans la première chaîne, on considère que le site s_4 est digne de confiance car : $P^0_{\Phi_{2/9}}(Chane1) =$

$202,7 \leq 210$; et dans la seconde chaîne on considère que le site s_4 n'est pas digne de confiance car $P^0_{\Phi_{2/9}}(Chane2) = 211,9 > 210$. Cette différence permet, dans le cas où l'exploration du graphe retrouve plusieurs chaînes de confiance, de sélectionner de manière plus précise la chaîne de confiance qui répond le plus à la politique de sécurité de la cible.

Ainsi, comme on peut le constater, la première chaîne est meilleure que la seconde, étant donné que la confiance entre s_0 et s_1 (dans la première chaîne) est supérieur à la confiance que s_3 a en s_4 (dans la seconde chaîne). La seconde méthode permet d'amplifier la méfiance d'un nœud lointain en le pénalisant par rapport à l'évaluation d'un nœud plus proche.

Cependant, tous les nœuds se trouvant à une distance supérieure au $\theta P^m_{s_0}$ sont tous considérés très éloignés et donc évalués de manière équivalente avec un coefficient reflétant une méfiance maximale. Dans ce cas, l'évaluation par la seconde méthode ne permet pas de faire cette distinction.

5.5.4 Sélection d'une chaîne de confiance

Dans un graphe de confiance, il peut exister différentes chaînes de confiance entre deux sites. L'évaluation de ces chaînes peut retourner plusieurs valeurs distinctes et la prise de décision peut alors s'avérer difficile. En effet, faut-il choisir le premier résultat obtenu, la chaîne la plus longue ou la plus courte en terme de nombre de sites, la chaîne qui retourne la plus grande ou la plus petite valeur de confiance, ou bien faire la moyenne des différentes valeurs calculées par la fonction de propagation ?

Comme pour l'évaluation directe et transitive, on peut considérer qu'un site méfiant a tendance à considérer le pire cas i.e., la chaîne la plus longue ou l'évaluation la plus grande. Par contre, un site confiant va plutôt choisir de sélectionner la première évaluation ou faire une moyenne de toutes les évaluations des chaînes de confiance existantes. Cependant, ce travail n'a pas été traité dans le cadre la thèse, mais se présente comme une perspective à notre modèle de confiance T2D.

5.6 Le simulateur "TrustSim"

Dans la littérature, plusieurs simulateurs réseaux[NS 3 09, OMNe 09] ont été proposés , mais aucun d'eux ne permet de modéliser un graphe traitant la disposition à la confiance. Pour cette raison, nous avons fait le choix d'implémenter notre propre simulateur. Ce dernier est développé dans le langage python. Il permet de charger ou de créer de manière aléatoire des réseaux en choisissant principalement leur taille et leur nature (confiante ou méfiante). Dans cette section, nous allons décrire l'interface et les différentes fonctionnalités qu'offre notre simulateur.

5.6.1 Interface

Le simulateur mis en place dans le cadre de notre modèle de confiance T2D, permet d'afficher le graphe de confiance $T_g(\mathbb{S}, \mathbb{E})$ défini précédemment. Comme le montre la figure 5.14, le simulateur T2D est composé de trois panneaux permettant d'afficher et de manipuler le graphe de confiance ainsi que de calculer et d'évaluer les chaînes de confiance entre n'importe quels nœuds du graphe.

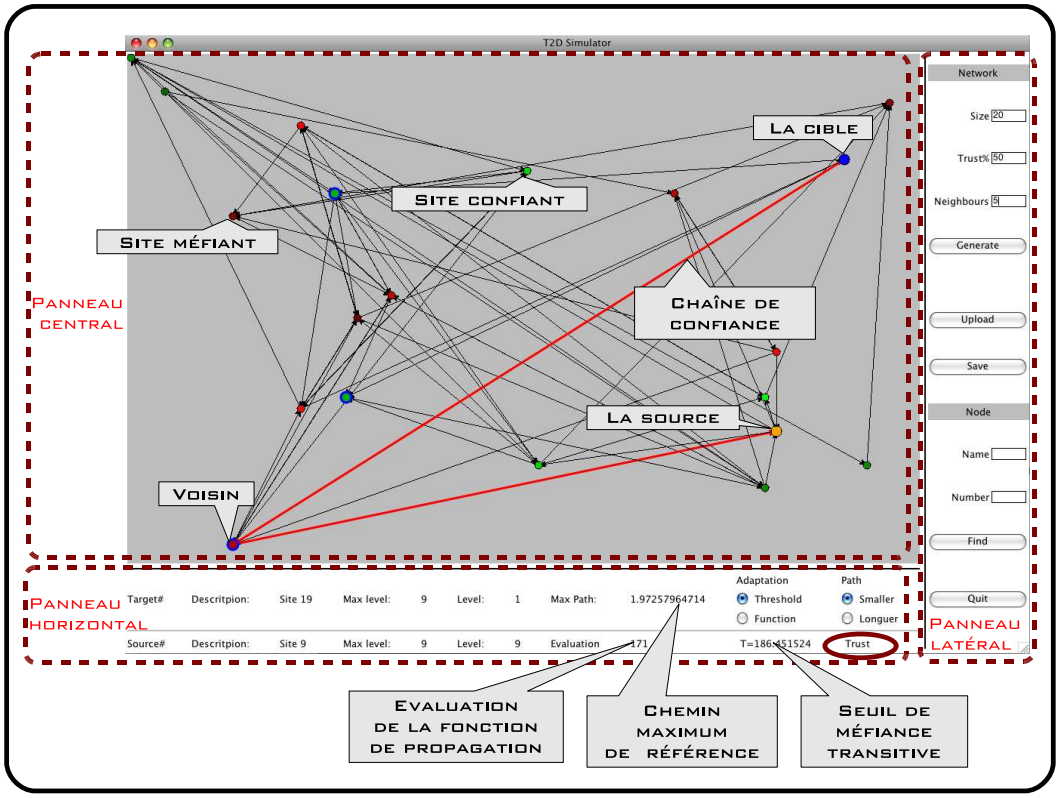

Figure 5.14 – Le simulateur T2D

5.6.1.1 Panneau latéral

Le panneau latéral sert principalement à initialiser le simulateur. Il contient l'ensemble des boutons qui permettent de :

- Générer un réseau aléatoire.
- Charger un réseau à partir d'un fichier.
- Sauvegarder le réseau affiché.

Ce panneau sert aussi à retrouver un site dans le réseau de confiance à partir de son nom ou de son numéro.

5.6.1.2 Panneau horizontal

Ce second panneau affiche les informations relatives aux différents nœuds du réseau. Comme le montre la figure 5.14 ce panneau est divisé en deux lignes :

- La première ligne désigne la cible, elle permet de fournir toutes les informations relatives à son nom, son numéro, son degré de confiance, son seuil de méfiance, etc.
- La seconde ligne désigne la source ; comme pour la première ligne celle-ci décrit le site source, et illustre si la cible lui fait confiance (cercle rouge -figure 5.14).

5.6.1.3 Panneau central

Ce panneau illustre graphiquement le graphe de confiance. Les nœuds sont représentés par des cercles et les arcs par des flèches.

Chaque cercle possède une couleur qui détermine sa personnalité :

– Un dégradé du vert pour les sites confiants (vert foncé pour un site ayant une confiance maximale i.e $l^0 = 0$)

– Un dégradé de rouge pour les sites méfiants (rouge foncé pour un site ayant une méfiance maximale i.e $l^0 = 1$)

Si la *cible* est définie, celle-ci est signalée d'un *cercle bleu*, et tous ses *sites de confiance* par un *contour bleu*. Ensuite, une fois la *source* sélectionnée, celle-ci est signalée par une couleur *orange*, et le *chemin de confiance* qui permet de relier la cible à la source est souligné en *rouge*.

5.6.2 Chargement du réseau

Comme illustré dans la section précédente, le panneau latéral permet d'afficher le réseau grâce aux deux boutons, *Generate* et *Upload*. Comme son nom l'indique, le premier bouton sert à générer un réseau aléatoire, et le second de charger un réseau de confiance à partir d'un fichier spécifique.

5.6.2.1 Génération automatique

Cette première option du simulateur est très utile pour créer rapidement un réseau de confiance de n'importe quelle taille. Comme on peut le voir dans la figure 5.14, afin de générer un réseau de confiance il faut définir les trois paramètres suivants :

– Size : Cette valeur représente la taille du réseau en nombre de sites.

– Trust% : Cette valeur est comprise entre 0 et 100 et permet de définir le pourcentage de sites confiants, par exemple, la valeur 40 signifie que 40% des sites du réseau ont un caractère confiant ($l^0 \leq 0,5$) et que les 60% des sites restants sont méfiants ($l^0 > 0,5$).

– Neighbours : Cette valeur détermine le nombre maximum de voisins par nœud.

Ainsi, en renseignant ces trois paramètres, le bouton *Generate* peut alors créer un graphe de confiance qui sera affiché dans le panneau central (voir figure 5.15).

5.6.2.2 Importation à partir d'un fichier

Dans le but de créer un réseau de nœuds illustrant notre modèle de confiance, nous avons modélisé notre graphe de confiance $T_g(\mathbb{S}, \mathbb{E})$ à l'aide des deux tableaux suivants :

– Une matrice carrée de taille $|\mathbb{S}| * |\mathbb{S}|$. Les cases de la matrice sont des réels qui représentent la valeur de confiance $t^0(i, j)$ correspondant à l'arc séparant le site i du site j.

– Un tableau qui a également une taille égale au nombre des nœuds du graphe. Les valeurs contenues dans le tableau représentent la disposition à la confiance l^0 des différents sites du réseau.

Afin de remplir ces deux structures, nous avons défini une syntaxe en format XML permettant de représenter les caractéristiques de chaque nœud du graphe de confiance. Ce fichier permet à

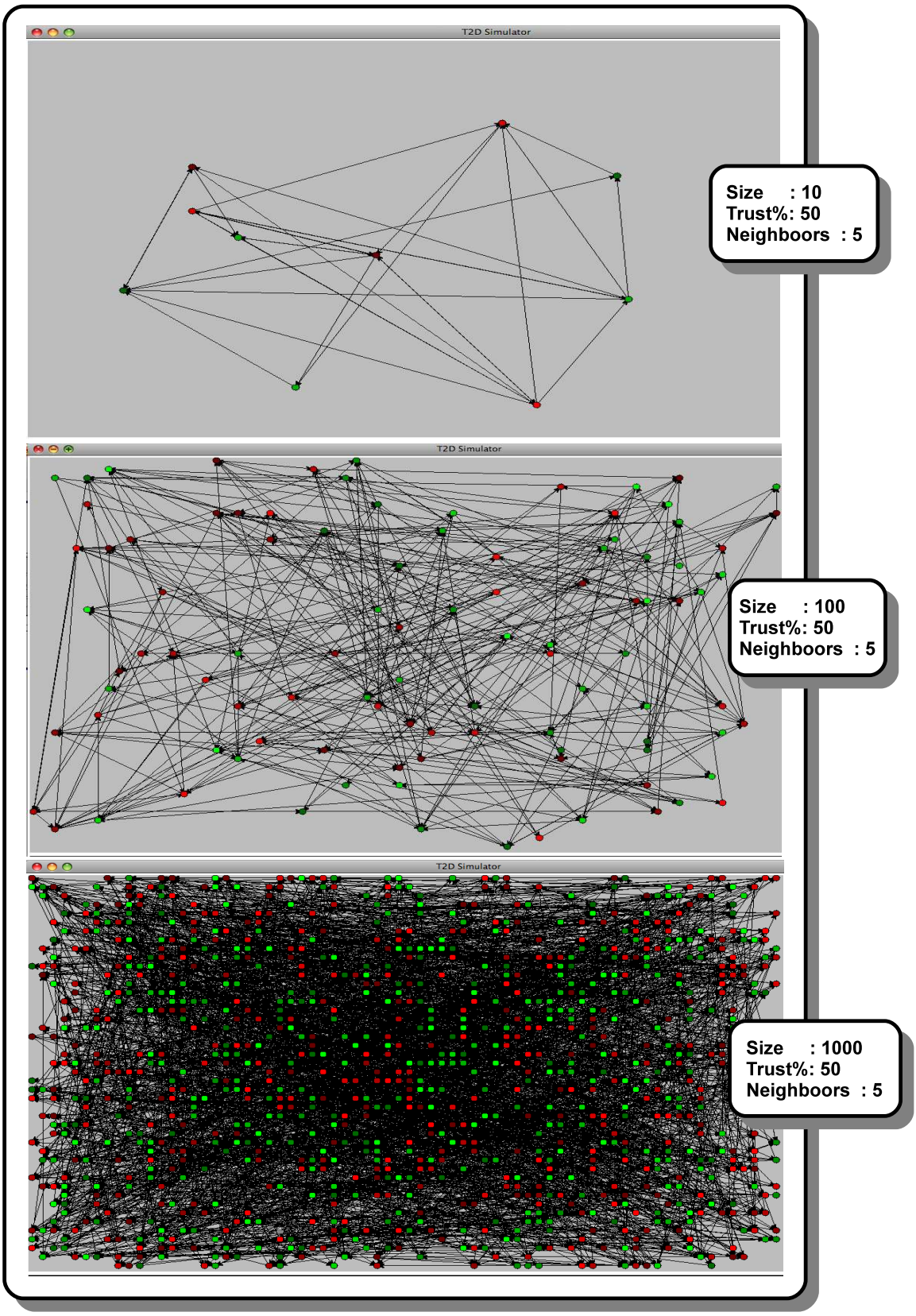

FIGURE 5.15 – Réseau de confiance

l'aide des boutons *Upload* et *Save* d'importer et d'exporter un graphe de confiance.

L'arbre XML décrivant un graphe de confiance se présente comme suit :

```
<TrustGraph>
00    <Description name= />
01    <Size value= />
02    <nodes>
03       (<node>
04            <Description name= />
05            <DispositionLevel l= />
06            <DistrustThreshold t= />
07            <neighborhoods>
08                   (<neighborhood
09                        <Node_Number id= />
10                        <TrustSort value= />
11                   <neighborhood id = >)*
12            <neighborhoods>
13        </node>)+
14    </nodes>
<TrustGraph>
```

- Ligne 00 : Cette balise permet de décrire le type ou le nom du graphe.
- Ligne 01 : Cette valeur définit la taille du réseau.
- Lignes 02-14 : Ces lignes définissent les différents nœuds du réseau. Ainsi, cette partie de l'arbre est composée d'une ou plusieurs balises *node* (+).
- Lignes 03-13 : Ces attributs décrivent les caractéristiques d'un nœud du réseau.
- Ligne 04 : Cette balise contient l'identifiant ou le nom du nœud.
- Lignes 05-06 : Ces balises dénotent le Disposition Level l^0 et le threshold.
- Ligne 07-12 : Dans cette section, sont représentés tous les voisins du nœud.
- Ligne 08-11 : Ces lignes décrivent les caractéristiques de chaque voisin, chaque site peut avoir plusieurs voisins (*).
- Ligne 09 : Dans cette balise se trouve le numéro du nœud selon l'ordre dans lequel ce dernier est représenté dans l'arbre XML.
- Ligne 11 : Selon le tri (groupé ou unitaire), cette balise contient la valeur correspondante au voisin dans la liste TrustSort.

5.6.3 Calcul d'un chemin de confiance

5.6.3.1 Définir les paramètres

Comme l'illustre le panneau horizontal, l'administrateur de la cible doit définir deux paramètres essentiels pour le calcul de la fonction de propagation.

- Le premier paramètre nommé *Ajustement* permet de définir si la cible évalue la confiance transitive en utilisant l'ajustement du seuil ou l'ajustement de la fonction de propagation (voir section 5.5.3).
- Le second paramètre nommé *Path* permet à la cible, dans le cas où plusieurs chemins de confiance sont trouvés, de choisir le chemin le plus court ou le chemin plus long (voir section 5.5.4).

5.6.3.2 Définir la cible et la source

Une fois le graphe créé et affiché dans le panneau central, actionner la première fois le bouton *Find* ou double cliquer sur un nœud du graphe, permet de sélectionner la cible et de calculer son chemin maximum de référence θP^m (voir annexe C.1). Une seconde sélection définit la source et permet de calculer et d'évaluer la chaîne de confiance la liant à la cible (le détail de l'implémentation est en annexe C.2).

5.7 Conclusion

Dans ce chapitre, nous avons défini un nouveau modèle de confiance qui permet à chaque site d'évaluer numériquement et de manière automatique les différents éléments de son cercle de confiance sortant, en fixant comme seul paramètre la variable (l^0) correspondant à la disposition à la confiance.

Le calcul de la fonction transitive a, lui aussi, bénéficié de l'apport de la disposition à la confiance afin de refléter au mieux les besoins d'une administration en terme de confiance dans la définition de sa politique de sécurité.

Cependant, deux aspects restent à approfondir. Le premier aspect concerne la méthodologie à suivre afin de sélectionner la chaîne de confiance (parmi les résultats possibles) qui correspond au mieux aux exigences et à la personnalité de chaque site. Le second aspect est lié à la mise en place d'un protocole assurant la confidentialité de la politique de confiance de chaque site. En effet, lors du processus d'évaluation de la confiance transitive (P^0), les évaluations de confiance (t^0) sont transmises de pair en pair. Par transitivité, un site donné peut alors déduire la politique de confiance d'un site distant sans que ce dernier ne donne son accord. Cette problématique fait l'objet d'un autre travail de thèse au sein de notre équipe [Hasa 08].

A la fin de ce chapitre, nous avons également détaillé l'implémentation de notre simulateur. Il nous a fourni dans le cadre de la thèse le moyen d'explorer les mécanismes de propagation de la confiance mis en place, et nous à permis de définir une méthode permettant d'initialiser de manière automatique le seuil de méfiance transitive en calculant la longueur de chemin maximum de référence θP^m (annexe C.1).

Ce simulateur peut également permettre à un administrateur de simuler un déploiement de notre solution en chargeant simplement le graphe représentant son réseau (fichier XML), qui peut être généré à l'aide de solutions tiers qui découvrent et explorent la topologie du réseau.

Cette implémentation est la première à exploiter la notion de confiance dans le calcul de chemin dans un graphe. Cependant, celle-ci n'est qu'au stade de développement, les algorithmes qui ont été implémentés pour tester le simulateur peuvent être optimisés (en terme de optimisation du trafique, de vitesse d'exécution, etc.). Le simulateur étant extensible, il peut servir de plate-forme pour tester d'autres algorithmes traitant de la confiance.

Troisième partie

Certification et Signature

Chapitre 6

Etat de l'art sur la signature numérique

Sommaire

6.1 Introduction

L'évolution de l'informatique nous plonge dans une nouvelle ère où "tout est numérique". Le support électronique s'impose de plus en plus comme une alternative aux documents papiers. Ceci nécessite l'élaboration de mécanismes qui offrent le même niveau de sécurité (voire plus) que l'équivalent papier. C'est ainsi que se sont développées plusieurs techniques cryptographiques dont les PKIs (Public Key Infrastructures), dont le but est de permettre l'identification, la réalisation d'une signature numérique fiable, d'assurer la non-répudiation et de protéger la confidentialité des données.

Dans le cadre de notre thèse, nous nous sommes intéressés au développement d'un nouveau type de certificats implémentant une nouvelle forme de signature numérique qui permet l'adaptation du contenu signé au contexte pervasif. Ainsi, et afin de motiver les choix pris pour la conception de notre certificat, nous allons dresser dans ce chapitre un état de l'art illustrant :

- Les modèles usuels de certificats numériques.
- Les différentes syntaxes d'une signature numérique.
- Les techniques de signature pouvant adapter le contenu d'un document signé aux besoins de son propriétaire.

6.2 Les certificats

Le certificat est l'attestation numérique signée la plus échangée et la plus utilisée par les systèmes d'authentification et de contrôle d'accès. Comme illustré dans le chapitre 2, deux types de certificats ont vu le jour, les certificats d'identité et les certificats d'attributs. Dans cette section nous allons présenter les modèles les plus utilisés dans la littérature qui implémentent ces deux types de documents.

6.2.1 Modèles de certificat

6.2.1.1 Certificat X509

X509 est le standard de certificat le plus implémenté et le plus utilisé dans la littérature et à travers le web. Il est formalisé dans le langage ASN.1[ITU 88, ITU 02] et en est à sa troisième version. La première et la seconde version le définissent comme un certificat d'identité. Par contre, dans sa dernière version, X509 a été pourvu d'une extension permettant de définir un certain nombre de renseignements sur l'usage de la clé publique et du certificat ainsi que les privilèges de l'utilisateur et de l'autorité de certification. X509 v3 peut être considéré comme un certificat d'identité et d'attributs composé principalement des données suivantes :

- La version et le numéro de série du certificat.
- l'identifiant de l'algorithme de signature (ex : RSA with SHA1).
- Le nom du signataire (Issuer) selon la norme X500[Chad 96] (ex : c=FR, o=Université cn=INSA).
- La date de validité.
- Le nom du sujet (selon la norme X500) et sa clé publique.
- La partie extension.
- La signature du Issuer.

6.2.1.2 Certificat PGP

PGP[Zimm 95] définit des certificats d'identité pouvant embarquer plusieurs signatures. Chaque utilisateur est libre de créer sa propre identité. Ensuite, c'est à la communauté de valider ou de rejeter le certificat (mécanisme de *web of trust*). Le processus de validation consiste à faire signer le certificat par des entités qui reconnaissent l'exactitude de son contenu. Ainsi, plus ce dernier est signé, plus il devient crédible. Le certificat PGP est composé principalement des informations suivantes :

- La version de PGP.
- La clé publique du titulaire du certificat.
- Les identifiants du titulaire (ex : nom, adresse mail, etc.).
- La date de validité du certificat.
- L'algorithme de chiffrement symétrique préféré.
- L'auto-signature du propriétaire, ainsi que les signatures des tiers qui ont confiance dans la paire clé publique / identifiant(s) du titulaire.

6.2.1.3 Certificat SPKI

Un certificat SPKI (Simple Public Key Infrastructure) peut exprimer l'identité et les attributs. A la différence du modèle X509, il est l'un des premiers à avoir intégré, lors de sa conception, les autorisations et le principe de la délégation. Lors de sa première définition en septembre 1999 (RFC 2693[Elli 99b]), il a été formalisé en utilisant le langage S-expression[Rive 97], mais par la suite un certain nombre de travaux[East 02] l'ont traduit en XML. Il est composé principalement des parties suivantes :

- L'identifiant du signataire (Issuer) : Ce paramètre est défini par une clé publique ou son empreinte (hash de la clé).
- L'identifiant du sujet : Il représente le titulaire du certificat et est défini de la même manière que l'identifiant du signataire.
- L'attribut délégation : C'est un booléen qui permet de donner au sujet le droit de déléguer l'accès aux ressources auxquelles il a droit.
- La partie autorisation : Elle est exprimée en S-expression et comporte un ensemble de règles définissant les droits du sujet.
- La date de validité.
- La signature de l'Issuer.

6.2.2 Synthèse

Le tableau ci-dessous compare les différents standards de certificat, définis précédemment, par rapport à nos besoins suivants :

- Généricité : la structure d'un certificat doit de préférence être définie en XML étant donné que ce langage de représentation est celui qui est adoptée par la majorité des standards.

- Multi-identification : avec la multiplication des dispositifs utilisés par les utilisateurs et la diversification des techniques d'authentification (ex : clés publiques, identifiants biométriques, etc.), le certificat doit pouvoir embarquer plusieurs identifiants.

- Chaîne de signatures : cette méthode permet d'attribuer plus de crédibilité au propriétaire du certificat mais également, comme on a pu le voir dans les chapitres précédents, d'exprimer et de valider une chaîne de confiance liant les différents signataires de manière transitive.

- Adaptabilité : la structure du certificat doit pouvoir s'adapter au contexte d'utilisation en permettant à son propriétaire de ne divulguer que les attributs jugés nécessaires lors des processus d'authentification et de contrôle d'accès (voir le second cas d'utilisation dans le premier chapitre).

	Généricité	Multi-identification	Chaîne de signature	Adaptabilité
X509	non	non	non	non
PGP	non	non	oui	non
SPKI	oui	non	non	non

TABLE 6.1 – Comparatif des certificats

Avec l'expansion des réseaux distribués, l'utilisation des certificats embarquant l'identité ainsi que les attributs des usagers devient incontournable. C'est pour cette raison que le certificat X509 a intégré dans sa dernière version une extension pouvant inclure des attributs.

Comme le montre le tableau ci-dessous (Généricité), le principal problème des modèles de certificats illustrés précédemment est l'utilisation de langages particulièrement complexes (ASN.1, S-expression etc.), ce qui obscurcit le contenu du certificat. Cette contrainte est restrictive, étant donné que la majeure partie des nouvelles normes traitant de la sécurité sont issues de XML (XACML, SAML, XCBF, etc.). Le seul modèle qui a essayé d'évoluer vers une écriture XML est SPKI. En outre, aucun des modèles de certificat n'a été défini afin d'accueillir plusieurs identifiants (Multi-identification).

Le certificat PGP est le seul à offrir la possibilité d'embarquer sur un même certificat plusieurs signatures, permettant ainsi de représenter des relations de confiance (web of trust). Cependant, la structure de celui-ci ne permet pas de retrouver l'ordre dans lequel les signatures s'enchaînent (Chaîne de signatures).

Enfin, tous les modèles actuels des certificats implémentent un modèle standard de signature (RSA, DSA, etc.) qui ne permet pas d'adapter leur contenu au contexte pervasif (Adaptabilité).

Nous considérons que la prochaine évolution des certificats concernera la signature. En effet, il devient indispensable de doter les certificats numériques d'une certaine flexibilité afin de permettre à son propriétaire de disposer de ses attributs de manière sélective et ainsi de s'exposer le moins possible.

Dans ce qui suit, nous allons nous intéresser à la signature numérique plus particulièrement à la syntaxe, aux algorithmes et aux méthodes qui permettent la gestion du contexte en environnement pervasif.

6.3 La signature numérique

6.3.1 Principe de la signature numérique

La signature numérique est l'équivalent électronique de la signature manuscrite. Elle est fondée sur la cryptographie asymétrique et permet de créer des documents infalsifiables. Elle repose sur un système de chiffrement utilisant une paire de clés publique et privée, dont aucune des clés ne peut être déduite de l'autre. La clé privée est gardée secrète et sert au processus de signature qui est le résultat du chiffrement du document à signer avec cette clé. Par contre, la clé publique est diffusée et permet à n'importe qui de vérifier l'authenticité de la signature générée par la clé secrète correspondante (figure 6.1). Le processus de vérification consiste à déchiffrer la signature avec la clé publique et de comparer le résultat obtenu avec le document original. En effet, le principe de la cryptographie asymétrique est que tous ce qui est fait avec une des clés ne peut être défait qu'avec l'autre. C'est ainsi que la confidentialité d'un message (cryptage) est assurée par le chiffrement inverse en utilisant la clé publique.

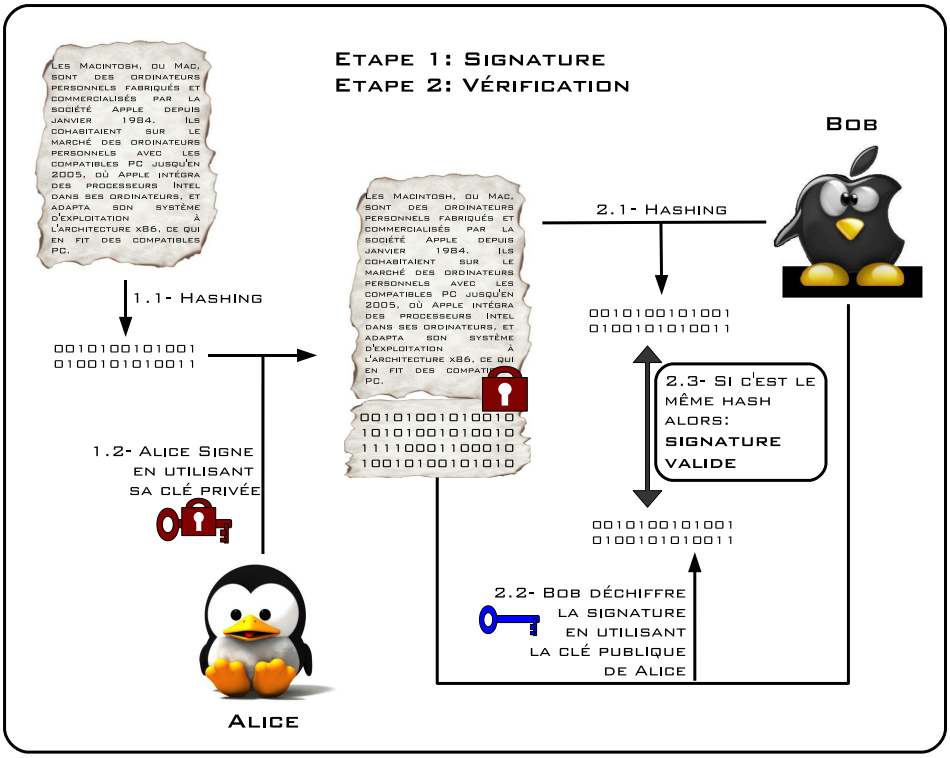

Figure 6.1 – Processus d'une signature numérique

Etant donné que le chiffrement asymétrique est très lourd, le processus de chiffrement de la signature est couplé avec une fonction de hash comme suit :

- Appliquer une fonction de hash sur le document à signer pour avoir un résidu.
- Chiffrer le résidu obtenu en utilisant la clé privée du signataire.

Cette méthode assure d'une part l'intégrité du document, car toute modification de celui-ci entraine la modification de la valeur du résidu et l'invalidation de la signature. D'autre part, elle assure l'authenticité de la signature car le signataire est le seul à détenir la clé privée qui a servi au processus de chiffrement.

6.3.2 La syntaxe d'une signature numérique

La signature numérique a été implémentée par divers modèles. Ainsi, la codification et la structuration qui permet de représenter le contenu d'une signature a été défini sous différentes formes. Dans ce qui suit, nous présentons un certain nombre de langages qui ont été utilisés pour décrire les paramètres généraux qui permettent de définir le contenu d'une signature numérique.

6.3.2.1 Tableau d'octets

Cette structure définit une codification sur un nombre d'octets afin d'identifier les différents paramètres d'une signature numérique. Cet encodage a été implémenté par PGP[Zimm 95] et tous les standards issus de ce dernier, tels que : OpenPGP[Call 07], GnuPG[GNU 09], etc.

Dans la RFC4880[Call 07] la version 3 de la signature OpenPGP est ainsi représentée par un

tableau d'octets comme suit :

1. un octet : version de la signature (par défaut la valeur 3).
2. deux octets : taille du bloc suivant qui renseigne la signature (par défaut égale à 5).
   - un octet : type de signature : (ex : 0X00 -> signature d'un document, 0X02 -> auto-signature, 0X11-> signature d'un certificat, etc.)
   - quatre octets : date de création.
3. huit octets : clé du signataire.
4. un octet : algorithme de signature (ex : 0X03 -> RSA, 0X11 -> DSA).
5. un octet : algorithme de hash (ex : 0X01 -> MD5, 0X02 -> SHA1).
6. deux octets : premiers 16 bits du résidu calculé par la fonction de hash du deuxième élément du tableau (dans le but de rejeter rapidement les signatures invalides).
7. plusieurs octets : résultat de la signature du document (la taille de cette partie dépend de l'algorithme utilisé).

6.3.2.2 ASN 1

ASN.1 (Abstract Syntax Notation One) est un standard international de destiné à l'origine à décrire les données échangées dans les protocoles de télécommunication. Standardisé en 1988, sous la norme X.208[ITU 88] et révisé en 1995 et 2002 sous la norme X.680[ITU 02], ASN.1 est un langage formel et flexible permettant de définir tout type de structure par exemple : de l'encodage, du décodage, des protocoles de transmission, etc. Il a été utilisé par un grand nombre de standards, tels que X509[Hous 99], S/MIME[Duss 98], PKCS#7[Kali 98], etc. ASN.1 est une notation qui permet de définir un ensemble de règles de codage normalisée. Ces règles de codage permettent de sérialiser les données sous forme de chaînes d'octets ayant un intitulé et un type. Dans PKCS#7, qui représente le standard de l'IETF permettant de définir la syntaxe des messages cryptographiques, une signature numérique est représentée comme suit :

```
SignedData ::= SEQUENCE {
     version Version,
     contentInfo ContentInfo, (Le contenu à signer)
     signerInfos SignerInfos  (Information sur le ou les signataires)
     }

SignerInfos ::= SET OF SignerInfo

SignerInfo ::= SEQUENCE {
     version Version,
     issuerAndSerialNumber IssuerAndSerialNumber,
     digestAlgorithm DigestAlgorithmIdentifier,
     digestEncryptionAlgorithm   DigestEncryptionAlgorithmIdentifier,
     encryptedDigest EncryptedDigest,
     }
```

où :

- issuerAndSerialNumber : identifiant du signataire
- digestAlgorithmIdentifier : algorithme de hash qui sert à calculer le résidu de ContentInfo (e.x. MD5).
- digestEncryptionAlgorithmIdentifier : algorithme de signature (e.x. RSA).
- encryptedDigest : résultat de la signature du résidu du hash de ContentInfo.

6.3.2.3 S-expression

S-expression [Rive 97] (expression Symbolique) est de la famille des langages LISP[Abra 66]. Il a été créé par Ron Rivest en 1997 et soumis à la RFC. Cependant, il n'a jamais été approuvé, bien qu'il ait été utilisé pour décrire le certificat SPKI [Elli 99b]. A l'instar des langage LISP, S-expression décrit les structures de données en forme de liste délimitée par des parenthèses, où tous les éléments sont séparés par des espaces. Chaque liste peut contenir d'autres listes de manière récursive. L'exemple suivant illustre la représentation d'une signature en S-expression pour le modèle SPKI :

```
<signature>:: "(" "signature" <hash> <principal> <sig> ")"
<hash>:: "(" "hash" <hash-alg-name> <hash-value> ")"
<principal>:: <pub-key> | <hash-of-key>
         <sig>:: "(" <sig-alg-name> <sig-value> ")"
```

où :

- "signature" et "hash" : chaînes de caractères permettant d'identifier les informations contenues entre parenthèses.
- <hash-alg-name> : nom de l'algorithme de hash (ex : sha1, md5, etc.)
- <hash-value> : résidu du document à signer.
- <principal> : soit la clé publique du signataire (<pub-key>) soit son hash (<hash-of-key>.
- <sig-alg-name> : nom de l'algorithme de siganture (ex : RSA, DSA, etc.)
- <sig-value> : résultat de la signature.

6.3.2.4 XML

XML (eXtensible Markup Language) est un langage balisé, recommandé par le W3C. Il est utilisé dans plusieurs standards tels que : XHTML[W3C 06], XMLDSIg[W3C 08], XMLEncryption [W3C 02], XCBF[Larm 03], XAdES[W3C 03] etc.

XMLDSig (XML Digital Signature) est une recommandation conjointe du W3C et de l'IETF (Internet Engineering Task Force) publiée le 12 Février 2002. Elle constitue aujourd'hui un standard sur lequel de nombreuses solutions se basent (ex. SAML). En effet, c'est le modèle de signature le plus abouti. Il englobe de nombreuses techniques de signature tout en offrant une flexibilité pour évoluer et intégrer de nouvelles fonctionnalités. Comme son nom l'indique, cette norme permet l'encodage d'une signature numérique sous forme XML. Dans ce standard, une

signature est représentée par la syntaxe suivante :

```
<Signature ID?>
     <SignedInfo>
       <CanonicalizationMethod/>
       <SignatureMethod Algorithm= />
       (<Reference URI? >
         (<Transforms>)?
         <DigestMethod Algorithm= >
         <DigestValue>
       </Reference>)+
     </SignedInfo>
     <SignatureValue>
    (<KeyInfo>)?
</Signature>
```

où :

- Annotations :
  - " ?" : la balise est optionnelle.
  - "+" : la balise apparaît au moins une fois.
- SignedInfo : paramètres qui permettent de générer la signature.
- CanonicalizationMethod : transformation du document XML en forme canonique (ex : remplacer <item > par <item>).
- SignatureMethod : Définit l'algorithme de signature, par exemple : <SignatureMethod Algorithm="http ://www.w3.org/2000/09/xmldsig#dsa-sha1"/>.
- Reference : permet d'identifier le ou les contenus à signer. Ainsi chaque contenu est représenté par :
  - URI : lien permettant d'indexer le contenu.
  - Transforms : définit la ou les transformations à appliquer sur le contenu avant de calculer la fonction de hash (voir chapitre 8).
  - DigestMethod : définit l'algorithme de hash appliqué sur ce contenu, par exemple : <DigestMethod Algorithm="http ://www.w3.org/2000/09/xmldsig#sha1"/>.
  - DigestValue : contient le résidu qui correspond au Hash de la partie référencée.
- SignatureValue : Contient le résultat de la signature de toutes les parties référencées.
- KeyInfo : Renseigne la clé publique du signataire.

Remarque : Chacun des trois standards détaillés précédemment utilise une notation normalisée qui lui est spécifique pour identifier les algorithmes de hachage et de signature.

6.3.2.5 Synthèse

Plusieurs langages ont été utilisés pour représenter, de manière structurée, une signature numérique et tous les paramètres servant à la générer. De tous les standards introduits précédemment, XMLDSig nous semble le plus lisible et le plus complet, mais surtout le plus flexible pour accueillir de nouveaux algorithmes nécessitant une transformation du document original (cf. balise $Transform$) en gardant une grammaire inchangée.

Dans le but d'intégrer la signature numérique dans le contexte pervasif, nous allons présenter, dans ce qui suit, les différentes solutions proposées dans la littérature qui permettent d'adapter le contenu d'un document afin de protéger les attributs de son propriétaire, tout en gardant la signature valide.

