Table des matières

Liste des figures

Chapitre 1 : Technologie d'agents mobiles

Chapitre 3 : Ontologie dans le domaine de sécurité des agents mobiles

Chapitre 4 : Modélisation des agents mobiles à base de composants et les agents de services

Chapitre 5 : Evaluation, Validation et Adaptation des agents mobiles par le SGSA

Chapitre 6 : Protection des données dynamiques des agents mobiles : Politiques XACML/ABAC

Chapitre 7 : Protection des agents mobiles et étude des performances

Liste des tableaux

Chapitre 2 : Sécurité d'agents mobiles

Chapitre 5 : Evaluation, Validation et Adaptation des agents mobiles par le SGSA

Chapitre 7 : Protection des agents mobiles et étude des performances

Introduction générale

Introduction

Ces dernières années sont témoins de la naissance d'une technologie nouvelle et prometteuse pour le développement des applications distribuées et ouvertes, celle des agents mobiles. Les principaux objectifs de l'utilisation de cette technologie concernent l'exécution asynchrone et autonome ainsi que la réduction de la charge dans le réseau.

La puissance de la technologie des agents mobiles dans la résolution des problèmes complexes résulte du fait que les agents, grâce à leur autonomie, mobilité et adaptabilité, peuvent réaliser leurs buts d'une manière flexible en se servant d'une interaction locale et/ou distante avec les autres agents sur le réseau. Cependant, cette flexibilité soulève des difficultés dans le développement des systèmes à base d'agents mobiles quant à la sécurité du système, la migration des agents et leurs communications, etc. Cependant, cette avancée a immédiatement mis en évidence un sérieux problème de sécurité qui a freiné son expansion.

Les travaux de recherche relatifs à la sécurité des agents mobiles suivent principalement deux axes. Le premier axe concerne la protection de la plate-forme contre des agents mobiles malicieux tandis que le second concerne la protection de l'agent mobile contre la malveillance des plates-formes visitées.

Les attaques d'un agent contre une plate-forme sont variées. Un agent hostile peut tenter d'avoir accès à des ressources sans autorisation de la plate-forme, consommer trop de ressources ou tenter de se faire passer pour un autre agent. Pour protéger la plate-forme, certains chercheurs ont développé plusieurs techniques dont la majorité est basée sur les approches de protection des systèmes conventionnels. D'autres chercheurs ont proposé des concepts nouveaux tel le code avec preuve qui consiste à prouver certaines propriétés sur le code d'un agent comme la protection de la mémoire et à fournir une preuve de ces propriétés.

Les attaques d'une plate-forme malveillante quant à elle, demeure un problème ouvert et difficile du fait que l'environnement d'exécution à un contrôle total sur l'agent mobile. Un agent se déplace sur plusieurs plates-formes, dans des domaines de sécurité différents. Une plate-forme malicieuse peut essayer d'attaquer un agent mobile afin d'obtenir des informations sensibles stockées dans sa mémoire ou analyser et manipuler son code critique. L'encryptage classique n'est valable ici que

pour le code et les données qui ne seront pas utilisés sur cette plate-forme. En outre, même si la plate-forme n'est pas capable d'obtenir directement de l'information par l'agent, elle peut en déduire par les services demandés. La destination de l'agent ou encore à partir des agents avec qui il communique. Cette plate-forme peut aussi altérer le code, les données sensibles et la communication de l'agent mobile. Cela serait désastreux dans le cas de transactions financières. Un agent prend des risques à chaque fois qu'il se déplace sur une nouvelle plate-forme. De plus, il peut devenir impossible de retrouver la plate-forme responsable d'une altération si elle n'est pas détectée tout de suite. Par conséquent, le propriétaire d'agent doit avoir des garanties concernant la protection de l'agent contre les menaces des plates-formes malicieuse. Ainsi, l'agent mobile doit se protéger contre n'importe quel acte visant à sa détérioration, sa destruction ou la manipulation de son code, de son état ou de ses données.

Plusieurs tentatives abordent ces menaces d'une manière totale ou partielle. Les techniques de protection d'un agent sont subdivisées essentiellement en deux catégories les techniques préventives (rendre l'accès et la modification de l'agent difficile) et les techniques de détection (détecte la modification illégale du code, et les données de l'agent).

Le problème de la protection de la plate-forme a reçu de considérables attentions de la part des recherches scientifiques, tandis que la protection de ces agents est un domaine qui reste ouvert. Tous ces paramètres ont attiré notre attention, et nous ont poussé à réfléchir sérieusement à ce sujet. La sécurisation de l'agent mobile contre les attaques des plates-formes malveillantes nous intéresse particulièrement. Les approches introduites aident à l'amélioration de la sécurité du code exécuté dans un environnement non crédible. Toutefois, elles présentent des inconvénients. C'est dans cette optique que s'inscrit notre contribution qui consiste à mettre en œuvre de nouveaux concepts pour renforcer la sécurité de l'agent mobile contre des plates-formes malveillantes.

Objectifs

Le concept d'agent mobile intègre deux participants : le propriétaire de l'agent et celui de la plate-forme. Chacun veut avoir des garanties que l'autre ne pourra pas l'attaquer. D'un côté, on a la plate-forme qui ne connaît pas toujours le programme d'agent mobile qu'elle reçoit et qu'elle doit exécuter. Il faut donc des techniques de protection de la plate-forme. Il est nécessaire de protéger l'environnement d'exécution et les ressources de la plate-forme contre plusieurs types d'attaques provenant d'un agent mobile. D'un autre côté, le propriétaire de l'agent ne veut pas

qu'une plate-forme puisse modifier l'agent ou les résultats qu'il a obtenus sur d'autres plates-formes. Pour parer à ces attaques, il faut trouver des techniques de protection de l'agent. En effet, deux types de problèmes se posent en ce qui concerne la sécurité dans le concept d'agent mobile. D'un côté, on veut protéger la plate-forme contre les agents mobiles malveillants et d'un autre, on veut protéger l'agent contre la plate-forme qui l'exécute.

La protection des plates-formes visitées contre des attaques menées par des agents mobiles malveillants est un problème qui est aujourd'hui assez bien maîtrisé. En effet, plusieurs solutions permettent maintenant de se prémunir contre d'éventuelles attaques. La première solution pour protéger la plate-forme de la malveillance des agents mobiles est de simplement limiter la fonctionnalité de l'environnement d'exécution afin de limiter les vulnérabilités. Les techniques de protection des plates-formes suivent, aujourd'hui deux directions. La première direction s'occupe de l'enrichissement graduel de l'infrastructure du code mobile par l'authentification, l'intégrité des données et par des mécanismes de contrôle d'accès. Par contre la seconde direction se consacre à la vérification de la sémantique du code mobile.

La protection d'un agent mobile à l'encontre des plates-formes malicieuses revient à protéger principalement son exécution, son intégrité et sa confidentialité. Plusieurs approches de protection de l'agent mobile ont été proposées. Elles tentent de garantir l'accès de l'agent mobile à des plates-formes dans lesquelles il peut avoir toute confiance ou de déceler ceux qui sont malveillants. Elles visent principalement à détecter les attaques ou à les rendre inefficaces.

L'objectif principal de cette thèse est de concevoir un mécanisme de sécurisation à la fois robuste et efficace permettant de protéger les agents mobiles contre les attaques d'analyse ou d'altération de leur code et ses résultats obtenus sur les plates-formes visitées. Notre approche répond à des questions importantes lors de la création d'agents mobiles, comme la modularité du code sous forme de composants, la spécification détaillée aux tâches, la mobilité et la sécurité de l'agent mobile. L'objectif principal de ces systèmes est de créer des agents mobiles à base des composants, fiables qui peuvent faire confiance et dont la confiance peut être vérifiée par une autorité de confiance.

Pour déterminer la correspondance entre les exigences de sécurité d'agent et les plates-formes d'exécution, nous apportons un intérêt à la modélisation des politiques de sécurité dans le système d'agents mobiles. Nous avons choisi un standard WS-Policy pour exprimer les exigences et les capacités de sécurité. Enfin, nous avons construit une ontologie dans le domaine de sécurité des agents mobiles,

afin d'éliminer les différences sémantiques qui existent entre ces politiques (agents et plates-formes).

Notre approche de sécurité est basée sur deux stratégies d'adaptation. La première est une adaptation statique permet de transformer l'agent à un agent mobile sécurisé de confiance. Cette adaptation repose sur trois sous-systèmes du système de gestion de la sécurité d'agents (SGSA) :

- Système d'Evaluation d'Agents (SEA) permet d'évaluer et de valider : l'identité, l'intégrité et l'assemblage des composants fonctionnels de l'agent.
- Système de Gestion des Services (SGS) son rôle est d'analyser la correspondance sémantique entre les politiques de sécurité (plate-forme/agent) pour créer l'itinéraire de l'agent mobile.
- Système d'Adaptation de Sécurité (SAS) joue un rôle important dans le processus d'adaptation, il estime le niveau de sécurité des composants de l'agent et le degré de confiance des plates-formes d'exécution. Ensuite, il sécurise les composants sensibles et définit les règles de contrôle d'accès aux données d'agent.

La deuxième est une adaptation dynamique permet d'appliquer la politique de sécurité de l'agent. Cette adaptation protège les données dynamiques collectées par l'agent durant son exécution.

Organisation de thèse

Les synthèses faites et les réflexions développées dans le cadre de ce travail de thèse ont fait l'objet de sept chapitres répartis comme suit :

Le premier chapitre commence par introduire le contexte d'application des agents mobiles afin d'appréhender la complexité liée à ce domaine. Ensuite, il synthétise les principaux travaux focalisés sur la modélisation des systèmes à base d'agents mobiles.

Le deuxième chapitre présente un état de l'art des principaux travaux liés à la sécurité des systèmes distribués afin de montrer à quel niveau ils peuvent être appliqués aux problèmes spécifiques aux systèmes à base d'agents mobiles. Ensuite, nous avons étudié les travaux liés à la sécurité des systèmes à base d'agents mobiles pour exhiber leurs limites en regard de leurs avantages.

Le troisième chapitre modélise les politiques de sécurité pour exprimer les exigences et les capacités de sécurité dans notre système d'agents mobiles. Ensuite,

nous avons construit une ontologie dans le domaine de sécurité des agents mobiles pour faciliter l'analyse automatique de la compatibilité sémantique entre les politiques de sécurité (agent/plate-forme).

Le quatrième chapitre présente le modèle d'agent mobile à base des composants. Ce modèle permet au concepteur d'un agent de spécifier à travers des concepts de haut niveau (composants, descriptions fonctionnelles, politique de sécurité…) quand et comment l'agent doit s'adapter à l'environnement d'exécution. Ce chapitre présente aussi l'architecture interne des plates-formes offrant des services aux agents mobiles. Chaque service est modélisé par un agent de service. Ensuite, il montre comment adapter le registre UDDI pour publier les services offerts par ces plates-formes. Le nouveau registre publie deux types de descriptions de service. La première est une description fonctionnelle pour les services offerts par la plate-forme. La deuxième est une politique de sécurité basée sur l'ontologie que nous avons définie dans le chapitre 3. Cette politique permet d'exprimer les exigences et les capacités de sécurité d'un service particulier.

Le sixième chapitre montre l'architecture générale de notre système d'adaptation. L'objectif de ce système est de transformer l'agent à un agent mobile sécurisé. Il repose sur trois systèmes de confiance. Le premier est le SEA, qui permet d'évaluer et valider l'identité, l'intégrité et l'assemblage des composants fonctionnels de l'agent. Le deuxième est le SGS, son rôle est d'analyser la correspondance sémantique entre les politiques de sécurité (plate-forme/agent) pour créer l'itinéraire de l'agent mobile. Le troisième est le SAS qui estime le niveau de sécurité des composants d'agent et le degré de confiance des plates-formes d'exécution. Enfin, ce chapitre montre comment protéger les composants sensibles et comment définir les règles de contrôle d'accès aux données d'agent.

Le sixième chapitre présente un modèle de contrôle d'accès basé sur le modèle ABAC et langage XACML. Le modèle ABAC est utilisé dans notre système pour représenter les politiques de contrôle d'accès aux données d'agent à base d'attributs. Ensuite, nous adaptons le langage XACML pour créer et gérer les politiques de contrôle d'accès entre les plates-formes d'exécution.

Le septième chapitre présente trois parties : la première concerne comment le propriétaire de l'agent initialise et authentifie ses agents. La deuxième montre comment authentifier, valider et décrypter les composants sensibles d'un agent par les plates-formes d'exécution. La troisième discute la fiabilité, la performance et la robustesse de notre protection vis-à-vis des différents types attaques.

Enfin, nous concluons ce manuscrit par une synthèse des contributions et dégageons les perspectives de recherches.

CHAPITRE 1

Technologie d'agents mobiles

1.1. Introduction

La plupart des applications sur le réseau Internet nécessitent l'interaction entre différentes entités à travers le réseau, afin d'échanger des données et de répartir les tâches. Aujourd'hui, le modèle « client/serveur » où les échanges se font par des appels distants à travers le réseau est le modèle le plus utilisé. Ce modèle présente l'inconvénient d'augmenter le trafic sur le réseau et il exige une connexion permanente entre le client et le serveur ce qui n'est pas le cas des terminaux mobiles qui sont exposés à la perte de la connexion.

Dans ce chapitre, nous présentons les différents modèles utilisés pour la répartition d'une application sur le réseau Internet. Nous nous focaliserons sur la description du modèle d'agent mobile, qui propose une solution facilitant le développement des applications réparties sur des réseaux à grande échelle. Cette description permet d'appréhender la complexité de ce domaine qui est exprimée par la variété des aspects à considérer dans un tel système.

Pour gérer cette complexité, plusieurs travaux de modélisation ont été proposés. Nous présentons, dans ce chapitre, une classification des travaux existants. Cette classification nous permet de préciser les points forts et les points faibles des différents modèles proposés.

1.2. Concepts de base des agents mobiles

Un processus informatique est constitué par une séquence d'instructions qui s'exécute sur une machine et qui utilise les ressources de cette machine. Dans le contexte de la mobilité, il faut envisager la possibilité d'interrompre l'exécution d'un processus afin de la poursuivre sur une autre machine. Le processus est représenté, en plus de son code, par son état d'exécution. Par état d'exécution nous entendons la valeur du compteur ordinal, celle de la pile d'exécution et les différents registres du processeur. Selon le schéma d'exécution de ce code et la localisation des différentes entités du système (ressource, code et état d'exécution), nous pouvons distinguer les notions d'évaluation distante, de code à la demande et d'agents

mobiles [Belkhelladi, 2010].

1.2.1. Évaluation distante

Dans une interaction par évaluation distante (figure 1.1), un client envoie un code à un site distant. Le site récepteur utilise ses ressources pour exécuter le programme envoyé. Éventuellement, une interaction additionnelle délivre ensuite les résultats au client. Dans ce schéma, seul le code est transmis au serveur et l'exécution du code se déroule uniquement sur ce dernier. Les interactions avec les imprimantes PostScript sont réalisées par ce modèle. Le code d'une requête SQL émis vers un serveur de base de données représente un autre exemple d'évaluation distante [Belkhelladi, 2010].

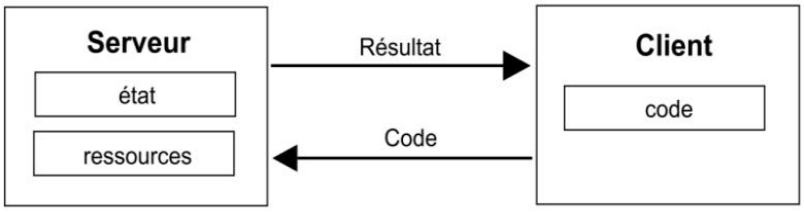

Figure 1. 1. Évaluation distante

1.2.2. Code à la demande

Dans ce schéma, le processus client interagit avec un site distant afin de récupérer le savoir-faire qui sera exécuté sur la machine cliente. Ainsi, le client télécharge le code nécessaire à la réalisation d'un service. Le rôle du site distant est de fournir le code du service qui sera exécuté sur le site client (figure 1.2). Les Applets Java reposent sur cette technologie du code mobile, il s'agit d'un programme chargé à partir d'une page Web pour être exécuté sur la machine du client [Belkhelladi, 2010].

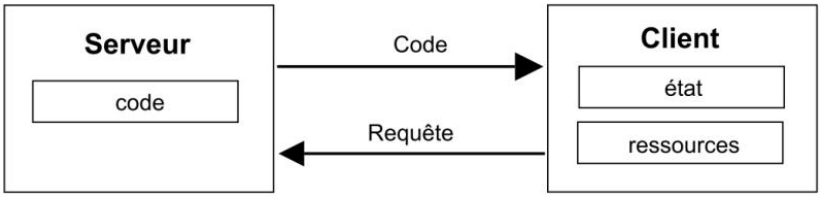

Figure 1. 2. Code à la demande

1.2.3. Agents mobiles

Par comparaison avec les deux schémas précédents, l'exécution du processus débute sur le site client. Dans la mesure où le client a besoin d'interagir avec le serveur, ce même processus (code, état d'exécution et données) se déplace à travers le réseau pour continuer son exécution et pour interagir localement avec les ressources du serveur (figure 1.3). Après exécution, l'agent mobile retourne

éventuellement vers son client afin de lui fournir les résultats de son exécution [Belkhelladi, 2010].

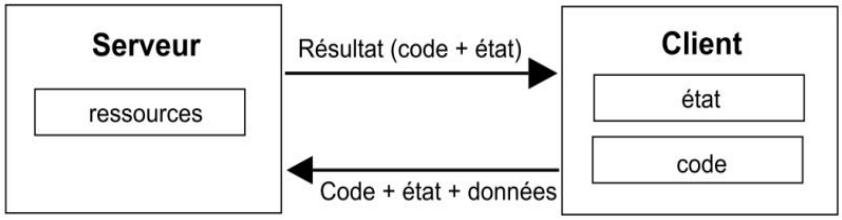

Figure 1. 3. Agent mobile

Dans ce schéma, le savoir-faire appartient au client, l'exécution du code est initiée côté client et continuée sur les différentes machines visitées.

1.2.3.1. Définition d'agent mobile

Un agent mobile est défini comme un élément autonome œuvrant généralement au nom d'un utilisateur ou d'une application [Leriche et al., 2006]. Il possède une activité interne avec ses propres ressources, il peut aussi accéder aux ressources de l'hôte d'accueil et communiquer avec d'autres agents afin de réaliser la tâche pour laquelle il a été créé. Il a la capacité de se déplacer de site en site en ayant conscience de son déplacement. Il est aussi capable de percevoir son environnement, de s'adapter à des conditions particulières et réagir à des événements préalablement décrits. De cette définition découlent les points essentiels suivants :

Comportement : un agent est pourvu d'un programme qui lui dicte la tâche à accomplir. On dit qu'il agit par délégation pour le client. Un agent peut ainsi venir avec une compétence particulière sur une machine hôte.

Autonomie : un agent agit indépendamment du client. Il décide lui-même là où il va se déplacer et ce qu'il doit y faire, en fonction du comportement qui lui a été donné. Cela implique que l'on ne peut pas toujours prévoir l'itinéraire des agents.

Mémoire : un agent mobile dispose d'une capacité de mémorisation lui permettant de récolter des informations sur les sites visités. Les informations mémorisées seront livrées par l'agent après son retour au client.

Environnement : l'environnement dans lequel l'agent évolue est constitué par :

- Les machines qui vont accueillir l'agent lors de son exécution, ainsi l'agent utilise les ressources (unité de calcul, mémoire, etc.) disponibles sur la machine afin d'accomplir la tâche demandée par le client.
- Les liens réseau que l'agent utilise pour se déplacer entre les différents sites.
- Les autres agents fixes et mobiles avec qui l'agent interagit.

1.2.3.2. Caractéristique d'un agent mobile

Pour pouvoir évoluer convenablement dans un système distribue, l'agent mobile doit satisfaire un certain nombre de propriétés. Principalement, il doit être :

- Autonome : un agent mobile doit être capable de prendre des décisions afin d'améliorer l'exécution de sa mission. Ainsi, il est capable de se déplacer d'un système à un autre afin de se rapprocher des ressources et/ou des services dont il a besoin. De plus, nous supposons qu'un agent soit capable de se cloner afin de lancer en parallèle l'exécution de ses tâches sur différents systèmes.
- Communicant : un agent mobile doit avoir la faculté de communiquer avec les autres agents du système (agents locaux ou distants), afin d'échanger des informations et bénéficier des connaissances et du savoir-faire des autres agents. En pratique, la mobilité ne se substitue pas aux capacités de communication des agents, mais elle les complète (la communication distante, moins couteuse dans certains cas, reste possible).
- Sécurisé : à sa migration, l'agent doit se protéger contre les systèmes d'accueil malveillants. Sans protection, le système d'accueil peut altérer ou détruire les informations sensibles de l'agent tel que les résultats partiels de son exécution.
- Adaptable : la mobilité de l'agent nécessite impérativement l'attitude d'adaptation chez l'agent. En effet, pour pouvoir évoluer convenablement entre différents systèmes hétérogènes l'agent doit être capable de s'adapter dynamiquement aux variations de son contexte d'exécution.

1.2.4. Intérêts des agents mobiles

Réduction de la charge réseau : la mise en place d'agents mobiles permet de réduire le trafic réseau étant donné qu'une majorité des traitements sont réalisés localement. C'est notamment le cas dans [Sahai et al., 1998] sur une application de supervision et de gestion de réseaux dans laquelle des agents mobiles ont une tâche à accomplir sur un ensemble de sites et la réalisent de manière autonome avant de retourner sur le site de leur initiateur. Dans [Gray et al., 2002], l'évaluation de la réduction de charge est exécutée sur une application militaire de collecte d'informations de terrain dans laquelle les agents mobiles font le lien entre les soldats sur le terrain et le poste de commandement.

Diminution du temps de latence : dans les échanges client/serveur un gain en temps de réponse des applications est obtenu s'il est possible de rapprocher physiquement

le client du serveur sur le réseau. On retrouve notamment ce problème dans le placement des caches dans les réseaux de distribution de contenu (Content Delivery Network). Une mise en pratique au moyen d'agents mobiles a été réalisée dans [Johansen, 1998] sur une application de serveur d'images satellites de grande résolution et sur une application de serveur de vidéos de grande taille. Dans ces deux applications, le client est un agent mobile qui se déplace sur le serveur pour effectuer les recherches dans ces collections de grande taille. Sur cette application, le gain en temps de réponse de l'application agents mobiles par rapport à l'application client/serveur est très significatif. Dans [El Falou et al., 2005], [El Falou, 2006], le gain en temps de réponse des approches par agents mobiles est illustré sur des problématiques de recherche d'informations. Dans ce contexte applicatif, le résultat de la requête d'un client est obtenu suite à l'interrogation en séquence d'un ensemble de serveurs. Dans cette approche, un agent mobile calcule sa politique de visite et d'interaction avec les serveurs à l'aide d'un processus de décision markovien, qui cherche à optimiser le temps de réponse de l'application. Cette politique est notamment calculée en fonction du débit sur les liens et de la taille de l'agent. Cette approche est hybride dans le sens où c'est la politique de l'agent qui détermine si tel ou tel échange se fera par une migration de l'agent ou par une communication client/serveur entre l'agent et le serveur concerné. Les résultats expérimentaux montrent que cette approche hybride par MDP apporte de meilleurs temps de réponse que des approches 100% client/serveur ou 100% mobiles.

Conception et génie logiciel : cette nouvelle technologie ouvre la porte à un nouveau paradigme de programmation. Dans JavAct [Arcangeli et al., 2001], par exemple, les agents sont implémentés au moyen d'objets (au sens de la programmation orientée objets) actifs s'exécutant dans un thread et possédant les propriétés d'agents. La porte est alors ouverte à un nouveau paradigme que l'on pourrait caractériser de programmation par agents mobiles.

1.2.5. Environnement d'exécution d'agents mobiles

Un environnement d'exécution d'agents mobiles fournit une interface de programmation et un environnement d'exécution offrant des primitives pour créer, lancer et exécuter un agent mobile. Cet environnement est constitué d'un ensemble de programmes statiques, appelés places, s'exécutant sur les sites du système susceptibles d'accueillir des agents.

Les environnements d'exécution d'agents mobiles offrent plusieurs services de base permettant l'exécution d'un agent mobile sur un site. Ces services sont les suivants : création et migration d'un agent mobile, communication et échange de messages

entre les agents mobiles, accès aux ressources locales par les agents mobiles et traçage des agents mobiles en cours d'exécution. Il faut ajouter à ces services, fournis par l'environnement d'exécution d'agents mobiles, des mécanismes mettant en œuvre des notions de qualité de service requise par les applications par exemple la sécurité ou la tolérance aux pannes.

1.3. Efforts de normalisation des plates-formes d'agents mobiles

Depuis l'apparition du concept d'agents mobiles, des nombreuses plates-formes telles que Aglets [Lange et al., 1999], JADE [Bellifemine et al., 2007] et Voyager [Recursion Software, 2011] ont été développées pour faciliter la mise en œuvre des applications à base de ce paradigme. L'architecture et la mise en œuvre de ces systèmes sont très différentes. Ceci empêche l'interopérabilité entre les plates-formes et freine ainsi la diffusion rapide de la technologie d'agents mobiles. Pour promouvoir l'interopérabilité et la diversité du système, plusieurs organisations ont proposé des standards. Deux d'entre elles sont la norme de *Mobile* Agent System Interoperability Facility (MASIF) [Milojicic et al., 1998] et celle de FIPA (Foundation for Physical Intelligent Agents) [Foundation for Intelligent Physical Agents, 2011].

1.3.1. Norme MASIF

MASIF [Milojicic et al., 1998] est un standard, pour les systèmes à base d'agents mobiles, qui a été spécifié par l'Object Management Group (OMG). L'objectif principal de ce travail était de proposer un ensemble de définitions et d'interfaces afin d'avoir un niveau d'interopérabilité entre différents agents mobiles. MASIF ne standardise pas les opérations locales des agents telles que l'interprétation de l'agent, sa sérialisation et son exécution. Plutôt, MASIF standardise :

- La gestion des agents par la définition des opérations pour créer un agent, suspendre son exécution, reprendre et terminer son activité.
- Le transfert des agents.
- Le nommage des agents et des systèmes d'accueil des agents par le fait de standardiser la syntaxe et la sémantique des noms.
- Les types des systèmes d'accueil et la syntaxe de leur localisation afin de vérifier si le type du système supporte l'accueil de l'agent.

La norme MASIF exige une infrastructure basée principalement sur les concepts suivants :

- Les agents sont des programmes autonomes qui agissent pour le compte d'une personne ou d'une organisation. Elles expriment l'autorité de l'agent.