6.3.3 Signature adaptable au contexte pervasif

Le principe de la signature numérique est que lorsqu'un document est signé, il ne peut être modifié, ni démuni d'aucune information. Cependant, prenons l'exemple du certificat X509 d'Alice (voir le scénario du consommateur -Chapitre 1-), contenant son nom, sa date de naissance, sa profession etc. Avec les mécanismes classiques de signature, Alice n'a pas la possibilité de masquer, par exemple, sa profession ; elle est dans l'obligation de divulguer tout le contenu de sa pièce d'identité (son certificat) à chaque fois qu'elle doit la présenter.

La résolution de cette problématique, consiste à définir un nouveau modèle de signature capable de doter la structure du document signé d'une certaine flexibilité. Ce mécanisme de signature a été définie sous différentes appellations telles que : Digitally signed document sanitizing schemes[Miya 06], Content extraction signature[Stei 01], ou encore Redactable signature[John 02].

La définition la plus courante de ce type de signature est celle de Steinfeld et al [Stei 01] : "*A Content Extraction Signature should allow anyone, given a signed document, to extract a publicly verifiable extracted signature for a specified subdocument of a signed document, without interaction with the signer of the original document*"

Ainsi, une signature adaptable au contexte doit permettre aux utilisateurs de manipuler leurs documents signés de manière sélective, en ne révélant que les attributs qu'ils jugent nécessaires pour satisfaire le contexte dans lequel ils se trouvent. Cependant, en plus de cette caractéristique, ce type de signature doit être capable de gérer les propriétés suivantes :

- La dissociation : Dans un certificat ou tout autre document signé, il est important de dissocier les parties autorisées à être masquées de celle qui ne le peuvent pas. Par exemple, la date de validité ou le numéro de série constituent des attributs indissociables d'un certificat. Dans ce cas de figure, quel que soit le contexte dans lequel l'utilisateur se trouve, son certificat doit toujours embarquer ces attributs.

- L'intégrité : Dans un certain nombre de documents signés, cacher ou enlever une sous-partie peut altérer leur structure. En prenant l'exemple d'une image BMP, le fait de masquer une

zone de l'image va décaler l'ordre des lignes et ainsi fournir une image altérée. Pour cette raison, la signature doit garantir l'intégrité de la structure du document signé quel que soit la partie masquée.

– La compatibilité : Ce nouveau type de signature doit être compatible avec les standards de signature existant, en s'intégrant sans avoir besoin de modifier en profondeur les processus de signature classiques.

Dans la suite de l'état de l'art, nous avons choisi de noter les différents intervenants dans le processus de signature comme suit :

– *Le signataire* est celui qui génère la signature du document original.
– *Le propriétaire* est celui qui est habilité à créer des sous-documents et à distribuer les différentes versions.
– *Le contrôleur* représente le dernier maillon de la chaîne. Il doit pouvoir vérifier l'authenticité de toutes les versions fournies par le propriétaire.

En pratique, le signataire signe un document avec ce nouveau type de signature. Le propriétaire reçoit le document et peut l'utiliser dans différentes transactions en adaptant le contenu à chaque contexte. Le propriétaire est parfaitement autonome et ne sollicite à aucun moment le signataire du document.

Dans ce qui suit nous allons présenter un état de l'art des solutions mathématiques et algorithmiques mettant en place une signature adaptable au contexte.

6.3.3.1 Signatures homomorphiques

Rivest et al. sont les premiers à avoir introduit la notion des signatures homomorphiques. Lors d'une série de présentations [Gold 88], Rivest a présenté deux schémas de signature : la signature transitive et la signature agrégée :

– En découpant un document D en trois sous-documents x,y et z, la signature transitive permet de déduire à partir des deux signatures Sig (x, y) et Sig (y, z), une signature valide Sig (x, z) sans avoir accès à la clé secrète. Avec ce type de signature, le propriétaire peut cacher le sous-document 'y' en ayant seulement les deux tranches Sig(x,y) et Sig(y,z).
– Avec la signature agrégée, chaque sous-document est signé par le signataire séparément. Ensuite, la concaténation de n'importe quelles sous-parties (ex : Sig(a).Sig(b)=Sig(a.b)) permet de générer une signature agrégée valide sans utiliser la clé secrète. Ce type d'opération ne s'applique qu'à une catégorie spécifique d'algorithme tel que RSA.

CES (context extraction signature)[Stei 01] utilise le mécanisme d'agrégation de signatures générées par une clé RSA. Ce même mécanisme a également été implémenté par Brands[Bran 02] qui l'utilise dans le cadre des certificats.

Toutes les méthodes précédentes découpent un document en plusieurs sous-documents qui sont traités par le processus de signature de manière identique, ce qui permet de masquer n'importe quelle partie sans pouvoir distinguer les parties statiques des parties dynamiques.

Nous avons fait le choix de ne pas trop détailler les algorithmes cités précédemment étant donné que ces derniers ne s'appliquent qu'à une catégorie bien précise d'algorithmes de signature à savoir ceux qui présentent des propriétés homomorphiques, ce que nous considérons restrictif. Ainsi, dans la section suivante, seront présentées des solutions qui ne se basent pas sur les propriétés mathématiques des signatures numériques mais modifient le processus algorithmique de ces dernières. Ceci a comme avantage de présenter des approches pouvant intégrer n'importe quel standard de signature (e.x. RSA, DSA etc.).

6.3.3.2 Signature de Johnson et al.

Comme illustré dans la figure 6.2, l'algorithme de Johnson et al. [John 02] découpe un document "D" en plusieurs sous-parties d_i. Ensuite, à partir d'une clé aléatoire k_0 un ensemble de clés (correspondant au nombre des sous-parties) est généré de manière récursive grâce à la fonction G tel que $G(k_n) =< k_{n0}, k_{n1} >$. Enfin, toutes les unités sont hachées par le fonction H deux à deux de manière récursive (en remontant l'arbre) jusqu'à l'obtention d'un seul résidu h_0 qui est signé par le signataire (voir figure 6.2). La valeur obtenue ainsi que la clé k_0 sont envoyées au propriétaire comme étant la signature du document D.

Dans le cas où le propriétaire veut masquer une partie du document (ex : d_{010}), il doit fournir la signature de h_0 ainsi que les valeur des noeuds se trouvant autours de la partie masquée dans l'arbre (k_0, k_{011} et h_{011} - voir figure 6.2-), afin de permettre au contrôleur de reconstruire le résidu final h_0. Cette méthode à l'avantage, la plupart du temps, d'agréger en un seul hash un certain nombre de parties masquées.

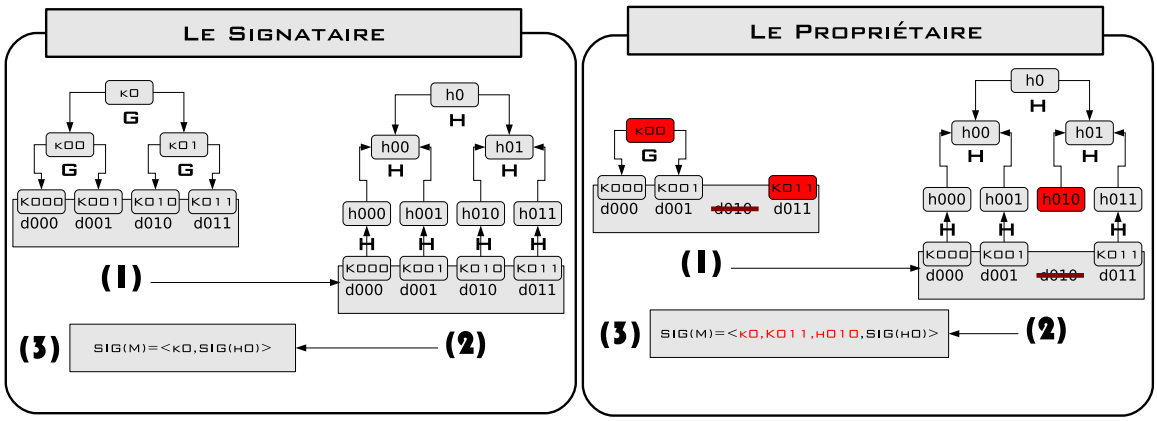

FIGURE 6.2 – Signature de Johnson et al.

Ce type d'algorithme a été utilisé par Haber et al. [Habe 08]. Mais, au lieu de considérer toutes les sous-parties dans le processus de hash, seules celles considérées comme étant amovibles sont concernées par le processus de hachage successif. Ceci permet de différencier les parties pouvant être masquées des parties persistantes.

Cependant cette méthode est coûteuse en terme de temps de calcul, que ce soit pour le processus de la signature ou pour procéder à la vérification, puisque pour n sous-documents il faut exécuter la fonction de hash $[(2 * n) - 1]$ fois.

6.3.3.3 Signature de SUMI

Miyazaki et al. [Kuni 03], ont défini en 2003 une version de leur algorithme SUMI qui permet d'augmenter la confidentialité des messages hachés afin de rendre plus difficile le reverse engineering. En effet, si le message haché est de petite taille (e-mail, numéro de téléphone etc.) une attaque à force brute (essayer toutes les combinaisons) peut facilement retrouver la texte en claire.

Donc, en prenant un document D décomposé en n sous parties notées d_i pour $0 \leq i < n$, le signataire génère la signature SUMI de la façon suivante :

1. Pour chaque sous-partie 'd_i' est généré aléatoirement un bloc d'octets noté r_i.
2. Pour chaque paire (d_i, r_i) calculer $h_i = HASH(d_i, r_i)$ qui représente le résultat de la fonction de hash appliquée à la concaténation de d_i avec r_i
3. Générer la signature S=Sig(H) à partir de l'ensemble H qui contient tous les h_i.
4. Enfin, la signature en sortie est représentée par le tuple suivant : $(d_0, ...d_{n-1}, r_0....r_{n-1}, S)$

De cette manière, si une partie d_i doit être cachée par le propriétaire, il suffit de remplacer d_i par le h_i correspondant, et d'enlever de la signature le r_i. Ceci se traduit en sortie par la signature suivante : $(d_0, ..., h_i, ...d_{n-1}, r_0, ..., r_{i-1}, r_{i+1}, ...r_{n-1}, S)$

PIATS[Izu 07] utilise ce même principe mais présente l'avantage d'obliger chaque version générée à partir du document original à être signée par son propriétaire.

En 2005, la méthode SUMI[Miya 05] a été améliorée ; elle prend désormais en compte la propriété de dissociation définie précédemment. Dans cette nouvelle version, le signataire procède comme suit :

1. Pour chaque sous-partie d_i d'un document D sont générés aléatoirement deux blocs d'octets notés respectivement r_i et t_i.
2. Pour chaque i-ème sous-partie est définie la ligne l_i qui passe par les deux points aux coordonnées respectives $(1, hash(d_i, r_i))$ et $(2, hash(t_i))$.
3. Calculer p_i et q_i tels que les points $(0, q_i)$ et $(3, p_i)$ appartiennent à la droite l_i.
4. Calculer la signature $S = Sig(q_0...q_{n-1}, p_0, ...p_{n-1})$.
5. Enfin, la signature du document D est représentée par le tuple suivant : $(d_0....d_{n-1}, r_0....r_{n-1}, t_0....t_{n-1}, p_0....p_{n-1}, S)$

Avec cette dernière version, si le propriétaire veut :

- masquer une partie d_i, il suffit d'enlever d_i du document fourni en sortie, car les valeurs t_i et p_i correspondantes suffisent à déduire la valeur q_i nécessaire pour calculer la signature.

- faire apparaître une partie mais ne pas autoriser à posteriori le masquage, il suffit de ne pas fournir au contrôleur le t_i correspondant étant donné que cette valeur est indispensable en cas de masquage. Cependant, cela ne va pas empêcher de vérifier quand même la signature, puisque les deux points $(3, p_i)$ et $(1, h_i)$ permettent de calculer la valeur de q_i.

6.3.3.4 Synthèse

Le tableau ci-dessous compare les méthodes citées précédemment par rapport aux trois propriétés introduites au début de la section. On remarque ainsi que seuls Haber et al. et SUMI traitent la contrainte de dissociation, car il ne suffit pas de découper un fichier en plusieurs blocs, il faut également dissocier les parties amovibles des parties statiques.

Malheureusement, aucune des solutions présentées dans l'état de l'art ne met en place une implémentation concrète (API, jeu de test) de leur approche dans un contexte de signatures standardisées, tel que XMLDSig. De plus, le découpage ou le masquage d'une partie d'un fichier, peut porter atteinte à son intégrité en le rendant non valide à l'exécution (par exemple, dans une image BMP il suffit de enlever une partie du fichier image pour que celui-ci ne puisse plus s'ouvrir avec un éditeur d'image).

	Dissociation	Intégrité	Compatibilité
johnson et al.	non	non	non
Haber et al.	oui	non	non
PIATS	non	non	non
SUMI	oui	non	non

TABLE 6.2 – Comparatif des signatures gérant l'adaptation au contexte

6.4 Conclusion

Il ressort de l'état de l'art que les certificats et les signatures numériques classiques manquent de flexibilité et ne répondent pas aux besoins émergents des utilisateurs et des infrastructures nomades, plus particulièrement à la problématique concernant la protection de la vie privée.

Ainsi, l'étude de l'état de l'art sur la signature numérique nous a poussé à mettre en place deux solutions traitant des formats de certificats, et du mode de signature.

Dans le chapitre suivant, nous allons donc définir un nouveau modèle de certificat formalisé en XML, ayant un format structuré lui donnant la capacité de représenter tout type de certificat (identité, attributs).

Celui-ci bénéficie également d'un nouveau modèle de signature qui lui permet de faire un masquage sélectif. Ce nouveau modèle qu'on nommera : "Fenrir morph signature" sera introduit dans les chapitres 8 et 9. Fenrir morph signature a été parfaitement intégrée dans XMLDSig et répond aux propriétés de dissociation, d'intégrité et de compatibilité.

Chapitre 7

Le certificat X316

Sommaire

7.1 Introduction

Dans les chapitres précédents, nous avons détaillé les trois standards de certification X509, SPKI et PGP (voir chapitre 5) ainsi qu'un certain nombre de variantes (voir chapitre 2) qui ont été mis en place dans le cadre d'infrastructures distribuées, telles que : Akenti[Thom 99], PERMIS[Chad 03], etc.

Comme on a pu le constater, aucune des solutions existantes ne répond pleinement à nos besoins. Ainsi, nous avions le choix entre utiliser un standard et le modifier ou bien en définir un nouveau. La définition d'un nouveau modèle de certificat s'avère une bonne alternative, étant donné que la surcharge d'un certificat de type X509 ou SPKI va générer de toutes les manières une nouvelle structure.

Dans ce chapitre, nous allons définir notre modèle de certificat nommé X316[7] : $_{13}Morph$ $_{1}Access$ $_{16}Pass$ Certificate. Il peut être vu comme un passe-partout, offrant à son propriétaire non seulement la possibilité d'interagir avec les ressources alentour, mais aussi la possibilité d'acquérir dynamiquement de plus en plus de droits en fonction des attributs d'origine.

Le certificat X316 a comme principaux objectifs :

- Définir une structure générique qui permet d'exprimer un large panel d'attributs de manière bien définie.
- Exprimer la confiance et les chaînes de certificats.

7. Indice : "A" est la première lettre de l'alphabet ...

- Autoriser la multi-identification vu que l'environnement pervasif implique l'utilisation d'une large gamme de dispositifs à capacité variable.
- Intégrer une nouvelle méthode de signature assurant une flexibilité contrôlée de l'utilisation des certificats numériques par leurs propriétaires.

7.2 Structure du certificat X316

La structuration de notre certificat X316 est totalement décrite en XML, faisant apparaître quatre nœuds à partir de la racine. Ces nœuds ont été définis dans le but de mettre en place une structure plus lisible facilitant ainsi l'indexation et la recherche, comme suit :

- **Header :** Ce nœud définit l'entête, autrement dit l'identité du certificat. Il contient toutes les informations susceptibles de renseigner l'identité du signataire, le type du certificat, la date de validité, etc.
- **Permission :** Ce nœud contient tous les attributs définissant les droits du propriétaire chez l'entité fournissant le certificat.
- **Identification :** Cette partie contient tous les types d'attributs pouvant identifier le propriétaire du certificat par exemple : la/les clé(s) publique(s) ou l'empreinte digitale.
- **Signature :** Cette partie ressemble au standard XMLDSIG mais intègre un certain nombre d'éléments la rendant plus adaptée à un usage pervasif.

La structure globale du certificat X316 est ainsi définie en format XML, comme suit :

```
<X316 xmlns="http://liris.cnrs.fr/~rsaadi/X316MCert#"
xmlns:ds[a]="http://www.w3.org/2000/09/xmldsig#"
  <HEADER>
  <PERMISSION>
  <IDENTIFICATION>
  <SIGNATURE>
</X316>
```

a. Cet espace de nom nous servira afin d'utiliser la syntaxe défini par le standard XMLDSig.

Dans le reste de ce document, pour une bonne lisibilité, nous allons utiliser la définition syntaxique du W3C pour décrire les différentes parties du certificat X316. Cette syntaxe utilise des opérateurs pour définir la fréquence d'apparition d'un élément comme suit :

- " ?" : l'élément apparaît au maximum une fois.
- "+" : l'élément apparaît au minimum une fois.
- "*" : l'élément peut apparaître entre 0 est une infinité de fois.
- "DN définition" signifie que l'élément est décrit selon la norme "Distinguished Name" de LDAP [Wahl 97] (ex : "o=liris, cn=Rachid Saadi").

7.2.1 HEADER

L'entête (Header) représente le descriptif du certificat. Comme le montre la figure 7.1, celle-ci décrit principalement trois parties :

- L'identité du certificat (Cert_Identity).
- Le signataire (Signer).
- Le propriétaire de ce certificat (Subject).

Par rapport à l'état de l'art, le Header du certificat X316 intègre fortement la notion de confiance en définissant les variables :

- CertificateDegree (contennue dans la balise Cert_Identity) : qui représente la valeur de confiance qu'accorde le signataire (Signer) au propriétaire du certificat (Subject).
- TrustPath (contennue dans la balise Signer) : qui permet de représenter un chaîne de confiance (i.e certificats).

Cert_Identity

```
01      <Cert_Identity>
02         <CertifiateID>
03         <Type>
04        (<CertificateDegree>
05             <ComputeMethode Algorithm=/>
06             <Values> <Value id=>+ </Values>
07         </CertificateDegree>)?
08         <Validity>
09             <NotAfter>
10             <NotBefore>
11         </Validity>
12      </Cert_Identity>
```

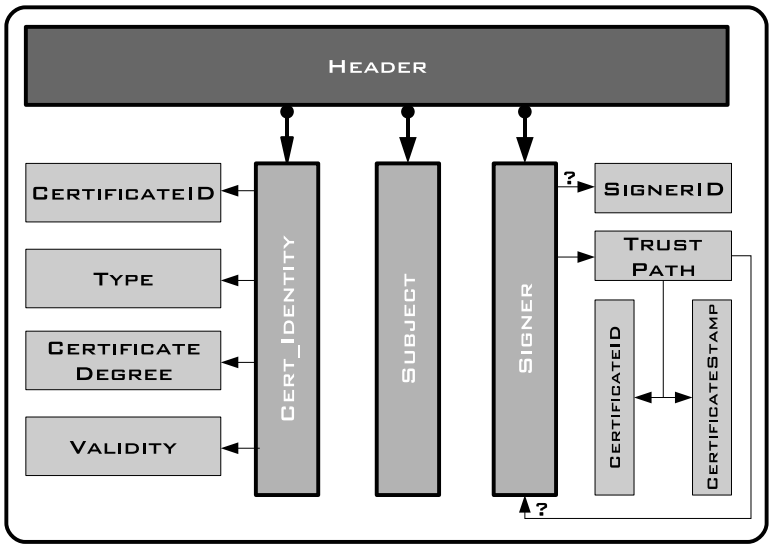

Figure 7.1 – Le Header du X316

Cette partie définit l'identité du certificat. Elle présente, comme pour tout certificat, l'identifiant du certificat et sa durée de validité. Mais, en plus de ces éléments, le *Cert_Identity* contient deux nouveaux items, qui sont le *Type* du certificat (voir le chapitre 9) ainsi que le degré du certificat *CertificateDegree*. Ce dernier représente l'évaluation du signataire. Elle peut être représentée par plusieurs indices, par exemple : la confiance (trust), la méfiance (distrust), etc.

Cette évaluation est très importante pour notre système Chameleon. Comme défini dans le chapitre 3, notre approche pervasive est basée sur la confiance qui peut exister entre les différents sites régissant l'environnement. Ce degré représente la confiance qu'accorde le signataire au propriétaire du certificat (Subject).

Exemple :

```
<CertificateDegree>
    <ComputeMethode Algorithm="http://liris.cnrs.fr/~rsaadi/X316MCert/degree#T2D"/>
    <Values> <Value id="TrustDegree"> 45 </Value> </Values>
</CertificateDegree>
```

Subject :
L'élément Subject désigne le propriétaire ou l'entité concernée par le certificat. Le *Subject* est défini en utilisant le standard LDAP "DN définition".

Signer :
Cette partie définit le signataire du certificat (ex : CA : autorité de certification) ainsi que les entités qui ont collaboré à la création de ce certificat (chaîne de confiance). Comme on peut le voir sur la figure 7.1, la partie *Signer* est composée de deux éléments, à commencer par le "SignerID" (ligne 14) qui est défini en utilisant "DN définition" et le "TrustPath" lignes (15-19) qui représente la chaîne de confiance reliant le site parent de l'utilisateur (la "source") et le signataire (la "cible").

Le $TrustPath$ est composé de trois éléments :

- Le $CertificateID$ et le $CertificateStamp$ qui définissent respectivement l'identifiant et l'empreinte du certificat illustrant la relation de confiance entre la cible et son site de confiance.

- L'élément *Signer* permet d'exprimer récursivement la totalité d'une chaîne de confiance.

```
13      <Signer>
14          <SignerID>
15         (<TrustPath>
16              <CertificateID>
17              <CertificateStamp>
18                <ds:DigestMethod>
19                <ds:DigestValue>
20             <Signer> ...</Signer>
21          </TrustPath>)?
22      </Signer >
```

Exemple : Dans cet exemple nous allons détailler le $Header$ qui compose les certificats X316 de professeur Bob (voir l'exemple du professeur -Chapitre 1-) membre de l'université A, sachant qu'il existe une chaîne de confiance reliant les universités A, B et C, telle que : $C\ Trust\ B$ et $B\ Trust\ A$.
L'université A délivre à Bob le certificat suivant :

```
C01  <X316  xmlns="http://liris.cnrs.fr/~rsaadi/X316MCert#"
C02     xmlns:ds="http://www.w3.org/2000/09/xmldsig#">
C03    <HEADER>
C04        <Cert_Identity>
C05           <CertificateID> 587 </CertificateID>
C06           <Type>...</Type>
C07           <CertificateDegree>
C08               <ComputeMethode Algorithm="http://liris.cnrs.fr/~rsaadi/X316MCert/degree#T2D"/>
C09               <Values>  <Value id="TrustDegree">0</Value>  </Values>
C10           </CertificateDegree>
C11           <Validity>
C12               <NotAfter>  ...  </NotAfter>
C13               <NotBefore> ... </NotBefore>
C14           </Validity>
C15        </Cert_Identity>
C16        <Subject> cn=Bob, o=UnivA </Subject>
C17        <Signer>
C18           <SignerID>  o=UnivA </SignerID>
C19        </Signer >
C20    </HEADER>
C21    <PERMISSION> ... </PERMISSION>
C22    <IDENTIFICATION> ... </IDENTIFICATION>
C23    <SIGNATURE> ... </SIGNATURE>
C24  </X316>
```

Comme on peut le voir, l'université A (ligne C18) délivre à Bob (ligne C16) un certificat X316 ayant comme identifiant "587" (ligne C05). Le degré de confiance est égale à 0 (ligne C09) puisque Bob appartient au domaine du signataire de ce certificat.

Ensuite, en utilisant ce certificat, Bob accède à l'université B et obtient donc le certificat suivant :

```
B01  <X316 xmlns="http://liris.cnrs.fr/~rsaadi/X316MCert#"
B02     xmlns:ds="http://www.w3.org/2000/09/xmldsig#">
B03    <HEADER>
B04        <Cert_Identity>
B05           <CertificateID> 8567 </CertificateID>
B06           <Type>...</Type>
B07          <CertificateDegree>
B08               <ComputeMethode Algorithm="http://liris.cnrs.fr/~rsaadi/X316MCert/degree#T2D"/>
B09               <Values>  <Value id="TrustDegree">25</Value>  </Values>
B10           </CertificateDegree>
B11           <Validity>
B12               <NotAfter>  ...  </NotAfter>
B13               <NotBefore> ... </NotBefore>
B14           </Validity>
B15        </Cert_Identity>
B16        <Subject> cn=Bob, o=UnivA </Subject>
B17        <Signer>
B18           <SignerID>  o=UnivB </SignerID>
B19           <TrustPath>
B20                <CertificateID>  587 </CertificateID>
B21                <CertificateStamp>
B22                     <ds:DigestMethod Algorithm="http://www.w3.org/2000/09/xmldsig#sha1"/>
B23                     <ds:DigestValue>dGhpcySBzaWduK...</ds:DigestValue>
B24                </CertificateStamp>
B25                <Signer>
B26                   <SignerID>  o=UnivA </SignerID>
B27                </Signer>
B28           </TrustPath>
B29        </Signer >
B30    </HEADER>
B31    <PERMISSION> ... </PERMISSION>
B32    <IDENTIFICATION> ... </IDENTIFICATION>
B33    <SIGNATURE> ... </SIGNATURE>
B34  </X316>
```

Comme on peut le constater ce nouveau certificat a comme identifiant "8567", il dénote la confiance qu'accorde l'université B à l'université A à savoir 25 (ligne B09). On peut remarquer également que la partie *TrustPath* est renseignée (lignes (B19-B28)). Cette dernière comporte l'identifiant (ligne B20), l'empreinte du certificat précèdent (lignes B21-B24) ainsi que la partie *Signer* du certificat qui a été délivré à Bob par son site d'appartenance (l'université A).

Enfin, en utilisant ce dernier certificat, Bob accède a l'université C et obtient le certificat suivant :

```
A01  <X316 xmlns="http://liris.cnrs.fr/~rsaadi/X316MCert#"
A02     xmlns:ds="http://www.w3.org/2000/09/xmldsig#">
A03    <HEADER>
A04        <Cert_Identity>
A05           <CertificateID> 39762 </CertificateID>
A06           <Type>...</Type>
A07          <CertificateDegree>
A08              <ComputeMethode Algorithm="http://liris.cnrs.fr/~rsaadi/X316MCert/degree#T2D"/>
A09              <Values>  <Value id="TrustDegree">55</Value>  </Values>
A10           </CertificateDegree>
A11           <Validity>
A12              <NotAfter>  ...  </NotAfter>
A13              <NotBefore> ... </NotBefore>
A14           </Validity>
A15        </Cert_Identity>
A16        <Subject> cn=Bob, o=UnivA </Subject>
A17        <Signer>
A18          <SignerID>  o=UnivC </SignerID>
A19          <TrustPath>
A20             <CertificateID>  8567 </CertificateID>
A21             <CertificateStamp>
A22                <ds:DigestMethod Algorithm="http://www.w3.org/2000/09/xmldsig#sha1"/>
A23                <ds:DigestValue>bd4AeRfySD67eR4d...</DigestValue>
A24             </CertificateStamp>
A25             <Signer>
A26                 <SignerID>  o=UnivB </SignerID>
A27                 <TrustPath>
A28                    <CertificateID>  587 </CertificateID>
A29                    <CertificateStamp>
A30                       <ds:DigestMethod Algorithm="http://www.w3.org/2000/09/xmldsig#sha1"/>
A31                       <ds:DigestValue>dGhpcySBzaWduK...</ds:DigestValue>
A32                    </CertificateStamp>
A33                    <Signer>
A34                       <SignerID>  o=UnivA </SignerID>
A35                    </Signer>
A36                 </TrustPath>
A37             </Signer>
A38          </TrustPath>
A39        </Signer >
A40    </HEADER>
A41    <PERMISSION> ... </PERMISSION>
A42    <IDENTIFICATION> ... </IDENTIFICATION>
A43    <SIGNATURE> ... </SIGNATURE>
A44  </X316>
```

Comme pour le précédent certificat celui-ci reprend la partie *Signer* du certificat délivré par l'université B (lignes (B17- B29, A22-A31)). Il dénote un degré de confiance égale à 55.

7.2.2 PERMISSION

Cette partie du certificat X316 est assez flexible et dépend de la politique locale de l'entité délivrant le certificat. La partie "PERMISSION" définit les droits du "Subject" chez le "Issuer", elle renseigne principalement le profil du propriétaire (ex : étudiant, médecin, etc.) mais aussi les ressources locales auxquelles ce dernier a droit.

Le but du certificat X316 est d'offrir à l'utilisateur une sorte de carte de visite, sur laquelle figure tous ses droits. Ceci permet au propriétaire d'un certificat X316 de faire valoir ces attributs au sein d'autorités de confiance. Il sera alors autonome et pourra négocier sa politique de sécurité sans faire appel aux signataires de ses certificats.

Afin d'intégrer l'existant, la partie "Permission" du certificat X316 ne décrit pas une nouvelle syntaxe pour les autorisations mais permet de les représentées dans une syntaxe XML standardisée en spécifiant l'URI définissant le standard.

Comme illustré dans la figure 7.2, la partie PERMISSION est composée de trois items :

- PolicyType : définit le type de la politique locale, ex : RBAC, MAC, DAC, etc.
- SubjectProfiles : définit le (les) profil(s) du Subject, ex : étudiant, médecin, etc.
- SubjectAttributes : comme illustré dans la figure 7.2, cette séquence peut contenir plusieurs attributs correspondant aux différents droits accordés par le signataire au Subject.

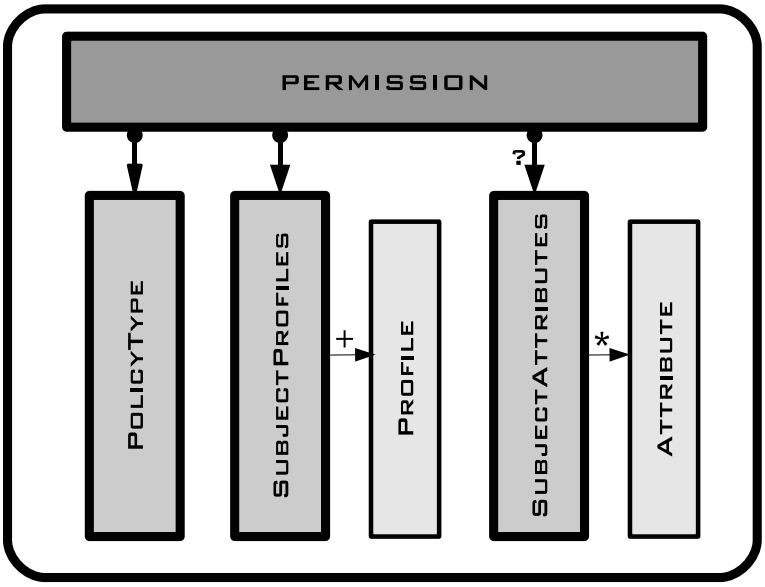

Figure 7.2 – The X316 Right

Syntaxe de la balise PERMISSION :

```
00  <PERMISSION>
01    <PolicyType>
02    <SubjectProfiles>
03      (<Profile Type=>
04          <Description>?
05          <ProfileID Type=>+
06      </Profile>)+
07    </SubjectProfiles>
08    (<SubjectAttributes Format=?>
09      <Attribute Format=>
10    </SubjectAttributes>)?
11  </PERMISSION>
```

Comme illustré dans la séquence XML ci-dessus, chaque *Subject* peut avoir plusieurs profiles. Chaque profil à un type (ligne 03), qui permet de savoir, par exemple, s'il est *Original* (attribué par le site d'appartenance) ou bien *Mapped* (attribué par un site de confiance grâce à la politique de mise en correspondance). Chaque profil peut être décrit (ligne 04) et peut être défini à son tour par plusieurs identifiants (ligne 05).

La partie *SubjectAttributes* est optionnelle. Elle contient l'ensemble des attributs qui définissent quelles sont les ressources sur lesquelles le *Subject* peut exercer un certain nombres de droits.

A la ligne 08, figure l'option *Format* ; celle-ci permet de renseigner le formalisme par défaut dans lequel les attributs sont décrits (ex : XACML). On peut également définir un format spécifique pour chaque attribut (ligne 09) ; dans le cas ou celui-ci n'est pas défini, c'est le format par défaut qui est pris en compte.

Exemple :

```
00  <PERMISSION>
01    <PolicyType> RBAC </PolicyType>
02    < SubjectProfiles>
03      <Profile Type="local">
04        <ProfileID Type="role"> Professeur </ProfileID>
05      </Profile>
06    </SubjectProfiles>
07    <SubjectAttributes>
08      <Attribute Format="urn:oasis:names:tc:xacml:1.0">
09        <Rule xmlns="urn:oasis:names:tc:xacml:1.0:policy"'
10                    ...RuleId="" Effect="Permit">
11           <Description>...</Description>
12           <Target>
13             <Subjects><Subject> Bob </Subject></Subjects>
14             <Resources><Resource>
15               <ResourceMatch MatchId="">
16                  <AttributeValue DataType="">
17                     PrinterID
18                  </AttributeValue>
19               </ResourceMatch>
20             </Resource></Resources>
21             <Actions><Action>
22               <ActionMatch MatchId="">
23                  <AttributeValue DataType="">
24                    print
25                  </AttributeValue>
26               </ActionMatch>
27             </Action></Actions>
28           </Target>
29         </Rule>
30      </Attribute>
31    </SubjectAttributes >
32  </PERMISSION>
```

Dans cette exemple, un attribut au format XACML est défini, donnant le droit à l'utilisateur Bob ayant le rôle $Professeur$ (ligne 04) d'effectuer l'action *print* (ligne 24) sur l'imprimante "PrinterID" (ligne 17).

7.2.3 IDENTIFICATION :

Comme son nom l'indique, cette partie permet d'identifier le propriétaire du certificat. En effet, la méthode classique de certification identifie le propriétaire d'un certificat en faisant appel à de l'authentification cryptographique se basant sur un challenge/réponse. D'un autre coté, de

nouvelles techniques émergent, basées principalement sur la biométrie. Ces dernières sont de plus en plus utilisées dans les aéroports ou sur les passeports en dépit de certains aspects négatifs dûs, par exemple, à l'impossibilité de révoquer une propriété physiques.

De nouvelles méthodes d'authentification utilisent plusieurs identifiants comme par exemple, une empreinte digitale plus une empreinte vocale. L'utilisation de plusieurs identifiants permet en effet d'offrir un moyen d'authentification plus fiable. Ce cas de figure a fait aussi ses preuves au niveau de l'utilisation des cartes bancaires qui nécessitent une double identification, le clé qui se trouve à l'intérieur de la carte combinée au code PIN. En outre, les dispositifs employés par les usagers sont de plus en plus nombreux et de capacité variable. Les certificats doivent être embarqués sur ces différents terminaux, et être adaptés à la capacité du chaque terminal. Pour cette raison, nous défendons l'idée que disposer de plusieurs identifiants dans un seul certificat permettraient à son propriétaire de s'authentifier en fonction de ce que peut lui offrir le terminal avec lequel la transaction est effectuée.

Selon le type d'identifiant, deux manières d'authentification sont possibles. Une authentification distante ou une authentification directe.

Authentification distante : Cette authentification peut être utilisée dans n'importe quelle condition en mode filaire ou non filaire (ex : wifi). Le protocole le plus connu est le challenge/réponse. Il utilise la cryptographie asymétrique et considère la clé publique comme étant le paramètre d'identification. Ce dernier assure une sécurité de bout en bout quel que soit le médium de communication. Le problème avec ce type d'authentification est la capacité de calcul des différents dispositifs. Pour cela, le certificat peut embarquer plusieurs clés de tailles différentes, par exemple : une clé RSA de 512 bits qui sera utilisée par les terminaux à faible capacité et une autre clé RSA de 2048 bits qui sera sélectionnée par ceux ayant une puissance suffisante.

Authentification directe : Cette authentification est spécifique à un certain type d'équipements. Elle est souvent utilisé par des support démunis de puissance de calcul, comme par exemple : une carte bancaire ou une clé usb. En effet, ces derniers peuvent stocker des données ou des certificats mais ne peuvent faire des calculs. Ainsi, l'authentification ne peut se faire sans la présence de terminaux spécifiques auxquels ces supports se connectent.

L'authentification directe peut également manipuler des identifiants biométriques. En effet, de plus en plus de récepteurs biométriques équipent les ordinateurs portables ou les assistants personnels (PDA). Ces nouveaux dispositifs peuvent être exploités comme étant des ATM. Prenons l'exemple d'un agent de sécurité. Celui-ci peut utiliser un PDA équipé d'un lecteur d'empreinte digitale pour contrôler l'accès à un bâtiment. Si une personne veut accéder à ce bâtiment, elle se présente auprès de l'agent de sécurité, lui envoie le certificat qui peut être stocké sur n'importe quel support (son téléphone portable ou sa clé usb) ; et ensuite s'authentifie en passant le pouce sur le lecteur du PDA de l'agent.

La majorité des identifiants biométriques (empreinte digitale, géométrie de la main, empreinte rétinienne ou ADN) sont formalisés en XML selon la norme d'OASIS XCBF (OASIS XML Common Biometric Format). En utilisant ce formalisme, le certificat X316 peut embarquer la plupart des identifiants appartenant à cette catégorie.