- Les agents sont hébergés et s'exécutent sur des places. Une place est un contexte défini dans un système d'agents (agent system).
- La place source et la place destination d'un agent mobile peuvent résider dans le même système d'agents.
- Un système d'agents est une plate-forme qui peut créer, interpréter, exécuter, transférer et arrêter l'agent.
- De même que l'agent, un système d'agents est associé à une autorité qui identifie la personne ou l'organisation pour laquelle il agit.
- Un type de système d'agents (agent system type) décrit le profile de l'agent. Un agent se déplace d'une place à une autre si son profil (type du système d'agents, langage et méthode de sérialisation) est reconnu par le système d'agents destination.
- Les systèmes d'agents de même autorité seront regroupés dans une région. Ce concept assure la sociabilité.

La norme MASIF est fondée sur des services CORBA à savoir le service de : nommage, cycle de vie, externalisation (sérialisation) et sécurité. Le service de sécurité définit principalement des fonctions permettant : l'authentification du client pour une création distante des agents, l'authentification mutuelle des systèmes d'accueil d'agents, l'authentification et délégation des agents. Les agents mobiles et les systèmes d'accueil disposent d'une ou plusieurs politiques de sécurité qui gouvernent leurs activités. Cette dernière est composée d'un ensemble de règles pour limiter ou accorder des capacités/accès à l'agent et définir des limites de consommation de ressource. En fait, une politique est définie en fonction de l'authenticité des parties communicantes, la classe d'agent, l'autorité de l'agent, et/ou plusieurs autres facteurs.

1.3.2. Norme FIPA

FIPA est une organisation de standardisation fondée en 1996 à Genève (Suisse) dont l'objectif est de spécifier des standards logiciels pour assurer l'interopérabilité entre les agents et les applications à base d'agents. Les spécifications FIPA sont réparties en cinq catégories :

- Les applications : représentant des exemples de domaines d'application sur lesquels peuvent être déployer des agents FIPA.
- Les architectures abstraites : ces spécifications s'intéressent à la définition des entités abstraites nécessaires pour le développement d'un

environnement d'agents.

- La gestion des agents : ces spécifications traitent le contrôle et la gestion des agents dans/entre les plates-formes d'agents.
- Le transport des messages d'agents : ces spécifications traitent le transport et la représentation des messages à travers différents protocoles réseaux.
- La communication des agents : ces spécifications traitent les messages d'ACL (Agent Communication Language), les protocoles d'échange de message, etc.

FIPA a publié plusieurs documents qui présentent les spécifications pour ces différentes catégories. Ces spécifications ont passé par plusieurs versions FIPA97, FIPA98 et FIPA2000. Dans notre étude nous nous sommes basés sur la version 2000 et plus particulièrement au document intitule "FIPA Agent Management Support for Mobility Specification" qui s'intéresse à la spécification des besoins et des technologies permettant à l'agent de tenir avantage de la mobilité. Principalement dans ce contexte, les efforts ont été investis dans la spécification de :

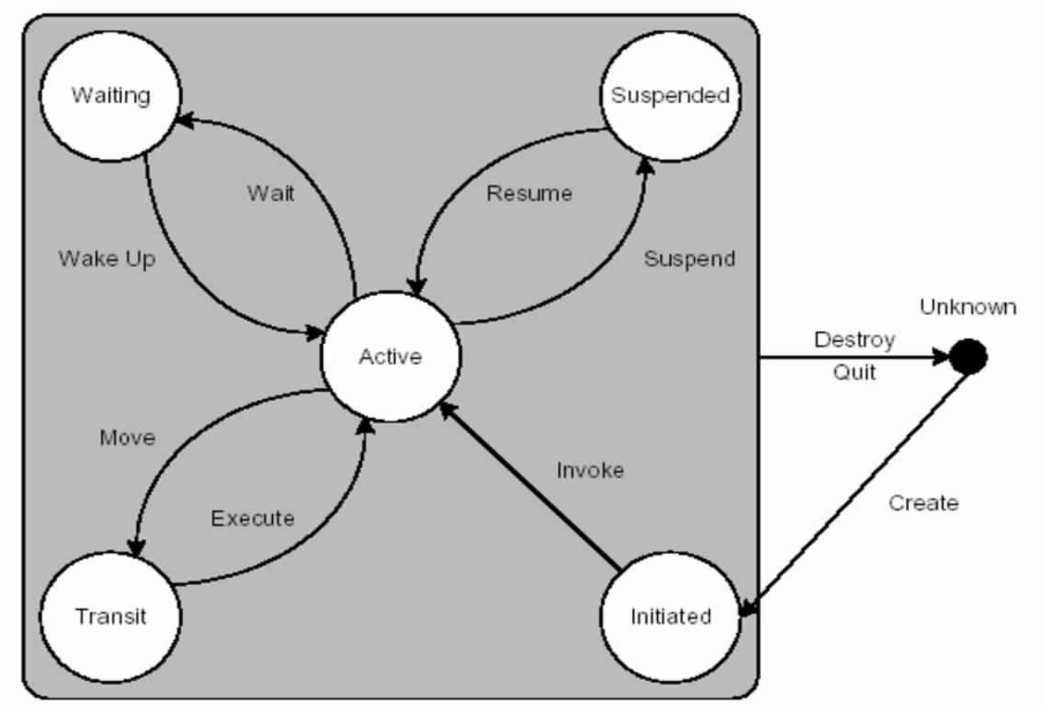

Figure 1. 4. Cycle de vie de l'agent mobile de FIPA

- Protocoles de mobilité : un nombre de protocoles pour couvrir les différentes formes de mobilité de l'agent. Ils s'adressent, spécialement à la migration, au clonage et à l'invocation de l'agent. En fait, chaque protocole représente graphiquement les actions et les réactions des entités impliquées (agents et plates-formes d'agents).

- Cycle de vie de la mobilité : il étend le cycle de vie de l'agent stationnaire par l'ajout d'un nouvel état ('*Transit'*) et deux nouvelles actions pour atteindre/partir de cet état ('*Move'* et '*Execute'*). Ce cycle de vie est illustre par la Figure 1.4.
- L'action 'Move' permet de faire passer l'agent dans un état transitoire. Cette action sera initiée par l'agent, par contre l'action '*execute'* sera initiée par le système d'exécution d'agents. L'action '*execute'* permet de faire sortir l'agent de l'état transitoire et active son exécution. Pour passer à l'état '*Transit'*, l'agent mobile doit initie l'exécution d'un protocole de mobilité pour aller à un nouveau système.
- Ontologie de mobilité de l'agent : elle présente une description textuelle d'un ensemble d'objets et de fonctions pour assurer la mobilité de l'agent.
- Tout objet est décrit par un ensemble de paramètres, dont chacun est représenté par une description (qui présente une description textuelle détaillée de la sémantique de ce paramètre), une présence (*mandatory*/*optional* : qui indique si le paramètre est obligatoire ou facultatif), un type (qui indique le type des valeurs de ce paramètre : *Integer*, *Word*, *String*, *URL*, *Term*, *Set* ou *Sequence*) et des valeurs réservées (représentent une liste de valeurs qui peuvent être supportées par ce paramètre).
- D'une manière similaire, une fonction est décrite d'un symbole (qui l'identifie), un type d'agent (qui supporte cette fonction), une description (description textuelle de la sémantique de cette fonction), un domaine (le domaine auquel est définie une fonction), un range, et une arité (indique le nombre des arguments de cette fonction)

Les problèmes liés à la sécurité, dans un contexte mobile, n'ont pas été traités ni dans cette spécification ni dans ses implémentations tel que FIPA-OS (FIPA Open Source) [Hadingham et al., 2000]. Pour ce faire, une tentative de définition d'architecture de sécurité a été proposée dans [Zhang et al., 2001]. Cette architecture définit une structure permettant d'implémenter deux services de sécurité : un service de communication sécurisée et un autre pour une exécution sécurisée. Le premier service fait face à l'écoute ou l'interférence du réseau extérieur. Le deuxième service protège les ressources du serveur d'exécution et les services de l'agent contre les accès non-autorisés.

Actuellement, plusieurs plates-formes d'agents mobiles supportent le standard de FIPA, telles que JADE [Bellifemine et al., 1999] et ZEUS [Nwana et al., 1999]. De

même, la norme MASIF a été largement utilisée dans le développement des plates-formes d'agents mobiles. Grasshopper [Baumer et al., 1999] a été la première plates-forme conforme au standard MASIF.

La complémentarité des aspects définis par MASIF et FIPA a incité plusieurs chercheurs à définir des plates-formes conformes aux deux normes [Kaffille et al., 2003].

1.3.3. Synthèse de normes MASIF et FIPA

Les normes *MASIF* et *FIPA* ne sont qu'un point de départ pour permettre l'interopérabilité des plates-formes d'agents et ainsi exploiter les paradigmes basés sur les agents mobiles. Nous constatons certaines similitudes et différences entre les normes FIPA et MASIF. En effet, les efforts de normalisation MASIF et FIPA ne sont pas complètement différents. Les deux normes mettent l'accent sur l'interopérabilité entre les systèmes d'agents. Cependant, chacune a sa propre façon de réaliser cette interopérabilité. La norme MASIF utilise la mobilité d'agents tandis que FIPA se concentre sur la communication d'agents. En ce qui concerne les caractéristiques de deux normes, nous avons constaté que les deux sont complémentaires [Islam et al., 2010].

Les normes MASIF et FIPA ont apporté un gain considérable à la communauté des agents mobiles. Cependant, ces normes présentent des limites. En effet, aucune de ces deux normes ne spécifie la structure interne de l'agent mobile, par exemple. Bref, les deux normes présentent une spécification incomplète d'agents mobiles. Dans la prochaine section, nous allons présenter les formalismes d'agents mobiles qui tentent d'apporter des solutions à cette situation.

1.4. Formalismes d'agents mobiles

Les agents mobiles constituent un concept utilisé dans différents domaines, il est donc difficile d'en trouver une formalisation unique. Dans la littérature, il existe de nombreux formalismes pour la modélisation des systèmes à base d'agents mobiles. En effet, ces systèmes peuvent être modélisés à différents niveaux d'abstraction (structurelle versus comportementale) en utilisant des notations très variées (semi-formelle versus formelle).

1.4.1. Modélisation structurelle

Une modélisation structurelle permet d'illustrer l'architecture générale d'un système, ses composants ainsi que ses différents types de liens d'interactions. Dans [Muscutariu et al., 2001], les auteurs ont, d'une part, défini un ADL (Architecture

Description Language) permettant de modéliser l'infrastructure afin de supporter l'exécution des agents mobiles conformément au standard MASIF [Object Management Group, 2000]. Ils ont, d'autre part, spécifié les éléments nécessaires pour la représentation des mécanismes permettant une transparence à la distribution comme l'accès, la persistance, la migration des agents et leur localisation. ADL est défini comme étant un profil UML (Unified Modeling Language) nommé «MASIF-DESIGN» qui spécifie les concepts de «région», «cœur d'agence» par des sous-systèmes stéréotypés.

Une plate-forme conforme à la norme MASIF définit les éléments suivants : une région, un système d'agents, un cœur d'agence, une place et un agent. Chaque élément de la plate-forme offre un ensemble des opérations qui représente les interactions entre les éléments. Ces interactions forment le raffinement nécessaire pour une spécification comportementale, le raffinement qui détaille la spécification opérationnelle d'un système d'agents.

Dans [Chhetri et al., 2006], les auteurs proposent une spécification d'une ontologie pour la modélisation des systèmes à base d'agents mobiles. Dans leur article, les auteurs identifient des concepts clés pour la modélisation de systèmes d'agents mobiles. L'identification des principaux concepts et leurs interrelations est généralement une étape préalable à tout exercice de modélisation conceptuelle et reste le précurseur de la spécification formelle. Les concepts identifiés comme base pour décrire la mobilité d'agents sont : les rôles, les agents, le domicile, les itinéraires, les visites, les tâches, Y emplacement, les contraintes de migration et les ressources.

1.4.2. Modélisation comportementale

La modélisation comportementale met l'accent sur la manière dont les agents mobiles évoluent et interagissent dans leur environnement réparti. Dans [Kusek et al., 2005], les auteurs utilisent des diagrammes de séquence d'UML pour modéliser la mobilité d'agents en s'appuyant sur trois concepts clés d'agents : l'emplacement courant, le trajet de mobilité et l'emplacement de création.

Dans [Loukil et al., 2006], les auteurs proposent une notation MA-UML (Mobile Agent UML) pour modéliser l'aspect dynamique des agents mobiles. MA-UML permet des extensions des diagrammes d'activités, d'état-transition et de séquence d'AUML (Agent UML). Pour établir le lien entre un agent et son emplacement, le concept de localisation a été introduit au niveau du diagramme d'activités. MA-UML étend le diagramme d'état-transition pour supporter la modélisation d'un plan d'itinéraire des agents. En plus, MA-UML étend le diagramme de séquence d'AUML

afin de supporter la spécification des différentes interactions qui peuvent se présenter entre les agents mobiles, les agents stationnaires, les places (emplacement d'exécution des agents mobiles) et les ressources.

D'autres travaux s'intéressent à des notations mathématiques (des méthodes formelles) pour formaliser l'aspect comportemental des agents mobiles. Les méthodes formelles sont des techniques permettant de raisonner rigoureusement, à l'aide de logique mathématique, sur les systèmes, afin de démontrer leur validité par rapport à une certaine spécification. Dans ce contexte, la formalisation d'agents mobiles fait appel d'une part aux systèmes de transitions [McCann et al., 1998] tels que les réseaux de Petri de haut niveau et d'autre part aux algèbres de processus avec comme base le n-calcul. Dans [Sewell et al., 1999], une communication entre agents mobiles est formalisée en utilisant une extension du n-calcul. Du point de vue de la mobilité, on peut aussi raisonner sur la localisation des agents. Ainsi dans le langage Klaim [Bettini et al., 2002], les propriétés des agents sont exprimées dynamiquement en fonction de l'évolution de leur localisation. Dans sa thèse, Loulou [Loulou, 2010] a proposé une spécification formelle des systèmes d'agents mobiles basée sur la notation Z [Woodcock et al., 1996].