Syntaxe de la balise IDENTIFICATION

```
00  <IDENTIFICATION>
01    (<Identifier Type>
02      <Parameters>
03        (<Parameter>
04           <Description>
05           <ParameterValue>
06         </Parameter>)+
07      </Parameters>
08   </Identifier>)+
09 </IDENTIFICATION>
```

Dans la syntaxe définie ci-dessus, la partie identification peut contenir, non pas un identifiant mais plusieurs (lignes 01-08). Chacun est défini par son type (ligne 01) (par exemple : RSA 512) et comporte un certain nombre de paramètres(lignes 02-07). En prenant l'exemple d'une infrastructure à clé publique, les paramètres sont ceux de la clé publique. Chaque paramètre est décrit dans la ligne 04, et renseigné par sa valeur à la ligne 05.

Exemple :

```
00 <IDENTIFICATION>
01  <Identifier Type="http://liris.cnrs.fr/~rsaadi/X316MCert/Identity#PublicKey">
02   <Parameters>
03    <Parameter>
04     <Description> 512 RSA PubKey </Description>
05     <ParameterValue>
06         <ds:RSAKeyValue>
07           <ds:Modulus>
08              vmJ2k9b3JrDhG5LQ8uFFg9h1N0=
09           </ds:Modulus>
10           <ds:Exponent>AQAB</ds:Exponent>
11         </ds:RSAKeyValue>
12     </ParameterValue>
13    </Parameter>
14   </Parameters>
15  </Identifier>
16  <Identifier Type=".../X316MCert/Identity#Biometric">
17   <Parameters>
18    <Parameter>
19     <Description> Fingerprint </Description>
20     <ParameterValue xmlns="http://schemas.xmlsoap.org/ws/2002/04/secext">
21         <biometricObject>
22           <biometricHeader>
23             <version> 1 </version>
24             <recordType> <id> 5 </id> </recordType>
25             <dataType> <processed/> </dataType>
26             <purpose> <verify/> </purpose>
27             <quality> 93 </quality>
28             <format>
29               <formatOwner> <id> 123 </id> </formatOwner>
30             </format>
31           </biometricHeader>
32           <biometricData>
33             98F71....84723472E472=
34           </biometricData>
35         </biometricObject>
36     </ParameterValue>
37    </Parameter>
38   </Parameters>
15  </Identifier>
16 </IDENTIFICATION>
```

Dans cet exemple, le propriétaire du certificat est identifié avec une clé publique RSA de 512 bits (lignes (06-11)) tel que défini dans le standard XMLDSig et avec une empreinte digitale (lignes (21-35)) définie selon le standard XCBF.

7.2.4 SIGNATURE :

Etant donné que le certificat X316 est formalisé en XML, il est tout fait logique d'utiliser le standard XMLDSig afin de calculer et de représenter la signature. Cependant, un certificat X316 contient beaucoup plus d'information qu'un certificat standard. L'augmentation et la diversité de ces attributs nous a motivé pour définir un nouveau mode de signature.capabale de s'adapter au contexte de l'utilisateur (ex : l'interlocuteur, le type de la transaction, les moyens de communication -médium et dispositifs-, etc.).

Dans le cadre du certificat X316, nous considérons les attributs contenus dans les balises : *Subject* du $HEADER$, $PERMISSION$ et $IDENTIFICATION$ comme étant les parties qui peuvent être masquées par le propriétaire du certificat.

Dans le chapitre suivant, nous allons introduire cette nouvelle signature. Elle a été parfaitement intégrée dans le standard XMLDSig via la définition d'une nouvelle méthode de transformation.

7.3 Conclusion

Dans ce chapitre, nous avons défini une nouvelle structure de certificat nommé X316. Celle-ci permet de définir aussi bien un certificat d'identité grâce à la partie HEADER qu'un certificat d'attributs en les embarquants dans la partie PERMISSION. De plus, la troisième partie du certificat ($IDENTIFICATION$) permet de définir non pas un seul mais plusieurs identifiants.

La décomposition de la structure du certificat X316 en trois catégories permet d'indexer le certificat et de retrouver plus facilement les attributs nécessaires au propriétaire pour effectuer ses transactions (ex : identification, autorisation etc.).

De plus, avec l'intégration d'un nouveau mode de signature (Chapitre 8 et 9), X316 devient le premier certificat morph compatible avec les exigences des environnements pervasifs .

Enfin, dans le chapitre 10, nous allons montrer la flexibilité du certificat X316 en le déployant dans l'architecture Chameleon sous différentes déclinaisons : certificats d'attributs, certificats d'exploration et certificat de révocation.

Chapitre 8

The Fenrir Morph Signature

Sommaire

8.1 Introduction

La méthode classiquement utilisée pour signer numériquement un document, assure l'intégrité de celui-ci, car toute modification de ce dernier entraîne le changement de la valeur du résidu et ainsi l'invalidation de la signature. Par conséquent, une fois le document signé, on ne peut masquer aucune de ses sous-parties sans en invalider la signature. Le standard actuel de signatures numériques ne répond donc pas à nos besoins, puisque le propriétaire d'un fichier signé, ne peut disposer des informations contenues dans ce dernier de manière sélective.

Dans ce chapitre, nous allons traiter ce problème en définissant une méthode de signature offrant la possibilité à l'utilisateur de distribuer ses fichiers signés en cachant certaines informations sans être obligé de régénérer une signature à chaque fois que certaines informations sont

sélectionnées. Cette méthode de signature a été baptisée : Signature **FeMoS** comme abréviation de "Fenrir Morph Signature" [8].

8.2 Principe de la signature FeMoS

Pour illustrer ce principe, utilisons les dénominations : Signataire, Propriétaire et Contrôleur, telles qu'elles ont été définies dans le chapitre 6.

Le principe de base de la signature FeMoS consiste à demander au signataire de différencier, dans le fichier à signer, les parties dynamiques des parties statiques.

- Les parties dynamiques représentent les données que le Propriétaire peut, selon le contexte, divulguer ou garder secrètes, par exemple le nom, le prénom, la date de naissance, etc.
- Les parties statiques représentent les parties qui apparaîtront obligatoirement dans toutes les versions du fichier. Dans le cas des certificats, le numéro de série, l'identité du signataire et la date de validité sont toutes des données statiques (ex : la balise *Cert_Identity* du certificat X316 -voir chapitre 7-).

Dans la signature FeMoS, nous n'identifierons que les parties dynamiques étant donnée que toute partie qui ne l'est pas est forcément statique. Les parties dynamiques seront alors indéxées dans une structure externe au document à signé qui portera le nom de "Morph Template"

Ainsi, dans le scénario Signataire, Propriétaire et Contrôleur, une signature FeMoS d'un fichier XML est signée, adaptée et vérifiée en suivant les étapes suivantes :

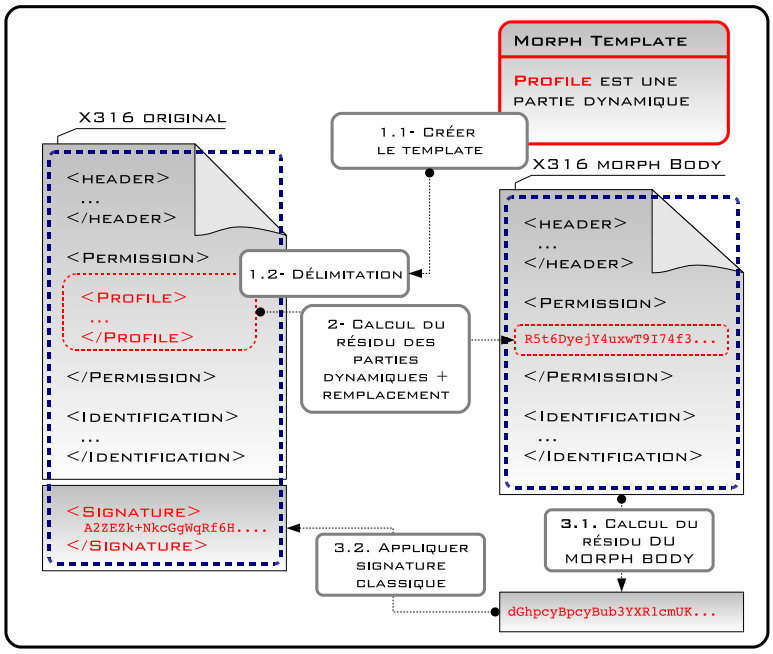

Figure 8.1 – Génération de la signature FeMoS par le Signataire.

8. Dans la mythologie nordique, le nom **Fenrir** représente un loup gigantesque. Ce nom a été choisi afin de mettre en avant une signature intelligente permettant au fichier signé de changer de forme (Morph).

Signataire :

1. Définir le Moph Template qui permet de délimiter les parties dynamiques du fichier.

2. Calculer le résidu de chaque partie dynamique et la remplacer par son résidu dans le fichier ; cette étape génère une forme intermédiaire du fichier appelé "MorphBody".

3. Calculer la signature de manière classique sur le MorphBody.

Enfin, le Signataire envoie le fichier original, désormais signé, accompagné du Morph Template ainsi que la signature du MorphBody au Propriétaire.

Propriétaire : Si le Propriétaire veut fournir le fichier original(voir figure 8.2) en masquant une partie particulière (ex : la partie $Profile$), il doit :

1. Créer le fichier Morph en amputant la partie dynamique (Profile).

2. Calculer le hash de la partie amputée et mettre le résultat ainsi que sa position dans le MorphBody (afin qu'on puisse reconstruire le MorphBody ultérieurement) dans une structure externe définit sous le nom de "Partie Flottante".

3. Reprendre la même signature du fichier original.

Enfin, fournir au Contrôleur le Morph fichier, le Morph Template, la partie flottante contenant les hashs de toutes les parties dynamiques masquées ainsi que la signature originale.

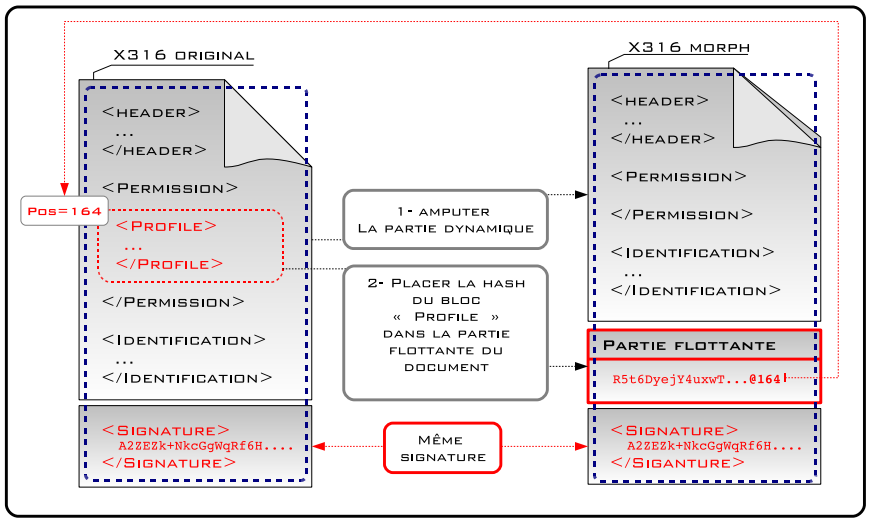

FIGURE 8.2 – Approche de signature Morph directe.

Contrôleur : Quand le Contrôleur reçoit le fichier Morph, il doit reconstruire le MorphBody qui constitue le point commun de toute version Morph issue du fichier original (voir figure 8.3).

Le Contrôleur procède alors comme suit :

1. Charger le Morph Template ainsi que la partie flottante.

2. Recréer le MorphBody en remplaçant les parties dynamiques par leur valeur hash et en replaçant les parties masquées (définies dans la Partie Flottantes) à leur position respective.

3. Une fois le MorphBody reconstitué, la vérification de la signature peut se faire de manière classique.

Dans la suite de chapitre nous allons définir les éléments nécessaires au fonctionnement de la siganture FeMoS, à savoir le Morph Template (section 8.3), la partie flottante (-FloatPart- section 8.4), et enfin dans la section 8.5 nous montrons la faisabilité de la signature en l'intégrant dans le standard XMLDSig sous forme d'une transformation.

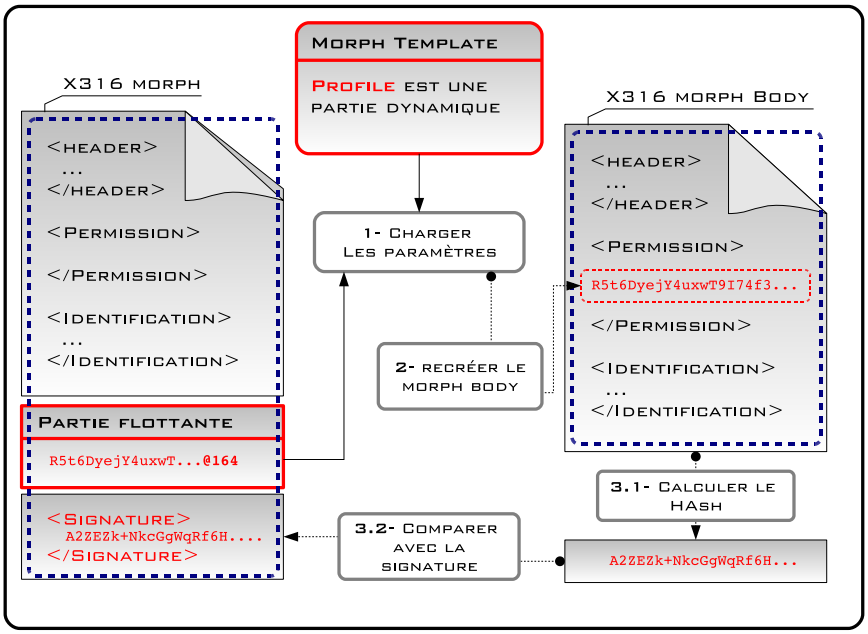

FIGURE 8.3 – Approche de signature Morph directe.

8.3 Le Morph Template

Le Morph Template constitue le cœur de la siganture FeMoS. Dans cette section nous allons détailler sa structure ainsi que ses différentes fonctionnalité qui permettent de répondre aux deux premières contraintes (voir chapitre 6 section 6.3.3) qui nous avons fixée dans l'état de l'art :

– Contrainte de dissociation en délimitants les parties dynamiques (section 8.3.3).

– Contrainte d'intégrité en définissant des données de remplacement (section 8.3.4) et des politiques de divulgation (section 8.3.5).

8.3.1 Spécification du Morph Template

Le "Morph template", ou M-Template, permet de décrire, dans une syntaxe spécifique, les différentes parties dynamiques d'un fichier. L'application d'un M-Template permet de séparer la définition des parties dynamique de la structure du fichier, en évitant ainsi de polluer le fichier avec des données superflues.

De manière plus générale, comme le montre les exemples suivants, chaque sous-partie d'un fichier peut être identifiée et délimitée en fonction de son format.

- Fichier XML : la syntaxe <item> partie dynamique </item> permet de référencer une partie d'un document XML.

- Fichier Latex : la syntaxe \begin{item} partie dynamique \end{item} permet de délimiter une sous-partie d'un document latex.

- Fichier BMP : les formes de type rectangle ou cercle permettent de représenter une partie de l'image (voir figure 8.4).

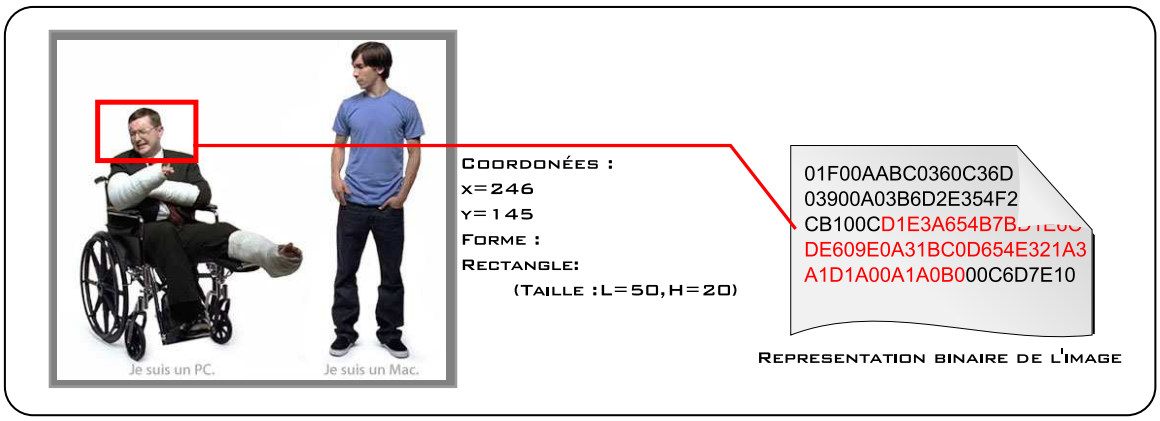

Figure 8.4 – Proposition de mécanisme de sélection pour les formats images

Avec la diversité des formats de fichiers existants, il est nécessaire d'opter pour une méthode qui permet de délimiter de manière générique (indexer) des parties dynamiques de n'importe quel type de fichier sans modifier son contenu et sans que cela implique l'implémentation de plusieurs algorithmes en fonction du type de fichiers à signer.

Nous considérons alors que tout fichier, indépendamment de son format, se présente comme un flux d'octets. Il est donc suffisant, pour définir une partie dynamique de fournir sa position de début dans le fichier ainsi que sa taille (ou la position de fin).

L'algorithme de signature FeMoS utilise un M-Template binaire noté BM-Template (Binary Morph Template), qui est indépendant de tout format de fichier étant donné qu'il indexe au niveau octet.

Néanmoins, comme le montre la figure 8.5, des M-Templates logiques peuvent être définis tels que : XML M-Template, Tex M-Template, BMP M-Template, etc. Ainsi, avant de pouvoir exécuter la signature FeMoS il est nécessaire de repasser par un M-Template binaire en le générant à partir du fichier à signer et de son M-Template logique.

Exemple : Dans un XML M-Template, on peut définir la balise "Profile" comme délimiteur d'un contenu dynamique. La traduction de cette syntaxe dans un BM-Template est définie en fonction de la position en octets de la balise <Profile> dans le document, et de la taille de la partie dynamique comme étant la différence en octets entre la balise ouvrante et la balise fermante </profile>. En prenant l'exemple de la figure 8.4, nous pouvons aussi définir un mécanisme de translation similaire au précédent en traduisant les coordonnées logiques des pixels délimitant la forme de la sélection (ex. rectangle) en positions absolues dans le fichier image.

Le rôle de chaque parseur d'un M-Template logique est de pouvoir sélectionner des parties dynamiques en utilisant des informations logiques liées au type de fichiers parsés et de retourner un BM-Template en associant à chaque partie logique sa position absolue dans le fichier.

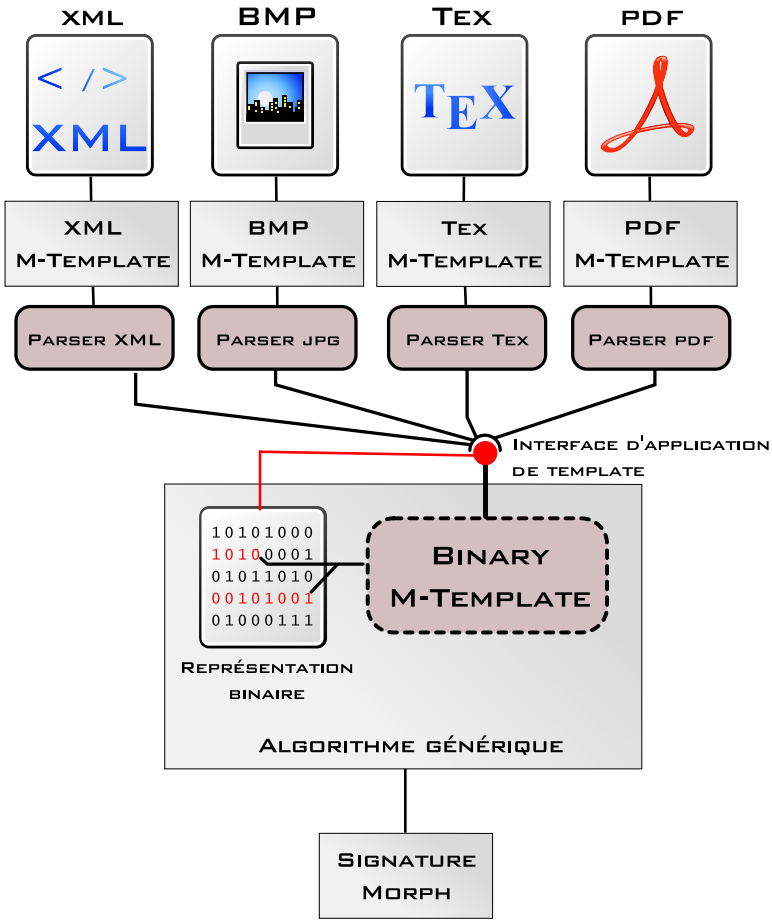

FIGURE 8.5 – Approche de signature Morph utilisant l'abstraction de formats.

Les mécanismes de sélection sont indépendants de l'algorithme de signature. Ceci permet de définir, pour chaque format de fichier, les mécanismes les plus pertinents tout en gardant l'algorithme aussi générique que possible.

8.3.2 Format du M-Template

L'objectif de cette spécification est de proposer un format extensible pour le M-Template. Le format retenu est le langage XML, et cela principalement pour des raisons de portabilité et de compatibilité avec le format de signature standard XMLDSig.

La syntaxe générale du format est la suivante :

```
<MorphTemplate templateURI=? serviceURI=? targetMimeType=>
    <Parameters>
    <DynamicParts>
    <DisclosePolicies>
</MorphTemplate>
```

Un Template se compose donc de trois parties distinctes. La première, délimitée par la balise `Parameters`, spécifie les différents paramètres passés à l'algorithme. La seconde (DynamicParts) permet de définir les différentes parties dynamiques et la troisième (Disclosepolicies) les politiques de divulgation.

L'élément principal `MorphTemplate` accepte trois attributs. L'attribut *templateURI* permet de définir si le "M-Template" est implicite ou explicite. Dans le cas d'un MorphTemplate implicite, l'attribut templateURI est renseigné par une URI[9]. Cette derrière fait référence à un M-Template générique utilisé pour une certaine classe de documents. Par contre dans le cas d'un Template explicite, le M-Template est spécifique au fichier à signer et est défini localement dans la signature.

L'attribut *targetMimeType* permet de définir le format de document ciblé. Afin de garantir une cohérence et d'éviter toute ambigüité par rapport aux différents noms de format existants, nous avons opté pour le standard MIME et plus particulièrement le MIME Media [Free 96a, Free 96b].

Cette première balise est commune à tout les M-Templates (binaires ou logiques). Elle permet à la signature FeMoS de savoir si le template est binaire, et donc le lire et le traiter, sinon solliciter l'API ou le service défini par l'attribut *serviceURI*.

L'élément `Parameters` regroupe les paramètres nécessaires à l'algorithme :

```
<Parameters>
    <UsedHashAlgorithm algorithmName/>
    <Parameter name value>*
</Parameters>
```

L'élément `UsedHashAlgorithm` permet de définir l'algorithme utilisé pour le calcul du résidu des parties dynamiques. Pour des raisons de consistance et d'homogénéisation, on utilise les mêmes URIs que ceux définis par la spécification W3C de XMLDSig (ex : algorithmName="http ://www.w3.org/2000/09/xmldsig#sha1").

Il est aussi possible de définir d'autres paramètres selon le type de document ciblé en utilisant l'élément `Parameter`. Il suffit pour cela de spécifier le nom d'un paramétre pour l'attribut *name* et la valeur correspondante dans *value*. Cet élément permet de garder la syntaxe du M-Template extensible et indépendante des différents formats de document.

9. Universal Ressource Identifier, défini dans la RFC3986 de l'IETF [Bern 05]

8.3.3 Délimitation des parties dynamiques

Après avoir défini les paramètres globaux de l'algorithme par l'élément `Parameters`, on déclare l'ensemble des parties dynamiques. Cette étape consiste à définir un élément `DynamicPart` pour chaque partie dynamique du document en utilisant la syntaxe suivante :

```
<DynamicParts>
  (<DynamicPart ID>
    <Label>?
    <Command commandName>
        <Parameter name value>+
        <Command commandName>*
    </Command>
    <ReplacementData>
  </DynamicPart>
</DynamicParts>)+
```

Chaque partie dynamique est identifiée par un identificateur unique spécifié par l'attribut *ID*. Cet identificateur doit être unique dans le M-Template où il apparaît, et servira à faire référence de manière non ambigüe à la partie dynamique dans le reste du template.

Pour permettre de décrire le contenu de la partie dynamique, l'élément facultatif `Label` peut être utilisé.

La sélection des parties dynamiques se fait à l'aide de la balise "Command". Ces commandes utilisent les mécanismes de sélection introduits précédemment et offrent une représentation très flexible.

Les commandes peuvent être simples ou imbriquées La commande principale est appelée commande mère et les commandes imbriquées sont dites commandes filles.

Les commandes peuvent accepter des paramètres. Pour les définir, on utilise des éléments `Parameter`. Ces derniers acceptent comme attribut le nom du paramètre (*name*) et sa valeur (*value*).

Dans le cadre de notre BM-Template nous avons défini deux commandes : "FrameSelect" et "Select". La première délimite les parties dynamiques en utilisant les paramètres "begin" et "end", et la seconde à l'aide des paramètres "begin" et "length".

L'exemple suivant illustre l'utilisation de ces commandes de manière imbriquée.

```
[01]<Command commandName="FrameSelect">
[02]    <Parameter name="begin" value="100"/>
[03]    <Parameter name="end" value="400"/>
[04]    <Command commandName="Select">
[05]        <Parameter name="begin" value="30"/>
[06]        <Parameter name="length" value="165"/>
[07]    </Command>
[08]</Command>
```

Dans cet exemple, on distingue deux commandes dont l'une est imbriquée dans l'autre. La première est la commande mère et est de type "FrameSelect" (ligne 01). Elle permet de sélectionner, dans un format brut, la zone contenant les parties dynamiques en spécifiant deux paramètres : les positions absolues de début (ligne 02) et de fin (ligne 03).

La commande imbriquée (fille) dans le "FrameSelect" est la commande "Select" (ligne 04). Son évaluation est relative à la commande mère. Dans ce cas, le "Select" s'évalue dans l'intervalle défini par le "FrameSelect", à savoir de 100 à 400. Ainsi, après interprétation de ces deux commandes, la partie dynamique résultante est délimitée par la position de départ=30+100=130 et de fin= 130+165=295.

8.3.4 Définition des données de remplacement

D'une part, les résidus sont généralement des groupes d'octets ne pouvant pas être intégrés directement dans un fichier sans en altérer sa présentation. Ainsi, si l'on se contente de remplacer la partie dynamique dans un fichier texte ou XML par son résidu, le fichier résultant pourrait contiendra un ensemble d'artefacts illisibles. Or, ce résultat n'est pas souhaité. D'autre part, le fait d'extraire une partie dynamique d'un document peut le rendre non valide et illisible.

Il est donc nécessaire de permettre le remplacement des parties dynamiques. Le principe, comme l'illustre la figure 8.6 (a), est de définir un bloc de données qui remplacera la partie dynamique si elle venait à être cachée. Pour pouvoir embarquer la valeur de remplacement dans la signature, il peut s'avérer nécessaire d'encoder les données de remplacement afin de garantir l'intégrité de la structure XML (Encodage en Base64, Base 16, ou Base2) ou bien de fournir une certaine lisibilité (encodage ASCII, UTF-8, etc.).

```
<ReplacementData sourceEncoding=>
       <CompletionValue>?
       <DataValue>
</ReplacementData>
```

L'attribut *sourceEncoding* de l'élément `ReplacementData` a été prévu à cet effet. Ainsi, les données représentées par les balises CompletionValue et DataValue sont traduites par la méthode d'encodage spécifiée par cet attribut. De cette manière, avant d'appliquer un quelconque remplacement, cette valeur est traduite en binaire en fonction de l'encodage utilisé.

L'élément `DataValue` contient la valeur de remplacement. Cette dernière, peut dans certains cas, avoir besoin d'être complétée afin de remplacer la partie dynamique correspondante par une suite de bits ayant une taille identique. La complétion est représentée dans le `DataValue` par la balise de complétion $< CD/ >$. Ainsi, dans la position où se trouve le caractère de complétion, la valeur contenu dans la balise `DataValue` est complétée par le contenu de la balise `CompletionValue` jusqu'à atteindre la taille de la partie dynamique.

Dans la figure 8.6(a), le ReplacementData du document texte s'écrit comme suit :

```
<ReplacementData sourceEncoding="UTF-8">
       <DataValue>Texte masqué</DataValue>
</ReplacementData>
```

Dans la figure 8.6(b), le ReplacementData de l'image s'écrit comme suit :

```
<ReplacementData sourceEncoding="Base2">
 <CompletionValue>0</CompletionValue>
```

```
      <DataValue><CD/></DataValue>
</ReplacementData>
```

8.3.5 Définition des politiques de divulgation

La signature FeMoS offre la possibilité de cacher certaines parties du document tout en le gardant valide. Cependant, il peut parfois être nécessaire d'introduire des contraintes d'utilisation. On peut, par exemple, forcer l'utilisateur à divulguer au moins une information parmi un groupe de parties dynamiques. On peut également le forcer à cacher ou divulguer un groupe de parties dynamiques en même temps. Ces contraintes peuvent être spécifiées dans le M-Template au moyen des politiques de divulgation (disclose policies).

Une partie dynamique peut avoir deux états de visibilité : divulgué ou visible (disclosed) ; ou bien caché (hidden). Les politiques de divulgation permettent de grouper un certain nombre de parties dynamiques et de lier leurs états de visibilité par un opérateur logique. Nous en avons défini deux :

- La politique "At least one" : Dans le groupe de parties dynamiques liées par cette politique, on considère que celle-ci est respectée si au moins une des parties dynamiques est divulguée. Cette politique correspond à un opérateur logique "OU".

- la politique "Same disclosure State" : Cette règle est considérée comme respectée, si l'ensemble du groupe, des parties dynamiques référencées est présent ou absent de manière non sélective. Cette politique correspond à l'opérateur logique "XNor".

La syntaxe de définition des politiques de divulgation est la suivante :

```
<DisclosePolicies>
  (<DisclosePolicy policyType>
    <PolicyInfo>?
    <PolicyParameter name value>*
    <BoundDynamicPart dynamicPartIdentificator>+
  <DisclosePolicy>)*
<DisclosePolicies>
```

Le type de la politique est définit par l'attribut *policyType*. Les deux types définis pour l'instant sont "AtLeastOne" et "SameDisclosureState".

Il est possible d'inclure des informations sur la politique en utilisant l'élément facultatif `PolicyInfo`. Ces informations seront principalement destinées au Propriétaire du document.

Dans un souci d'extensibilité, d'autres politiques de divulgation peuvent être définies par la suite ; ces politiques peuvent avoir besoin de paramètres définis à l'aide de l'élément `PolicyParameter`.

Les parties dynamiques liées à la politique sont définies en utilisant l'élément `BoundDynamicPart`, l'attribut *dynamicPartIdentificator* étant l'identificateur de la partie dynamique liée (voir section 8.3.3).

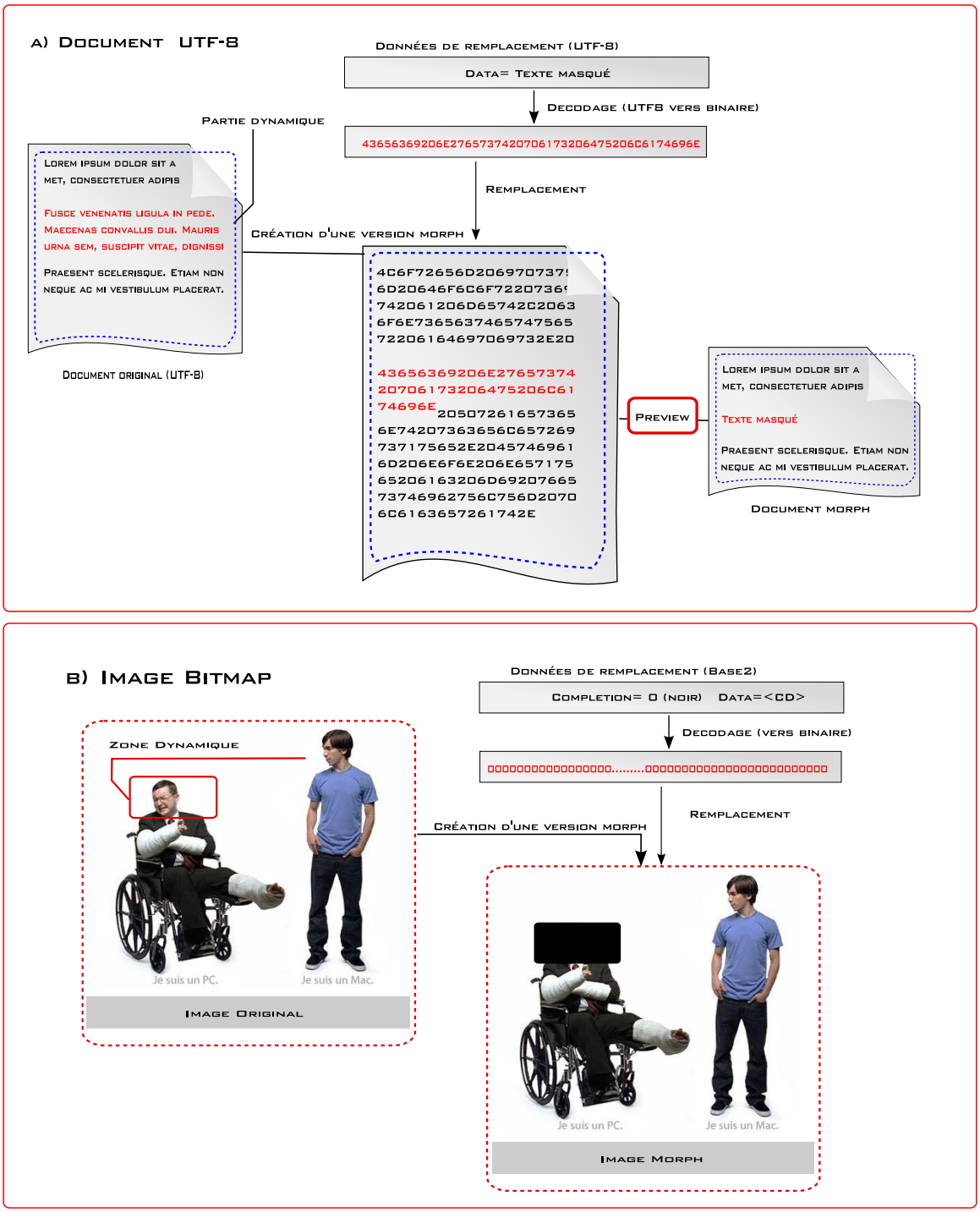

Figure 8.6 – Remplacement et complétion.

Exemple : Prenons l'exemple de l'image illustrée dans la figure 8.6. Sachant que dans un fichier BMP l'image est stockée ligne par ligne, la sélection d'un rectangle se fait alors en regroupant, grâce à la politique "SameDisclosureState" (lignes 27-33), tous les segments parallèles qui le dessine(lignes 04-24), comme suit :

```
00 <MorphTemplate  targetMimeType="image/bmp">
01     <Parameters>
02      <UsedHashAlgorithm algorithmName="http://www.w3.org/2000/09/xmldsig#sha1"/>
03     </Parameters>
04     <DynamicParts>
05       <DynamicPart id="ligne1">
06          <Command commandName="Select">
07            <Parameter name="begin" value="456"/>
08            <Parameter name="length" value="15"/>
09         </Command>
10         <ReplacementData sourceEncoding="Base2">
11           <CompletionValue>0</CompletionValue>
12           <DataValue><CD></DataValue>
13         </ReplacementData>
14       <DynamicPart>
15         .....
16       </DynamicPart>
17       <DynamicPart id="ligne16">
18          <Command commandName="Select">
19            <Parameter name="begin" value="720"/>
20            <Parameter name="length" value="15"/>
21          </Command>
22          <ReplacementData sourceEncoding="Base2">
23            <CompletionValue>0</CompletionValue>
24            <DataValue><CD></DataValue>
25          </ReplacementData>
26       </DynamicPart>
27     </DynamicParts>
28     <DisclosePolicies>
29        <DisclosePolicy policyType="SameDisclosureState">
30            <PolicyInfo> Rectangle <PolicyInfo>
31            <BoundDynamicPart dynamicPartIdentificator="ligne1"/>
32            <BoundDynamicPart dynamicPartIdentificator="ligne2"/>
33            ....
34            <BoundDynamicPart dynamicPartIdentificator="ligne16"/>
35        <DisclosePolicy>
36     <DisclosePolicies>
37 </MorphTemplate>
```

Dans cet exemple, chaque bloc *DynamicPart* (lignes 04-27) correspond à un ligne du rectangle de taille de 15 octets. Ces derniers sont cachés et remplacés par des 0 correspondant à du noir dans l'image. De plus, la partie *DisclosePolicies* (lignes 29-36) impose que soit toutes les lignes apparaissent ensemble, soit elles sont cachées ensemble.

8.4 La partie flottante

La parties flottante d'un fichier signé représente les parties dynamiques qui ont été masquées par le Propriétaire. Elles sont regroupées dans un élément `FloatParts` définit en utilisant la syntaxe suivante :

```
<FloatParts ID discloseLevel>
    (<FloatPart ID=? startPosition>..Valeur de hash..</FloatPart>)*
</FloatParts>
```

La partie flottante du document font référence aux parties dynamiques masquées grâce à l'attribut *ID*. Cet identificateur doit être le même que celui défini dans le M-Template. L'attribut *discloseLevel* définit le niveau de divulgation du document. Il peut prendre l'une des deux valeurs suivantes : full (complète) et partial (partielle). Un niveau de divulgation "full" signifie que le document n'a aucune partie masquée, tandis qu'un niveau "partial" signifie qu'au moins une partie dynamique est masquée.