1.4.3. Synthèse de formalismes d'agents mobiles

Les travaux formels focalisent sur la description des agents mobiles ne possèdent pas les mêmes visions certes, néanmoins, ils cherchent tous à pouvoir offrir aux développeurs la possibilité de raisonner sur certaines propriétés. Cependant, nous avons décelé quelques limites.

En effet, la plupart de ces travaux sont orientés plutôt comportement et ceci par la modélisation des protocoles de communication et de migration des agents. En outre, ces modélisations souffrent d'un manque de couverture des concepts qui décrivent un système à base d'agents mobiles (e.g. les services, les ressources, etc.). Il y a même des concepts différents (agent mobile et système d'agents) qui ont été modélisés par la même représentation. Nous avons remarqué, également, que la majorité de ces travaux limitent la représentation de l'agent mobile à un simple objet ou processus. Une telle représentation est dépourvue de tous les concepts nécessaires pour exprimer l'autonomie et l'aspect cognitif de l'agent (telles que les croyances, les connaissances et les compétences).

Nous avons aussi remarqué essentiellement que la majorité de ces travaux s'intéressent à modéliser la mobilité des agents sans toutefois accorder une importance aux problèmes de la sécurité.

1.5. Conclusion

Dans ce chapitre, nous avons présenté les différentes technologies utilisées pour la répartition et la communication entre les différentes entités d'une application répartie. Ainsi, nous avons situé la technologie d'agents mobiles par rapport aux différentes techniques présentées. Nous avons aussi présenté les efforts de standardisation des plates-formes d'agents mobiles, à savoir les normes MASIF et FIPA. Nous avons aussi présenté les travaux sur la modélisation des systèmes à base d'agents mobiles selon deux grandes classes : la modélisation structurelle et la modélisation comportementale. Les agents mobiles offrent des avantages, mais présentent aussi des difficultés par rapport aux paradigmes de programmation plus classiques. Pour illustrer les avantages offerts par l'utilisation des agents mobiles, nous avons présenté les domaines d'application comme la réduction du trafic réseau, les calculs indépendants et la tolérance aux fautes physiques. Ils permettent aussi de mieux caractériser certaines applications.

Certes la technologie des agents mobiles offre des avantages, mais elle vient aussi avec son lot d'inconvénients dont les principaux sont :

- Le manque de standardisation entre le grand nombre de plates-formes d'agents mobiles. Malgré les efforts de standardisation de MASIF et FIPA, cette vaste gamme de plates-formes ne facilite pas l'interopérabilité d'agents issus de différents organismes. Il en résulte une très grande incompatibilité entre les différentes plates-formes [Romito, 2012].
- La complexité de déverminer des programmes se déplaçant d'une machine à une autre.
- La difficulté de sécuriser les plates-formes d'agents mobiles : Le plus gros obstacle actuel à l'utilisation de cette technologie [Roth, 2004] demeure le manque d'une protection d'agents mobiles contre les attaques de plates-formes malveillantes.

La sécurité des agents mobiles constitue donc un problème qui n'est pas encore intégralement résolu. C'est d'ailleurs le principal argument avancé pour expliquer la faible utilisation de ce paradigme. En effet, les agents mobiles représentent un nouveau champ d'investigation pour le domaine de recherche en sécurité. Dans le chapitre suivant, nous étudions le problème de sécurité des agents mobiles en mettant en évidence les différents types d'attaques possibles provenant des hôtes ou bien des agents mobiles. Nous discutons ensuite la protection des deux acteurs en mettant l'accent sur celle de l'agent mobile.

CHAPITRE 2

Sécurité d'agents mobiles

2.1. Introduction

La puissance de la technologie des agents mobiles dans la résolution des problèmes complexes résulte du fait que les agents, grâce à leur autonomie, mobilité et adaptabilité, peuvent réaliser leurs buts d'une manière flexible en se servant d'une interaction locale et/ou distante avec les autres agents sur le réseau. Cependant, cette technologie a introduit plusieurs problèmes sérieux tels que le problème de l'hétérogénéité, celui de maintien les communications ou encore celui de la gestion des ressources partagées et enfin le problème de sécurité des agents mobiles.

Notamment, le problème de sécurité constitue un frein à l'expansion de cette technologie. Dans un système d'agents mobiles, les plates-formes, par le fait qu'elles exécutent le code des agents mobiles, sont menacées par leurs attaques. D'autre part, lors de leurs déplacements à travers le réseau, les agents mobiles peuvent visiter des plates-formes malveillantes et sont vulnérables aux nombreuses attaques qui peuvent nuire à leur bon fonctionnement. De nombreux travaux de recherche ont vu le jour pour traiter le problème de la sécurité des agents mobiles.

Dans ce chapitre, nous exposons le problème de la sécurité dans les systèmes d'agents mobiles. Dans le but d'une meilleure compréhension, nous présentons en premier lieu, l'aspect sécuritaire dans le cadre des systèmes d'agents mobiles. Nous verrons les différentes attaques possibles. Nous décrivons ainsi, les différentes mesures de sécurisation envisageables après avoir dénoter les besoins de la sécurité requis. Ensuite, nous exposons les approches les plus connues ayant traité le problème dont nous parlons. Enfin, nous achevons le chapitre avec une conclusion.

2.2. Exigences de sécurité pour les agents mobiles

Comme dans tout système d'information, pour assurer la sécurité d'un système d'agent mobile, il faut répondre aux besoins que l'organisation "ISO" (International Standard Organisation) a fait ressortir dans ses études sur les sécurités des réseaux. Il s'agit de l'authentification, la confidentialité, l'intégrité, la disponibilité, et la responsabilité. Cette section fournit une vue d'ensemble brève de ces exigences de

la sécurité [Alfalayleh et al., 2004], [Pillou, 2005].

2.2.1. Authentification

L'authentification est la procédure qui consiste à vérifier l'identité de l'agent mobile ou la plate-forme d'exécution afin d'autoriser l'accès de cette entité à des ressources. L'authentification est souvent décrite comme le service le plus urgent à intégrer dans les réseaux de télécommunication de nos jours. Elle est aussi indispensable pour d'autres services de sécurité telle que le contrôle d'accès. Il y a trois méthodes permettant la vérification de l'identité. Elles reposent sur la connaissance :

- D'un élément connu de l'individu : qui est à la fois secret et difficile à deviner comme un mot de passe.
- D'un élément possédé par l'individu : peut-être une carte magnétique.
- Un élément au sujet de l'individu : ce sont en général les systèmes biométriques.

2.2.2. Confidentialité

N'importe quelle donnée privée enregistrée sur une plate-forme ou portée par un agent doit demeurer confidentielle. Les oreilles indiscrètes peuvent recueillir des informations au sujet des activités de l'agent non seulement du contenu des messages échangés, mais également du flux de messages d'un agent à un autre agent ou à des agents différents. Un agent acheteur peut, par exemple, envoyer trois messages à un agent vendeur, suivi d'un message à sa plate-forme d'origine, et conclure la transaction avec deux messages à l'agent vendeur. L'agent vendeur peut envoyer deux messages à une agence de vérification de crédit et conclure avec un message à son agent d'opérations bancaires. Bien que le contenu des messages n'ait été jamais révélé, une oreille indiscrète peut pouvoir impliquer que les deux agents ont avec succès terminé une vente d'un certain produit.

Les agents mobiles peuvent également vouloir maintenir leur emplacement confidentiel. On doit permettre à des agents de décider si leur présence sera publiquement disponible, et les plates-formes peuvent imposer différentes politiques de sécurité sur les agents qui ont choisi d'être anonymes. Des agents mobiles qui font un shopping pour des biens et des services souhaitent le faire ainsi l'intimité, mais quand une transaction financière doit être effectuée la plate-forme peut exiger une certaine forme d'authentification.

2.2.3. Intégrité

La plate-forme d'agent doit protéger des agents contre la modification non autorisée de leur code, état, et leurs données et s'assurer que seulement les agents autorisés effectuent des modifications sur les données partagées. L'agent lui-même ne peut pas empêcher une plate-forme malveillante d'altérer son code, état, ou données, mais l'agent peut prendre des mesures pour détecter cette altération.

L'exécution sécurisée des systèmes des agents mobiles dépend également de l'intégrité des plates-formes locales et distantes. Un hôte malveillant peut facilement compromettre l'intégrité d'un agent mobile qui visite une plate-forme à distance. L'utilisateur et le développeur de l'agent mobile peut ne pas avoir aucune connaissance du comportement d'une plate-forme malveillante et les effets qu'il aura sur le code de l'agent, l'état et les données qui sont sous le contrôle complet de la plate-forme. Une plate-forme malveillante peut altérer de façon subtile la séquence d'exécution du code de l'agent et engendrer des modifications difficiles à détecter aux résultats calculés. Une plate-forme malveillante peut s'ingérer dans les transactions entre les agents représentant différents organismes et falsifier les vérifications rétrospectives. Une des conséquences est que chaque organisation impliquée dans la transaction blâme l'autre de ne pas avoir fourni les biens ou les services promis.

Les attaques orientés-objectif contre les communications d'agent visent à compromettre l'intégrité des messages en changeant leur contenu, en substituant le message entier, en réutilisant un vieux message, en effaçant le message, ou en changeant la source ou la destination de message. Les attaques malveillantes sur l'intégrité des transmissions d'agent peuvent être plus graves que des erreurs de transmission résultant des voies de transmission défectueuses ou limitées.

2.2.4. Disponibilité

La plate-forme d'agent doit pouvoir assurer la disponibilité des données et des services aux agents mobiles. La plate-forme doit être capable de fournir le contrôle de concurrence, le support pour l'accès simultané, la gestion de blocage, et l'accès exclusif au besoin. Les données partagées doivent être disponibles sous une forme utilisable, la capacité doit répondre aux besoins de service. La plate-forme d'agent doit être capable de détecter et récupérer les pannes du matériel et logiciel. En outre, la plate-forme peut exiger des agents d'assumer la responsabilité de leur propre faute.

La plate-forme d'agent doit pouvoir traiter les requêtes des centaines ou des milliers de visites d'agents mobiles, sinon il y aura un risque de créer un déni de service involontaire. Au cas où une plate-forme ne pourrait pas manipuler son chargement de calcul et de transmission, elle doit fournir le fonctionnement en mode dégradé des services et informer les agents mobiles qu'elle ne peut plus fournir le niveau et la qualité du service prévu.

L'assurance de la confidentialité et l'intégrité créent des restrictions sur la disponibilité et également le temps d'exécution. Le cryptage des agents et messages en transit peut imposer des retards inacceptables dans les environnements où la réponse à temps est exigée.

2.2.5. Contrôle d'accès et responsabilité

Le contrôle d'accès détermine qui doit avoir accès à quelle ressource, autrement dit garantir qu'une ressource n'est accessible que par les entités autorisées. Toutefois, contrôler l'accès ne se résume pas à limiter l'accès. Un bon contrôle d'accès nécessite que les entités autorisées puissent accéder à la ressource demandée très facilement tout en assurant les plus grandes difficultés d'accès aux utilisateurs non autorisés. Ainsi, chaque entité plate-forme ou agent dans le système doit être tenue responsable pour ses actions. Pour cela chacune de ces entités doit être identifiée de façon unique, certifiée, et authentifiée. Cette propriété exige la maintenance d'un fichier "audit" d'événements critiques. Ces événements sont définis par la politique de sécurité adoptée. Ces fichiers d'audit doivent être eux-mêmes protégés avec soin contre les accès non autorisés et les altérations.

2.3. Problèmes de sécurité d'agent mobile

Dans cette section, nous discutons les problèmes de sécurité d'agent mobile [Borselius, 2002] basés sur les caractéristiques décrites dans la section 1.2.3.2.

2.3.1. Exécution d'agent

Les agents mobiles ont le besoin de s'exécuter quelque part. Un hôte, l'environnement immédiat d'un agent, est finalement responsable de l'exécution et la protection correctes de l'agent. Ceci mène à la question d'où des décisions de contrôle d'accès devraient être exécutées et imposées. Est-ce que l'agent comporte toutes les informations exigées ? Et si oui, est-il autorisé ?

L'environnement doit également avoir besoin d'une certaine protection contre les agents qu'il accueille. Un agent doit, par exemple, être empêché de lancer une attaque par déni de service en consommant toutes les ressources sur un hôte

(serveur), et ainsi empêcher l'hôte d'effectuer autres travaux. Les problèmes de sécurité liés à l'hôte exécutant deviennent plus évidents pour les agents qui sont mobiles.

2.3.2. Autonomie

L'autonomie peut introduire des problèmes de sécurité sérieux. L'agent peut, à titre d'exemple, être capable de réaliser des transactions et être la cible de différentes attaques qui pourraient l'empêcher de remplir ses engagements. Il est important de mentionner aussi que les virus Internet possèdent également la propriété d'autonomie, qui leur permet de se propager efficacement sans exiger aucune interaction humaine. L'autonomie, donc, peut être utilisée pour des buts malveillants si elle n'est pas correctement contrôlée.