L'attribut *ID* représente l'identificateur qui permet de référencer les parties flottantes. L'attribut *startPosition* permet d'identifier et de restituer la partie dynamique dans le document signé. La valeur du résidu de la partie dynamique est contenue dans l'élément `FloatPart` (voir l'exemple de la section 8.6 à la fin du chapitre).

8.5 Intégration dans XMLDSig

8.5.1 Fonctionnement de XMLDSig

Pour rappel (voir chapitre 6 section 6.3.2.4), une signature au format XMLDSig se présente sous la forme suivante :

```
01 <Signature>
02     <SignedInfo>
03     <CanonicalisationMethod>
04     <SignatureMethod>
05     (<Reference URI=?>
06         (<Transforms>?
07           <Transform>+
08         <Transforms>)?
09         <DigestMethod>
10         <DigestValue>
11     </Reference>)+
12     </SignedInfo>
13     <SignatureValue>
14     <KeyInfo>?
15     <Object>*
16 </Signature>
```

La génération de la signature se fait en deux étapes. Il faut d'abord générer les éléments `Reference` puis calculer la signature sur l'élément `SignedInfo`.

8.5.1.1 Génération des références

Pour chaque référence présente dans la signature, on doit appliquer les étapes suivantes :

1. Appliquer les transformations (lignes 06-08) sur l'objet référencé par l'URI (ligne 05).
2. Calculer la valeur de résidu en utilisant un algorithme de hash (lignes 09-10).
3. Créer un élément `Reference` regroupant l'URI de l'objet signé, les transformations subies par l'objet (`Transform`s), l'algorithme de hash utilisé (`DigestMethod`) et la valeur du résidu calculée (`DigestValue`).

C'est sur l'élément `Reference` résultant que se fera le calcul de la signature.

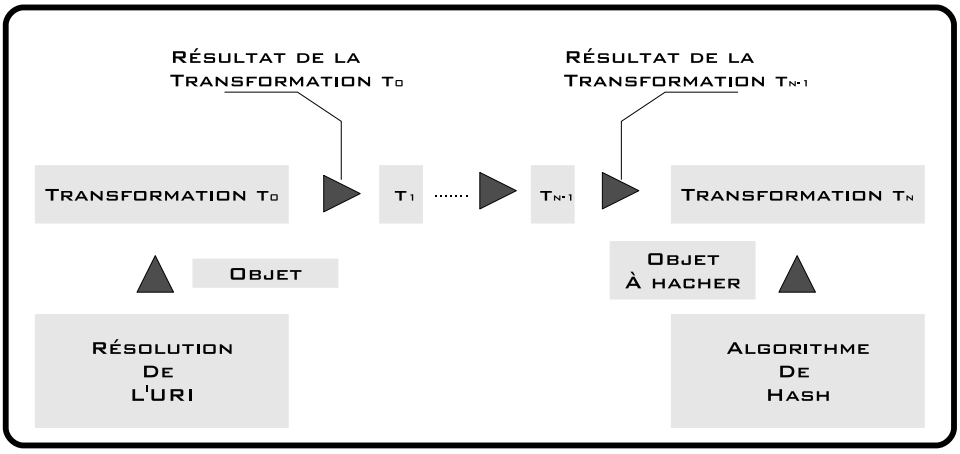

FIGURE 8.7 – Chaîne de tranformation dans XMLDSig

8.5.1.2 Génération de la signature

Le standard prévoit la génération de la signature en trois étapes :

1. Créer un élément `SignedInfo` incluant la méthode de signature (`SignatureMethod`) et l'algorithme de canonisation (`Canonicalization method`) [10] utilisé, ainsi que les références (`Reference`).
2. Canoniser l'élément `SignedInfo` puis calculer la signature en utilisant l'algorithme spécifié.
3. Générer l'élément `Signature` contenant les éléments `SignedInfo`, `Object`, `KeyInfo` et `SignatureValue`.

8.5.2 La transformation dans XMLDSig

Le standard XMLDsig permet de manipuler, par le biais de l'élément `Transforms`, l'objet avant de le signer. Ce dernier contient une liste ordonnée d'éléments `Transform`. Ceci permet de préparer l'objet avant de faire le calcul du résidu. Le résultat de chaque transformation sert d'entrée à la prochaine transformation et forme ainsi une chaîne comme l'illustre la figure 8.7. L'entrée de la première transformation est fournie par la résolution (déréférencement) de l'URI présent dans l'élément `Reference` et la sortie du dernier élément de la chaine représente l'objet sur lequel sera appliqué la méthode spécifiée dans `DigestMethod` [W3C 08].

Un certain nombre d'algorithmes de transformation ont été définis et doivent obligatoirement être implémentés par les applications voulant se conformer aux standards. Le *filtrage XPath* ou bien la transformation *base64* sont des exemples. La syntaxe de l'élément `Transform` permet d'inclure des éléments d'un autre espace de noms et ainsi de pouvoir définir une syntaxe propre pour les paramètres destinés à l'algorithme.

8.5.3 L'algorithm MorphBodyTransform

Afin d'intégrer la signature FeMoS dans XMLDSig, nous définissons la transformation MorphBodyTransform qui permet de transformer le fichier en entrée en Morphbody. Ainsi, la modélisation de l'algorithme comme algorithme de transformation offre plusieurs avantages. L'in-

10. Met en forme le document XML, par exemple : normaliser le retour chariot par $\#xA$...

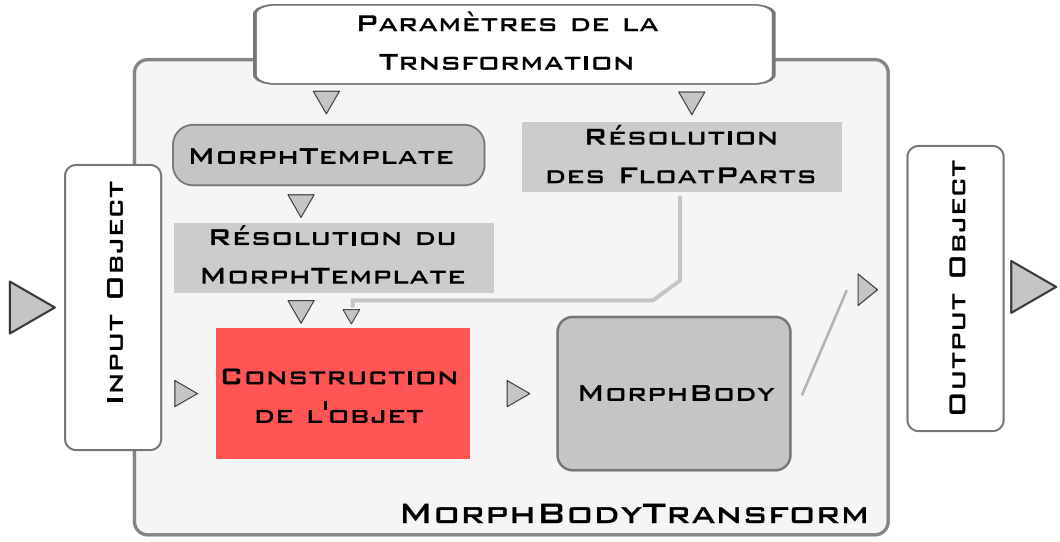

Figure 8.8 – modèle de traitement de la transformation MorphBody

tégration au standard existant se fait de manière transparente et l'ajout de la transformation au reste de l'implémentation se fait simplement en suivant le modèle de transformation proposé, sans apporter de modification au fonctionnement interne de l'implémentation de XMLDSig.

Néanmoins l'utilisation de l'algorithme dans une chaîne de transformations pose certaines contraintes. L'algorithme MorphBody génère différentes versions d'un même fichier. Son utilisation doit donc prendre en compte que la chaîne de transformations doit produire un même objet final. Ainsi, si la position de la transformation dans la chaine est i, on doit s'assurer que l'objet intermédiaire produit par la transformation du résultat de T_{i-1} est constant pour toute version du document. Nous préconisons donc que notre transformation soit appliquée en premier afin que les autres transformations s'appliquent sur le même objet quelque soit la version moph du fichier à contrôler.

Nous définissons un modèle de traitement par référence qui implémente l'algorithme en restant dans l'optique d'intégration au standard XMLDSig. Ce modèle est un ensemble de méthodes qui manipulent l'objet entrant durant la création et la validation de la transformation comme résumé par la figure 8.8 et explicité dans la section suivante.

8.5.3.1 Objets manipulés durant la transformation

La transformation MorphBody manipule un certain nombre d'objets, nous les définissons et décrivons leurs rôles comme suit :

Les objets d'entrée/sortie : Ils représentent respectivement l'objet issu de la transformation précédente et l'objet généré par notre transformation. Le standard XMLDSig définit deux types possibles pour les objets issus des transformations, une liste de nœuds XML (node set) ou un flux d'octets (octet stream). Notre transformation accepte, comme entrée, l'un ou l'autre des deux types, et génère toujours un flux d'octets en sortie.

Les paramètres de l'algorithme : Cet objet contient les paramètres nécessaires à l'applica-

tion de l'algorithme. Ces paramètres sont passés à la transformation sous forme d'un élément XML. Ainsi, l'algorithme de transformation extrait deux informations principales : l'emplacement des parties flottantes et le MorphTemplate.

8.5.3.2 Algorithme de la transformation

L'algorithme MorphBodyTransform effectue un traitement spécifique sur les differents objets de la transformation. Comme décrit dans la figure 8.8, cet algorithme se déroule comme suit :

1. Charger l'objet d'entrée.
2. Résoudre les paramètres de la transformation (M-Template et parties flottantes).
3. Appliquer le M-Template à l'objet d'entrée en le combinant éventuellement avec les parties flottantes pour générer un MorphBody.

Ces différentes étapes restent les mêmes pour le processus de signature, de création de document Morph et de vérification de la signature. La figure 8.9 représente l'enchaînement des différentes étapes de l'algorithme, comme suit :

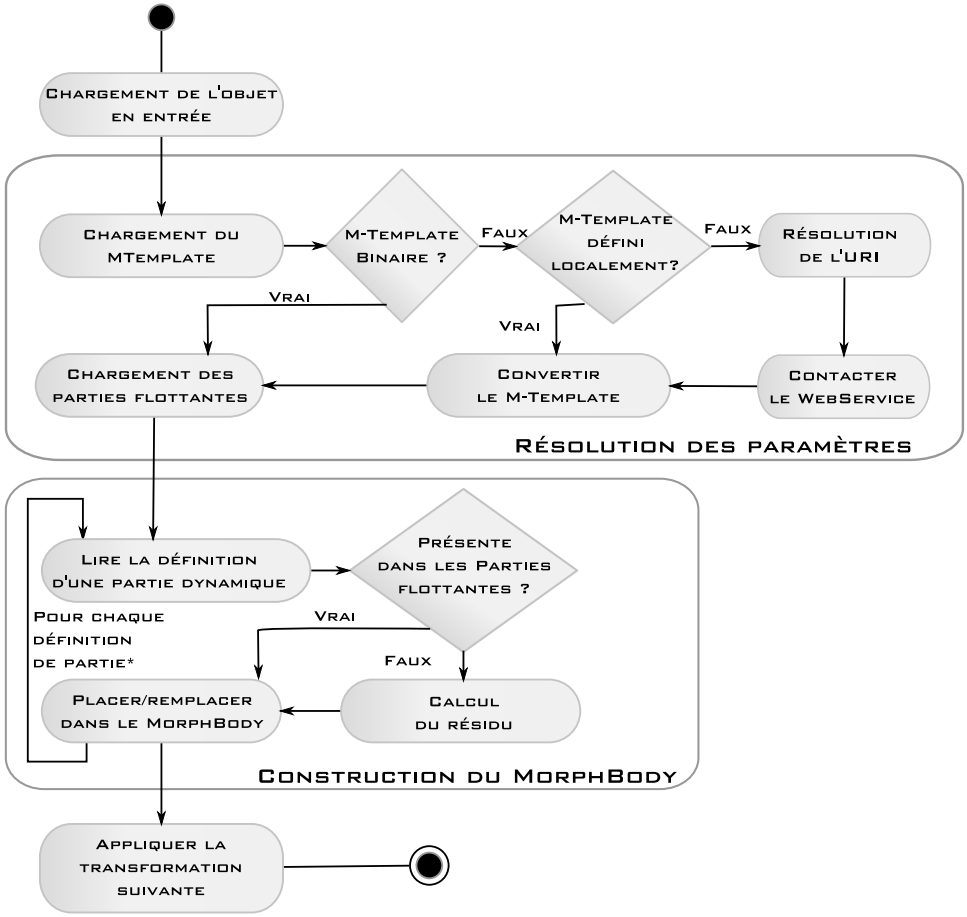

Figure 8.9 – Diagramme d'activité de l'algorithme de transformation.

1. **Chargement de l'objet d'entrée :** Le standard XMLDSig prévoit deux types d'objets : XML ou flux d'octets. Notre transformation peut accepter l'un ou l'autre. Le contenu de l'objet est traité selon le format du fichier cible définit dans le M-Template par l'attribut *targetMimeType*.

2. **Résolution des paramètres :** La résolution des paramètres se fait en lisant les informations de la balise `Transform` et se fait en deux étapes :

   2.a. Résolution du M-Template : Dans le cas d'un M-Template logique, il suffit de charger l'API qui permet de parser le document et de générer le BM-Template. Dans le cas où l'API n'existe pas en local, il suffit de résoudre l'URI et de contacter le WebService référencé afin de procéder à la translation du template logique.

   2.b. Résolution des parties flottantes : Les parties flottantes d'un document doivent être chargées avant d'appliquer le template et sont utilisées conjointement avec celui-ci pour générer le MorphBody. Comme l'algorithme de transformation doit s'appliquer à la fois dans le cas d'un document original ou Morph, il se peut que le document ne contienne aucune partie flottante.

3. **Génération du MorphBody :** Pour générer le MorphBody, on parcourt le M-Template séquentiellement. Pour chaque partie dynamique :
   - si celle-ci figure dans les parties flottantes, elle est placée dans le MorphBody à sa position initiale,
   - sinon elle est remplacée par son résidu.
4. **Signature :** Sérialiser l'objet représentant le MorphBody et l'envoyer à la transformation suivante.

8.5.4 Syntaxe de la Transformation "MorphBodyTransform"

```
<Transform Algorithm="http://fr.fms.wikia.com/wiki/MorphBodyTransform#">
    <FloatPartsDescriptor URI=/>
    <MorphTemplate>
</Transform>
```

Les règles de traitements des signatures XMLDsig déterminent le modèle de référence à utiliser pour la validation des signatures[W3C 08]. Le standard spécifie que la validation se fait sur l'élément `SignedInfo` contenant entre autre les éléments `Transform`. Chaque algorithme `Transform` est identifié par une URI et prend comme paramètres les données délimitées par les balises ouvrante et fermante de cette transformation.

Ainsi, l'algorithme de transformation *MorphBody* est identifié par l'URI :
"http ://fr.fms.wikia.com/wiki/MorphBodyTransform#".

Comme le montre la syntaxe de la transformation ci-dessus, l'algorithme "MorphBodyTransform" prend en entrée les deux paramètres suivant :

– L'élément FloatPartsDescriptor : Cet élément est dynamique par nature et est différent d'une version du fichier à signer à une autre. On ne peut donc pas l'inclure dans l'élément `Transform`, néanmoins nous pouvons introduire une référence statique à son emplacement par le biais d'une URI qui fait référence aux parties flottantes du document (défini précédemment dans l'élément `FloatParts`).

– L'élément `MorphTemplate` : Il est employé pour signer le document. Il faut noter que dans le cas d'un M-Template explicite, celui-ci est défini directement dans la signature et ne peut donc pas être modifié sans invalider cette dernière.

8.6 Exemple d'intégration

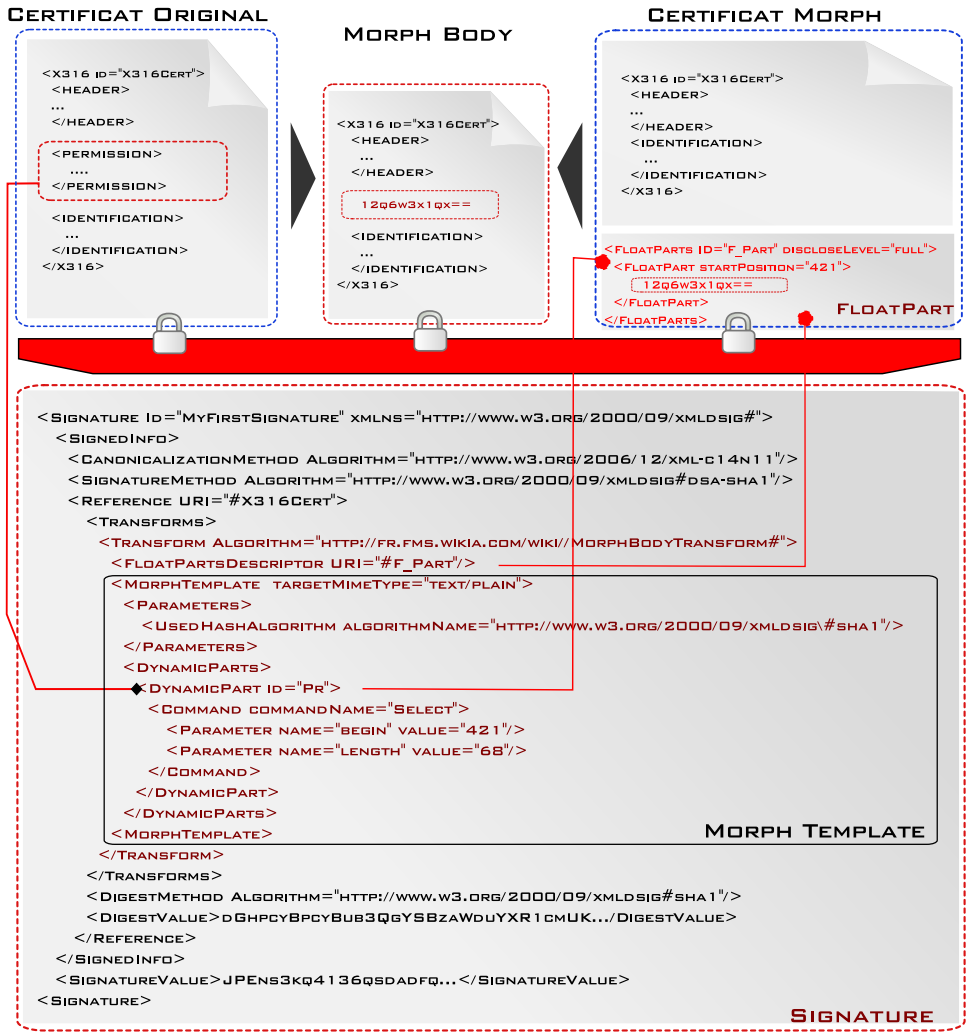

FIGURE 8.10 – Exemple d'une signature FeMoS sur un certificat X316.

Dans l'exemple illustré par la figure 8.10, nous avons détailler la signature FeMoS d'un certificat X316 ayant comme partie dynamique la balise `Permission`, le MorphBody correspondant

ainsi qu'une version Morph de ce X316 en masquant sa partie `Permission`.

Nous pouvons remarquer que la signature est la même pour les trois versions du certificat (le certificat original, le certificat Morph et le MorphBody). Concernant le certificat Morph, la partie *Permission* à été cachée et déplacée dans le bloc contenant les parties flottantes qui se situent en dehors du certificat. Ceci permet d'avoir une nouvelle structure qui ne contient que les attributs qui ont été choisis pour être divulgués.

8.7 Conclusion

Nous avons apporté dans ce chapitre une solution au problème de divulgation d'information sensible de fichiers signé. Nous avons ainsi défini la signature FeMoS qui permet aux utilisateurs de pouvoir cacher certaines informations sensibles dans un document selon le contexte, tout en garantissant la validité de la signature.

L'algorithme Morph est défini avec un Morph template binaire pour rendre la signature indépendante des différents formats de fichiers. Nous avons également spécifié une structure particulière du Template afin de garder une optique d'extensibilité et de généricité.

Cependant, le masquage des parties dynamiques par leur valeur de hash peut être vulnérable quand les parties dynamiques sont de petite taille. Dans ce cas, une méthode de reverse engineering peut exécuter une attaque itérative et retrouver la partie dynamique en un laps de temps acceptable. Cette contrainte a été traitée dans l'état de l'art (voir chapitre 6) en rajoutant une valeur aléatoire qui est concaténée à la partie dynamique avant d'appliquer la fonction de hash. En s'inspirant de ces solutions, dans la prochaine évolution de la signature FeMoS, nous comptons traiter cette contrainte en générant pour chaque partie dynamique une clé de session. Ainsi, avant de procéder au masquage d'une partie dynamique, celle-ci sera chiffrée avec la clé de session qui lui correspond. Ceci permet une obfuscation des partie flottantes rendant ainsi le processus de reverse engineering plus complexe.

La signature FeMoS peut être utilisée comme méthode de transformation dans XMLDSig. Cette intégration est faite de manière standard.

Dans le chapitre suivant, nous allons montrer de manière pratique le développement qui a été réalisé en rajoutant une entrée dans l'API XMLDSig de Mono C# (Version Open Source de .Net). Cette implémentation est le fruit d'un encadrement que j'ai effectué dans le cadre d'un projet de fin d'études d'ingénieur au sein de l'Université USTHB [11] d'Alger.

11. USTHB : Université des Sciences et la Technologie Houari Boumedienne.

Chapitre 9

Implémentation de FeMoS

Sommaire

9.1 Introduction

Nous avons défini, dans le chapitre précédent, un modèle de traitement par transformation pour la signature FeMoS afin de l'intégrer dans le standard XMLDSig.

Un grand nombre de librairies, frameworks et produits supportant ce standard de signature existent. Pour pouvoir démontrer la faisabilité de notre solution, nous avons choisi de considérer les critères suivants :

– L'accessibilité du code source de l'implémentation et la possibilité de modifier certaines de ses parties.

– La portabilité de l'implémentation, qui doit être multi-plateformes/langages.

Le développement multiplateformes à moindre coût nécessite de recourir à un environnement d'exécution virtuel, ou machine virtuelle. Ceci permet d'abstraire l'architecture sous-jacente, de n'écrire le code qu'une seule fois et de pouvoir le déployer sur plusieurs plateformes. Il existe un grand nombre de ces environnements dont la JVM (Java Virtual Machine) de SUN et le CLI (Common Language Infrastructure)[ECMA 06] de Microsoft. Pour implémenter notre solution,

nous avons opté pour le CLI sous une version OpenSource de celui-ci, à savoir l'environnement Mono[MONO 07].

Dans ce chapitre, nous allons présenter la plate-forme Mono, l'API XMLDSig, et enfin l'intégration de FeMoS.

9.2 Environnement de développement

9.2.1 Mono Developement Platform

Le projet Mono, lancé par les développeurs de l'environnement graphique GNOME et sponsorisée par Novell, a pour but d'offrir une alternative Open Source au framework .NET de Microsoft. Les principaux constituants de la plateforme sont :

- Un compilateur `C#` [12]
- Le Mono Runtime[MONO 07] (implémentation du CLI)
- Une implementation de la BCL [13] compatible avec celle fournie par le .Net Framework.
- La Mono Class Library (classes pour Gtk+, Zip, OpenGL, etc...) destinée à fournir une alternative à la FCL [14].

Mono a été conçu dans une optique multiplateformes. Ainsi, grâce a son runtime (qui peut traduire le code CLI vers de nombreuses architectures matérielles), il est, supporté par de nombreux systèmes d'exploitation (versions 32 et 64bits) dont :

- Linux
- BSD
- MacOS X, et iPhoneOS
- Sun Solaris
- Microsoft Windows

La popularité de cette plateforme, sa gratuité et l'enthousiasme de la communauté Open Source, ainsi que la facilité du portage sur les plates-formes mobiles (utilisant windows mobile) nous a motivés à choisir cette solution pour implémenter notre signature morph.

La dernière version de la plateforme Mono est 2.0 et date du 6 Octobre 2008, néanmoins notre intégration s'est faite sur la version 1.9.1 à travers le langage C#.

9.2.2 Support de XMLDSig dans Mono

Nous avons voulu intégrer notre format de signature dans une implémentation existante de XMLDSig et montrer ainsi que notre modèle de traitement par référencement peut s'intégrer de manière simple dans ce standard. La solution proposée a été pensé pour être aussi peu intrusive que possible en n'impliquant pas de modifications majeures dans l'implémentation de XMLDSig.

L'espace de noms `System.Security.Cryptography.Xml` contient des classes destinées à la prise en charge, la création et la validation des signatures numériques au format XMLDSig. La classe `SignedXml` est la classe principalement utilisée. Les autres classes de cet espace de noms ont été modélisées comme le montre le diagramme de classes de la figure 9.1. On retrouve ainsi les différentes classes correspondant aux éléments XML de la signature XMLDSig (discutés dans le chapitre précédent), telles que : les classes `SignedInfo`, `Reference` et `Transform`.

12. Compatible avec les versions 1.0, 2.0 et partiellement avec la version 3.0 de .net
13. Base Class Library
14. Framework Class Library

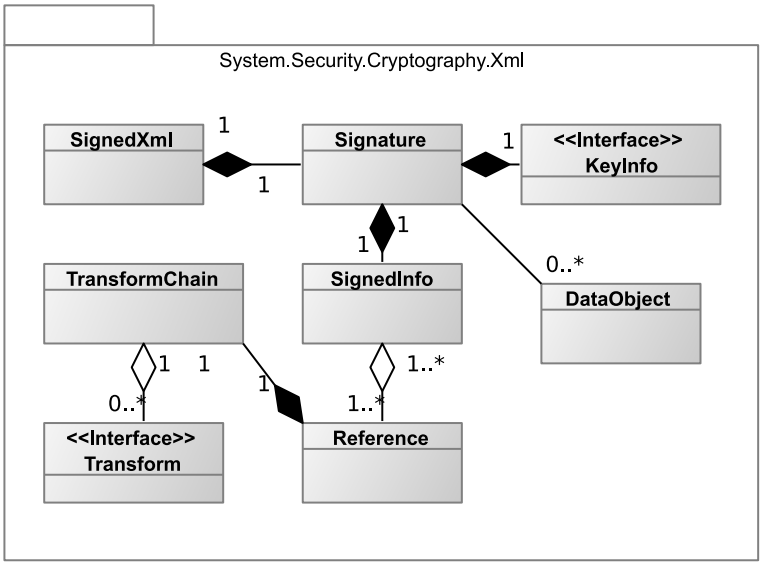

FIGURE 9.1 – Diagramme de classes partielles de l'espace de noms System.Security.Cryptography.Xml

9.2.3 Point d'intégration

La classe `Transform` représente la classe de base abstraite à partir de laquelle sont dérivés tous les algorithmes de transformation comme le montre le diagramme de classes décrit par la figure 9.2. Ainsi chaque algorithme de transformation est représenté par une classe dérivée de `Transform`, comme la classe `XmlDsigEnvelopedSignatureTransform` qui représente la transformation de signatures enveloppées [W3C 08], ou encore la classe `XmlDsigBase64Transform` qui représente la transformation ou encodage en base64.

La surcharge à effectuer dans l'API XMLDSig consiste à intégrer notre signature comme un algorithme de transformation et donc comme une dérivée de la classe `Transform`. Ainsi, l'insertion de notre algorithme sous le nom de "XmlDsigMorphBodyTransform" se fait de manière transparente dans l'API étant donné que notre classe implémente la même interface que n'importe quel autre algorithme de transformation (voir figure 9.2).

9.3 L'application Fenrir

9.3.1 Description

Pour démontrer l'utilisation de notre API, nous avons développé une application cliente en *C#* utilisant la signature Morph pour signer des documents textes. Elle est disponible en libre téléchargement [15] et nécessite l'installation de ".NET framework".

L'application se présente comme un éditeur de texte classique, accompagné d'un ensemble de fonctions offrant la possibilité de signer et de générer des morphs documents. Cette application est construite autour des acteurs Signataire, Propriétaire et Contrôleur.

15. http ://fr.fms.wikia.com/

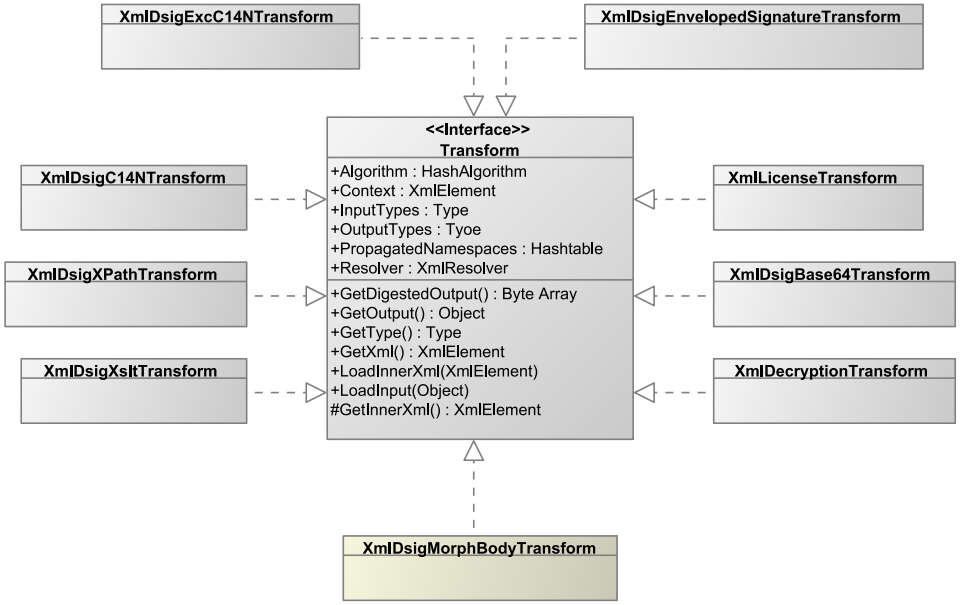

FIGURE 9.2 – Hierarchie des classes `Transform`

L'application permet principalement de réaliser les scenarii d'utilisation suivants :

1. Signature : la signature du fichier du propriétaire par le Signataire.

2. Versionnement : la création d'une version morph du fichier original par le Propriètaire.

3. Vérification : la vérification de la signature par le Contrôleur.

9.3.2 Scenarii de l'application

9.3.2.1 Signature

Le Signataire, après avoir saisi le document original (figure 9.3), signe le document en utilisant la signature morph. Pour cela il a la possibilité de générer une clé aléatoire, ou bien d'en charger une.

Une fois la clé de signature spécifiée, il peut construire les parties dynamiques en sélectionnant une partie du texte et en la marquant comme dynamique (voir figure 9.3 : encadrement continu). Une boîte de dialogue apparaît afin de saisir un certain nombre d'informations telles que les données de remplacement et leur identifiant. Les parties dynamiques sont regroupées dans une liste pour pouvoir être éditée et apparaissent avec un fond bleu dans le texte.

Le signataire peut également ajouter des politiques de divulgation (voir figure 9.3 : encadrement discontinu) avant de signer le document. L'application enregistre ensuite le fichier ainsi que la signature qui l'accompagne dans un fichier séparé portant l'extension ".sig". Ce sont ces deux fichiers qui seront manipulés par le Propriétaire.

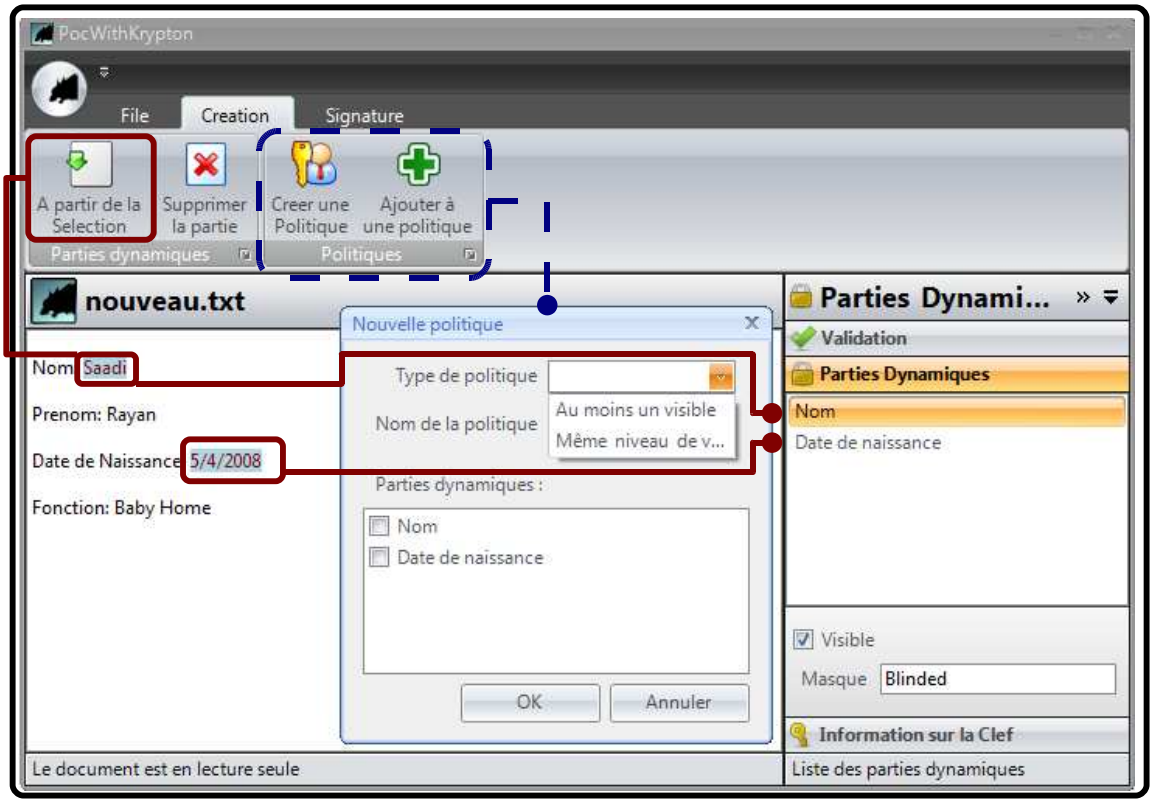

FIGURE 9.3 – Interface du Signataire.

9.3.2.2 Versionnement

En ouvrant un document signé (voir figure 9.4), l'application recherche automatiquement la signature et vérifie sa validité. Dans le cas où le test est vérifié, l'application se met en mode "Versionnement" et affiche les différentes parties dynamiques dans la liste des DPs (dans le panneau de droite) en marquant le texte correspondant avec un fond bleu. Le Propriétaire a la possibilité de séléctionner les parties qu'il veut cacher en décochant la case "Visible"(figure 9.4 : encadrement au centre). Pour une meilleure visualisation, les parties dynamiques cachées sont annotées par le couleur rouge. Pour garantir la non-répudiation du Propriétaire, ce dernier doit signer les modifications (les parties flottantes) qu'il a apportées au document original en utilisant sa clé privée qu'il charge ou qu'il génère. Une fois signé, un nouveau document morph est sauvegardé (voir figure 9.4 : encadrement en haut) ainsi qu'un nouveau fichier de signature contenant la signature du Signataire ainsi que l'ensemble des parties flottantes signées par le Propriétaire.

9.3.2.3 Vérification

Le processus de vérification par le Contrôleur se résume à réceptionner le document Morph. Lors de l'ouverture du document, la vérification se fait automatiquement. Ainsi, dans le cas où la signature du signataire est valide, l'application vérifie la signature du distributeur ainsi que le respect des politiques de divulgation. Toute erreur dans l'un de ces processus entraîne l'invalidation de la signature et la non ouverture du document (voir figure 9.5).

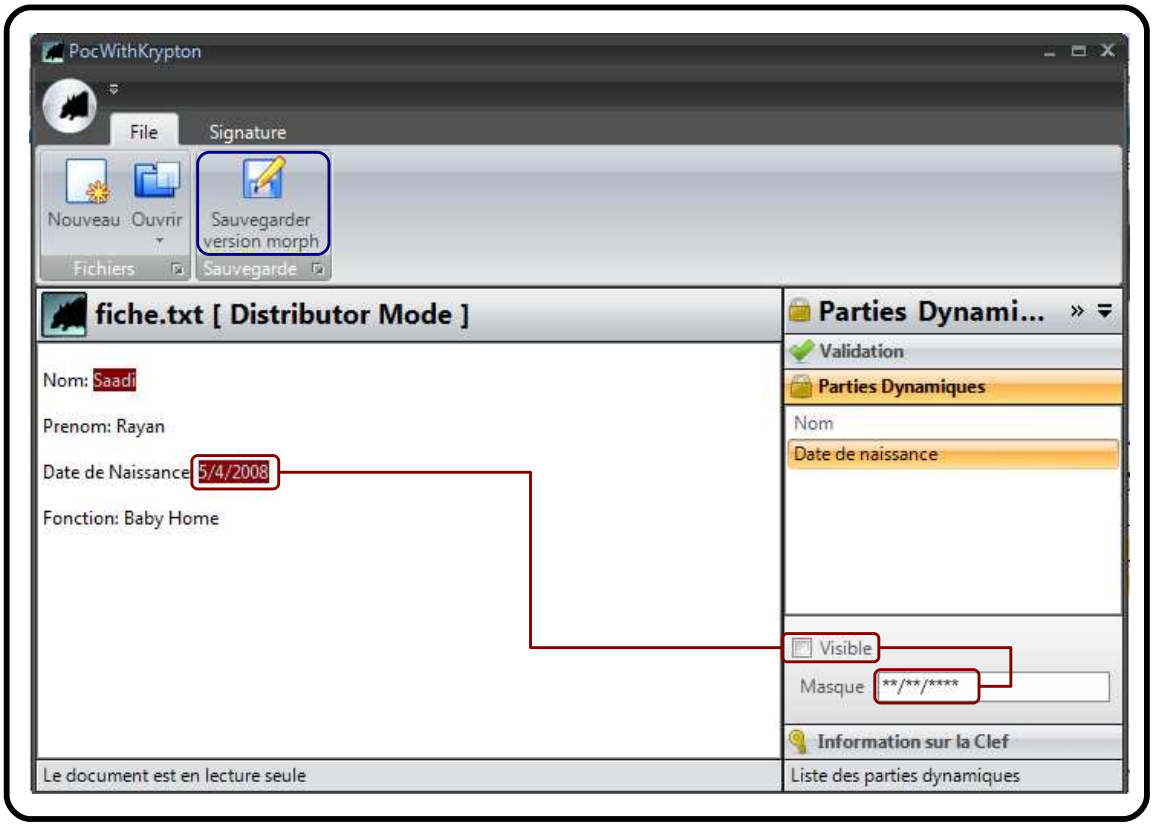

FIGURE 9.4 – Interface du Propriétaire.