2.3.3. Communication

Quelques implémentations des systèmes multi-agents MAS (Multi Agent System) supposent que la sécurité est fournie d'une manière transparente par une couche inférieure. Cette approche est suffisante dans un système fermé où les agents peuvent se faire confiance et le seul souci concerne des parties malveillantes externes. Cependant, les agents dans un système ouvert ont besoin d'être sensible à la sécurité (security awareness), c.-à-d. ils doivent pouvoir prendre des décisions basées sur d'où l'information provient et comment elle est protégée. He [He, 1998] suggère que la cryptographie à clé publique et une infrastructure supportant la clé publique peuvent être utilisées comme outils importants de transmission inter-agents.

Avec la mise en place d'une infrastructure à clé publique en place, les protocoles et les mécanismes de sécurité déjà développés pour d'autres applications peuvent être amenés à se conformer aux conditions des systèmes multi-agents de fournir l'authentification, la confidentialité, et l'intégrité des données.

2.4. Types d'attaques

Les attaques classiques dans les systèmes distribués sont diverses. Elles peuvent être passives ou actives. En ce qui concerne, le système à base de code mobile et particulièrement le cas de l'agent mobile où la capacité de mobilité de cet agent sur le réseau engendre d'autres types d'attaques, ces dernières peuvent être causées aussi bien par l'agent ou par l'hôte accueillant cet agent. Nous focalisons dans ce qui suit sur celles liées à l'interaction entre l'agent mobile et l'hôte visité.

On trouve dans la littérature une multitude de classifications des attaques dans les systèmes d'agents mobiles [Jansen et al., 2000a]. Mais on distingue principalement trois grandes classes d'attaques.

2.4.1. Attaques des plates-formes malveillantes

Ce type d'attaque vient de l'environnement de calcul accueillant et responsable de l'exécution de l'agent. Elle souligne que si un hôte exécute l'agent il aura l'autorisation d'accéder à son code et à son état. Par conséquent, l'agent est exposé aux différentes opérations légales ou illégales.

Fritz Hohl [Hohl, 1998] définit un hôte malveillant comme étant un hôte capable d'exécuter un agent provenant d'un autre hôte, et qui essaye de nuire au bon fonctionnement de l'agent mobile d'une manière quelconque. Dans [Hohl, 1998], les différentes attaques d'un hôte contre un agent mobile sont identifiées. Nous détaillons ces différentes attaques dans ce qui suit :

Espionnage (Eavesdropping) : la menace de l'espionnage classique comporte l'interception et la surveillance des transmissions secrètes. Cependant, cette menace est encore aggravée dans les systèmes des agents mobiles parce que l'hôte peut surveiller non seulement la transmission, mais aussi chaque instruction à exécuter par l'agent, toutes les données non codées ou publiques apportées, et toutes les données produites sur l'hôte. Puisque l'hôte a accès au code, à l'état, et aux données de l'agent, l'agent visiteur doit être circonspect du fait qu'il peut exposer des algorithmes de propriété industrielle, des secrets commerciaux, des stratégies de négociation, ou d'autres informations sensibles. Nous détaillons les différents types d'attaques d'espionnage dans ce qui suit :

- Espionnage du code : pour exécuter l'agent mobile, l'hôte doit être en mesure de lire son code. La connaissance du code de l'agent mobile mène à la connaissance de la stratégie de l'exécution de l'agent, la connaissance des structures physiques du code et des données dans la mémoire de l'hôte et parfois, la connaissance d'une partie des données de l'agent mobile.
- Espionnage des données : la lecture des données privées d'un agent mobile par un hôte est très critique vu que cette attaque ne laisse pas de trace contrairement à l'attaque de modification des données. C'est un problème qui concerne certaines classes de données dont la simple connaissance implique une perte de confidentialité. La clé privée est un exemple de cette catégorie de données.

- Espionnage de l'état : dès que l'hôte prend connaissance du code de l'agent et de ses données, il peut déterminer l'étape suivante de son exécution et même si on protège les données utilisées par l'agent mobile. Il est difficile de protéger l'information concernant son état d'exécution. C'est un problème important parce qu'avec la connaissance du code, l'hôte malveillant peut déduire des informations sur l'état de l'agent. Par exemple, il peut savoir si une offre est meilleure ou non en observant simplement l'état d'exécution sans connaître les données de l'agent.

Altération (Alteration) : puisqu'un agent peut visiter plusieurs hôtes sous divers domaines de sécurité durant sa vie, les mécanismes doivent être en place pour assurer l'intégrité du code, état et données de l'agent. Un hôte malveillant doit être empêché de modifier le code, l'état, ou les données d'un agent sans être détecté. La détection des modifications malveillantes sur l'état d'un agent pendant son exécution ou les données produites lors de sa visite d'un hôte malveillant n'a pas encore trouvé de solution définitive.

Mascarade (Masquerade) : une plate-forme malveillante peut masquer son identité en trompant un agent mobile et s'identifier comme sa vraie destination. La mascarade est une des attaques qui introduisent d'autres types d'attaques, tels que l'espionnage, l'extraction des informations secrètes, l'exécution incorrecte… Une plate-forme qui se masque comme une troisième réception de confiance peut en plus de leurrer des agents naïfs, nuire aux plates-formes dont l'identité a été exploitée.

Déni de service (Denial of Service) : quand un agent arrive à une plate-forme cliente, il attend qu'elle exécute les demandes de l'agent fidèlement, en lui allouant les ressources demandées, et se conformer aux contraintes exigées dans le contrat mis en accord. Cependant, une plate-forme malveillante peut ignorer les demandes de l'agent mobile, introduire des délais inacceptables pour les tâches critiques, ou simplement ne pas exécuter l'agent mobile, ou même terminer l'agent sans notification dans le but de gâcher le travail de l'agent mobile.

Exécution incorrecte du code : sans avoir à changer ni le code ni l'état d'exécution de l'agent mobile, une plate-forme malveillante peut modifier la façon dont elle exécute le code de l'agent. Elle peut par exemple retourner une fausse valeur quand elle procède à la comparaison de son prix avec un prix nominal fixé par la plate-forme d'origine. Ou bien ré-exécuter l'agent après avoir copié, une partie de lui ou un message de celui-ci. Par exemple, une plate-forme malveillante peut ré-exécuter un message d'achat d'un produit ou de payement d'une facture, ou tout simplement ré-exécuter l'agent mobile avec des données différentes.

2.4.2. Attaques des agents malveillants

Cette première situation est perçue quand l'agent peut mettre en danger l'hôte accueillant sur lequel il s'exécute. Ce genre d'attaque a été largement étudié depuis de nombreuses années. On distingue principalement quatre types d'attaques [Bob, 2000] :

Mascarade : quand un agent non autorisé utilise l'identité d'un autre agent, il est décrit une mascarade (déguisement). L'agent en déguisement peut se présenter comme étant un agent autorisé dans l'effort d'accéder aux services et aux ressources auxquels il n'a pas droit. L'agent en déguisement peut également se présenter comme étant un autre agent non autorisé pour éviter le blâme de toutes actions pour lesquelles il ne veut pas être jugé responsable. Un agent en déguisement peut endommager la confiance que l'agent légitime a établie dans une communauté d'agent et sa réputation associée.

Déni de service : le déni de service DoS est, d'une manière générale, l'attaque qui vise à rendre une application informatique incapable de répondre aux requêtes de ses utilisateurs. Les agents mobiles peuvent lancer une attaque par déni de service en consommant une quantité excessive des ressources de l'hôte. Cette attaque peut être lancée intentionnellement avec des scripts en exploitant les vulnérabilités du système, ou involontairement par des erreurs de programmation.

Accès non autorisé : des mécanismes de contrôle d'accès sont employés pour empêcher les utilisateurs non autorisés d'avoir accès aux services et ressources. L'application des mécanismes appropriés de contrôle d'accès exige à l'hôte d'authentifier d'abord l'identité d'un agent mobile avant qu'il soit instancié sur l'hôte. Un agent qui a l'accès à un hôte et à ses services sans avoir l'autorisation appropriée peut nuire à d'autres agents et à l'hôte lui-même. Un hôte qui accueille des agents représentant divers utilisateurs et organismes doit s'assurer que les agents n'ont pas la possibilité de lire ou d'écrire des données pour lesquelles ils n'ont aucune autorisation.

Répudiation (Repudiation) : la Répudiation ou encore le Reniement se produit quand un agent, participant à une transaction, arrive à causer une rupture de la communication qui a eu lieu entre les éléments du système. La répudiation que ce soit causer d'une manière accidentelle ou planifiée peut mener à de sérieux problèmes qui ne peuvent pas être résolus facilement à moins que les contre-mesures adéquates soient mises en place. Si une plate-forme de l'agent ne peut pas prévenir les agents malveillants de répudier une transaction, elle doit être en mesure pour résoudre les désaccords occasionnés.

2.4.3. Attaques d'agents contre agent

Cette catégorie représente l'ensemble de menaces dans lesquelles les agents exploitent des faiblesses de sécurité ou lancent des attaques contre d'autres agents. Cet ensemble de menaces inclut : la mascarade, le déni de service et l'accès non autorisé.

Mascarade : la transmission inter-agent peut avoir lieu directement entre deux agents ou peut exiger la participation de l'hôte et ses services fournis. Dans ces deux cas, un agent peut essayer de déguiser son identité dans un effort de tromper l'agent avec lequel il communique. Un agent A peut se présenter, par exemple, en tant que vendeur des marchandises et de services bien connus et essaie de convaincre un autre agent B confiant de lui fournir les numéros de carte de crédit, les données de compte bancaire, une certaine forme d'argent numérique, ou d'autres informations personnelles. Le fait de se déguiser étant un autre agent nuit à l'agent B qui est trompé et l'agent dont l'identité a été assumée, particulièrement dans des sociétés d'agent où la réputation est valorisée et utilisée comme un moyen d'établir la confiance.

Déni de service : les agents peuvent lancer l'attaque par déni de service contre d'autres agents. Par exemple, envoyer des messages à un autre agent à plusieurs reprises peut saturer les sous-programmes de gestion de messages du destinataire. Les agents qui sont spammés peuvent choisir de bloquer des messages des agents non autorisés, mais même cette tâche exige certains traitements par l'agent ou son proxy de transmission. Si un agent est chargé par le nombre de cycles CPU qu'il consomme sur une plate-forme, spammer un agent peut causer à l'agent spammé de payer un coût monétaire en plus d'un coût d'exécution.

Accès non autorisé : si l'hôte n'a pas un mécanisme de contrôle, un agent peut directement gêner un autre agent en appelant ses méthodes publiques (par exemple, tentative de débordement de tampon, remise à l'état initial, etc.), ou en accédant et en modifiant les données ou le code de l'agent. La modification du code d'un agent est une forme d'attaque particulièrement insidieuse, puisqu'elle peut changer radicalement le comportement de l'agent (par exemple, transformant un agent de confiance en agent malveillant). Un agent peut également obtenir des informations sur les activités d'autres agents en utilisant des services de plate-forme pour écouter clandestinement leurs transmissions.

2.5. Techniques de protection dans les systèmes d'agents mobiles

Le problème de sécurité représente le talon d'Achille des agents mobiles. Il constitue un frein à l'utilisation réelle de cette technologie [Vigna, 1998] [Borselius, 2002], [Alfalaleh et al., 2004]. Leur emploi dans un système basé agent peut exiger, d'une part la protection des ressources et des données des machines hôtes en limitant les droits d'accès et la consommation des ressources, et d'autre part, la préservation de l'intégrité et de la confidentialité des agents eux-mêmes et de leurs communications.

La sécurité des agents mobiles constitue donc un problème qui n'est pas encore intégralement résolu. C'est d'ailleurs le principal argument avancé pour expliquer la faible utilisation de ce paradigme [Roth, 2004], [Vigna, 2004]. En effet, les agents mobiles représentent un nouveau champ d'investigation pour le domaine de recherche en sécurité. Il existe principalement deux axes de recherches : d'une part, la protection des hôtes vis-à-vis des agents malveillants et d'autre part, la protection des agents vis-à-vis des hôtes malveillants. Les sections 2.5.1 et 2.5.3 de ce chapitre abordent en détail ces deux volets de la sécurité des agents mobiles.

2.5.1. Techniques de protection des plates-formes

La protection des hôtes visités contre des attaques menées par des agents mobiles malveillants est un problème qui est aujourd'hui assez bien maîtrisé. En effet, un système d'exécution devient vulnérable dès qu'il exécute un code venant de l'extérieur s'il n'est pas protégé. Un agent peut facilement violer les propriétés de confidentialité ou d'intégrité s'il a accès à la mémoire ou aux fichiers de la machine hôte. L'agent peut bloquer le système en réalisant des opérations non conformes, consommer de manière illimitée et incontrôlée des ressources, se cloner indéfiniment ou bien migrer sans fin.

Les techniques de protection des hôtes suivent, aujourd'hui deux directions. La première direction s'occupe de l'enrichissement graduel de l'infrastructure du code mobile par l'authentification, l'intégrité des données et par des mécanismes de contrôle d'accès. Tandis que la seconde direction se consacre à la vérification de la sémantique du code mobile. Nous retrouvons ces deux directions dans les techniques présentées ci-dessous :

L'authentification des agents : le problème de l'authentification des agents mobiles est similaire à celui qui se pose en milieu distribué. Ici, on a deux buts : vérifier l'intégrité du code et authentifier ses auteurs et utilisateurs. Greenberg et Byington [Greenberg et al., 1998] proposent de signer l'agent mobile numériquement avec

un algorithme cryptographique à clé publique. Cet algorithme peut soit générer un certificat qui assure l'intégrité du code et qui permet d'authentifier le propriétaire et le producteur de l'agent mobile pour l'hôte, soit crypter l'agent de sorte que seul le réceptionnaire puisse le décoder. Cela assure la confidentialité de l'agent mobile.

Le carré de sable (Sandboxing) [Vigna, 1998] : consiste à isoler un programme dans son propre espace de fautes de façon logicielle. L'environnement est restreint en termes de privilèges et de ressources. Le code est exécuté dans une sorte de « carré de sable ». Un agent ne pourra pas modifier la plate-forme ni les autres agents qui roulent sur la plate-forme. Par exemple, un Applet Java ne peut pas accéder au système de fichier local.

Contrôle d'accès et d'autorisation [Schneider et al., 2001] : donne des autorisations à un agent mobile pour un certain nombre de ressources en fonction du résultat de son authentification. Pour savoir quelles autorisations donner à quel agent, le moniteur de référence applique les règles de la politique de sécurité. Ces règles touchent toutes les ressources dont un agent peut avoir besoin : accès au réseau, aux disques, etc. Cette technique est différente du « carré de sable », car elle permet, d'une part, d'avoir une granularité plus fine sur la protection et l'autorisation et, d'autre part, une adaptation des droits d'accès en fonction des agents. Le problème du carré de sable ne se pose plus, car une application avec des besoins en ressources forts ou particuliers pourra les avoir si la politique de sécurité le permet.

La vérification du code : permet d'obtenir une garantie sur sa sémantique à travers l'analyse de sa structure, ou de son comportement pour un agent, en fonction d'une politique de sécurité donnée. Elle permet de modifier en conséquence son statut (par exemple de "fiable" à "non fiable"). Pour cela on peut utiliser l'approche PCC (Proof Carrying Code) [Necula et al., 1998]. Lors de la mise en route de l'agent, son créateur fournit un ensemble de preuves intégrées transportées par l'agent. Ces preuves garantissent le comportement de l'agent en fonction de critères de sécurité des hôtes à visiter [Jansen et al., 2000a]. Elles utilisent un ensemble d'axiomes et de règles de réécriture suivant la logique choisie et partagée par le producteur du code et son consommateur. Lorsque l'agent commence une nouvelle visite, l'hôte récupère la preuve et vérifie si elle correspond à sa politique de sécurité. L'hôte n'exécute le code que seulement si la preuve est correcte. Ceci peut être vu comme une forme de vérification du type du programme puisque la preuve est directement dérivée du code. Cette technique se base pour l'instant sur des propriétés de typage et la preuve est fournie par le compilateur. PCC garantit la sûreté du code reçu en supposant qu'il n'y ait aucune faille dans le générateur de vérification de conditions,

dans les axiomes, dans les règles de typage et dans le contrôleur de preuve [Appel, 2001].

L'évaluation d'état : est utilisée pour garantir qu'un agent n'est pas devenu malicieux ou modifié en raison de l'altération de son état au niveau d'un hôte non fiable. Cette technique est employée autant pour la protection de l'hôte que celle de l'agent mobile lui-même. Elle utilise des fonctions d'évaluation (appraisal functions) [Farmer et al., 1996a] qui déterminent les privilèges accordés à un agent en se basant sur des facteurs conditionnels et sur des invariants : un agent dont l'état viole un invariant peut n'avoir aucun privilège alors qu'un agent dont l'état ne correspond pas à des facteurs conditionnels peut obtenir un ensemble restreint de privilèges.

Historique des hôtes (Path Histories) [Ordille, 1996] : l'idée est d'évaluer la confiance qu'a un agent dans un hôte à partir de son identité et des plates-formes sur lesquelles il s'est exécuté auparavant. Pour cela, il est nécessaire de mettre à jour un journal avec l'identité de toutes les plates-formes qui ont été visitées par l'agent. Ce journal est protégé avec des algorithmes cryptographiques pour éviter toutes tentatives de modifications : à chaque fois qu'un agent mobile migre, la plate-forme signe le journal. Lorsqu'un agent arrive sur une nouvelle plate-forme, cette dernière décide si elle l'exécute ou non et quelles ressources elle lui accorde en fonction de l'historique des hôtes traversés par l'agent. Cette technique est efficace dans le sens où, avec la signature numérique, une entrée dans l'historique est non répudiable. Le problème est que lorsque le nombre d'hôtes visités augmente, l'historique peut prendre un poids prohibitif.

2.5.2. Synthèse

De nombreux travaux ont souligné l'importance de la protection des hôtes contre des agents mobiles malicieux [Bella vista, 2004], [Bierman et al., 2002]. Certaines de ces techniques de protection telles que celle du Sandboxing, ont été employées pendant longtemps et sont bien comprises. Toutefois, aucune des techniques existantes ne fournit une solution optimale pour tous les scénarios. Par exemple, le Sandboxing fournit un niveau élevé de sécurité, mais il est très restrictif puisque très peu d'applications peuvent fonctionner dans un environnement aussi restreint. Néanmoins, une combinaison de diverses techniques peut déboucher sur des solutions assez puissantes. Par exemple, dans Java 2 le Sandboxing a été employé en combinaison avec la signature du code et un contrôle d'accès fin.

Dans l'optique de permettre à cette thèse d'ouvrir des horizons de recherche touchant au problème de sécurité suscité par l'emploi des agents mobiles, nous

avons tenu à présenter les principales techniques utilisées dans la protection des hôtes visités par des agents mobiles potentiellement malveillants. Ce problème n'est cependant pas traité dans cette thèse. La protection de l'agent mobile contre les hôtes malveillants est par contre, un problème ouvert qui nous intéresse.

2.5.3. Techniques de protection des agents mobiles

Les agents mobiles sont exposés à diverses menaces de la part des hôtes visités [Bierman et al., 2002]. Ce problème est difficile, car l'environnement visité a un contrôle total sur l'exécution de l'agent mobile. La protection d'un agent mobile à l'encontre d'hôtes malicieux revient à protéger principalement son exécution, son intégrité et sa confidentialité. Protéger l'exécution d'un agent mobile signifie qu'il faut s'assurer qu'il n'est pas à la merci de l'hôte et qu'il ne peut pas être détourné vers d'autres destinations, être privé de ressources ou être terminé de façon prématurée. Protéger l'intégrité d'un agent exige la détection de la modification de son code et de son état due à une mauvaise exécution par un hôte malicieux. Enfin, protéger la confidentialité revient à cacher le code et l'état de l'agent mobile vis-à-vis du site qui l'exécute. Les techniques de protection d'un agent sont subdivisées essentiellement en deux catégories les techniques préventives et les techniques de détection.

2.5.3.1. Techniques de détection

Dans cette section, nous présentons une analyse bibliographique des mécanismes de sécurité les plus importants pour détecter les failles de sécurité contre les agents mobiles. Le mécanisme de détection vise la détection des modifications illégales du code, de l'état et du flux d'exécution de l'agent mobile. Bien que le code statique puisse être facilement protégé par la signature digitale, l'état et le flux d'exécution sont des composants dynamiques difficiles à protéger. C'est pourquoi un ensemble d'approches de détection sont proposées :

Signatures numériques (Digital signatures) : les signatures numériques peuvent être utilisées pour détecter la modification de code et des données d'agent mobile. La signature de code a été proposée par Janson [Jansen, 2000b] comme une technique avec laquelle les plates-formes d'agent peuvent détecter l'altération de code et les données d'agent mobile. Récemment, Michel et Brazier [Michel et al., 2010] présentent également une technique pour détecter l'altération de code et des données de l'agent mobile par les plates-formes malveillantes. La protection de code et les données des agents mobiles sont des tâches difficiles. La signature du code intervient lors de la création d'un agent, son créateur le signant

numériquement afin qu'il puisse s'identifier durant ses déplacements. Cette technique permet d'obtenir une authentification de haut niveau pour les hôtes. Elle assure aussi l'intégrité du code pour l'hôte visité. Une signature digitale sert donc comme moyen de confirmation de l'authenticité de l'agent mobile, de son origine et de son intégrité. Le signataire du code est soit son créateur ou une autre entité ayant révisé le code. Cette technique est efficace pour les éléments statiques de l'agent mobile, comme le code et les données qui ne changent pas durant l'exécution de l'agent mobile, mais certains éléments de l'agent mobile sont dynamiques comme l'état d'exécution et les données qui changent durant l'exécution de l'agent. De plus, cette solution ne résout pas le risque fondamental lié à l'exécution incorrecte de code de l'agent mobile par des plates-formes malveillantes.

L'évaluation d'état (State appraisal) : Jansen [Jansen, 2001a, 2001b, 2001c] propose une technique qui permet de séparer la structure de données définissant le comportement de l'agent de son propre code. Cette structure sera définie dans un certificat conforme à la norme X.509. Dans ce certificat, on précise les droits et les responsabilités d'un agent sur un site donné. On distingue les certificats d'attributs dans lesquels sont définis le comportement de l'agent et les certificats de politique servant à décrire le comportement du site envers tous les agents qu'il accueille. L'approche protège l'agent contre une utilisation illicite en fournissant une preuve des intentions réelles de son producteur/émetteur, mais il n'offre aucune protection des données que l'agent transporte. Une autre approche proposée par Sayeb [Sayeb et al., 2002] consiste à utiliser un arbre de diagnostic permettant la détection d'une attaque possible. Dans cette approche les auteurs associent un symptôme à chaque attaque et la combinaison de ces symptômes définit un arbre. Comme symptômes nous citons : la longue durée d'exécution qui informe sur une analyse du code ou des données au cours de l'exécution, l'enregistrement temporaire de l'agent qui alerte sur une analyse possible tandis qu'un comportement anormal de l'agent suppose une manipulation du code de l'agent. L'approche se limite aux attaques dont l'origine ne prend pas en compte un éventuel mécanisme de détection, sinon il cherchera à masquer les symptômes et simuler un comportement normal. Cette approche permet la détection automatique des manipulations qui mettent un agent mobile dans un état inacceptable. Ces manipulations doivent donc être prévues et les fonctions de vérification correspondantes doivent être mises en application dans l'agent mobile, ce qui constitue une tâche difficile. De plus, la détection des manipulations par l'agent mobile durant son exécution est conditionnée par l'exécution réelle des fonctions d'évaluation, ce qui est le cas uniquement d'une plate-forme d'agent honnête.

Enregistrement d'itinéraires avec des agents coopérants (Itinerary recording with with cooperating agents) : les techniques utilisant des agents coopérants ont leurs racines dans le domaine de la tolérance aux fautes. Dans le cas où un seul agent est lancé pour accomplir une certaine tâche, il a en général beaucoup de chance de se faire attaquer par une plate-forme malveillante. Roth [Roth, 1999] présente une approche permettant l'enregistrement d'itinéraires avec des agents coopérants. Il propose de donner une tâche à deux agents qui vont parcourir un ensemble de plates-formes et qui ont des itinéraires disjoints, un des deux agents mobiles enregistrant l'itinéraire de l'autre. Le premier agent migre seulement vers les plates-formes du premier groupe tandis que le second agent migre uniquement vers les plates-formes appartenant au second groupe. Un des deux groupes est composé uniquement d'hôtes fiables. De cette façon, un des deux agents s'exécute seulement sur des plates-formes fiables. Dans ce scénario à deux agents coopérants, un agent protège l'autre des manipulations de son itinéraire à travers le réseau. Ceci est spécialement important si cet itinéraire n'est pas prédéfini par leur propriétaire, auquel cas, il est alors très difficile de détecter des tentatives de manipulation. Avant de migrer, un agent envoie à l'autre agent, à travers un canal authentifié, l'identité de la dernière plate-forme, celle de la plate-forme actuelle, ainsi que celle de la plate-forme où il se rend. L'autre agent maintient un journal de l'itinéraire et vérifie qu'il n'y a pas de contradictions. En l'an 2005, All'ee et al. [All'ee et al., 2005] ont présenté un protocole de protection de l'itinéraire amélioré pour empêcher les attaques par rejeu qui étaient indétectables par le système d'itinéraire commun présenté par Roth [Roth, 1999]. Plus récemment, Garrigues et al. [Garrigues et al., 2008] ont proposé un mécanisme de protection des itinéraires dynamiques en utilisant un itinéraire explicite et le contrôle de code. La réalisation pratique de cette technique semble être très limitée en raison de problèmes liés à l'attente du programmeur de connaître toutes les plates-formes visitées par l'agent mobile.

Ces techniques sont intéressantes. Toutefois, elle est entachée d'un certain nombre d'inconvénients. Si un agent est tué, le protocole n'est pas capable de savoir lequel des deux hôtes est responsable. Si une plate-forme déclare avoir reçu un agent d'une plate-forme et que cela n'est pas vrai, le protocole ne peut décider qui est coupable. Enfin, si un hôte reçoit deux agents qui viennent du même hôte, il est possible d'inter-changer les agents qui enregistrent l'itinéraire des deux autres. De plus, la condition de la non-coopération d'hôtes malicieux est difficile à prouver ou à vérifier. Par ailleurs, la décomposition des tâches critiques peut ne pas être facile à automatiser. Ce qui limite considérablement le déploiement de cette approche. Enfin, un agent peut être modifié par un hôte malveillant de façon à ce que l'agent coopérant ne puisse le détecter.