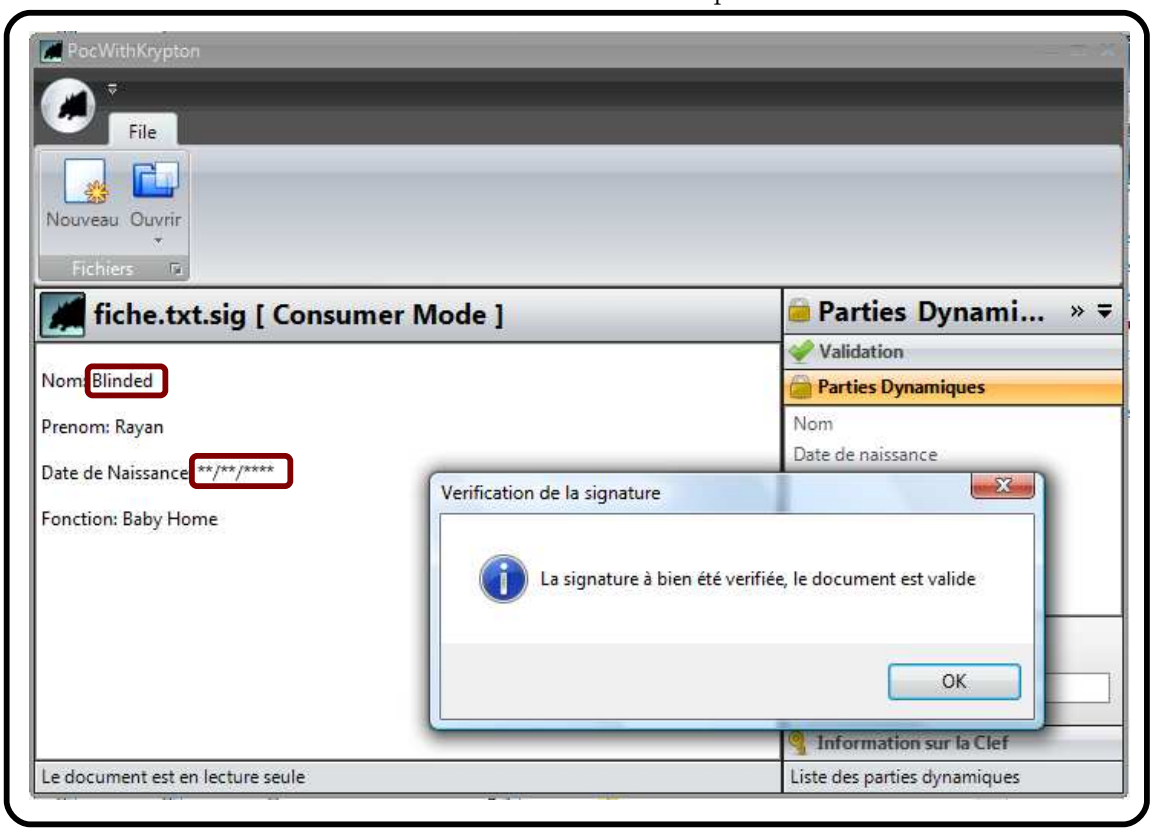

FIGURE 9.5 – Interface du Contrôleur.

9.4 Déploiement sur terminaux mobiles

Un portage de la partie signature de l'API FeMoS sur Windows mobile nous a permis de réaliser un test de performance sur le temps de calcul nécessaire pour générer une signature morph.

Afin de réaliser ce test, nous avons choisis trois plateformes différentes :

- Un ordinateur fixe sous Windows XP équipé d'un processeur Intel 3GHZ.
- Un PDA (HP HX4700) sous Windows mobile 2003 équipé d'un processeur ARM 624MHZ.
- Un SmartPhone (SPV M3000) sous Windows mobile 5 équipé d'un processeur ARM 198MHZ.

Le test de performance définit aléatoirement des parties dynamiques dans un fichier XML choisi ayant une taille de 200KO. Comme le montre la figure 9.6, en faisant varier le nombre des parties dynamiques (DP) de 20 à 200 (voir l'axe des abscisses), nous remarquons que le temps nécessaire pour générer une signature FeMoS reste faible étant donné que pour toutes plateformes confondues le calcul ne dépasse pas 0,7 seconde pour 200 parties dynamiques (ce qui constitue déjà un très grand nombre de parties -typiquement, un document va posséder au plus une dizaine de parties dynamiques-). De plus, tel que l'illustre la figure 9.6, le temps de calcul augmente en fonction du nombre des DPs de manière linéaire.

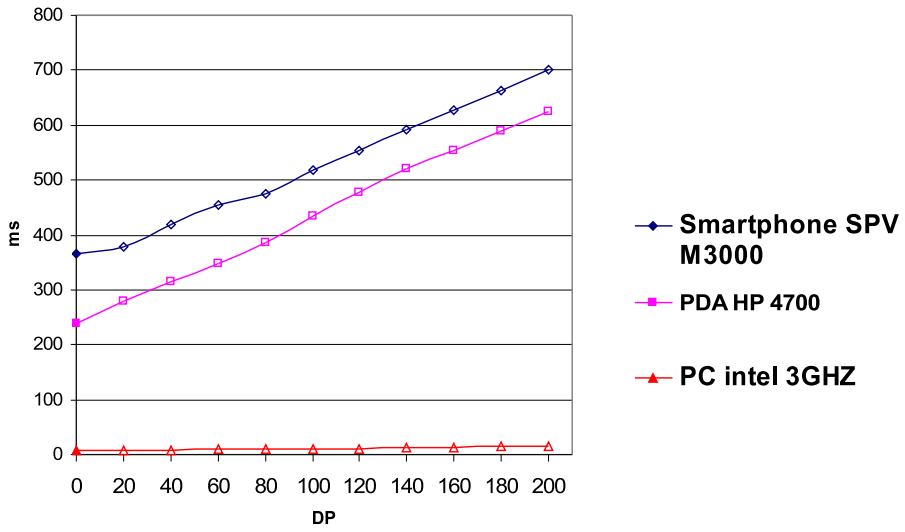

FIGURE 9.6 – Temps de Calcul d'une signature FeMoS.

9.5 Conclusion

Nous avons exposé dans ce chapitre la méthodologie suivie pour implémenter la signature FeMoS. Cette réalisation a permis à notre modèle de signature et de traitement, de s'intégrer aux implémentations existantes en respectant les spécifications du standard XMLDSig. En effet,

le code nécessaire à la vérification de la signature n'a pas besoin d'être modifié après introduction de notre transformation.

Par ailleurs, nous avons décrit l'API Fenrir que nous avons voulu extensible et ouverte à de futurs développements, ainsi que les points principaux sur lesquels elle se base.

Dans la dernière partie de ce chapitre est illustrée l'application qui nous sert de démonstration de l'API et de ses fonctionnalités. Cette application utilise le modèle que nous avons utilisé tout au long de notre travail et met en évidence l'intérêt et l'aspect applicatif de l'utilisation de l'API.

Enfin, l'API Fenrir Signature et son code source, la présentation de la signature FeMoS et l'application ont été déployés en ligne à travers un wiki à l'adresse "http ://fr.fms.wikia.com/".

Quatrième partie

Intégration

Chapitre 10

Mise en œuvre du système Chameleon

Sommaire

10.1 Introduction

Dans le chapitre 3, nous avons mis en place notre système Chameleon et avons défini les besoins en terme de mécanismes de confiance et de certification. Ainsi, tous le long du manuscrit nous avons étudié dans les deux parties précédentes le modèle de confiance (T2D) ainsi que le modèle de certification (X316) et sa signature (FeMoS). Comme le montre la figure 10.1, toutes ces contributions viennent s'intégrer dans notre système Chameleon comme suit :

- Les Certificate Checker et Certificate Provider du PEP se basent sur le certificat $X316$; ils sont donc renommés respectivement X316 Checker et X316 Provider.
- Le module Trust & Discovery utilise le modèle de confiance T2D et donc porte actuellement le nom $T2D$.

Nos travaux sur la politique de mapping sont encore au stade de développement. C'est pour cette raison que cet aspect du système Chameleon sera abordé comme perspective dans le dernier chapitre. Néanmoins, dans l'état actuel du développement, le mapping est implémenté de manière basique à l'aide d'une table de correspondance qui permet de réaliser le mapping entre

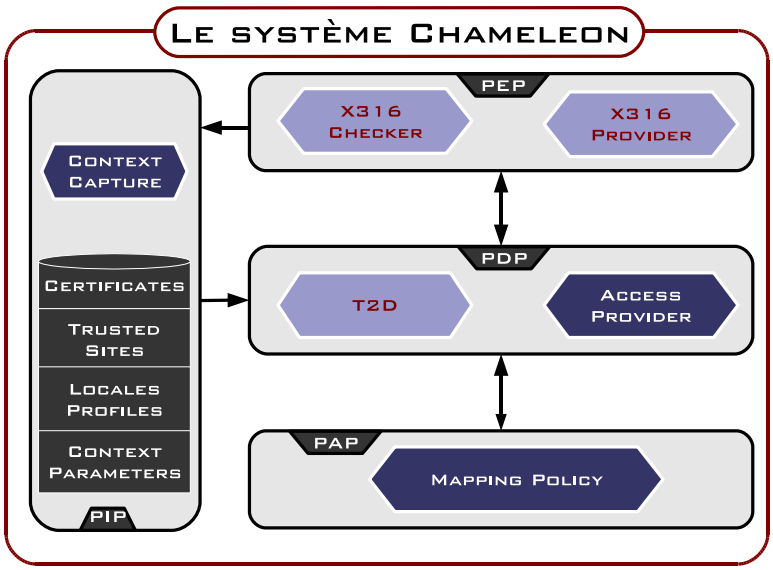

Figure 10.1 – Le système Chameleon

les différents profils des sites de confiance.

Dans ce chapitre nous illustrons le système *Chameleon* d'un point de vue applicatif. En premier lieu, nous allons présenter comment le certificat X316 ainsi que le modèle T2D sont intégrés dans l'implémentation. Ensuite, nous allons décrire les différents modules ainsi que les interfaces qui permettent la configuration et la mise en route du système.

10.2 Intégration dans le système Chameleon

10.2.1 X316 et Chameleon

Pour rappel, le certificat "X316" est composé des quatre parties suivantes :

– "Header" : Cette partie représente l'entête. Elle contient les informations nécessaires pour traiter et identifier le certificat.

– "Permissions" : Comme son nom l'indique, cette partie regroupe tous les droits du propriétaire du certificat, par exemple son rôle.

– "Identifiers" : Cette partie contient les attributs nécessaires pour identifier le propriétaire du certificat. En comparaison avec les modèles de certificats existants, X316 embarque non pas un identifiant (généralement la clé publique du sujet), mais plusieurs.

– "Signature" : Le certificat X316 utilise la signature XMLDSig avec la transformation FeMoS.

Pour les besoins du système Chameleon, le modèle de certificat X316 a permis de représenter trois catégories de certificats : les certificats utilisateur, les certificats d'exploration et les certificats de révocation.

10.2.1.1 Les certificats utilisateur

Les certificats utilisateur sont des certificats d'identité et d'attributs. Ils fonctionnent comme une attestation qui permet d'identifier son propriétaire et de faire valoir son profil et ses privilèges au sein de différentes organisations.

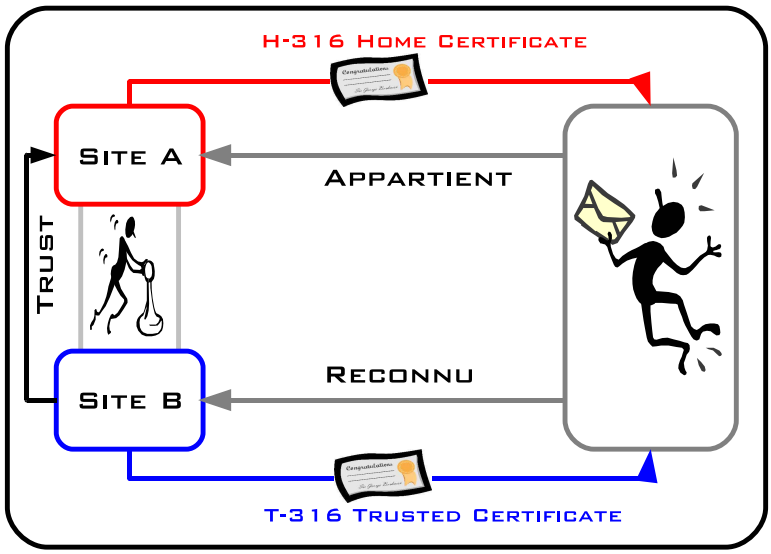

Figure 10.2 – Les types de certificat X316

Comme le montre la figure 10.2, un certificat X316 utilisateur peut être délivré à deux catégories d'usagers :

- Les membres du domaine : tout utilisateur peut obtenir auprès de son site d'appartenance un "Home Certificate" ou certificat d'appartenance noté "H316" dénotant son statut au sein du domaine dont il est membre. Il se caractérise par une longue durée de vie et est délivré par l'autorité de certification de chaque site à tous ses membres.

- Les visiteurs nomades dignes de confiance : chaque site peut délivrer des "Trusted Certificate" ou certificats de confiance noté "T316" à tout utilisateur reconnu comme une entité de confiance pouvant interagir avec les ressources et les utilisateurs locaux. Ce dernier possède une durée de vie plus courte que le H316. Il peut être obtenu à partir d'un certificat de confiance ou d'appartenance, à condition que l'autorité de certification (site source) du H316 ou du T316 soit reconnue localement comme une entité de confiance.

L'usage du certificat de confiance a principalement deux avantages. Premièrement, il réduit le trafic réseau, car le site accueillant l'usager nomade ne va évaluer que la chaîne le séparant de l'autorité qui a délivré le T316 et n'a plus besoin de propager la requête de découverte jusqu'au domaine d'appartenance (site source). Deuxièmement, comme le montre la figure 10.3, les sites qui se font confiance peuvent être connectés et ont la possibilité de communiquer pour propager la requête de découverte, mais peuvent aussi n'être liés par aucun lien physique. Dans le ce second cas de figure, le seul moyen pour qu'un site distant puisse communiquer avec un site de confiance est de délivrer aux utilisateurs nomades un certificat qui va embarquer le degré de confiance de

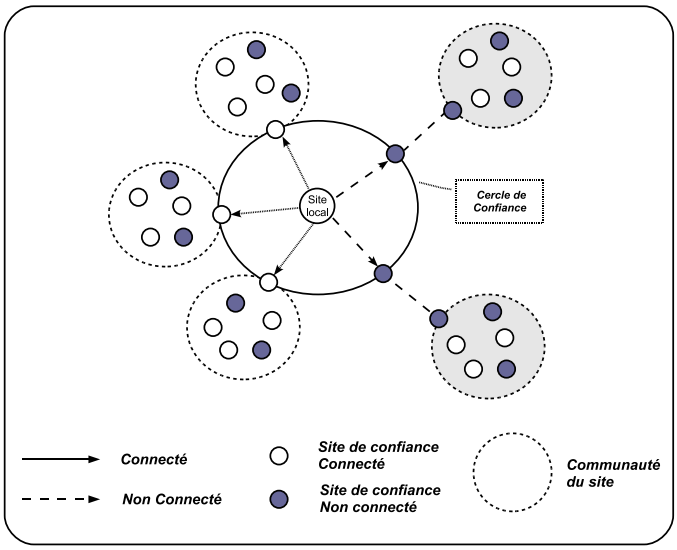

FIGURE 10.3 – La communauté de confiance

la chaîne le liant au site d'appartenance de l'utilisateur.

Tel que l'illustre la figure 10.4, la structure des deux types de certificats X316 est presque identique, sauf pour la partie "Header" du certificat de confiance qui contient, en plus des attributs standards :

- la chaîne de confiance qui a permis l'attribution de T316,
- le degré de confiance correspondant à l'évaluation du site d'appartenance du propriétaire de ce certificat.

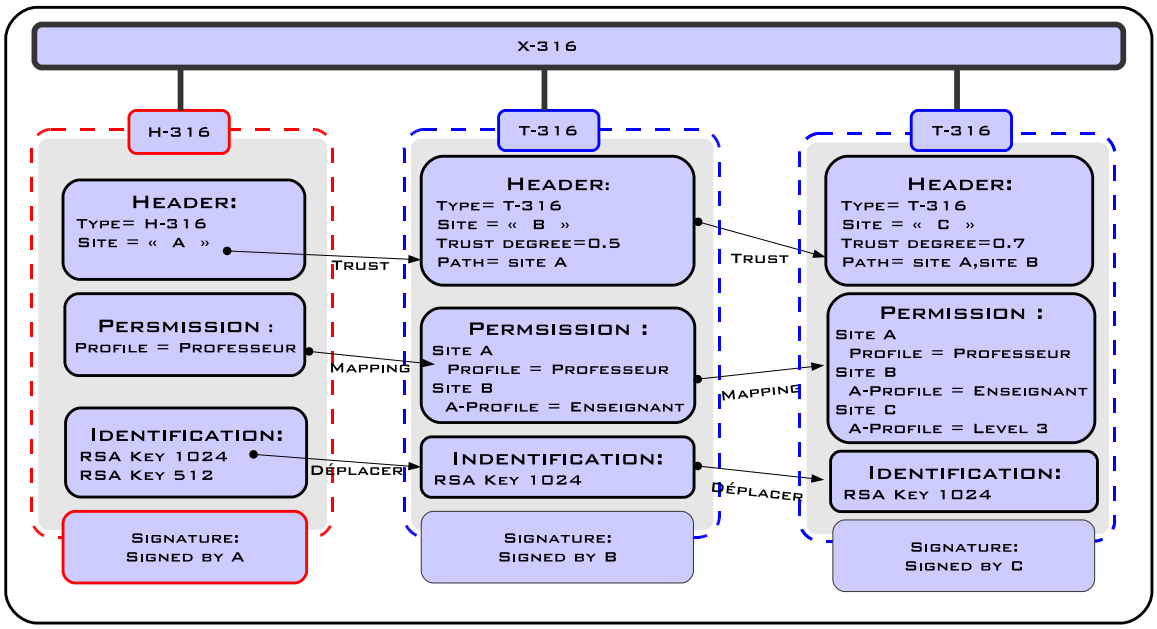

FIGURE 10.4 – La syntaxe du X316

Dans le système Chameleon, la structure du X316 permet la génération automatique des certificats pour le compte des utilisateurs nomades pendant leur déplacement. En prenant l'exemple du $Professeur$ (Voir Chapitre 1), on peut remarquer dans la figure 10.4 que le Professeur Bob peut obtenir une attestation de l'université B, générée à partir de son certificat d'appartenance (délivré par l'Université A), en reportant les droits figurants sur le H316 et les identifiants jugés valides par l'autorité locale B. Ainsi, comme pour le "Sigle Sign On", le fait d'utiliser toujours les mêmes identifiants permet à l'usager de ne gérer qu'une seule identité, en l'occurrence celle définie par son domaine d'appartenance.

De la même manière, en utilisant ce dernier certificat obtenu par le professeur chez le site B, il peut également obtenir un nouveau T316 avec un profil analogue de "niveau trois" lui permettant de participer à la conférence se déroulant sur le site C. Dans ce cas de figure, on peut remarquer que la présence du profil Professeur sur le premier certificat T316 de Bob ne lui est pas nécessaire pour obtenir le second certificat T316 étant donné que le site C aura besoin du profil Enseignant afin d'appliquer la politique de mise en correspondance. Ainsi, en utilisant la propriété de la signature FeMoS, le professeur Bob peut masquer le profil $Profeseur$ avant de solliciter un accès au site C.

On peut remarquer également que sur chacun des certificats (T-316) figure le degré de confiance (P^0) qu'attribuer le signataire au site d'appartenance de Bob (ex : sur le premier certificat T316 de la figure 10.4 le Trust degree $P^0(B, A) = 0,5$).

10.2.1.2 Les certificats d'exploration

Afin de rechercher dans le réseau de confiance l'identité d'un domaine qui n'est pas connu (la source), le site cible diffuse une requête à travers le réseau, et les tiers de confiance qui reconnaissent ce site source renvoient une réponse. Afin de garantir la traçabilité et la non-répudiation des informations échangées entre pairs, nous allons définir deux types de certificats X316 qu'on nommera des certificats d'exploration formulant les requêtes et les réponses.

Ainsi les certificats d'exploration sont : le certificat de découverte "D316" et le certificat de réponse "R316".

Dans la figure 10.5 est illustré le processus de génération et de propagation de certificats $D316$ et $R316$ à travers une chaîne de confiance constituée dans l'ordre par les quatres sites A,B,C et D. On remarque que le D316 et le R316 respectent la structure d'un certificat X316.

Le professeur Bob sollicite un accès au site D. Il envoie son certificat H316 qui lui a été fourni par son site d'appartenance (le site A). Un D316 est alors généré par le site cible (le site D) pour formuler une requête qui contient le nom du site source (site A) à évaluer ainsi que le profil (professeur) de l'utilisateur nomade. Ce D316 est propagé sur le réseau de manière récursive à travers les sites membres du cercle de confiance sortant (T_oS). Le D316 a une très courte durée de vie qui est de l'ordre de quelques secondes, ce qui permet de ne plus le renvoyer s'il est périmé.

Une fois que le sujet de la requête (site source) est reconnu par un tiers de confiance (site C), ce dernier formule sa réponse dans un certificat R316. Ce certificat comporte l'identifiant du D316 comme sujet, le chemin qui à permis de générer ce certificat (path), la confiance reliant la source au signataire de R316 (trust deg), et le profil analogue (a-Profile), qui est le profil qui correspond localement au rôle professeur, selon la politique de mapping du tiers de confiance.

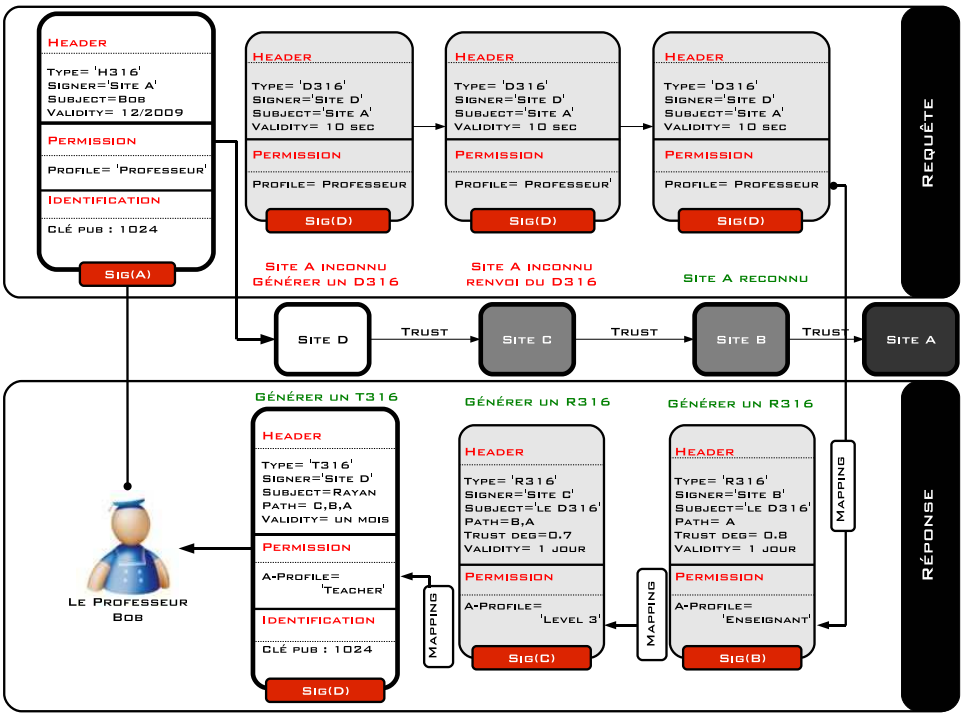

FIGURE 10.5 – Les certificats d'exploration

Ce certificat est renvoyé à travers le chemin inverse de celui qui a permis d'acheminer le D316 correspondant. Ainsi, et jusqu'à arriver à destination (site cible), chaque nœud régénère un R316 en mettant à jour, principalement, le profil analogue ainsi que le degré et le chemin de confiance.

Une fois que le site cible reçoit la réponse, il calcule le degré de confiance reflétant sa relation avec la source et décide, en fonction de cette valeur, d'accorder ou de refuser l'accès à l'usager nomade. Dans le cas où l'accès est accordé, un certificat de confiance T316 est alors délivré à cet utilisateur dénotant un profil local (Teacher) qui lui permet d'accéder aux ressources de la cible (site D).

10.2.1.3 Le certificat de révocation

Afin de permettre le processus de révocation nous avons défini, toujours à partir du modèle X316, le certificat de révocation noté V316. Il est composé principalement d'une partie *Header* où le *Subject* identifie principalement le certificat à révoquer (ID du certificat). Celui-ci est alors signé par l'autorité du domaine et est envoyé de manière récursive à tous les sites qui font confiance au site initiateur du certificat de révocation V316 (les membres du cercle de confiance entrant T_iS).

A l'instar du certificat de découverte, le certificat de révocation possède une très courte durée de vie, ce qui permet d'arrêter la diffusion une fois ce dernier périmé.

10.2.1.4 Synthèse

A l'inverse des systèmes de sécurité existants, le système Chameleon utilise un modèle de certificats unifié. En effet, le X316 présente une structure flexible qui nous permet de représenter

tous les types de certificats nécessaires pour le fonctionnement de notre implémentation (voir tableau 10.2.1.4).

X316	H316	T316	D316	R316	V316
Header	oui	oui	oui	oui	oui
CertificateID	oui	oui	oui	oui	oui
CertificateDegree	non	oui	non	oui	non
Validity	oui	oui	oui	oui	non
Subject	User	User	Site	CertificateID	CertificateID
Issuer	oui	oui	oui	oui	oui
TrustPath	non	oui	non	oui	non
Permission	User Profile	User Profile(s)	User Profile	User a-Profile	non
Identifiers	oui	oui	non	non	non

TABLE 10.1 – Les différents types de certificats X316

10.2.2 Le modèle T2D dans le Chameleon

Dans le système Chameleon, le modèle T2D implémente le module Trust & Discovery. Il se charge de découvrir un chemin de confiance et de l'évaluer. Ainsi, deux processus sont identifiés comme suit :

- Processus de découverte : Lors de cette étape le T2D commande au X316 provider d'envoyer un D316 afin de rechercher un site inconnu localement. Ainsi, chaque fois que le D316 transite à travers un site de confiance, son identifiant est enregistré afin de permettre le renvoi d'une réponse (R316) dans le cas où le chemin de découverte arrive à retrouver le site source. Ainsi, chaque nœud du réseau propage le certificat D316 une seule fois à tous ses sites de confiance, et donc ne diffuse un D316 que si c'est la première fois qu'il le reçoit et que sa date de validité n'a pas encore expiré.

- Processus de réponse : Cette étape utilise l'algorithme défini dans l'annexe C.2 (basé sur Dijkstra). Dans notre implémentation la réponse est renvoyée par chaque nœud aux sites appartenant à son T_iS ayant diffuser le requête (D316) sujette de la réponse.

Dans le cadre de l'implémentation actuelle du Chameleon, le calcul du chemin de référence θP^m qui permet de fixer le seuil de méfiance transitive (voir annexe C.1) n'a pas encore été intégré ; il est pour l'instant fixé en fonction de la taille approximative du réseau.

10.3 Implémentation du Chameleon

Le système Chameleon est implémenté par un service nommé *Service Chameleon* qui est accessible via une adresse IP et écoute sur un port spécifique. Ainsi, tel que l'illustre la figure 10.6 ce service est accessible à travers les trois interfaces de connexion suivantes :

- "Interface de Contrôle" qui permet à l'administrateur de configurer la plate-forme.
- "Interface du Service" qui sert de moyen d'interconnexion entre les sites de confiance.

– "Interface utilisateur" qui permet aux utilisateurs de se connecter et d'acquérir des certificats.

Ces interfaces permettent d'accéder et de manipuler l'application *Service Chameleon*. Dans ce qui suit, nous allons présenter les trois interfaces, afin d'illustrer le fonctionnement interne de notre service.

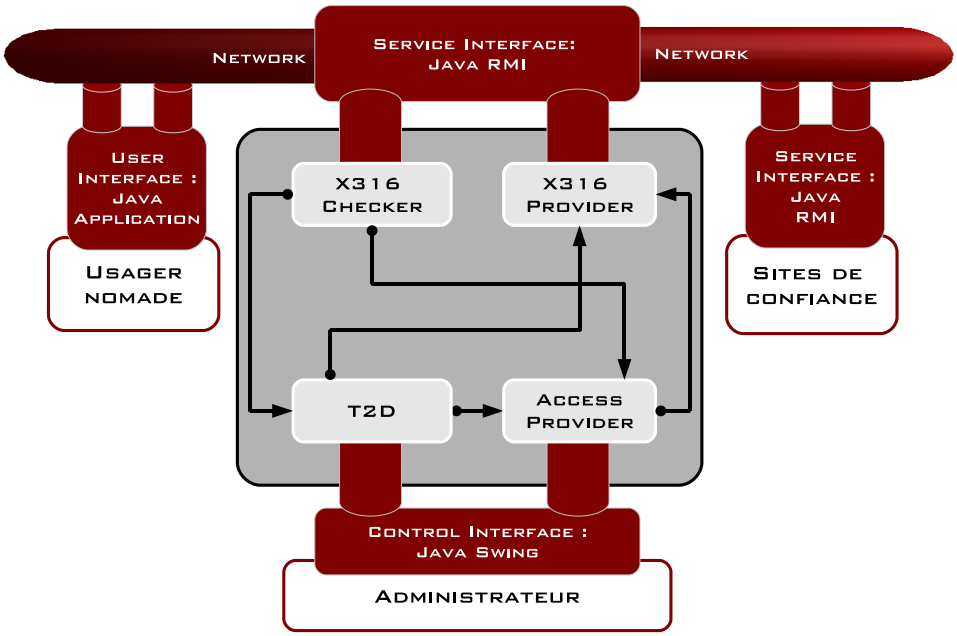

Figure 10.6 – Implémentation du Chameleon

10.3.1 Interface de Contrôle (*ControlInterface*)

Dans la figure 10.6, on peut remarquer que l'interface de contrôle est dédiée à l'administrateur. A travers cette application, l'administrateur a la possibilité de configurer le système Chameleon. Celle-ci est codée en Java et présente l'interface graphique illustrée dans la figure 10.7. Cette dernière offre principalement les fonctions suivantes :

– Initialiser le cercle de confiance (figure 10.7 : cadran 1) : grâce aux boutons *add* et *remove*, l'administrateur compose son cercle de confiance. L'ordre des sites dans la liste est important car il représente le tri défini par le processus du TrustSort.

– Initialiser les paramètres nécessaires au fonctionnent du modèle T2D (figure 10.7 : cadran 2) : dans ce cadran l'administrateur définit le disposition level l^0, le seuil local de méfiance T^0, ainsi qu'un certain nombre de paramètres (taille du réseau, taille maximum du cercle de confiance sortant (dmax) et couverture réseau souhaitée (access average)) afin d'initialiser le seuil de méfiance transitive θT^0.

– Afficher l'évaluation des sites de confiance par la fonction BV (figure 10.7 : bouton "Display Graph").

– Envoyer une requête de révocation (figure 10.7 : cadran 3).

– Lancer le *Service Chameleon* (figure 10.7 : bouton "Launch Service").

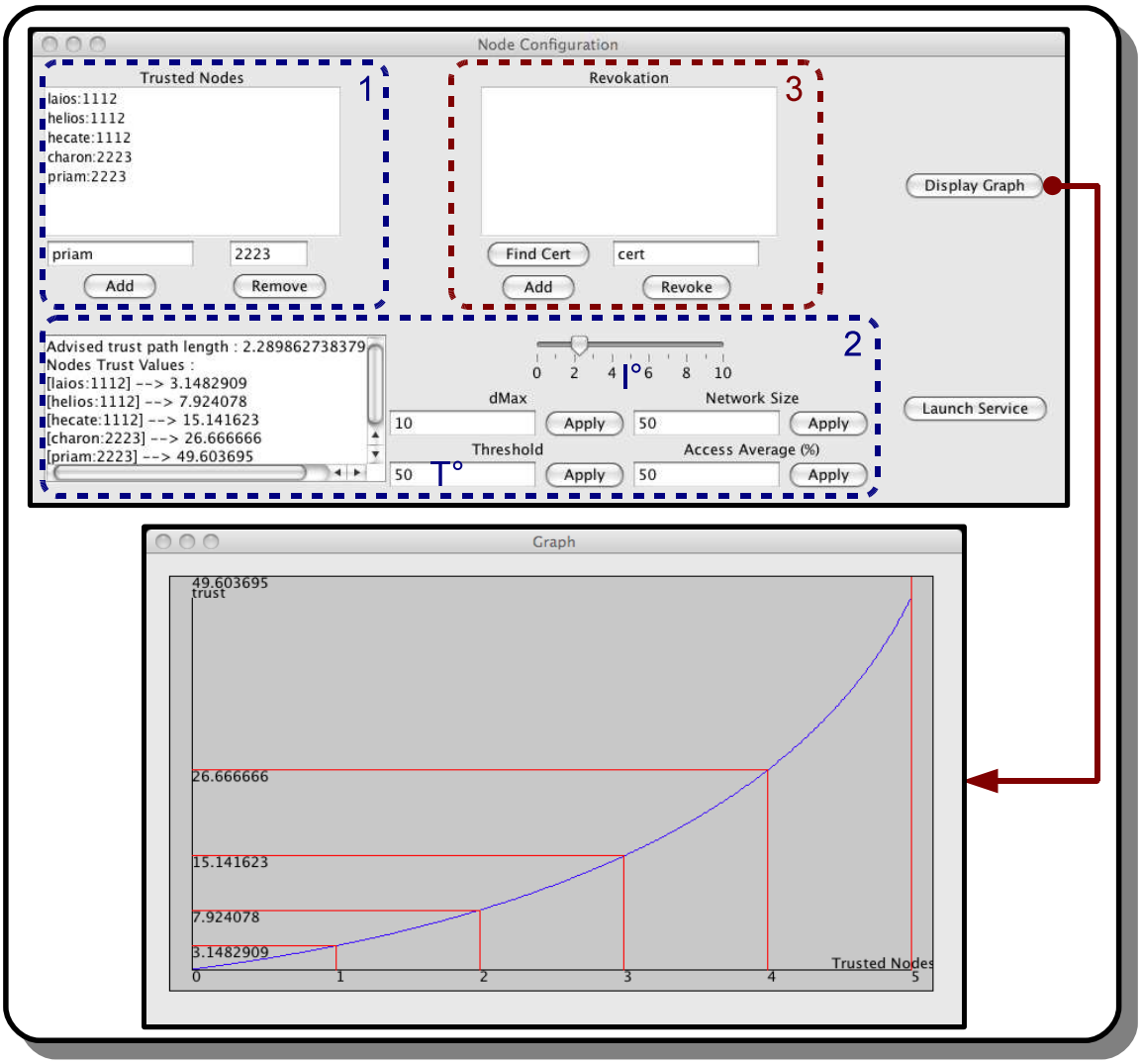

FIGURE 10.7 – L'Interface de contrôle

10.3.2 Interface utilisateur ($UserInterface$)

Dans la figure 10.6, le service Chameleon peut s'interfacer depuis le Web afin de proposer un portail de connexion pour les utilisateurs nomades.