Encapsulation des résultats partiels (Partial result encapsulation) : la technique d'encapsulation des résultats partiels est une technique qui a pour but de découvrir toutes les infractions de sécurité possibles sur un agent mobile durant son exécution sur différentes plates-formes. Cette technique est utilisée pour encapsuler les résultats de l'exécution d'un agent au niveau de chacune des plates-formes. L'information encapsulée est utilisée plus tard pour vérifier que l'agent n'a pas été attaqué par une plate-forme malicieuse. Le processus de vérification peut être réalisé lorsque l'agent retourne vers sa plate-forme d'origine ou bien en des points intermédiaires de son itinéraire. Cette technique possède différentes implémentations. Dans certains scénarios, c'est l'agent qui réalise lui-même l'encapsulation, tandis que dans d'autres ce sont les plates-formes qui s'en chargent. Afin d'adresser certaines exigences de sécurité telles que l'intégrité, la responsabilité et l'intimité de l'agent, l'encapsulation des résultats partiels utilise différentes primitives cryptographiques telles que le cryptage, la signature digitale, l'authentification du code et les fonctions de hachage. Yee [Yee, 1999] a introduit les PRACs (*Partial Result Authentification Codes*). L'idée est de protéger l'authenticité de l'état d'un agent intermédiaire ou les résultats partiels produits par l'exécution sur l'hôte. Quand l'agent migre d'une plate-forme à une autre, il transporte avec lui les résultats de son exécution dans les plates-formes précédentes. L'encapsulation des résultats partiels permet d'aider à détecter la modification ou l'élimination de ces résultats. L'inconvénient majeur de ces approches sont la taille de l'agent mobile augmente très vite à chaque migration et peut compromettre l'intérêt de son utilisation, si le nombre de plates-formes visitées est trop grand, ainsi si l'agent retourne par la suite à une plate-forme déjà exécutée l'agent, celle-ci peut modifier son offre ainsi que toutes les offres des plates-formes suivantes.

Copies d'agent (Copying agent) : dans l'approche «*k-out-of-n threshold scheme*» [Beimel et al., 2000], le secret de la transaction est distribué entre n agents mobiles dupliqués. Ces derniers sont émis vers différents hôtes. Si la tâche à effectuer par l'agent est idempotente, c.-à-d. n'est pas affectée par le nombre de fois qu'elle est effectuée, plusieurs copies de l'agent peuvent être lancées dans le réseau. Cette technique de tolérance aux fautes fonctionne avec la supposition que la majorité des plates-formes de migration ne sont pas hostiles. Le problème de la confidentialité est résolu puisqu'aucun agent mobile ne connaît la totalité de l'information. Cette approche ne prend pas en compte l'intégrité. Néanmoins, elle est un exemple du concept du calcul distribué sécurisé : des parties peuvent conjointement calculer le résultat d'une fonction particulière sans révéler les entrées. Par ailleurs, cette technique ne s'applique qu'aux applications où les agents

peuvent être dupliqués, où la tâche peut être décomposée sans problème et où la survie est une question importante. Cependant, l'inconvénient majeur est, bien sûr, l'existence des ressources additionnelles qui sont consommées par la duplication. Un problème possible est de vérifier qu'un des prédicats ne diverge pas, car, dans ce cas, ce prédicat n'enverrait pas de résultats aux prochaines plates-formes. En effet, la technique utilisant les machines à états n'est pas applicable ici, car les réplications peuvent exécuter des requêtes différentes ou des requêtes dans un ordre différent.

Trace d'exécution (Execution tracing) [Vigna, 1998] a été proposée pour la détection de tout comportement malicieux tel que la modification du code de l'agent mobile, de son état et de son flux d'exécution. Elle est fondée sur un enregistrement fidèle de l'exécution de l'agent mobile au niveau de chaque plate-forme. Cette technique est basée sur des traces cryptographiques (une sorte de résumé d'exécution) collectées durant l'exécution de l'agent mobile sur différents hôtes. Elle exige la création et la maintenance d'une trace non répudiable (signée par la plate-forme visitée) comportant les opérations effectuées par l'agent au cours de son exécution sur la plate-forme. C'est un mécanisme de non-répudiation par le fait qu'une plate-forme ne peut pas nier avoir exécuté un certain agent, s'il a résulté en une certaine trace. L'ensemble des traces recueillies par un agent, durant ses exécutions en dehors de sa plate-forme d'origine, constitue l'historique de son exécution. Cette approche permet de garanti l'intégrité de code, empêcher la modification de l'état et le flux d'exécution de l'agent mobile, mais il souffre plusieurs limitations telles que la taille de la trace potentiellement grande, sa maintenance, le nombre de messages échangés, ou le lancement tardif du processus de vérification. En effet, le propriétaire de l'agent mobile devra attendre l'obtention de résultats suspects avant de lancer le processus de vérification. Par ailleurs, des problèmes de confidentialité peuvent surgir lorsque ce type d'information est enregistré.

Amélioration de trace d'exécution est proposée par Tan et Moreau [Tan et al., 2001] [Tan et al., 2002b], modifie l'approche originale en assignant la vérification de la trace à une tierce partie de confiance, le serveur de vérification, plutôt que de dépendre du propriétaire de l'agent. Quand un agent mobile migre vers une nouvelle plate-forme, une copie de l'agent est soumise au serveur de vérification correspondant. La plate-forme visitée reçoit l'agent et produit la trace d'exécution associée. Avant la migration de l'agent vers une autre plate-forme, la plate-forme courante envoie la trace au serveur de vérification correspondant. Celui-ci simule l'exécution de l'agent sur la plate-forme en employant la trace d'exécution correspondante et la copie de l'agent. Le procédé de simulation est répété pour

chaque plate-forme de l'itinéraire par le serveur de vérification correspondant, jusqu'à ce que l'agent revienne de nouveau à sa plate-forme d'origine. Tan et Moreau [Tan et al., 2002a] ont fourni un protocole détaillé des échanges de messages, aussi bien que de la modélisation et la vérification formelle du protocole. La vérification de la trace d'exécution est forcée et c'est un avantage par rapport à la technique de trace d'exécution originale où le procédé de vérification est déclenché seulement à la suite de résultats suspicieux [Tan et al., 2002a]. Cependant, cette technique souffre toujours de la limitation de la taille de la trace et de la nécessité de maintenir une taille et un nombre potentiellement grands de notations. De plus, chaque plate-forme choisit un serveur de vérification et cela pourrait encourager et faciliter une collaboration malveillante possible entre une plate-forme et le serveur.

Les agents sédentaires (Sedentary agents) : En 2007, Ouardani et Pierre [Ouardani et al., 2007] proposent une approche combine quatre concepts : l'exécution de référence au sein d'une plate-forme de confiance, la coopération entre les agents, la signature numérique et la communication cryptée. Ce protocole utilise trois entités différentes : un agent mobile (MA) et deux agents sédentaires (SA et SAR). La MA migre entre un ensemble d'hôtes (P_i, qui pourrait être soit malveillant ou de confiance). Il suppose l'existence de deux plates-formes de confiance au sein de son protocole : la première est appelée T qui permet d'exécuter l'agent SA, et une deuxième plate-forme pour la récupération appelée TR sur laquelle la SAR s'exécute. Le code de l'agent mobile est divisé en deux parties : le sensible est appelé le code critique et le code non critique. Le code critique est également conservé par le SA. Il suppose que les messages échangés entre les agents soient transportés à travers des canaux authentiques, pour assurer la communication, un système de cryptographie asymétrique doit être mis à la disposition de sorte qu'il sera possible pour ces agents pour obtenir leurs clés privées et publiques nécessaires pour le chiffrement, ainsi qu'une fonction de hachage à sens unique pour permettre d'envoyer des données signées.

Ce protocole s'adapte à l'environnement et des conditions dans lesquelles il exerce ses activités. Le protocole utilise le comportement de référence qui permet partiellement (code critique) ou entièrement l'agent mobile de s'exécuter dans les plates-formes de confiance. Le principe de coopération associé au premier aspect permet la vérification de l'exécution dynamique de l'agent. Ainsi, la détection de plusieurs attaques instantanées. Le chiffrement et la signature numérique de la communication entre les agents participants font la coopération authentique et la confidentialité.

La réactivité du protocole à son environnement (être capable de s'adapter dynamiquement aux caractéristiques de la plate-forme actuelle) permet au système d'estimer la valeur du Timeout et de détecter la réexécution du code de l'agent. Cette estimation porte sur les paramètres de la plate-forme (performance, charge, etc.). Ainsi, l'intégrité et la confidentialité de l'agent mobile ne sont pas compromises. En outre, le protocole a été équipé d'un contre mesure contre l'attaque de modification permanente du code. Cette méthode réactive permet d'atténuer et de corriger les dommages causés par la modification du code permanent. Depuis une panne éventuelle de la plate-forme de confiance va certainement arrêter la coopération, un second agent a été introduit pour surveiller la répartition du premier agent coopérant. Cela permettra de garantir une continuité du protocole puisque l'agent poursuivra sa mission en pleine sécurité. Les inconvénients majeurs de cette approche sont le nombre important des messages échangés entre l'agent mobile et les agents sédentaires, ainsi la limite de temps d'exécution qui ne peut pas aider d'éviter l'exécution illégitime de l'agent.

2.5.3.2. Techniques de prévention

La prévention s'exprime par une politique de cloisonnement des informations d'une part et par des règles de comportement des utilisateurs du système informatique, d'autre part. Le mécanisme de prévention a pour but de rendre l'accès et la modification de l'agent difficile. Parmi ces approches on peut trouver :

Technique d'obscurcissement (Technical obfuscation) : les approches utilisant les techniques d'obscurcissement sont explorées afin de protéger, durant un temps minimal, le code d'un agent mobile contre la décompilation (reverse engineering). L'obscurcissement transforme un programme en un autre programme ayant un comportement équivalent, mais qui est plus difficile à comprendre. L'obscurcissement du code est basé sur l'application de transformations du code. Ces transformations sont évaluées selon leur puissance de confusion (le degré de confusion d'un lecteur humain), leur résilience (la capacité de résister à des attaques automatiques de « dé-obscurcissement »), et leur coût. Hohl [Hohl, 1998] propose une approche logicielle qui représente une solution de sécurisation de l'agent par son propre code. L'approche consiste à convertir l'agent d'origine en une boîte noire (black box) en utilisant des algorithmes d'obscurcissement appelés aussi algorithmes de confusion. Puisque cette technique ne permet pas la protection absolue du code [Barak et al., 2001], [D'Anna et al., 2003], on suppose l'existence d'un intervalle de temps durant lequel l'agent est protégé, c.-à-d. jusqu'à ce que la boîte noire ne soit plus fonctionnelle. Une date d'expiration est donc attachée à celle-ci. Dans ce cas, le code de l'agent change à chaque début de l'intervalle de protection. Cette technique peut être utilisée pour protéger le code et les données durant une certaine période de validité. Barak et al. [Barak et al., 2001] ont étudié

les limites théoriques des techniques d'obscurcissement et ont prouvé qu'en général il était impossible de réaliser un obscurcissement complètement sécurisé. Larry D'Anna et al. [D'Anna et al., 2003] indiquent que l'obscurcissement peut empêcher un hôte malicieux d'observer ou de fouiller le code ou les données, particulièrement si des algorithmes de confusion différents sont employés pour assurer une protection plus forte. Cependant, il ne peut empêcher la décompilation du programme. La technique d'obscurcissement pourrait être utilisée afin d'obtenir différentes versions d'une même tâche. Elle permet donc de fournir à l'agent mobile la possibilité d'avoir plusieurs comportements équivalents qui génèrent le même résultat.

Fonction cryptographique (Encrypted functions) : les techniques de fonction cryptographique sont destinées à fournir un mécanisme sûr par lequel les agents peuvent exécuter des comportements cryptés dans des environnements peu sûrs, tout en maintenant l'intégrité et la confidentialité de leurs calculs. Dans [Lee et al., 2004] présenté un système de protection d'agent mobile contre les plates-formes malveillantes. Les auteurs visent à répondre à l'exigence de la confidentialité et l'intégrité de l'agent mobile contre les plates-formes malveillantes en utilisant les fonctions cryptées. Le schéma présenté par Lee et al. [Lee et al., 2004] sont plus pratiques en raison de l'utilisation des fonctions composites et cryptographie homomorphique pour chiffrer les données, le code de l'agent et l'état d'exécution. Leur approche repose sur l'exécution, sur une plate-forme d'agent mobile, d'un programme représentant une fonction encryptée. Le code de l'agent mobile est une sorte de programme encrypté qui peut être exécuté en utilisant des données chiffrées sans décrypter ni le code ni les données. Le but du calcul avec des fonctions cryptographiques est de fournir aux agents mobiles une confidentialité de calcul pour les primitives cryptographiques. L'inconvénient de cette technique n'empêche ni la réexécution, ni l'extraction de code ou le déni de service.

Génération des clés à partir de l'environnement (Environmental key generation) : Riordan et Schneier [Riordan et al., 1998] proposent une méthode pour qu'un agent fasse une certaine action quand une condition de l'environnement est vraie. Une clé est générée quand cette condition est vraie. Elle peut être utilisée pour décrypter du code de l'agent, pour décrypter un message destiné à l'agent dont l'expéditeur ne veut qu'il soit déchiffrable que dans certaines conditions. La plate-forme ne doit pas savoir de quoi dépend cette condition, car elle maîtrise tout l'environnement. Filion et al. [Filion, 2005] illustrent un exemple de cas d'utilisation "virus blindé" pour la génération de clés de l'environnement pour empêcher l'analyse de code qui peut être appliqué aux agents mobiles. Une faiblesse de cette approche est qu'une plate-forme qui contrôle complètement l'agent pourrait simplement modifier l'agent pour

imprimer le code exécutable à la réception du déclenchement, au lieu de l'exécuter [Jansen 2000a]. Un autre inconvénient est qu'une plate-forme d'agent limite typiquement les possibilités d'un agent pour exécuter le code créé dynamiquement, puisqu'elle le considère en tant qu'opération peu sûre.