Cette interface est une application développée en JAVA. Dans le menu d'accueil, deux modes d'authentification sont disponibles, soit en utilisant un username/password (u/p), soit en envoyant un X316 d'attributs (H/T316). Un certificat X316 peut comporter jusqu'à trois identifiants, deux clés publiques (512 et 1024) et un identifiant obtenu à partir d'une photo de l'utilisateur et d'un mot de passe. Cet identifiant a été défini pour être utilisé à partir d'un

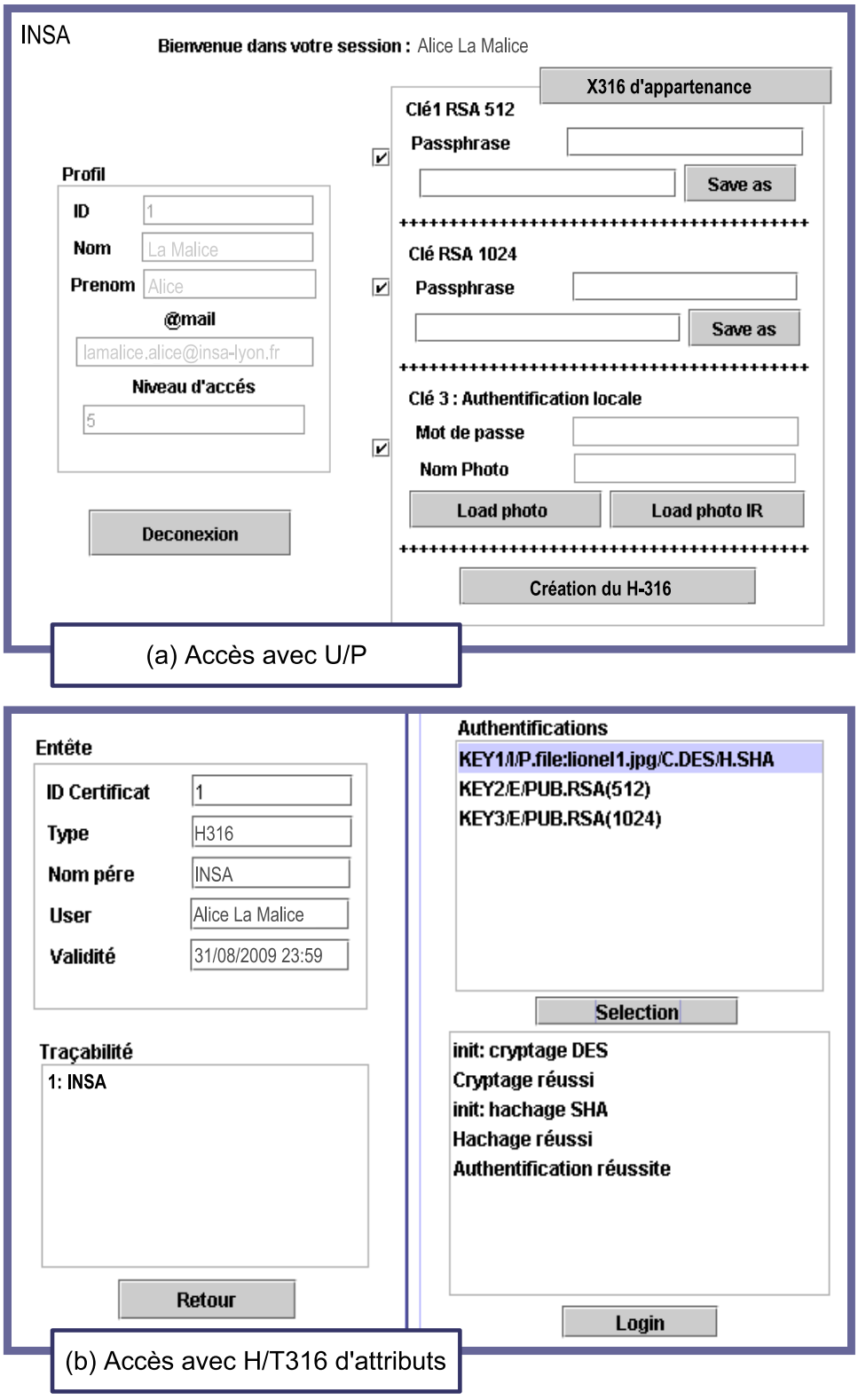

FIGURE 10.8 – L'interface utilisateur

téléphone portable équipé d'un appareil photo et d'un port Infrarouge (IR).

L'appareil photo du téléphone sert à prendre une photo de l'utilisateur. Celle-ci est ensuite chiffrée avec un mot de passe choisi par l'utilisateur. Enfin, le résultat est injecté dans une fonction de Hashing afin d'obtenir un résidu qui est stocké dans le certificat comme identifiant du propriétaire. La photo utilisée dans la génération de ce résidu permet d'identifier l'utilisateur de manière unique, étant donné qu'elle ne peut être prise que par son dispositif, et par là-même ne peut être acquise de nouveau.

Ce type d'authentification est analogue à celui employé par les lecteurs des cartes bancaires (la carte prenant le rôle de la photo et le code pin celui du mot de passe). Pour simuler cet environnement, nous avons utilisé un ordinateur portable doté d'un récepteur infrarouge qui joue un rôle analogue à celui du lecteur de cartes. Le téléphone portable utilisé possède un port infrarouge pour assurer la communication avec notre pseudo-terminal.

Ainsi les différents processus d'authentification des usagers nomades sont les suivants :

- Authentification utilisant un Username/Password : Ce mode d'accès classique est valable dans le cas où l'usager est membre du domaine. Ainsi, en introduisant ses identifiants, l'usager accède à son espace personnel. L'interface web lui propose alors de générer son certificat H316 en lui donnant la possibilité de choisir les identifiants qu'il veut embarquer (voir figure 10.8(a)).

- Login utilisant un H316 : Une fois que l'utilisateur possède un certificat H316, il peut solliciter un certain nombre de privilèges de manière automatique chez les sites qui ont confiance(de manière transitive) en son site d'appartenance. L'usager nomade peut ainsi accéder au portail du site cible et s'authentifier en utilisant son H316 (voir figure 10.8(b)). Si la procédure d'authentification est validée, le portail propose alors à l'usager de lui fournir un certificat de confiance T316 généré à partir de son H316.

- Login utilisant un T316 : L'authentification utilisant le T316 se fait de la même manière qu'avec le H316. Cependant, chaque site est libre d'évaluer différemment un H316 d'un T316, en attribuant, par exemple, un T316 avec une date de validité plus importante s'il est obtenu à partir d'un H316.

L'interface utilisateur sera amenée à évoluer vers une interface WEB. En effet, le service Chameleon peut s'interfacer depuis le réseau permettant ainsi de proposer un site web capable de procéder à l'authentification et à l'attribution de droits d'accès via les certificats X316.

10.3.3 L'interface du Service ($ServiceInterface$)

Cette interface définit les objets et les méthodes du *Service Chameleon*. L'interface du service propose la méthode $ServiceInterface.cert(X316)$ accessible depuis le réseau (via les RMI [16] de Java). A travers cette interface, le service réceptionne les différents certificats X316.

Chaque site qui adhère au système Chameleon déploie ce service afin d'offrir aux usagers nomades la possibilité d'acquérir un accès local. Les services déployés dans l'environnement communiquent exclusivement en utilisant des certificats, ce qui garantit une traçabilité et un système de non-répudiation pour tout échange effectué entre pairs.

L'accès au service est sécurisé, puisque de l'extérieur, seule la méthode $ServiceInterface.cert(X316)$

16. Remote Method Invocation

est accessible. Ainsi la manipulation des objets du $X316$ *Provider* (production de certificats) se fait exclusivement depuis le service lui-même (voir figure 10.9).

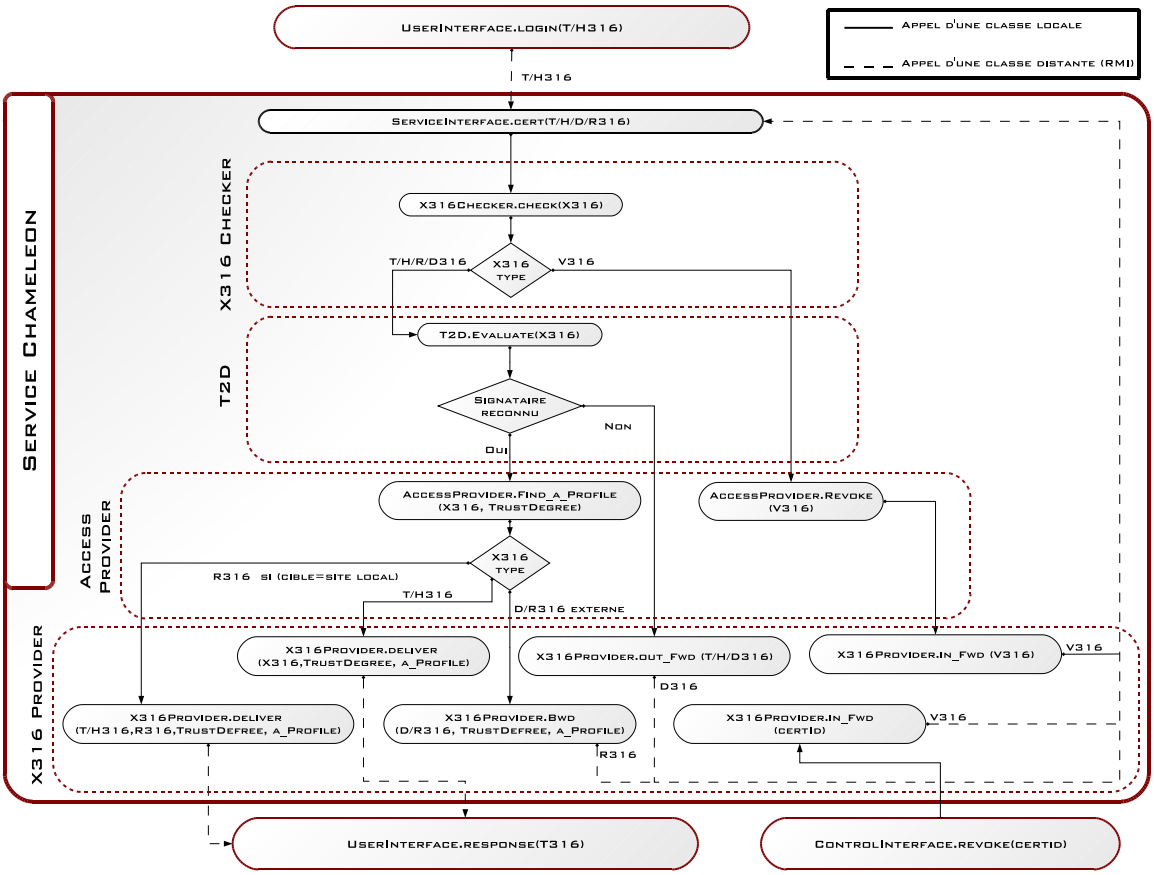

Figure 10.9 – Le service Chameleon

Comme le montre la figure 10.9, le service Chameleon peut recevoir et émettre les structures de données suivantes :

- En entrée :
  - Un H316 ou T316 : Dans le cas d'un login d'un utilisateur nomade.
  - Un D316 : Dans le cas d'une requête de découverte.
  - Un R316 : Dans le cas d'une réponse à une requête D316.
  - Un V316 : Dans le cas d'un requête de révocation.

- En sortie :
  - Un T316 : Il est fourni à un utilisateur nomade si son site source est reconnu comme étant un site de confiance.
  - Un D316 : Il est soit généré soit renvoyé si le site source n'est pas connu localement.
  - Un R316 : Il est généré dans le cas où le site source est reconnu. Il comporte l'évaluation (en terme de confiance) de ce dernier, ainsi que le profil analogue (a-Pr) qui peut être attribué à l'utilisateur nomade.
  - Un V316 : Il est soit généré soit renvoyé aux sites membres de cercle de confiance entrant.

Le fonctionnement du service Chameleon est illustré par le digramme de la figure 10.9. Il définit le flux de données qui est construit par les quatre modules du service comme suit :

1. Une fois que l'utilisateur a été authentifié par le $UserInterface$ en utilisant son certificat H/T316, ce dernier est envoyé au service en utilisant la méthode $login(X316)$.

2. la méthode *Login* de l'interface utilisateur sollicite l'interface du service en RMI à travers la méthode *cert*.

3. La méthode $ServiceInterface.cert(X316)$ est appelée depuis le réseau. Celle-ci reçoit en entrée un certificat X316. Cette méthode transmet le certificat reçu au $X316Checker$ en invoquant la méthode $Check(X316)$ et ensuite accuse la réception du certificat afin de libérer l'interface appelante étant donné que le service fonctionne en mode asynchrone.

4. La méthode *Check* contrôle la validité du certificat. Dans le cas où le certificat est valide, celui-ci est envoyé à l'$AccessProvider$ s'il s'agit d'un certificat V316, sinon c'est le module de confiance $T2D$ qui le réceptionne à travers la méthode *Evaluate*. Cette dernière a la charge d'évaluer la source qui représente :

   – le signataire du certificat s'il s'agit d'un certificat T/H/R316.

   – le sujet du certificat s'il s'agit d'un certificat D316.

5. Dans le cas où le signataire (T/H/R316) ou le sujet du certificat (D316) n'est pas reconnu, la méthode *Evaluate* fait appel au $X316Provider$ afin qu'il lance une requête de découverte à travers un D316. Dans le cas contraire, c'est l'$AccessProvider$ qui est instancié à travers la méthode $Find\_a\_Profile$.

6. La méthode $Find\_a\_Profile$ reçoit en entrée le X316 (T/H/R/D316) évalué ainsi que le $TrustDegree$ qui représente la valeur de confiance qui est attribuée par le site local à la source. Celle-ci récupère alors, en fonction de la politique de mapping, le profil analogue qui correspond à celui annoté par le X316. Ensuite, en fonction du type du certificat, le $X316Provider$ est sollicité afin de transmettre un X316 à l'utilisateur, à un service ou à un ensemble de services.

7. Le $AccessProvider$ peut également être sollicité directement par le $X316Checker$ (voir étape 4) à travers la méthode *Revoke*. Cette dernière se charge de révoquer le certificat annoté par le V316. Ensuite, en fonction de la politique locale du site, le $X316Provider$ peur être appelé à retransmettre ce certificat aux sites faisants confiance au site local (les site membres du T_IS).

8. Le $X316Provider$ représente le point de sortie du service Chameleon. Il a la responsabilité de créer et de transmettre un X316 à quatre classes d'entités, comme suit :

- **Les services des sites membres du T_OS :** Dans ce cas de figure, c'est la méthode *out_Fwd* qui est instanciée par le module de confiance T2D (voir étape 5). Cette méthode se charge de diffuser un D316, qui est soit reçu en entrée, soit créé à partir du H/T316 reçu en paramètre, à toutes les interfaces de service des sites de T_OS afin de rechercher le site source de l'utilisateur.

- **Le service appelant :** Dans le cas où le site source à pu être évaluée, la méthode *Bwd* s'occupe de générer un R316 et de le transmettre au site ayant émis la requête (D316) correspondante.

- **L'utilisateur :** La méthode *Deliver* est instanciée par le *AccessProvider* afin de délivrer à l'utilisateur un certificat de confiance T316 puisque le site source a été reconnu par la cible comme étant une entité digne de confiance parce que :

  - la source est membre du T_OS de la cible.

  - ou la cible a reçu un R316 contenant la chaîne de confiance permettant d'évaluer la source.

  Après avoir généré le T316 de l'utilisateur nomade, la méthode *Deliver* envoie ce certificat en faisant appel (par RMI) à la méthode *Response* de l'interface Utilisateur.

- **Les sites membres du T_IS :** L'interface du service de ces sites est instanciée dans le cas d'une requête de révocation. Cette requête est soit initiée par l'administrateur soit retransmise par le service à travers la méthode *in_Fwd* (voir étape 7).

10.4 Apports et limites du système Chameleon

Afin de valider la pertinence du système Chameleon, nous allons voir, dans ce qui suit, en quoi la mise en œuvre du système Chameleon répond aux besoins définis dans le chapitre 2.

B1 -Décentralisation- : Dans le système Chameleon, les différents sites composant l'environnement pervasif ne partagent pas une politique centralisée. Au contraire, chaque administration gère sa propre politique de confiance et définit également sa propre politique de mapping. Ceci gère de manière décentralisée la politique de sécurité de l'environnement.

B2 -Interopérabilité- : Afin d'interconnecter les différentes administrations, nous avons implémenté le modèle de confiance T2D. Il permet de représenter l'environnement comme un réseau pair à pair où chaque nœud du réseau (administration) peut communiquer et s'interconnecter avec des nœuds distants à travers ses voisins.

B3 -Gestion de la confiance- : Le fondement de base du système Chameleon est la confiance. A travers ce concept, nous avons implémenté une politique de sécurité qui prend en charge la transitivité de la confiance ainsi que l'aspect subjectif dans le processus d'évaluation de la confiance (disposition à la confiance). Cette mise en place permet à l'administrateur de définir une politique de sécurité étendue capable de rechercher et d'attribuer automatiquement des droits d'accès aux utilisateurs nomades jugés comme étant dignes de confiance. Cependant, dans l'état

actuel du système, le modèle de confiance ne permet pas de faire varier les droits d'accès qui sont accordés aux utilisateurs nomades en fonction de l'évaluation de leur contexte (dispositif, médium de communication, etc.).

B4 -Traçabilité et non répudiation- : Toutes les transactions effectuées par notre système sont réalisés via des certificats. Le modèle de certificat utilisé est le X316 ; il permet de gérer la non-répudiation du signataire mais également tous les sites qui ont collaboré à l'évaluation de la source (chaîne de confiance).

B5 -Autonomie- : Le système Chameleon fonctionne principalement en mode *push* en délivrant à l'utilisateur un certificat d'attributs dénotant son identité et tous ses droits. Ceci permet d'attribuer une autonomie certaine à l'usager puisque ce dernier a connaissance de tous ses privilèges et peut aussi s'identifier et obtenir des accès de manière totalement autonome.

B6 -Transparence et proactivité- : Les certificats T316 peuvent être générés de manière dynamique sans ré-authentifier leur propriétaire étant donné que les identifiants sont les mêmes pour tous les certificats T316 issus du même H316. La fiabilité de ces derniers n'est pas remise en cause car un certificat (H/T 316) doit, de toutes les manières, être authentifié au moment de son utilisation. Ceci permet au dispositif nomade de procéder à la récupération de certificats valides de manière transparente. Cela procure une certaine proactivité, étant donné qu'avant même que l'utilisateur nomade ne sollicite l'accès à une ressource ou à un service, le dispositif a déjà préalablement récupéré le certificat nécessaire.

B7 -Flexibilité- : Le certificat X316 nous a permis de définir toutes les attestations (certificats utilisateur, certificat d'exploration et certificat de révocation) nécessaires pour le fonctionnement de notre service Chameleon. Ainsi, la flexibilité et la généricité de la structure X316 permet d'exprimer plusieurs types d'attestation tout en gardant la même syntaxe. D'un autre coté, la pluralité des identifications embarquées dans le certificat permet à son propriétaire de l'utiliser, à partir divers terminaux en fonction de leurs capacités et du contexte d'utilisation (ex : accès à partir d'un automate).

B8 -Protection de la vie privée- : Dans le chapitre 8, nous avons défini la signature FeMoS, permettant de doter les certificats utilisés par le système Chameleon d'une fonction de masquage qui offre la possibilité de protéger certains attributs (ex : les profils de l'utilisateurs) présents dans le certificat. Outre le certificat X316, la signature FeMoS peut s'appliquer sur n'importe quel fichier, offrant ainsi un outil permettant de protéger l'identité, les attributs et les données signées d'un utilisateur contre le problème de divulgation non souhaitée.

B9 -Passage à l'échelle- : l'interconnexion pair à pair des différentes organisations par la confiance permet de mettre en place une politique de sécurité collaborative à large échelle. Ainsi, le réseau généré offre à tous les utilisateurs membres des différentes organisations un accès plus étendu et contrôlé aux différentes ressources et services déployés par les différents sites. Cependant, des algorithmes de routage plus optimisés que ceux utilisés par le simulateur (voir annexe C), peuvent êtres développés afin de permettre d'exploiter la méthode de calcul du seuil de méfiance transitive, ainsi qu'une meilleure diffusion de l'information afin de minimiser le trafic réseau.

10.5 Conclusion

Dans ce chapitre nous avons montré la richesse de notre approche grâce à l'intégration de nos précédentes contributions, notamment le modèle de certification (X316) et le modèle de confiance (T2D).

Le déploiement de notre système Chameleon se fait de manière aisée sous forme de service. Ce dernier est configuré par l'administrateur à travers l'interface de contrôle ; ainsi une fois lancé, celui-ci devient autonome, capable de gérer aussi bien les utilisateurs nomades que le flux de données généré par les autres services qui s'exécutent sur le réseau de confiance.

Cependant, il nous reste à mettre en place la fonction de calcul du chemin de référence θP^m illustrée dans le chapitre 5 ainsi que la mise en place d'une politique de mapping plus évoluée (voir la section perspectives du chapitre suivant).

Dans le cadre du projet *GreenNet* [17] le service Chameleon a été utilisé afin de permettre, de manière sécurisée, le déploiement d'autres services de la grille sur les différents serveurs la composant. En effet, dans le contexte du projet, il est important de choisir un serveur digne de confiance (la cible) acceptant d'accueillir le service d'un utilisateur accédant à partir d'un autre serveur (la source).

Tout au début de la thèse, nous avons introduit notre système Chameleon qui a constitué le point de départ de toutes les réflexions qui ont fait ce manuscrit. Nous avons conclu notre travail de thèse par la mise en œuvre de ce même système qui constitue le point de convergence englobant toutes les contributions définies dans les parties précédentes.

Dans le chapitre suivant nous allons présenter notre conclusion générale et nous finirons par introduire les principales perspectives découlant des différents contributions de la thèse.

17. http ://www.ens-lyon.fr/LIP/RESO/Projects/GREEN-NET/

Chapitre 11

Conclusion générale et perspectives

Sommaire

11.1 Conclusion

La mobilité de l'utilisateur et la multiplication des dispositifs légers et autonomes nous a motivés dans l'élaboration de notre système Chameleon.

Le système Chameleon permet d'interconnecter et de faire collaborer, de manière pair à pair, les politiques de sécurité des différents sites composant un environnement pervasif. Il se déploie sur chaque site sous forme d'un service composé de modules effectuant l'authentification et le contrôle d'accès des utilisateurs nomades.

Ces modules s'appuient principalement sur un modèle de confiance T2D et un modèle de certificat X316 utilisant la signature morph FeMoS.

Le modèle de confiance T2D, défini en chapitre 5, se base sur la disposition à la confiance de chaque site afin d'évaluer de manière automatique son cercle de confiance et de calculer puis de valider les chaînes de confiance.

La disposition à la confiance (personnalité) a été modélisée grâce à la fonction quadratique (de degré 2) de Bézier. Cependant, cette dernière a été transformée d'une équation paramétrée en une équation cartésienne ce qui nous a permis de représenter la personnalité par des courbes allant d'une forme concave à une forme convexe représentant ainsi la variation du caractère d'un type confiant à un type méfiant.

Le modèle T2D a été implémenté pour les réseaux pervasifs mais a été également utilisé en environnement de Grille dans le cadre du projet *GreenNet*. La définition de notre modèle est assez flexible et peut ainsi convenir a tout déploiement en réseau de type pair à pair.

Le certificat X316, qui représente le second pilier du système Chameleon, est un certificat générique totalement décrit en format XML. Il permet de définir aussi bien un certificat d'identité grâce à la partie HEADER qu'un certificat d'attributs via la partie PERMISSION. La partie IDENTIFICATION permet de définir non pas un seul identifiant (ex. clé publique) mais plusieurs. Cette décomposition en trois catégories permet au propriétaire d'indexer le certificat et de retrouver plus facilement les attributs nécessaires pour effectuer ses transactions (ex : identification, autorisation, etc.). Le système Chameleon bénéficie de la généricité du certificat X316 en l'utilisant pour implémenter toutes les attestations signées et échangées entre les différents services Chameleon déployés dans l'environnement. Cependant, la conception du système Chameleon pourraît être adaptée afin d'intégrer des certificats X509 surchargés par certains attributs dans la partie extension existante dans les certificats X509.

Le certificat X316 intègre la signature morph FeMoS autorisant son propriétaire (Subject) à masquer certaines parties, jugées confidentielles, sans invalider la signature. La signature FeMoS est une signature XMLDSig utilisant notre algorithme de transformation ($MorphBodyTransform$). On peut noté ici la généricité de notre signature. En effet, elle peut être calculée sur n'importe quel type de fichier en offrant la possibilité de masquer certaines parties sans altérer sa structure.

Cependant, le masquage des parties dynamiques par leur valeur de hash peut être vulnérable quand elles sont de petite taille. Dans ce cas, une méthode de reverse engineering peut exécuter une attaque par dictionnaire ou à texte brut et retrouver les parties masquées (hashées) en un laps de temps acceptable. Cette contrainte n'a pas été traitée dans le cadre de la thèse par manque de temps mais sera intégrée prochainement. (une piste sera présentée dans la section perspectives 11.2.3)

La politique de mapping implémentée dans la thèse est basique. Elle consiste en une mise en correspondance manuelle, par les administrateurs, des profils locaux avec tous les profils des sites de confiance. Cette procédure présente un certain nombre d'inconvénients :

- Complexité du mapping : Selon le nombre des profils à mapper ainsi que la taille du cercle de confiance, la mise en correspondance peut s'avérer complexe posant ainsi un problème de passage à l'échelle.

- Confidentialité : Pour le bon fonctionnement de cette approche, chaque site doit exposer le maximum d'informations sur ses profils locaux. Ceci peut induire des problèmes de confidentialité.

11.2 Perspectives

11.2.1 Le système Chameleon

Jusqu'à présent, nous avons défini une architecture qui permet à des sites au sein d'environnements distribués et collaboratifs d'interfacer leur politique de sécurité. Ceci permet aux

utilisateurs nomades de se déplacer et d'accéder aux ressources environnantes dans un environnement inconnu.

Néanmoins, la philosophie de l'informatique pervasive tend à créer un monde où l'usager est en parfaite osmose avec son environnement. Autrement dit, l'utilisateur et son environnement doivent interagir mutuellement.

La première étape consiste à developer le mécanisme de négociation entre l'utilisateur et le service Chameleon. En effet, le certificat X316 permet, à son propriétaire, de sélectionner les informations qui sont nécessaires pour effectuer une transaction particulière. Par contre, aucun mécanisme n'a été implémenté afin de permettre au dispositif de l'usager d'analyser le contexte et de sélectionner les bons attributs de manière automatique. Afin d'effectuer cette tâche, nous allons étudier en perspective la possibilité d'utiliser les solutions de négociation proposées par TrustBuilder[Seam 04] et Trust-X [Bert 04] couplées à des méthodes d'apprentissages afin de doter les dispositifs des usagers d'une certaine proactivité.

En seconde étape, nous envisageons d'étendre le système Chameleon afin d'offrir aux utilisateurs nomades la possibilité d'adapter la politique de sécurité de leurs dispositifs selon les contraintes organisationnelles, structurelles et temporelles de l'environnement local.

Par exemple, l'utilisateur "Bob" accède à un site inconnu et acquiert un nouveau profil grâce à la politique de confiance du système Chameleon. Réciproquement, *Bob* veut partager certains services et ressources qu'il a en sa possession avec les usagers locaux. Le problème qui se pose est le suivant :

Comment Bob peut-il contrôler, filtrer et personnaliser la politique de sécurité de ses dispositifs en fonction du contexte environnant ?

Ainsi, nous avons identifié trois contraintes :

- La politique dé sécurité utilisée par les dispositifs des usagers nomades est différente des environnements visités. Par exemple, la politique de contrôle d'accès de l'ordinateur portable d'un visiteur est configurée pour donner un certain nombre de privilèges aux personnes ayant comme rôle “Professeur”. Cependant, l'identifiant correspondant au rôle professeur diffère d'une politique à une autre, Il peut être noté “Teacher”, “Enseignant” ou “Professeur”, ce qui rend difficile voire impossible d'identifier ce rôle dans n'importe quel contexte.

- Pour un utilisateur nomade, les environnements visités lui sont totalement inconnus d'un point de vue organisationnel (ex : RBAC, DAC, etc.) et structurel (ex : salle 220, bât Blaise Pascal, etc.).

- L'adaptation de la politique des dispositifs des utilisateurs nomades doit être transparente et intuitive et prendre en considération les connaissances et le contexte de l'usager.

Dans le but de répondre à ces contraintes et pour mettre en place une politique de sécurité souple et dynamique adaptable au contexte, trois types d'adaptation de politiques peuvent être développées, comme suit :

– Adaptation par mapping : Cette option offre à l'utilisateur la possibilité d'utiliser les mêmes règles de contrôle d'accès définies préalablement dans les dispositifs. Par conséquent, le mapping doit se faire sur les règles (par exemple sur les rôles) dans le but d'obtenir une politique capable d'interagir avec le nouvel environnement.

– Adaptation par substitution : Dans ce cas de figure, l'environnement fournit à l'utilisateur un ensemble de règles pré-définies pour ses besoins. Par exemple : lorsque l'usager Bob souhaite partager un service X, le site accueillant peut lui proposer les règles d'un service local Y équivalent. Ceci permet à Bob de mettre en place, rapidement et à moindre coût, un contrôle d'accès adapté.

– Adaptation personnalisée : Cette adaptation nécessite une interface interactive permettant de générer de manière intuitive un ensemble de règles adaptées à la fois aux besoins de l'utilisateur et à la politique locale. En effet, l'interface peut être composée par des services responsables de décrire l'environnement. Comme le montre la figure 11.1, un service organisationnel peut représenter l'organisation de la politique de sécurité sous forme d'arborescence. Ainsi, l'usager est guidé dans l'expression de ses besoins, en sélectionnant les attributs nécessaires pour définir des règles de contrôle d'accès adéquates. Un autre service de localisation peut schématiser la structure du site ou des locaux en arborescence ou bien sous forme de carte, où, là aussi, il suffit de sélectionner la zone voulue pour récupérer son nom et ses cordonnées. Le système peut également utiliser les connaissances des usagers. Par exemple : Bob est un usager nomade qui veut partager ses ressources uniquement avec les utilisateurs locaux ayant comme profil celui de son amie Alice. Ceci peut être effectué par le service organisationnel, qui se chargera de retrouver le profil correspondant en faisant une recherche sur le nom.

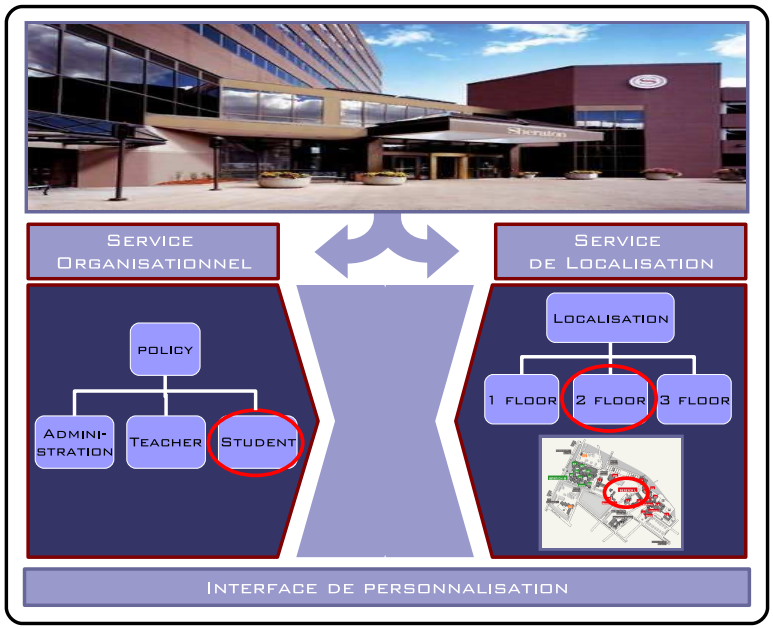

Figure 11.1 – Adaptation personnalisée

Ces adaptations peuvent être implémentées plus facilement dans une organisation de type

OrBAC[Kala 03]. En effet, le modèle OrBAC définit deux aspects importants pour notre approche à savoir la gestion du contexte et la hiérarchisation des rôles, permettant ainsi le développement de services capables de représenter les dimensions organisationnelle et contextuelle d'une organisation (figure 11.1).

11.2.2 La politique de confiance

Dans le modèle T2D, deux aspects restent à approfondir. Le premier aspect concerne la politique à suivre afin de sélectionner la chaîne de confiance (parmi les résultats possibles) qui correspond au mieux aux exigences et à la disposition à la confiance de chaque site. Le second aspect est relatif à la prise en compte de plusieurs variables comportementales. Dans le modèle T2D seule la confiance a été évaluée, d'autres variables pourraient l'être aussi telles que la méfiance, la certitude, etc.

L'étape suivante du développement de notre modèle de confiance consistera à améliorer la politique de mise en correspondance afin d'attribuer aux usagers nomades considérés comme dignes de confiance, un profil analogue qui reflète :

1. la confiance qui est accordée au site source (i.e. l'évaluation de la chaîne de confiance),
2. la confiance que le site source accorde au profil du visiteur,
3. la confiance qui peut être octroyée à la fiabilité du contexte de ce visiteur, par exemple : le médium de communication employé, le niveau de sécurité supporté par le dispositif utilisé, etc.

Ainsi, comme perspective, nous prévoyons d'explorer une méthode qui permet d'exploiter le $TrustSort$ (voir chapitre 5) afin de représenter dans un espace multidimensionnel l'organisation (ex : les rôles) de chaque site (voir figure 11.2).

L'approche sur laquelle nous travaillons consiste à définir plusieurs axes où chacun d'eux représente une activité particulière. Sur chacune des ces activités, un processus de tri ($Trustsort$) sur les profils projetés peut être appliqué, car les profils qui partagent cette activité ont le même objectif et vont donc vouloir accéder à un environnement commun et solliciter des ressources similaires. De ce fait, un profil ou un rôle ne sont plus décris par un intitulé, mais par un ensemble de coordonnées qui représentent le rang du profil dans chaque activité.

Exemple : Dans la figure 11.2, au sein du domaine universitaire, on peut distinguer trois activités différentes : activité éducative (ex : enseignement), activité sociale (ex : restauration) et activité administrative (ex : gestion des ressources humaine). Dans cet exemple, le rôle $Professeur$ est représenté par les coordonnées $(3, 0, 1)$. Ce qui correspond à 3 sur l'axe social (le professeur est le mieux payé et donc le mois avantagé sur cette activité), 0 sur l'axe $Administratif$ (Le professeur n'exerce pas d'activité administrative) et 1 sur l'axe $Educatif$ (le professeur est le responsable de cette activité ce qui le met au premier rang).

De cette manière, la mise en correspondance n'est plus implémentée entre les profils mais entre les intitulés des axes. Ceci permet de sauvegarder la confidentialité de l'organisation interne de chaque administration en ne divulguant que les activités.

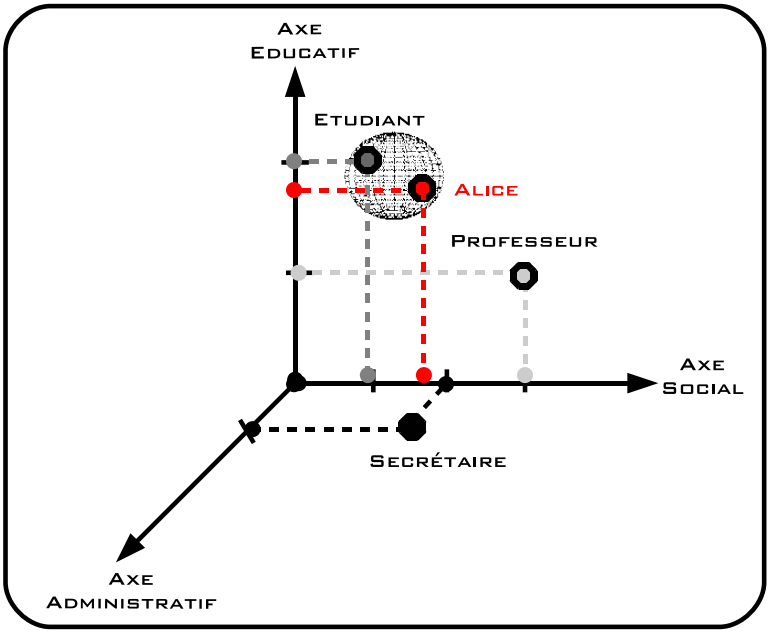

Figure 11.2 – Représentation multidimensionnelle d'une organisation

Dans le cas où la politique de confiance considère le site source de l'utilisateur comme étant un site de confiance, il faudra alors projeter les coordonnées du profil d'origine sur les axes mis en correspondance par la chaîne de confiance afin de représenter le profil de l'usager nomade dans l'espace multidimensionnel représentant l'organisation du site cible. Ainsi, l'usager nomade pourra recevoir un profil analogue correspondant au profil local le plus proche, par exemple, en terme de distance euclidienne, ou bien un profil analogue composé des profils supérieurs ou égales sur chacun des axes, afin de préserver le minimum de privilèges.

Ainsi, comme le montre la figure 11.2, en projetant le profil du visiteur Alice dans l'espace représentant la politique locale de la cible, la politique locale lui attribue le plus proche profil en termes de distance euclidienne à savoir le profil *Etudiant*, ou bien le profil analogue composé ayant les coordonnés (2 (Etudiant), 2 (Secrétaire),0).

Cette méthode pourrait également prendre en considération les évaluations du site source de l'utilisateur et de son contexte en pondérant les coordonnées par un coefficient (proportionnel à l'évaluation) permettant ainsi de déplacer le profil dans l'espace afin d'augmenter ou de diminuer les droits d'accès.

11.2.3 La signature FeMoS

Afin d'empêcher le reverse engineering de retrouver les parties dynamiques masquées, nous allons nous inspirer de l'état de l'art (voir chapitre 6) en les brouillant avant d'appliquer la fonction de hash. Ainsi, dans la prochaine évolution de la signature FeMoS, nous contant traiter cette contrainte en générant pour chaque partie dynamique une clé de session qui sera utilisée pour la chiffrer avant de la hasher. Ce mécanisme rend alors le reverse engineering plus complexe.

La signature FeMoS se base principalement sur le morph template afin de procéder au calcul et à la vérification d'une signature. Dans les chapitres 8 et 9 seul le template binaire a été défini. Ce dernier permet de délimiter des parties dynamiques en utilisant la position physique en octet dans le fichier. Cependant, afin de proposer une plus grande flexibilité au procédé de signature, il serait intéressant de développer, pour chaque type de média, une classe de template utilisant une interface de description spécifique, comme suit :

- Les templates pour le texte : la définition des parties dynamiques peut être plus pertinente en utilisant les expressions régulières afin de rechercher de manière sémantique le texte qui peut être masqué.

- Les templates pour l'image : la délimitation des zones dynamiques dans une image doit se faire de préférence de manière visuelle. Ceci nécessite de définir un certain ensemble d'objets qui permettent la sélection (ex : rectangle, cercle, etc.).

- Les templates pour l'audio et la vidéo : pour ce type de média il est plus intéressant de délimiter les parties dynamique en fonction du temps (ex : heure de début, heure de fin).

Le mot de la fin

Le travail effectué durant la période de la thèse m'a permis d'approfondir mes connaissances en sécurité informatique et ainsi pouvoir apporter ma modeste contribution à la recherche scientifique dans les domaines touchant aux notions de confiance, d'authentification, de contrôle d'accès, de certification et de signature (voir mes contributions bibliographiques en annexe A). Ce travail de thèse servira de matière première pour aller plus loin en développant les perspectives citées précédemment.

"Nous allons de commencement en commencement par des commencements qui n'ont pas de fin" *St Gregoire de Nysse.*

To be continued...

Bibliographie

[Abdu 97] A. Abdul-Rahman and S. Hailes. "A distributed Trust Model". In : *NSPW : New Security Paradigms Workshop*, pages. 48–60, ACM Press, New York, USA, 1997.

[Abi 06] D. Abi Haidar, N. Cuppens-Boulahia, F. Cuppens, and H. Debar. "An extended RBAC profile of XACML". In : *SWS '06 : 3rd ACM workshop on Secure web services*, pages. 13–22, ACM Press, New York, USA, 2006.

[Abob 04] B. Aboba, J. Vollbrecht, J. Carlson, and H. Levkowetz. "RFC3748 : Extensible Authentication Protocol (EAP)". RFC Editor, United States, 2004.

[Abra 66] P. W. Abrahams, J. A. Barnett, E. Book, D. Firth, S. L. Kameny, C. Weissman, L. Hawkinson, M. I. Levin, and R. A. Saunders. "The LISP 2 programming language and system". In : *AFIPS : fall joint computer conference*, pages. 661–676, ACM Press, New York, USA, 1966.

[Al K 02] M. A. Al-Kahtani and R. Sandhu. "A Model for Attribute-Based User-Role Assignment". In : *ACSAC : Annual Computer Security Applications Conference*, pages. 353–362, IEEE Computer Society, Washington, DC, USA, 2002.

[Alfi 04] R. Alfieri, R. Cecchini, V. Ciaschini, L. dell'Agnello, Á. Frohner, A. Gianoli, K. Lörentey, and F. Spataro. "VOMS, an Authorization System for Virtual Organizations". In : *European Across Grids Conference*, pages. 33–40, LNCS 2970, Springer-Verlag, Spain, 2004.

[Alme 04] F. Almenarez, A. Marin, C. Campo, and C. Garcia-Rubio. "PTM : A Pervasive Trust Management Model for Dynamic Open Environments". In : *First Workshop on Pervasive Security, Privacy and Trust PSPT'04 in conjuntion with Mobiquitous*, Boston, MA, USA, 2004.

[Alsa 06] M. Alsaleh and C. Adams. "Enhancing Consumer Privacy in the Liberty Alliance Identity Federation and Web Services Frameworks". In : *Workshop on Privacy Enhancing Technologies*, pages. 59–77, LNCS 4258, Springer-Verlag, Cambridge, UK, 2006.

[Arda 06] C. A. Ardagna, E. Damiani, S. D. C. di Vimercati, F. Frati, and P. Samarati. "CAS++ : An Open Source Single Sign-On Solution for Secure e-Services". In : *21st International Information Security Conference*, pages. 208–220, LNCS 201, Springer-Verlag, Karlstad, Sweden, 2006.

[Arin 06] R. Aringhieri, E. Damiani, S. D. C. Di Vimercati, S. Paraboschi, and P. Samarati. "Fuzzy techniques for trust and reputation management in anonymous peer-to-peer systems : Special Topic Section on Soft Approaches to Information Retrieval and Information Access on the Web". *JASIST : Journal of the American Society for Information Science and Technology*, Vol. 57, No. 4, pages. 528–537, 2006.

[Band 02] O. L. Bandmann, B. S. Firozabadi, and M. Dam. "Constrained Delegation". In : *IEEE Symposium on Security and Privacy*, page. 131, IEEE Computer Society, Oakland, California, USA, 2002.

[Bell 04] A. Belleil and D. Kaplan. "Confiance et sécurité sur les réseaux". Tech. Rep., la Fing : Fondation Internet Nouvelle Génération, 2004.

[Bern 05] T. Berners-Lee, R. Fielding, and L. Masinter. "RFC3986 : Uniform Resource Identifier (URI) : Generic Syntax". United States, 2005.

[Bert 01] E. Bertino, P. A. Bonatti, and E. Ferrari. "TRBAC : A temporal role-based access control model". *ACM Transactions on Information and System Security*, Vol. 4, No. 3, pages. 191–233, 2001.

[Bert 04] E. Bertino, E. Ferraria, and A. Squicciarin. "Trust-X : A Peer-to-Peer Framework for Trust Establishment". *IEEE Transactions on Knowledge and Data Engineering*, Vol. 16, No. 7, pages. 827–842, 2004.

[Beth 94] T. Beth, M. Borcherding, and B. Klein. "Valuation of trust in open networks". In : *The European Symposium on Research in Computer Security*, pages. 1–18, LNCS 875, Springer-Verlag, London, UK, 1994.

[Bird 07] N. Bird, C. Conrado, J. Guajardo, S. Maubach, G. J. Schrijen, B. Skoric, A. M. H. Tombeur, P. Thueringer, and P. Tuyls. "ALGSICS - Combining Physics and Cryptography to Enhance Security and Privacy in RFID Systems". In : *Fourth European Workshop on Security and Privacy in Ad hoc and Sensor Networks*, pages. 187–202, LNCS 4572, Springer-Verlag, Cambridge, UK, 2007.

[Bran 02] S. Brands. "A technical Overview of Digital Credentials". Tech. Rep., white paper in credentica.com, 2002.

[Butl 84] J. Butler and R. Cantrell. "A behavioral decision theory approach to modeling dyadic trust in superiors and subordinates". *Psychological reports*, Vol. 55, No. 1, pages. 19–28, 1984.

[Call 07] J. Callas, L. Donnerhacke, H. Finney, D. Shaw, and R. Thayer. "RFC4880 : OpenPGP Message Format". United States, 2007.

[Camp 02] R. Campbell, J. Al-muhtadi, P. Naldurg, G. Sampemane, and M. D. Mickunas. "Towards security and privacy for pervasive computing". In : *International Symposium on Software Security*, pages. 1–15, LNCS 2609, Springer-Verlag, Tokyo, Japan, 2002.

[Cart 09] O. Carton. "Courbes de Bézier et B-splines". `http://www.liafa.jussieu.fr/~carton/Enseignement/InterfacesGraphiques/MasterInfo/Cours/Swing/splines.html`, Consulté le 20.01.2009.

[Chad 03] D. W. Chadwick and A. Otenko. "The PERMIS X.509 role based privilege management infrastructure". *Future Generation Computer Systems Journal*, Vol. 19, No. 2, pages. 277–289, 2003.

[Chad 96] D. W. Chadwick. *Understanding X.500 (The Directory)*. International Thompson Publishing, London, 1996.

[Chan 01] K. Chan, J. Seligson, D. Durham, S. Gai, K. McCloghrie, S. Herzog, F. Reichmeyer, R. Yavatkar, and A. Smith. "RFC3084 : COPS Usage for Policy Provisioning (COPS-PR)". United States, 2001.

[Char 05] M.-H. Charki. "Le paradoxe de la confiance initiale". In : *Conférence Internationale de management stratégique*, Angers, France, 2005.

[Chen 03] R. Chen and W. Yeager. “Poblano : A Distributed Trust Model for Peer-to-Peer Networks”. Tech. Rep., Sun Microsystems, 2003.

[Cole 90] J. S. Coleman. *Foundations of Social Theory.* The Belknap Press of Harvard University, 1990.

[Coma 08] C. Coma, N. Cuppens-Boulahia, F. Cuppens, and A.-R. Cavalli. “Context Ontology for Secure Interoperability”. In : *ARES ’08 : Third International Conference on Availability, Reliability and Security*, pages. 821–827, IEEE Computer Society, Washington, DC, USA, 2008.

[Cupp 06] F. Cuppens and C. C. Nora Cuppens-Boulahia. “O2O : Managing Security Policy Interoperability with Virtual Private Organizations”. In : *ICISS 2006 : Second International Conference on Information Systems Security*, pages. 101–115, LNCS 4332, Springer-Verlag, Kolkata, India, 2006.

[Dier 99] T. Dierks and C. Allen. “RFC2246 : The TLS Protocol Version 1.0”. United States, 1999.

[Duss 98] S. Dusse, P. Hoffman, B. Ramsdell, L. Lundblade, and L. Repka. “RFC2311 : S/MIME Version 2 Message Specification”. United States, 1998.

[East 02] D. Eastlake, J. Reagle, and D. Solo. “RFC3275 : (Extensible Markup Language) XML-Signature Syntax and Processing”. United States, 2002.

[ECMA 06] ECMA International. “Standard ECMA-335 - Common Language Infrastructure (CLI)”. `http://www.ecma-international.org/publications/standards/Ecma-335.htm`, June 2006.

[Elli 99a] C. Ellison. “RFC2692 : SPKI Requirements”. United States, 1999.

[Elli 99b] C. Ellison, B. Frantz, B. Lampson, R. Rivest, B. Thomas, and T. Ylonen. “RFC2693 : SPKI Certificate Theory”. United States, 1999.

[Ferr 01] D. F. Ferraiolo, R. Sandhu, S. Gavrila, D. R. Kuhn, and R. Chandramouli. “Proposed NIST standard for role-based access control”. *ACM Transactions on Information and System Security*, Vol. 4, No. 3, pages. 224–274, 2001.

[Fost 01] I. Foster, C. Kesselman, and S. Tuecke. “The Anatomy of the Grid : Enabling Scalable Virtual Organizations”. *International Journal of High Performance Computing Applications*, Vol. 15, No. 3, pages. 200–222, 2001.

[Free 96a] N. Freed and N. Borenstein. “RFC2045 : Multipurpose Internet Mail Extensions (MIME) Part One : Format of Internet Message Bodies”. RFC Editor, United States, 1996.

[Free 96b] N. Freed and N. Borenstein. “RFC2046 : Multipurpose Internet Mail Extensions (MIME) Part Two : Media Types”. United States, 1996.

[Gian 07] P. D. Giang, L. X. Hung, S. Lee, Y.-K. Lee, and H. Lee. “A Flexible Trust-Based Access Control Mechanism for Security and Privacy Enhancement in Ubiquitous Systems”. In : *MUE ’07 : International Conference on Multimedia and Ubiquitous Engineering*, pages. 698–703, IEEE Computer Society, Washington, DC, USA, 2007.

[Glob 09] Globus Aliance. “`http://www.globus.org/`”. Consulté le 20.01.2009.

[GNU 09] GNU Privacy Guard. “`http://www.gnupg.org/`”. Consulté le 20.01.2009.

[Gold 88] S. Goldwasser, S. Micali, and R. L. Rivest. “A digital signature scheme secure against adaptive chosen-message attacks”. *SIAM Journal on Computing*, Vol. 17, No. 2, pages. 281–308, 1988.

[Gole 75] R. T. Golembiewski and M. McConkie. *The centrality of interpersonal trust in group processes*. G.L. Cooper, Wiley, London, 1975.

[Gran 85] M. Granovetter. "Economic action and social structure : the problem of embeddedness". *American Journal of Sociology*, Vol. 91, No. 3, pages. 481–510, 1985.

[Grid 09] GridShib. "`http://gridshib.globus.org/`". Consulté le 20.01.2009.

[Habe 08] S. Haber, Y. Hatano, Y. Honda, W. Horne, K. Miyazaki, T. Sander, S. Tezoku, and D. Yao. "Efficient signature schemes supporting redaction, pseudonymization, and data deidentification". In : *ASIACCS : The ACM symposium on Information, computer and communications security*, pages. 353–362, ACM Press, New York, NY, USA, 2008.

[Haid 07] D. A. Haidar, N. Cuppens, F. Cuppens, and H. Debar. "Resource Classification Based Negotiation in Web Services". In : *IAS '07 : Third International Symposium on Information Assurance and Security*, pages. 313–318, IEEE Computer Society, Washington, DC, USA, 2007.

[Harr 06] R. Harrison. "RFC4513 : Lightweight Directory Access Protocol (LDAP) : Authentication Methods and Security Mechanisms". United States, 2006.

[Hasa 08] O. Hasan, J. M. Pierson, and L. Brunie. "Access Control in Ubiquitous Environments Based on Subjectivity Eliminated Trust Propagation". In : *TRUST08 : International Symposium on Trustworthiness, Reliability, and services in Ubiquitous and Sensor neTworks, in conjunction with EUC 2008*, pages. 603–609, IEEE Computer Society, Shanghai,China, 2008.

[Hous 99] R. Housley, W. Ford, W. Polk, and D. Solo. "RFC2459 : Internet X.509 Public Key Infrastructure Certificate and CRL Profile". United States, 1999.

[ITU 02] ITU-T. "X.680 : OSI networking and system aspects – Abstract Syntax Notation One (ASN.1)". 2002.

[ITU 88] ITU-T. "X.208 Specification of Abstract Syntax Notation One (ASN.1)". 1988.

[Izu 07] T. Izu, N. Kanaya, M. Takenaka, and T. Yoshioka. "PIATS : A Partially Sanitizable Signature Scheme". In : *ICICS : International Conference on Information and Communications Security*, pages. 72–83, LNCS 3783, Springer-Verlag, Zhengzhou, Henan Province, China, 2007.

[John 02] R. Johnson, D. Molnar, D. X. Song, and D. Wagner. "Homomorphic Signature Schemes". In : *CT-RSA '02 : The Cryptographer's Track at the RSA Conference on Topics in Cryptology*, pages. 244–262, LNCS 2271, Springer-Verlag, London, UK, 2002.

[Josa 05] A. Jøsang and S. Pope. "Semantic constraints for trust transitivity". In : *APCCM : 2nd Asia-Pacific conference on Conceptual modelling*, pages. 59–68, Australian Computer Society, Inc., Newcastle, New South Wales, Australia, 2005.

[Josh 03] J. B. D. Joshi, B. Shafiq, A. Ghafoor, and E. Bertino. "Dependencies and separation of duty constraints in GTRBAC". In : *SACMAT : The eighth ACM symposium on Access control models and technologies*, pages. 51–64, ACM Press, Como, Italy, 2003.

[Kala 03] A. Abou El Kalam, S. Benferhat, A. Miège, R. E. Baida, F. Cuppens, C. Saurel, P. Balbiani, Y. Deswarte, and G. Trouessin. "Organization based access control". In : *POLICY '03 : 4th IEEE International Workshop on Policies for Distributed Systems and Networks*, page. 120, IEEE Computer Society, Washington, DC, USA, 2003.

[Kala 06] A. Abou El Kalam and Y. Deswarte. “Multi-OrBAC : un modèle de contrôle d’accès pour les systèmes multi-organisationnels”. In : *Third Conference on Security of Information Systems*, pages. 67–85, Seignosse, Landes, France, 2006.

[Kali 98] B. Kaliski. “RFC2315 : PKCS #7 : Cryptographic Message Syntax Version 1.5”. United States, 1998.

[Kuni 03] M. Kunihiko, S. Seiichi, I. Mitsuru, M. Tsutomu, and S. Ryoichi. “Digital Document Sanitizing Problem”. *IEIC Technical Report*, Vol. 103, No. 195, pages. 61–67, Institute of Electronics, Information and Communication Engineers, 2003.

[Larm 03] J. Larmouth and I. Member. “XML Common Biometric Format”. Tech. Rep., OASIS : Organization for the Advancement of Structured Information Standards, 2003.

[Lemo 07] LemonLDAP : :NG. “`http://wiki.lemonldap.objectweb.org`, (Consulté le 20.01.2009)”. 2007.

[Levi 09] R. Levien. “Advogato’s Trust Metric, `http://www.advogato.org/trust-metric.html`”. Consulté le 20.01.2009.

[Lewi 85] D. Lewis and A. Weigert. “Trust as a social reality”. *Sociales Forces*, Vol. 63, No. 4, pages. 967–985, 1985.

[Lisc 06] R. Liscano, A. K. UOIT, and U. O. Ottawa. “User-Credential Based Role Mapping in Multi-domain Environment”. In : *PST : Privacy, Security, Trust*, pages. 2–8, ACM Press, Markham, Ontario, Canada, 2006.

[Losc 98] P. A. Loscoccoa, S. D. Smalley, P. A. Muckelbauera, R. C. Taylor, S. J. Turner, , and J. F. Farrell. “The Inevitability of Failure : The Flawed Assumption of Security in Modern Computing Environments”. In : *The 21st National Information Systems Security Conference*, pages. 303–314, Arlington, Virginia, USA, October 1998.

[Luhm 88] N. Luhmann. “Familiarity, confidence, trust : Problems and alternatives”. *Trust, making and Breaking Cooperative relations*, pages. 94–107, D. Gambetta, Blackwell, New York, 1988.

[Mang 98] V. Mangematin. “La confiance : un mode de coordination dont l’utilisation dépend de ses conditions de production”. In : *Confiance et entreprise*, G. Morin, Paris, France, 1998.

[Mars 94] S. Marsh. *Formalising Trust as a Computational Concept*. PhD thesis, University of Stirling, Scotland, 1994.

[Maye 95] R. Mayer, J. Davis, and D. Schoorman. “An integration model of organizational trust”. *Academy of Management Review*, Vol. 20, No. 3, pages. 709–734, 1995.

[McKn 98] D. H. McKnight, L. L. Cummings, and N. L. Chervany. “Initial trust formation in new organizational relationships”. *Academy of Management Review*, Vol. 23, No. 3, pages. 473–490, 1998.

[Mend 05] F. A. Mendoza, A. M. López, C. Campo, and R. C. García. “TrustAC : Trust-Based Access Control for Pervasive Devices”. In : *International Conference on Security in Pervasive Computing*, pages. 225–238, LNCS 3450, Springer-Verlag, Boppard, Germany, 2005.

[Mend 06] F. A. Mendoza, A. M. López, D. Diaz, and J. Sanchez. “Developing a Model for Trust Management in Pervasive Devices”. In : *PerCom : Pervasive Computing and Communications Workshops*, pages. 267–271, IEEE Computer Society, Pisa, Italy, 2006.

[Mich 04] B. Michele, C. Dario, and C. Silvia. "Type based discretionary access control". In : *In CONCUR'04 : Concurrency Theory*, pages. 225–239, LNCS 3170, Springer-Verlag, London, UK, 2004.

[Miya 05] K. Miyazaki, M. Iwamura, T. Matsumoto, R. Sasaki, H. Yoshiura, S. Tezuka, and H. Imai. "Digitally Signed Document Sanitizing Scheme with Disclosure Condition Control". *IEICE Transactions on Fundamentals of Electronics, Communications and Computer Sciences*, Vol. 88-A, No. 1, pages. 239–246, 2005.

[Miya 06] K. Miyazaki, G. Hanaoka, and H. Imai. "Digitally signed document sanitizing scheme based on bilinear maps". In : *ASIACCS : The ACM Symposium on Information, computer and communications security*, pages. 343–354, ACM Press, New York, NY, USA, 2006.

[MONO 07] MONO. "`http://www.mono-project.com/`, (Consulté le 20.01.2009)". 2007.

[Neum 05] C. Neuman, T. Yu, S. Hartman, and K. Raeburn. "RFC4120 : The Kerberos Network Authentication Service (V5)". July 2005.

[Neve 04] V. Neveu. "La confiance organisationnelle : définition et mesure". In : *Congrès annuel de l'AGRH*, Montréal, Québec, Canada, septembre 2004.

[NS 3 09] NS-3 network simulator. "`http://www.nsnam.org/`". Consulté le 20.01.2009.

[OASI 05] OASIS : Organization for the Advancement of Structured Information Standards. "eXtensible Access Control Markup Language (XACML), `http://www.oasis-open.org/committees/tc_home.php?wg_abbrev=xacml#XACML20`, (Consulté le 20.01.2009)". 2005.

[OASI 06] OASIS : Organization for the Advancement of Structured Information Standards. "Web Services Security Specification (WS-Security, WS-Security 2004), `http://xml.coverpages.org/ws-security.html`, (Consulté le 20.01.2009)". 2006.

[OASI 07] OASIS : Organization for the Advancement of Structured Information Standards. "WS-Trust 1.3, `http://docs.oasis-open.org/ws-sx/ws-trust/200512`, (Consulté le 20.01.2009)". 2007.

[OASI 08] OASIS : Organization for the Advancement of Structured Information Standards. "Security Assertion Markup Language (SAML), `http://xml.coverpages.org/saml.html`, (Consulté le 20.01.2009)". 2008.

[OMNe 09] OMNeT++ Community Site. "`http://www.omnetpp.org/`". Consulté le 20.01.2009.

[Page 99] L. Page, S. Brin, M. Rajeevand, and W. Terry. "The PageRank Citation Ranking : Bringing Order to the Web". In : *7th International World Wide Web Conference*, pages. 161–172, Brisbane, Australia, 1999.

[Pash 03] A. Pashalidis and C. J. Mitchell. "A Taxonomy of Single Sign-On Systems". In : *8th Australasian Conference on Information Security and Privacy*, page. 219, LNCS 2727, Springer-Verlag, Wollongong, Australia, 2003.

[Pear 02] L. Pearlman, V. Welch, I. Foster, C. Kesselman, and S. Tuecke. "A Community Authorization Service for Group Collaboration". In : *POLICY '02 : Third International Workshop on Policies for Distributed Systems and Networks (POLICY'02)*, pages. 50–59, IEEE Computer Society, Washington, DC, USA, 2002.

[Poug 98] J. P. Pouget, G. Demengel, and P. Bezier. *Modèle de Bézier, des B.Splines et des NURBS*. Ellipses Paris, 1998.

[Reco 06] D. Recordon and D. Reed. "OpenID 2.0 : a platform for user-centric identity management". In : *Second ACM workshop on Digital identity management*, pages. 11–16, ACM Press, Alexandria, Virginia, USA, 2006.

[Rign 97a] C. Rigney. "RFC2139 : RADIUS Accounting". United States, 1997.

[Rign 97b] C. Rigney, A. Rubens, W. Simpson, and S. Willens. "RFC2138 : Remote Authentication Dial In User Service (RADIUS)". United States, 1997.

[Rive 97] R. Rivest. "S-Expressions draft, `http://people.csail.mit.edu/rivest/Sexp.txt`". 1997.

[Robi 94] S. Robinson and D. Rousseau. "Violating the psychological contract : not the exception but the norm". *Journal of Organizational Behavior*, Vol. 15, pages. 145–159, 1994.

[Roma 02] M. Román, C. Hess, R. Cerqueira, A. Ranganathan, R. H. Campbell, and K. Nahrstedt. "Gaia : A Middleware Infrastructure for Active Spaces". *IEEE Pervasive Computing*, Vol. 1, No. 4, pages. 74–83, IEEE Educational Activities Department, 2002.

[Rott 67] J. B. Rotter. "A new scale for the measurement of interpersonal trust". *Journal of Personality*, Vol. 35, No. 4, pages. 651–665, 1967.

[Sale 04] A. Saleem, M. Krznaric, J. Cohen, S. Newhouse, and J. Darlington. "Using the VOM Portal to Manage Policy within Globus Toolkit, Community Authorisation Service ICENI Resources". In : *UK e-Science All-Hands Meeting (AHM2004), September 2004, Nottingham, UK*, pages. 418–423, August 2004.

[Salo 01] T. Salo. "Security in Pervasive Computing". In : *Reasearch Seminar on Digital Media Pervasive Computing*, Helsinki University, Finland, 2001.

[Sand 99] R. Sandhu and Q. Munawer. "The ARBAC99 Model for Administration of Roles". In : *ACSAC : 15th Annual Computer Security Applications Conference*, pages. 229–238, IEEE Computer Society, Los Alamitos, CA, USA, 1999.

[Seam 04] K. Seamons. "TrustBuilder : Automated Trust Negotiation in Open Systems". In : *CERIAS Security Seminar*, Purdue University, 2004.

[Secu 09] SecuriteInfo. "La Biométrie". `http://www.securiteinfo.com/conseils/biometrie.shtml`, Consulté le 20.01.2009.

[Seit 05] L. Seitz, J. M. Pierson, and L. Brunie. "Sygn : A certificate based access control in Grid environments". Tech. Rep., INSA de Lyon, LIRIS Lab, 2005.

[Shib 09] Shibboleth Development Team. "Shibboleth Project, `http://shibboleth.internet2.edu/`". Consulté le 20.01.2009.

[Shoc 00] P. Shockley-Zalabak, K. Ellis, and G. Winograd. "Organizational trust : what it means, what it matters". *Organization Development Journal*, Vol. 18, No. 4, pages. 35–48, 2000.

[SICS 07] SICSACML. "XACML 3.0, `https://www.sics.se/node/2465`, (Consulté le 20.01.2009)". 2007.

[Simp 96] W. Simpson. "RFC1994 : PPP Challenge Handshake Authentication Protocol (CHAP)". United States, 1996.

[Stei 01] R. Steinfeld, L. Bull, and Y. Zheng. "Content Extraction Signatures". In : *ICISC : International Conference on Information Security and Cryptology*, pages. 285–304, LNCS 2288, Springer-Verlag, Seoul, South Korea, 2001.

[Step 08] K. Stephen and S. Stefan. "Public-Key Infrastructure (X.509) (pkix), http://www.ietf.org/html.charters/pkix-charter.html, (Consulté le 20.01.2009)". 2008.

[The 04] The DataGrid Project. "http://eu-datagrid.web.cern.ch/eu%2Ddatagrid/, (Consulté le 20.01.2009)". 2004.

[Theo 04] G. Theodorakopoulos and J. S. Baras. "Trust evaluation in ad-hoc networks". In : *3rd ACM workshop on Wireless security*, pages. 1–10, ACM Press, New York, NY, USA, 2004.

[Thom 99] M. Thompson, W. Johnston, S. Mudumbai, G. Hoo, K. Jackson, and A. Essiari. "Certificate-based access control for widely distributed resources". In : *SSYM : 8th conference on USENIX Security Symposium*, pages. 17–17, USENIX Association, Berkeley, CA, USA, 1999.

[Voll 00] J. Vollbrecht, P. Calhoun, S. Farrell, L. Gommans, G. Gross, B. de Bruijn, C. de Laat, M. Holdrege, and D. Spence. "RFC2904 : AAA Authorization Framework". USA, 2000.

[W3C 02] W3C. "XML Encryption Syntax and Processing, http://www.w3.org/TR/xmlenc-core/, (Consulté le 20.01.2009)". 2002.

[W3C 03] W3C. "XML Advanced Electronic Signatures (XAdES), http://www.w3.org/TR/xmlenc-core/, (Consulté le 20.01.2009)". 2003.

[W3C 06] W3C. "XHTML 2.0 Working Draft, http://www.w3.org/TR/xhtml2/, (Consulté le 20.01.2009)". 2006.

[W3C 07] W3C. "SOAP Version 1.2 Part 1 : Messaging Framework (Second Edition), http://www.w3.org/TR/soap12-part1/, (Consulté le 20.01.2009)". 2007.

[W3C 08] W3C. "XML Signature Syntax and Processing (Second Edition), http://www.w3.org/TR/xmldsig-core/, (Consulté le 20.01.2009)". 2008.

[Wahl 97] M. Wahl, S. Kille, and T. Howes. "RFC2253 : Lightweight Directory Access Protocol (v3) : UTF-8 String Representation of Distinguished Names". United States, 1997.

[Watt 06] J. P. Watt, O. Ajayi, J. Jiang, J. Koetsier, and R. O. Sinnott. "A Shibboleth-Protected Privilege Management Infrastructure for e-Science Education". In : *CCGRID : Sixth IEEE International Symposium on Cluster Computing and the Grid*, pages. 357–364, IEEE Computer Society, Washington, DC, USA, 2006.

[Weis 01] M. Weiser. "Whatever happened to the next-generation Internet ?". *Communications of the ACM*, Vol. 44, No. 9, pages. 61–68, 2001.

[Wiki 09a] Wikipedia. "Authentification unique". http://fr.wikipedia.org/wiki/Authentification_unique,, Consulté le 20.01.2009.

[Wiki 09b] Wikipedia. "Web of trust, http://en.wikipedia.org/wiki/Web_of_trust". Consulté le 20.01.2009.

[Will 93] O. Williamson. "Calculativeness, trust, and economic organization". *Journal of Law and Economics*, Vol. 36, No. 1, pages. 453–486, 1993.

[Xion 03] L. Xiong and L. Liu. "A reputation-based trust model for peer-to-peer ecommerce communities". In : *4th ACM conference on Electronic commerce*, pages. 228–229, 2003.

[Zahe 98] A. Zaheer, B. McEvily, and V. Perrone. "Does trust matter ? Exploring the effects of interorganizational and interpersonal trust on performance". *Organization Science*, Vol. 9, No. 2, pages. 141–159, 1998.

[Zand 72] D. E. Zand. "Trust and managerial problem solving". *Administrative Science Quarterly*, Vol. 17, pages. 229–239, 1972.

[Zieg 04] C.-N. Ziegler and G. Lausen. "Spreading Activation Models for Trust Propagation". In : *IEEE International Conference on e-Technology, e-Commerce and e-Service*, pages. 83–97, IEEE Computer Society, Taipei, Taiwan, 2004.

[Zimm 95] P. R. Zimmermann. *The official PGP user's guide*. MIT Press, Cambridge, MA, USA, 1995.

[Zuck 86] L. Zucker. "Production of trust : institutional sources of economic structure, 1840-1920". *Research in Organizational Behavior*, Vol. 8, pages. 53–111, 1986.

Annexe A

Mes contributions bibliographiques

A.1 Gestion de la confiance -Le modèle T2D-

1. R.Saadi and JM.Pierson and L. Brunie, Authentification par la méfiance dans les systèmes pervasifs, UbiMob : French-speaking conference on Mobility and ubiquity computing ACM (2005), 93-96.
2. R.Saadi and JM.Pierson and L. Brunie, (Dis)trust Certification Model for Large Access in Pervasive Environment, JPCC : Journal of Pervasive Computing and Communications 1(4) (2005), 289-299.
3. R.Saadi and O.Hasan and JM.Pierson and L. Brunie, Establishing Trust Beliefs based on a Uniform Disposition to Trust, SITIS : IEEE international Conference on Signal Image and Technologie Based Systems. Section : Web-Based Information Technologies and Distributed Systems IEEE Computer Society (2007), 221-228.

A.2 Certification et signature -Le certificat X316 et la signature FeMoS-

4. R.Saadi and JM.Pierson and L. Brunie, APC : Access Pass Certificate. Distrust Certification Model for Large Access in Pervasive Environment, ICPS : IEEE International Conference on Pervasive Services IEEE Computer Society (2005), 361-370.
5. R.Saadi and JM.Pierson and L. Brunie, A new Signature Framework for e-Documents in Pervasive Environment, ICPS 2007 IEEE International Conference in Pervasive Services. IEEE Computer Society (2007), 185-188.
6. R.Saadi and JM.Pierson and L. Brunie, A New Certificate Signature for Pervasive Environment, SecPerU : International Workshop on Security Privacy and Trust in Pervasive and Ubiquitous Environment, (Taux d'acceptation : 35%) IEEE Computer Society (2007), 43-48.
7. R.Saadi and JM.Pierson and L. Brunie, X316 Security Toolbox for New Generation of Certificate, TrustBus : International Conference on Trust, Privacy and Security in Digital Business, (Taux d'acceptation : 35%) Springer Verlag (Vol 4657/2007) (2007), 248-258.

8. R.Saadi and JM.Pierson and L. Brunie, Context Adapted Certificate Using Morph Template Signature for Pervasive Environment, UCS : International Symposium in Ubiquitous Systems, (Taux d'acceptation : 17%) Springer Verlag (Vol 4836/2008) (2007), 17-32.

A.3 Réseaux collaboratifs -Le système Chameleon-

9. R.Saadi and JM.Pierson and L. Brunie, Authentication and Access Control Using Trust Col laboration in Pervasive Grid Environment, GPC : International Conference on Grid and Pervasive Computing, (Taux d'acceptation : 28%) Springer Verlag (Vol 4459/2007) (2007), 348-361.
10. R.Saadi and JM.Pierson and L. Brunie, The Chameleon : A Pervasive Grid Security Architecture, ICNS : International Conference on Networking and Services, (Taux d'acceptation : 32%) IEEE Computer Society (2007), 48-54.
11. R.Saadi and JM.Pierson and L. Brunie, Security in distributed col laborative environments : limitations and solutions, Book Chapter on Emergent Web Intelligence (to be published) Springer Verlag (2009).

A.4 Algorithme génétique

12. A. Saadi and R.Saadi, La famille s'agrandit : naissance de notre fils Rayan, GNM : Grossesse Naissance Maternité, Centre Hospitalier de Mont de Marsan, France (05/04/2008 à 0h12).

Annexe B

Etude du comportement

La disposition à la confiance est la propension inhérente d'un individu à faire confiance ou à se méfier. Cette caractéristique est une propriété unique qui dépend directement de la personnalité de chaque entité. Celle-ci influe fortement sur la vision, de tout à chacun, à percevoir les différents niveaux de confiance.

Dans le but de modéliser la personnalité ou le comportement des entités vis-à-vis de la confiance, nous nous sommes intéressé à définir un schéma de fonctions. Il attribue à chaque comportement, une fonction mathématique qui prend en entrée une valeur de confiance neutre et fournis en sortie une évaluation pondérée par la personnalité correspondante à la fonction utilisée.

Afin de montrer l'impact de la personnalité dans l'appréciation et l'évaluation des actions et des relations entre les différents acteurs de l'environnement, nous avons réalisé un questionnaire comportant deux tests. Comme le montre la figure B.1, le premier test consiste à quantifier les appréciations (Excellent, Moyen et Mauvais) de stages sur une échelle de 0 à 10. Le second a pour but de montrer les différentes évaluations possibles, correspondantes au fait d'avoir en une personne : très confiance, confiance, moyennement confiance ou peu confiance sur une échelle de 0 à 20.

Le questionnaire à été diffusé sur le réseau universitaire. Le premier test a pour but d'introduire le second, puisqu'il concerne l'évaluation des stages d'étudiants qui constitue le quotidien de tout enseignant. La deuxième raison pour laquelle le premier test à été mis en place (malgré le fait qu'il ne concerne pas directement la confiance), est de montrer que le caractère influe de manière similaire quand il s'agit d'évaluer la confiance, le travail ou autre.

Afin de montrer l'impact de la disposition à la confiance dans l'évaluation, on a mis en place une représentation graphique en deux dimensions où l'axe des abscisses représente l'évaluation neutre et l'axe des ordonnées l'évaluation pondérée.

- L'évaluation neutre : est composée des mentions qui sont classées par ordre de mérite et positionnées de manière linéaire sur l'axe des abscisses comme suit :
  - Premier test : Evaluation des stages.
    - X=1 : Mauvais (Mv).
    - X=2 : Moyen (My).
    - X=3 : Excellent (Ex).

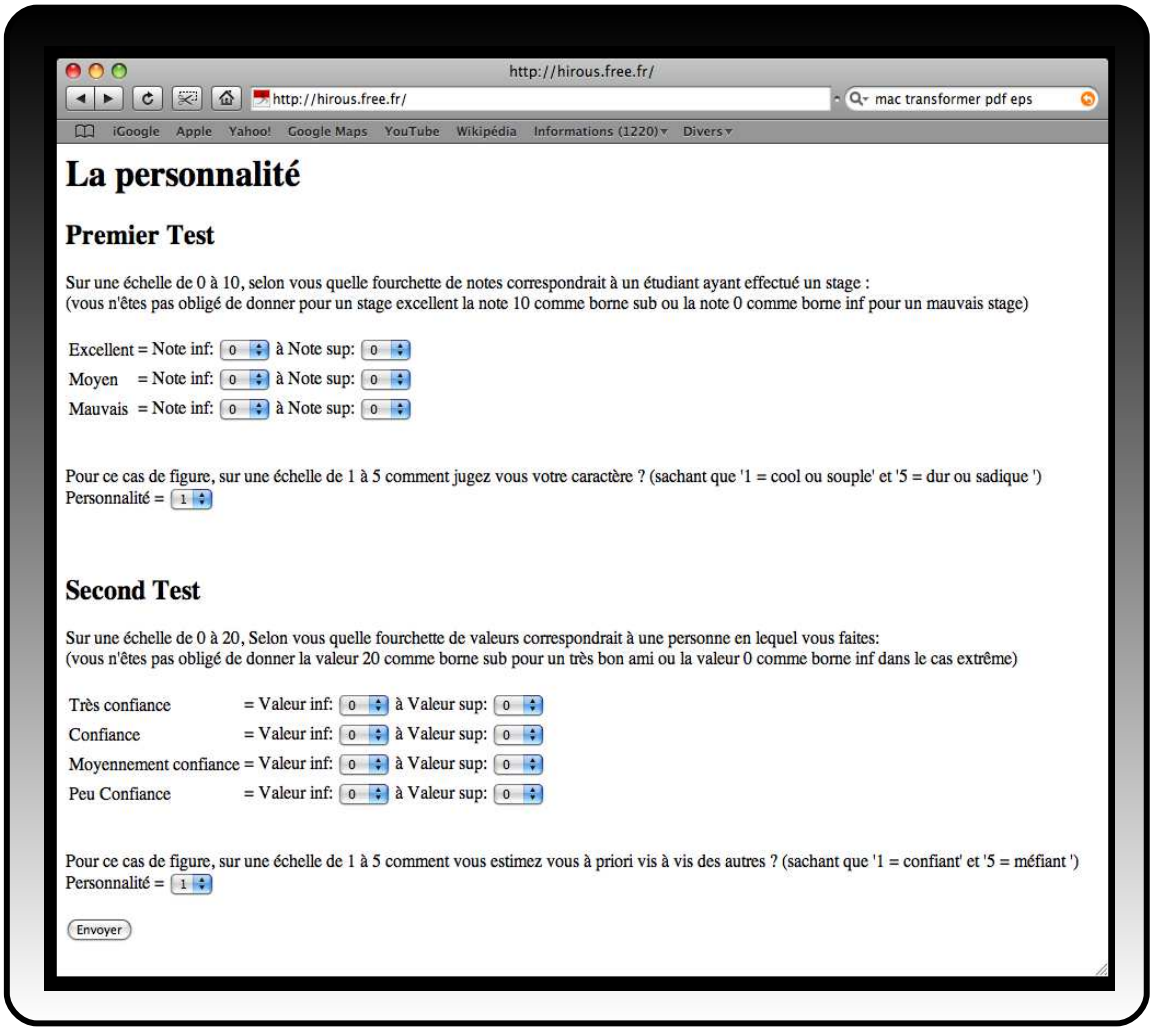

FIGURE B.1 – Questionnaire

  - Second test : Evaluation des personnes.
    - X=1 : Peu confiance (PC).
    - X=2 : Moyennement confiance (MC).
    - X=3 : Confiance (C).
    - X=4 : Très confiance (TC).

- L'évaluation pondérée : représente l'évaluation numérique propre à l'appréciation de chacun. Ainsi, pour toute évaluation neutre correspond sur l'axe des ordonnées la moyenne des bornes inférieures et supérieures introduites lors du questionnaire.

  - Premier test : Evaluation des stages.
    - X=0 => Y=0. (Stage nul)
    - X=1 : => Y= $Moyenne_{Mv}(sup, inf)$.
    - X=2 : => Y= $Moyenne_{My}(sup, inf)$.
    - X=3 : => Y= $Moyenne_{Ex}(sup, inf)$.
    - X=4 : 10.(Stage parfait)

- Second test : Evaluation des personnes.
  - X=0 : => Y= 0.(Absence de confiance)
  - X=1 : => Y= $Moyenne_{PC}(sup, inf)$.
  - X=2 : => Y= $Moyenne_{MC}(sup, inf)$.
  - X=3 : => Y= $Moyenne_{C}(sup, inf)$.
  - X=4 : => Y= $Moyenne_{TC}(sup, inf)$.
  - X=5 : => Y= 20.(Confiance absolue)

A l'issu du questionnaire, 195 réponses ont été enregistrées. Dans la figure B.2, sur chaque graphique les réponses sont représentées sous forme de courbes. Ce qui ressort est que lors du processus d'évaluation, la personnalité de chacune des personnes, ayant répondu au formulaire, apparaît le plus, lors de l'évaluation d'un stage mauvais ou d'une personne en laquelle en fait peu confiance. Par contre, pour l'évaluation d'un stage excellent ou d'une personne confiante le contraste n'est pas visible. Ce résultat peut être expliqué par la fait qu'un enseignant sévère ou peu confiant considère, comme toutes autres personnes, qu'un stage excellent mérite une excellente note (par exemple : 18). Cependant, la mention 'excellent' ne peut ou très difficilement être atteinte. Ce qui, dans la réalité, positionnera la majorité des entités évaluées sur une échelle bornée par une valeur inférieure à ce qui a été avancée lors du test.

En conséquence, dans une évaluation plus réaliste, le contraste visible au début des courbes doit également être visible à la fin des courbes. Ainsi, l'ensemble des comportements peut être compris entre la courbe bleue (qui représente les personnes ayant un caractère souple et confiant), et la courbe rouge (qui reflète le comportement de personnes ayant un caractère dur et méfiant).

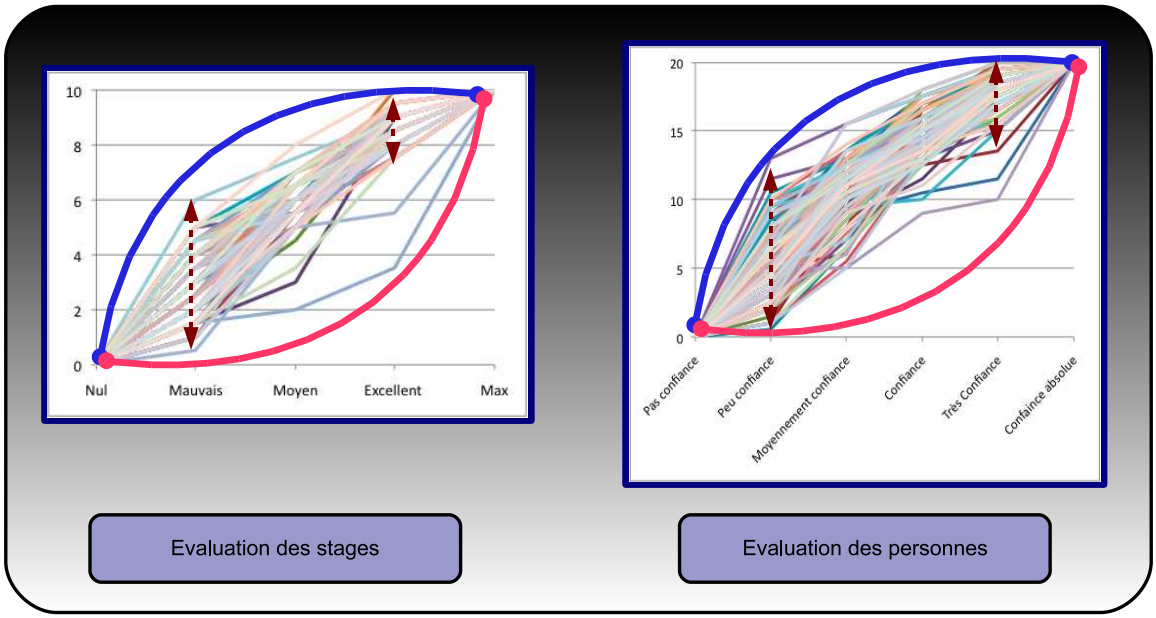

FIGURE B.2 – Résultats du questionnaire

En résumé comme le montre la figure 5.5, en se replaçant dans une échelle allant de la confiance (valeur =0) à la méfiance (valeur =max), deux catégories de personnes correspondant à deux classes de courbe ont été identifiées, comme suit :

1. **Classe confiante :** Elle représente les entités ayant une personnalité ouverte à caractère confiant. Nous définissons cette caractéristique par une fonction ayant une forme convexe. Comme c'est illustré dans la figure 5.5(a), ces entités auront tendance à faire confiance et à attribuer des valeurs plus proche du zéro (la valeur de confiance maximum) que de la valeur maximum de méfiance.

2. **Classe méfiante :** Elle regroupe toutes les entités présentant de la méfiance. Ce comportement peut se traduire par une courbe concave, où les entités sont plutôt réservées, accordant difficilement leur confiance en attribuant des valeurs plus proches de la borne maximum de méfiance que du zéro (voir figure 5.5(b)).

Annexe C

Algorithmes utilisés par le Simulateur

C.1 Calcul de la longueur du chemin maximum de référence θP_s^m

Cette valeur de référence a été introduite dans le chapitre 5 (section 5.5.3) afin de calculer la longueur maximum P_s^m nécessaire pour définir le seuil de méfiance transitive.

Rappel : La longueur du chemin maximum de référence θP_s^m du nœud "s" peut être déduite à partir de l'évaluation minimum (calculer grâce à la fonction de propagation) du chemin reliant "s" au nœud le plus éloigné du réseau dans un scénario où tous les nœuds du réseau exhibent une personnalité totalement confiante (i.e, $l^0 = 0$).

Autrement dit, *si tous les nœuds du réseau exhibent une personnalité très confiante(i.e. $l^0 = 0$), alors n'importe quelle source du graphe liée par un chemin de confiance à une cible est considérée par cette dernière comme étant une entité de confiance.*

Pour le simulateur nous avons évalué la confiance en utilisant la fonction de propagation P^0 définie dans le chapitre 5 (voir section 5.5.1). Concernant l'algorithme de recherche nous avons implémenté une version modifiée de l'algorithme de Dijkstra afin de calculer les chemins minimums, en terme de degré de confiance, pour atteindre tous les nœuds accessibles à partir de la cible. Ainsi, le chemin le plus long des chemins les plus courts (en terme de confiance) permet de définir le chemin maximum de référence θP^m , étant donné que la plus grande valeur de confiance retournée constitue la valeur minimum nécessaire pour que le nœud le plus éloigné soit considéré comme une entité de confiance.

Afin de réaliser ce calcul, le site voulant évaluer son θP^m propage une requête via les sites de son cercle de confiance sortant (inondation avec un parcours en profondeur) . Ensuite, tout site recevant cette requête évalue la fonction de propagation comme s'ils exhibaient un caractère totalement confiant. Cette évaluation ne remet pas en cause le TrustSort de chaque site mais nécessite juste de calculer la valeur correspondante à chaque site préalablement classé par le TrustSort en utilisant la fonction BV avec cette fois-ci un Disposition level $l^0 = 0$.

Dans ce qui suit, nous allons définir l'algorithme de propagation qui permet de remplir le tableau *eval* qui contient l'évaluation des chemins minimums pour atteindre tous les sites du réseau accessible à partir de la cible.

De cette manière, une fois le tableau rempli, la valeur maximum du tableau, $max(eval)$, permet de déduire la valeur θP^m. En effet, afin que tous les sites accessibles à partir de la cible soient considérés comme étant des sites de confiance, il suffit de considérer ce maximum comme étant le seuil de méfiance transitive :

$max(eval) = \theta t^0_{cible} = T^0_{cible} * \theta P^m$.

Ainsi, le θP^m peut être déduit en effectuant la division suivante :

$\theta P^m = \frac{max(val)}{T^0_{cible}}$

Notons :
- C_Trust(i,j) retourne l'évaluation par la confiance du site i envers le site j ($t^0(i,j)$), avec $l^0 = 0$.
- $T_OS(s)$ est un tableau qui contient tous les sites auxquels "s" fait confiance.

La fonction qui permet de calculer θP^m se présente comme suit :

```
#######################################################
# Entrée:                                             #
#      La cible:                                      #
#              id: numéro du site                     #
#              seuil: T^0_cible                       #
# Sortie:                                             #
#     Longueur du chemin maximum de référence θP^m    #
#######################################################


float Calcul_Chemin_Reference(cible)

    # Initialisation du tableau des seuils minimums
    pour n allant de 0 à |𝕊|
             eval[n]=+∞

    #Initialisation de la cible
     eval[cible.id]=0

    #Envoyer la requête à tous les site de confiance
    pour chaque i  membre du T_OS(cible.id)
              Chemin_Reference(i, C_Trust(cible.id,i), cible, eval )

    #Calcul de la valeur maximum de confiance
    max= cible.seuil  #chemin de référence supérieur ou égale à 1
    pour n allant de 0 à  |𝕊|-1
```

```
            Si (eval[n]> max) et  (eval[n]<> +∞) alors
                        max=eval[n]

    #Retourner le chemin de référence
    retourner( max/cible.seuil)
```

```
######################################################################
# Entrée:                                                            #
#     Local_Site:                                                    #
#             id: numéro du site  local qui exécute la fonction      #
#             seuil: T^0_Local_Site                                  #
#     Trust  : la confiance qu'attribut la cible au  Local_Site.     #
#     La cible.                                                      #
#     eval: tableau des chemins minimums                             #
# Sortie:                                                            #
#    eval. (Tableau --> passage par adresse)                         #
######################################################################
```

```
void Chemin_Reference(Local_Site, Trust, cible, eval)

    #Vérifier si ce chemin est le plus court (chemin optimal)
    Si (Trust < eval[Local_Site])  alors

            #Mettre à jour la table des chemins minimums
            eval[Local_Site]=Trust

            #Propager la requête à tous les voisins
            pour chaque i  membre du T_OS(Local_Site)

                    #Calculer la fonction de propagation
                    Trust= Trust+ cible.seuil/Local_Site.seuil * C_Trust(Local_Site.id, i)
                    Chemin_Reference(i, Trust, cible, eval)
```

Remarque : La récursivité de la fonction $Chemin\_Reference$ est bornée, car l'exécution s'arrête dès lors où le site local est sollicité par un chemin de confiance qui n'est pas optimal (chemin plus long en terme de degré de confiance). Ceci est également valable pour les circuits, puisque dans ce cas de figure l'évaluation de la chaîne de confiance va impérativement augmenter avant de boucler

Cependant, le calcul du θP^m peut être très long et générer beaucoup de trafic réseau, puisque l'algorithme doit découvrir tous les nœuds accessibles à partir de la cible afin de retrouver le nœud le plus éloigné. Pour cette raison cette procédure peut être effectuée occasionnellement (ex : une fois par jour, de préférence la nuit).

Exemple : Considérons l'exemple illustré par la figure C.1 avec les paramètres suivants :

- La cible est le site A
- Le seuil de méfiance local T^0 de tous les sites du graphe est égale à 10.
- Les valeurs [18] C_Trust sont positionnées sur les arcs du graphe (ex : C_Trust(A,C)=5).

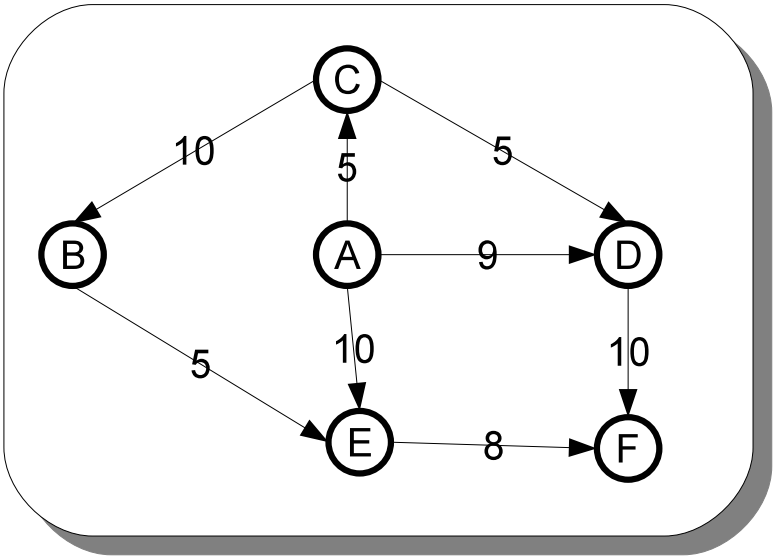

FIGURE C.1 – Exemple d'un graphe de confiance

Ainsi, en appliquant l'algorithme $Calcul\_Chemin\_Reference(A)$, on obtient la tableau suivant, qui illustre l'évolution du tableau "eval" :

	A	B	C	D	E	F
Initialisation	0	$+\infty$	$+\infty$	$+\infty$	$+\infty$	$+\infty$
Local_Site=A	0	$+\infty$	5	$+\infty$	$+\infty$	$+\infty$
Local_Site=C	0	15	5	$+\infty$	$+\infty$	$+\infty$
Local_Site=B	0	15	5	$+\infty$	20	$+\infty$
Local_Site=E	0	15	5	$+\infty$	20	28
Local_Site=C	0	15	5	10	20	28
Local_Site=D	0	15	5	10	20	20
Local_Site=A	0	15	5	9	20	20
Local_Site=D	0	15	5	9	20	19
Local_Site=A	0	15	5	9	10	19
Local_Site=E	0	15	5	9	10	18

TABLE C.1 – Exemple du calcul du chemin référence

18. Les valeurs de C_Trust ainsi que les arcs ont été définis de telle sorte à montrer le fonctionnement de l'algorithme et ne sont donc pas nécessairement conformes à une représentation utilisant le Trustsort.

A la fin de la propagation, la valeur maximum du tableau *eval* est égale à 18. Donc la fonction $Calcul\_Chemin\_Reference(A)$ retourne la valeur :

$\theta P^m = 18/10 = 1,8.$

Ainsi, ce θP^m appliqué dans un scénario où tout les sites sont confiant et compris A, permet à A de considérer F et tous les autres sites comme des sites de confiance, étant donné que F est le site le plus éloigné.

C.2 Calcul de la fonction de propagation

La fonction de propagation permet d'évaluer la relation de confiance liant le site cible au site source. Comme illustré dans le chapitre 5 (section 5.5.1), cette fonction est calculée à partir de la source. Ainsi, quand la cible récupère les différentes évaluations elle peut choisir de favoriser la plus petite ou la plus grande (voir panneau horizontal).

Comme pour le programme programme précèdent, nous avons utilisé l'algorithme de Dijkstra [19] afin de calculer les chemins les plus courts pour atteindre la cible. L'algorithme, qui est détaillé ci-dessous, calcule les chemins optimaux (les plus courts en terme de confiance) afin d'atteindre la source pour chaque site de confiance membre du T_OS de la cible. Ainsi, en fonction de la politique de la cible, l'algorithme peut alors retourner le plus long ou le plus court chemin des chemins optimaux.

Notons :

- C_path : liste qui comporte, dans l'ordre, l'ensemble des identifiants des nœuds composant le chemin parcouru à n'importe quel instant du programme.
- path : enregistrement composé de :
  - id : identifiant du site de confiance membre du T_OS.
  - ev : évaluation de la chaîne de confiance liant la cible à la source via le site de confiance identifié par "id".
  - ch : tableau contenant dans l'ordre les différents sites composants cette chaîne de confiance.
- paths : tableau d'enregistrements "path" dénotant l'ensemble des chemins optimaux par site de confiance.
- $T_IS(s)$ tableau qui contient tous les sites qui font confiance à "s".
- l_Trust(i,j) : fonction qui retourne l'évaluation par la confiance du "site i" envers le "site j" en utilisant la fonction BV avec le disposition level l^0 du "site i".
- Empilier et Dépiler, sont deux fonctions qui permettent respectivement de rajouter ou d'enlever un élément à la fin d'une liste.

19. Pour l'instant notre but est de montrer le fonctionnement du modèle de confiance T2D. Cependant, des algorithmes plus optimisés ferons l'objet de futurs travaux.

```
#########################################################################
# Entrée:                                                               #
#      source: identifiant de la source                                 #
#      cible:                                                           #
#              id: identifiant de la cible                              #
#              seuil: T^0_cible                                         #
# Sortie:                                                               #
#     Le chemin de confiance reliant la cible à la source:  "path"      #
#########################################################################
path Calcul_Fonction_Propagation(source, cible)

    # Initialisation du tableau des seuils minimums
      pour i allant de 0 à nombre_de_nœud
             eval[i]=+∞

     #Initialisation de la source et de la cible
       eval[source]=0
       eval[cible.id]=0

     #Initialisation du chemin de parcours (C_path)
       Empiler (C_path, source)

     #Initialisation du tableau des chemins minimums partant de la cible.
       j=0
       pour chaque i  membre du T_OS(cible)
               #stocker le numéro du site
               paths[j].id=i
               # évaluation de la source à partir du tiers de confiance i.
               paths[j].ev=+∞
                # chemin de confiance pour atteindre la source à partir de i.
               paths[j].ch=null
               j++

    #Envoyer la réponse à tous les sites qui font confiance à la source.
      pour chaque i  membre du T_IS(source)
              Propagation(i,source, 0, cible, eval, C_path, paths)

    #Sélectionner le chemin de confiance
    max=0
    min=0
    pour i allant de 0 à T_OS(cible)
          Si (paths[i][1]> paths[max][1]) et  (paths[i][1]<>+∞) alors  max=i
          Si (paths[i][1]< paths[min][1]) alors  min=i

     #retourner le plus court ou le plus long chemin
```

```
       Si (Sélectionner chemin le plus court)  alors   retourner (paths[min])
       Sinon retourner (paths[max])


#############################################################################
# Entrée:                                                                   #
#     Local_Site: l'indice du site qui exécute la fonction.                 #
#     Trusted_Site: l'indice du site qui répond au Local_Site.              #
#     Trust  : La confiance qu'attribue le Trusted_Site à la source.        #
#     cible: La cible                                                       #
#     eval: Tableau des chemins minimums.                                   #
#     C_path: Chemin parcouru                                               #
#     paths: Tableau des chemins optimaux                                   #
# Sortie:    (passage par adresse)                                          #
#    eval.                                                                  #
#    C_path.                                                                #
#    paths.                                                                 #
#############################################################################

Propagation(Local_Site, Trusted_Site, Trust, cible, eval, C_path, paths)

     #Calcul de la fonction de propagation
     Trust=Trust+cible.seuil/T^0_Local_Site * l_Trust(Local_Site, Trusted_Site)

     #Rajouter le site local à la chaîne de confiance
     Empiler(C_path, Local_Site)


     Si(Local_Site==cible.id)

                #Se positionner sur le site de confiance "Trusted_Site"
                   i=0
                   Tant que paths[i].id<>Trusted_Site alors i++

                #Vérifier si ce chemin est le plus court (chemin optimal)
                 Si(paths[i].eval>Trust) alors
                              paths[i].ev=Trust
                              paths[i].ch=C_path

     Sinon Si(eval[Local_Site]>Trust)

             #Diffuser l'évaluation aux membres du T_IS(Local_Site)
               eval[Local_Site]=Trust
               pour chaque i  membre du T_IS(Local_Site)
                    Propagation(i,Local_Site, Trust, eval, cible, C_path, paths)

     Depiler(C_path)
```

Exemple : En reprenant le même graphe illustré dans l'exemple précèdent (voir figure C.1) et en fixant les paramètres suivants :

- la source est le site F,
- la cible est le site A,
- les valeurs l_Trust représentent les poids des arcs du graph (ex : l_Trust(A,D)=9).

L'algorithme $Calcul\_Fonction\_Propagation(F, A)$, se déroule alors comme suit :

	eval						C_path	paths		
	A	B	C	D	E	F		id	ev	ch
Initialisation	0	$+\infty$	$+\infty$	$+\infty$	$+\infty$	0	F			
Local_Site=E	0	$+\infty$	$+\infty$	$+\infty$	8	0	FE			
Local_Site=A	0	$+\infty$	$+\infty$	$+\infty$	8	0	FEA	E	18	FEA
Local_Site=B	0	13	$+\infty$	$+\infty$	8	0	FEB			
Local_Site=C	0	13	23	$+\infty$	8	0	FEBC			
Local_Site=A	0	13	23	$+\infty$	8	0	FEBCA	C	28	FEBCA
Local_Site=D	0	13	23	10	8	0	FD			
Local_Site=A	0	13	23	10	8	0	FDA	D	19	FDA
Local_Site=C	0	13	15	10	8	0	FDC			
Local_Site=A	0	13	15	10	8	0	FDCA	C	20	FDCA

TABLE C.2 – Exemple du calcul de la fonction de propagation

On remarque qu'à partir du site C il existe deux chaînes de confiance, la chaîne "FEBCA" évaluée à 28 et la chaîne "FDCA" évaluée à 20. Dans ce cas de figure, l'algorithme choisit la seconde possibilité étant donné que son évaluation est la meilleur (plus petite).

Dans cet exemple, trois chaînes de confiance sont alors retenues (i.e. une chaîne pour chaque site de confiance de A). Par conséquent, la chaîne la plus courte en terme de valeur de confiance est la chaîne "FEA" accessible à partir de E et évaluée à 18 et la plus longue chaîne est "FDCA" accessible à partir de C et évaluée à 20.

Annexe D

Implémentation de l'API Fenrir

D.1 Vue générale de l'implementation

Nous avons choisi de regrouper l'API Fenrir dans le nom d'espace `Fenrir.Security` et de répartir les différentes classes dans deux sous-espaces : `MorphBodyObjects` et `MorphSignature` (voir figure D.1). Le premier est destiné à la création et à la manipulation des documents intermédiaires. Le second regroupe les classes nécessaires à la création et à la vérification d'une signature morph. Le diagramme de paquetage de la figure D.1 montre l'organisation logique des différents composants de l'API. Ainsi, la création et la vérification de la signature passent par la classe `XmlDsigMorphBodyTransform`. Par contre, la manipulation du document signé par le Propriétaire se fait à travers les classes définies dans le paquetage `MorphBodyObjects`.

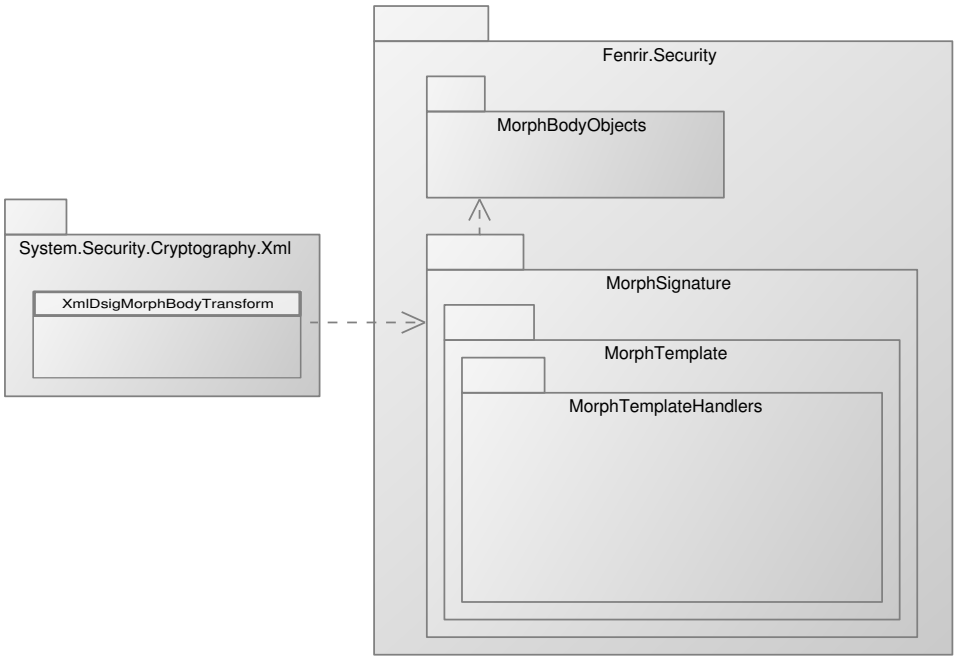

Figure D.1 – Diagramme de paquetages de l'API Fenrir

L'espace de noms `MorphSignature` englobe le sous espace `MorphTemplate` regroupant les différentes classes utilisées pour la manipulation des M-Templates. Il définit l'interface `IMorphTemplateHandler` qui permet, à l'image des transformations de l'API XMLDSig, de spécifier des M-Templates pour chaque type de document (XML, ASN1, etc.). Il définit aussi la classe `MorphTemplateManager` qui permet de sélectionner dans le sous-espace `MorphTemplateHandlers` le parseur qui correspond au type du M-Template et du document.

Afin de mieux illustrer les briques de base de notre implémentation, dans ce qui suit, nous allons décrire les trois classes suivantes :

- La classe `XmlDsigMorphBodyTransform` qui constitue le point d'ancrage.
- La classe `MorphBone` du paquetage `MorphBodyObjects`. qui permet au Propriétaire de manipuler le MorphBody.
- La classe `IMorphTemplateHandler` qui permet de définir le modèle d'un M-Template.

D.2 La classe `XmlDsigMorphBodyTransform`

La classe `XmlDsigMorphBodyTransform` peut être vue comme le point d'ancrage entre Mono et l'API Fenrir. Cette classe est utilisée pour deux types de comportement :

- Lors de la création d'une signature (voir figure D.2) : Une instance de la transformation est créée par le Signataire. Ensuite grâce à la méthode `AddMorphTemplate(T)`, le template T spécifié par le signataire est rajouté à la signature.
- Lors de la génération ou la vérification de la signature (voir figure D.3) : Le chargement de cette transformation est le même que pour toutes les autres transformations. L'instance de la classe `XmlDsigMorphBodyTransform` est créée par l'objet `SignedInfo` en faisant appel à la classe `Reference`. Celle-ci exécute la méthode `LoadInput(obj)` afin de charger l'objet (*obj*) de la dernière transformation et ensuite exécute `GetOutput()` afin d'appliquer la transformation qui s'occupe de charger le template (`LoadXmlTemplate`) et les parties flottantes (`LaodFloatPart` -dans le cas d'une vérification-) et ainsi pouvoir générer le MorphBody.

D.3 La classe `MorphBone`

La classe `MorphBone` est sollicitée par la classe de transformation et par le Propriétaire. Elle modélise l'objet MorphBody et dérive de la classe `HashableObject` comme le décrit le diagramme de classe de la figure D.4. Cette classe (`HashableObject`) permet d'implémenter l'algorithme en calculant le résidu d'une région dans un flux d'octets.

Les parties dynamiques sont modélisées par la classe `DynamicPartDescriptor` qui est dérivée de `HashableObject`. Elle définit en plus un certain nombre de propriétés et de méthodes comme l'identifiant de la partie dynamique, les données de remplacement, etc.

La classe `LockedDynamicPartDescriptor` dérivée de la classe `DynamicPartDescriptor` modélise, quand à elle, les parties dynamiques cachées dans une version morph du document et introduit

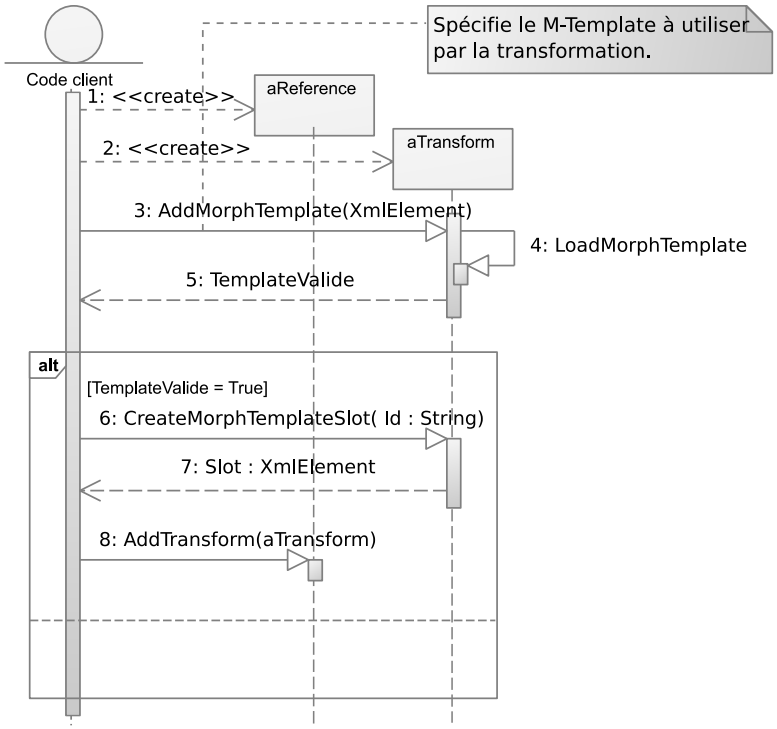

FIGURE D.2 – Création de la transformation par le code client

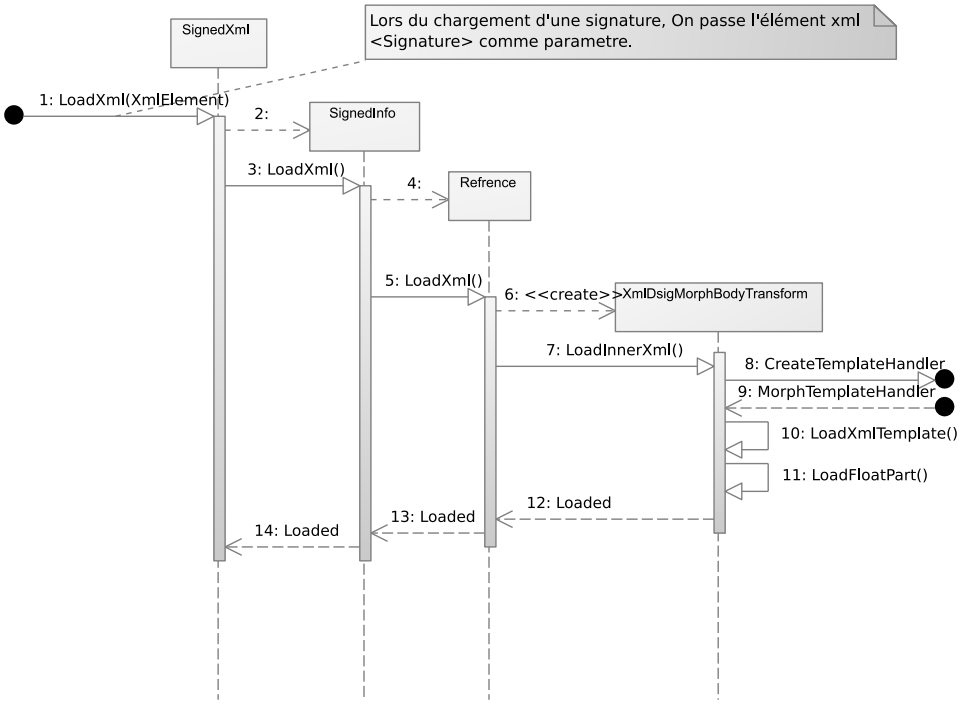

FIGURE D.3 – Comportement de la transformation lors du chargement d'une signature

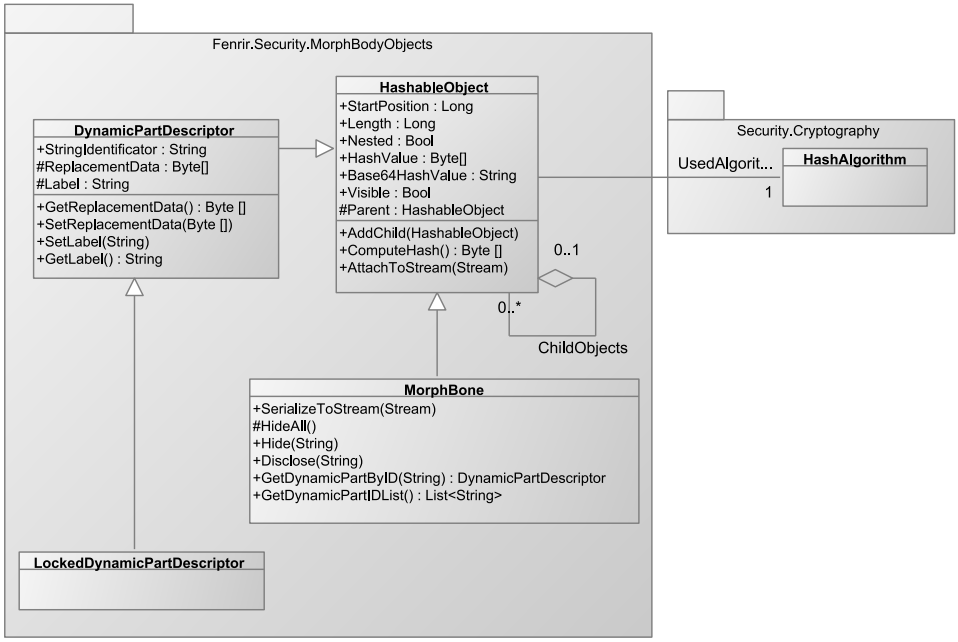

FIGURE D.4 – Diagramme de classes du paquetage MorphBodyObjects.

des contraintes sur la visibilité et la valeur de hash (valeur fixe définie dans les parties flottantes).

Le Propriétaire manipulera plus souvent les instances de la classe `MorphBone` qui permettent de créer différentes versions de l'objet signé, en activant ou désactivant la visibilité de certaines parties dynamiques à l'aide des méthodes `Hide()` et `Disclose()`. Le code client peut aussi accéder aux différentes instances de `DynamicPartDescriptor` grâce aux méthodes `GetDynamicPartByID()` et `GetDynamicPartIDList()`.

D.4 L'interface `IMorphTemplateHandlers`

Le modèle de traitement référence définit l'algorithme générique en utilisant une interface pour le traitement des M-Templates. Les objets réalisant cette interface sont des parseurs appelés MorphTemplateHandlers.

L'interface `IMorphTemplateHandler` définit le comportement que doit fournir un parseur et permet ainsi d'abstraire les détails concernant le traitement du M-Template sur le document cible. Le diagramme de classes de la figure D.5 décrit cette interface. La création des différentes instances des handlers se fait de manière indirecte en utilisant l'usine d'objets [20] `MorphTemplateManager`. Cette classe se charge aussi de répertorier et de gérer les differents type de handlers. La classe `MorphTemplateManager` permet ainsi de sélectionner le `IMorphTemplateHandler` qui correspond au type du M-Template. Par exemple pour un BM-Template, le `MorphTemplateManager` va créer un objet de type `RawMorphTemplateHandler`; par contre, dans le cas d'un XML M-Template c'est l'objet `XmlMorphTemplateHandler` qui doit être créé.

20. Factory object : objet permettant la création d'autres objets.

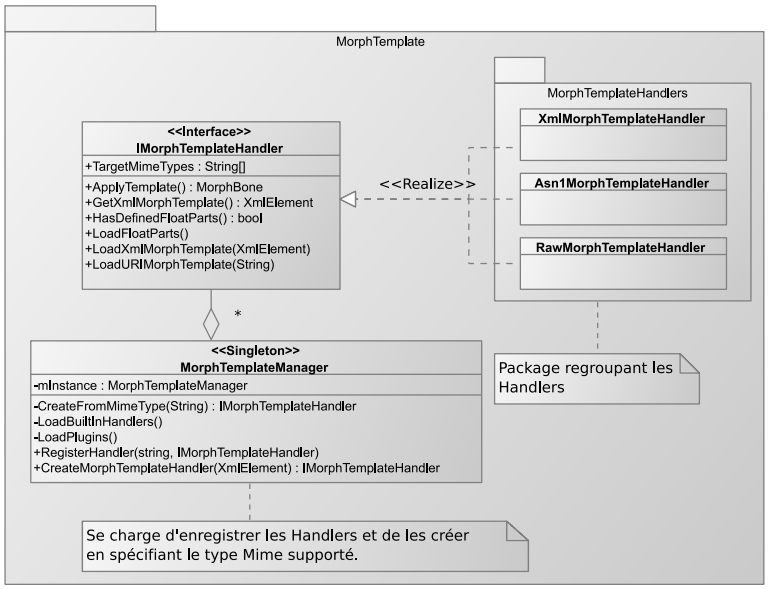

FIGURE D.5 – Diagramme de classes pour la prise en charge des M-Template.