TAMAP (Trust-based Approach for Mobile Agent Protection) : la technique de Hacini et al. [Hacini et al., 2007] consistent à protéger le code de l'agent mobile contre toute analyse visant sa divulgation. Hacini et al. supposent que la sécurité de l'agent mobile à un lien intrinsèque avec le degré de confiance de plate-forme visité. La sécurité de l'agent exige l'établissement dynamique des relations de confiance entre lui et les hôtes visités. L'idée est donc d'utiliser le concept de l'adaptabilité afin de déterminer le comportement adéquat de l'agent mobile selon le degré de confiance détecté. Pour cela, Hacini et al. proposent une architecture adaptative de l'agent mobile qui le dote de la capacité de réagir avec un comportement imprévisible. Le comportement adaptatif d'un agent est souvent inattendu, celui-ci est donc protégé puisqu'il n'est pas possible d'attaquer une entité dont le comportement est inconnu. D'ailleurs, pour choisir le comportement le plus approprié, l'agent mobile doit vérifier au préalable s'il peut faire confiance à l'hôte visité. Sa réaction est basée sur son aptitude à estimer le degré de confiance de ce dernier, à contrôler l'historique de son comportement et à exploiter les diverses qualités de service fournies par l'application. Cette approche permet d'empêcher l'analyse statique ou dynamique de code de l'agent mobile, mais il présente plusieurs inconvénients, le premier est la communication supplémentaire entre le pourvoyeur de service et l'agent mobile. En effet, mettre en place le canal de communications à chaque migration est une opération coûteuse. La deuxième est comment définir les différentes métriques de la confiance afin de protéger l'agent mobile en lui permettant de s'exécuter uniquement dans des environnements de confiance. Enfin, cette approche ne traite pas la protection des données transportées avec les agents mobiles.

Matériel de confiance (Trusted hardware) : Mana et Pimentel [Mana et al., 2002] ont proposé l'utilisation de cartes à puces. Leur idée consiste à subdiviser le code de l'agent en sections dont certaines seront encryptées par la clé publique d'une carte à puce. Ces dernières seront remplacées par des appels procéduraux vers la carte. Sur site d'exécution, on transmet à la carte, comme arguments, les sections encryptées qui seront décryptées puis exécutées à l'intérieur de cette dernière, la paire de clés est générée à l'intérieur de la carte à puce. La clé publique est publiée tandis que la clé secrète réside exclusivement à l'intérieur de la carte et ne sera jamais révélée [Mana et al., 2002]. De cette façon, la confidentialité du code est assurée et l'exécution de la totalité de l'agent sur le périphérique de sécurisation est

évitée. En cas d'exécution de plusieurs agents sur le site, la carte peut jouer le rôle d'une ressource critique dont l'accès est géré par les moyens connus. Malgré la robustesse apparente de l'approche, le code est partiellement caché et le code restant est lisible ce qui permet l'analyse du code clair et la déduction des fonctionnalités du code dissimulé. Les approches basées sur un matériel de confiance sont jugées être parmi les plus puissantes. Néanmoins, elles présentent un coût trop élevé.

Approches basées sur les nœuds de confiance (Trusted nodes) : un environnement de confiance d'exécution serait la contre-mesure la plus adéquate pour la protection de l'agent mobile. Elle est basée sur l'installation d'un ensemble d'hôtes de confiance. Ceci peut être fait en encryptant l'agent mobile durant son cheminement (entre les hôtes) et en authentifiant l'hôte avant que l'agent mobile ne lui soit transmis. Créer un environnement de confiance dans lequel un agent mobile erre librement sans être menacé par un hôte malveillant permet de se débarrasser de la plupart des classes de menaces [Bierman et al., 2002]. Cependant, cette méthode va à l'encontre de la notion d'agent mobile. Si un agent mobile à un itinéraire préétabli, l'avantage de la vaste quantité de ressources disponibles sur l'Internet, ainsi que la bonne concurrence peut être perdu ou sévèrement obstrué. D'un autre côté, en introduisant des nœuds de confiance au sein d'une infrastructure, vers laquelle les agents mobiles peuvent, si nécessaire, migrer, il est possible d'éviter que des informations sensibles ne soient émises vers des hôtes non fiables. De plus, il est aisé d'obtenir la trace de certains mauvais comportements de ces hôtes. L'hôte d'origine est souvent supposé être un hôte de confiance. Cette approche ne semble pas avoir été totalement explorée. Elle peut être potentiellement très valable dans des réseaux composés de nœuds mobiles et fixes, offrant aux utilisateurs la possibilité de distribuer les agents mobiles au niveau du réseau fixe en se basant sur les nœuds de confiance afin de traiter l'information sensible. L'agent mobile peut aussi être conçu de façon à ne s'exécuter que sur un hôte jugé de confiance.

Séparation des privilèges (Separation of privileges) : cette technique est utilisée dans le cadre de la négociation électronique à base d'agent mobile [Al-Jaljouli et al., 2007]. L'approche consiste à répartir les tâches entre trois agents. Ce qui permet de minimiser les risques des plates-formes malveillantes. Les tâches les plus critiques de l'agent sont exécutées par des plates-formes de confiance, tandis que les tâches non critiques sont exécutées par des plates-formes non fiables. Bien sûr, ce système est sans preuve complète. Les tâches essentielles à la mission dépendent également des tâches dites «non critiques». Cela implique que, même avec la séparation des rôles, des tâches critiques peuvent encore être affectées (même si peu) lorsque certains composants de l'agent sont compromis ou détruites. Le protocole proposé

utilise trois agents mobiles : agent du contrôleur (CA), agent travailleur (WA), et agent du registre de l'itinéraire (IRA).

- Agent contrôleur (CA) stocke les données critiques telles que : liste des offres, date d'expiration des fonctions soumission, notation et décision.
- Agent travailleur (WA) stocke les données non critiques telle que les fonctions de tactique qui forment le modèle de prise de décision de l'agent.
- Agent du registre de l'itinéraire (IRA) stocke les adresses des fournisseurs visités et le temps nécessaire à l'exécution de l'agent mobile dans l'hôte qui fournit le service. Si l'IRA est détruite en portant des adresses stockées des fournisseurs visités, le protocole d'exécution ne serait pas terminé en raison du manque d'informations de validation. En outre, ce qui est considéré comme non-critique, à savoir, le modèle de prise de décision est contrôlé par le CA, est apparemment une condition préalable à exécution de tous des transactions commerciales.

Pour résoudre les problèmes de sécurité associés au processus d'e-négociation, il propose un protocole de sécurité qui maintient les propriétés de sécurité sur les informations échangées au cours d'e-négociation : la confidentialité, l'authentification, la non-répudiation, l'anonymat et l'intégrité sont assurés par ce système. Le protocole proposé à des mécanismes de sécurité suffisante qui pourraient empêcher ou au moins de détecter les différentes attaques de sécurité et préserve les propriétés essentielles de sécurité pour les systèmes de commerce électronique à base d'agents mobiles. En outre, le protocole est efficace par rapport aux protocoles existants. Il ne nécessite pas de calculs complexes et intensifs.

2.5.4. Synthèse

Une synthèse discutant les avantages et les inconvénients des différentes approches de protection de l'agent mobile s'impose. Elle permet de mettre en évidence les lacunes à combler et qui doit servir de base à notre réflexion. En effet, à l'opposé de la protection des hôtes, celle des agents mobiles contre des hôtes malveillants ne dispose pas de solution sûre et reste encore aujourd'hui un champ de recherche ouvert.

Une comparaison entre les approches précédemment présentées (Tableau 2.1) est établie en s'appuyant sur des critères relatifs à la robustesse de la sécurité des agents et de ceux liés au coût de cette sécurité. Nous examinons les apports de chaque approche en fonction de plusieurs critères relatifs à la sécurité dans le système d'agent mobile :

- La communication : permet de réduire ou augmenter la performance d'une approche de sécurité, la communication supplémentaire entre les agents coopérants ou entre l'agent et leur propriétaire augmente le trafic sur le réseau. Celle-ci s'oppose face à l'intérêt de l'utilisation des agents mobiles.
- Le code de l'agent mobile : fournit plusieurs critères relatifs à la protection de code de l'agent mobile. Le premier est l'intégrité qui exige un agent de détecter une altération de son code ou de son état d'exécution par une plate-forme malveillante. Le deuxième est la confidentialité de code qui exige l'agent de cacher le code critique (tâches sensibles) dans les environnements d'exécutions non sûrs. Le troisième est la taille du code engendré, si la taille de code augmente, il s'ajoute une surcharge lors de la migration de l'agent, ainsi un temps supplémentaire d'exécution d'agent. Le quatrième est le temps nécessaire pour exécuter le code de l'agent.
- Les données de l'agent mobile : on a trois critères pour sécuriser les données d'agent mobile. Le premier est l'intégrité des données, l'agent peut empêcher la modification de certaines informations transportées avec lui ou ajoutée par des plates-formes d'exécution. Le deuxième est la confidentialité des données. Le troisième est le contrôle d'accès aux données d'agent (empêcher certaines plates-formes d'accéder aux données produites par d'autres plates-formes).

Les approches présentées ont été implicitement subdivisées en deux catégories, la première est basée sur le matériel de confiance, la deuxième est basée sur le développement logiciel. Nous sommes arrivés à la conclusion que les approches basées sur le matériel de confiance sont puissantes. Cependant, elles ne sont pas évolutives et présentent un coût prohibitif dû principalement à l'utilisation du périphérique de sécurisation. Par ailleurs, les approches logicielles, moins robustes, réduisent le coût de la sécurité et facilitent sa maintenance. C'est donc vers ces dernières que nous nous orientons. Par ailleurs, diverses menaces de sécurité peuvent émaner d'hôtes malveillants. Nous citons les attaques d'intégrité, de la confidentialité, le contrôle d'accès et l'authentification. Nous avons dressé le tableau 2.1 qui montre la comparaison entre les différentes approches de protection dans le système d'agents mobiles.

Les techniques de protection	Catégorie	Communication additionnelle	Code de l'agent mobile				Données de l'agent mobile		
			Intégrité	Confidentialité	Taille du code	Temps d'exécution	Intégrité	confidentialité	Contrôle d'accès
Signatures numériques	Détection	Aucune	Assurée	Non assurée	Non modifiée	Augmenté	Assurée	Non assurée	Non assuré
Évaluation d'état	Détection	Aucune	Assurée	Non assurée (code lisible pour tous les hôtes)	Augmentée avec la fonction d'évaluation d'état	Faible (ralentissement à cause de la fonction d'évaluation)	Non assurée	Non assurée (tous les hôtes peuvent lire les données)	Assuré (l'accès après l'évaluation d'état)
Enregistrement d'itinéraires avec des agents coopérants	Détection	Trop de communications additionnelles	Assurée (les tâches critiques exécutées dans les hôtes de confiance)	Non assurée	Relativement accepté	Relativement accepté	Assurée (les données sensibles transportées entre les hôtes de confiance)	Non assurée (tous les hôtes peuvent lire les données)	assuré (les tâches critiques exécutées dans les hôtes de confiance)
Encapsulation des résultats partiels	Détection	Aucune	Peut assurer (Signature numérique)	Non assurée (tous les hôtes peuvent lire le code)	Augmentée à cause des résultats partiels	Relativement accepté	Fortement assurée (les données signées)	Assurée (les données sont cryptées)	Oui (les données sont cryptées)
Copies d'agents	Détection	Aucune	Non assurée	Assurée (le code de l'agent est divisé en plusieurs agents)	Non modifiée	Non modifié	Non assurée	Assurée (les données sensibles transportées par plusieurs agents)	Assuré (l'accès est limité aux données transportées par l'agent)
Trace d'exécution	Détection	Trop de communications additionnelles	Assurée (trace d'exécution d'agent mobile)	Non assurée (tous les hôtes peuvent lire le code)	Augmentée à cause de la trace d'exécution	Faible (ralentissement à cause de la réexécution de l'agent)	Assurée (trace d'exécution de l'agent mobile)	Non assurée	Non assuré

Les techniques de protection	Catégorie	Communication additionnelle	Code de l'agent mobile				Données de l'agent mobile		
			Intégrité	Confidentialité	Taille du code	Temps d'exécution	Intégrité	confidentialité	Contrôle d'accès
Les agents sédentaires	Détection	Trop de communications additionnelles	Assurée (le code sensible est conservé dans l'agent sédentaire)	Assurée (le code sensible est exécuté dans un hôte de confiance)	Non modifiée	acceptable	Non assurée	Non assurée	Non assuré
La technique d'obscurcissement	Prévention	Aucune	Forte pendant une courte durée de vie de l'agent	Forte pendant une courte durée de vie de l'agent	Importante en cas d'insertion de code mort	Non modifié	Non assurée	Peut assurer pendant une courte duré	Non assuré
Fonction cryptographique	Prévention	L'agent doit retourner à son hôte après chaque visite.	Peut assurer par la signature numérique	Forte (l'exécution d'un agent crypté)	Relativement acceptable	Faible (l'exécution d'une fonction qui exécute un code crypté)	Non assurée	Assurée (les données de l'agent sont cryptées)	Peut assurer
Génération des clés à partir de l'environnement	Prévention	Aucune	Non assurée	Forte (génération des clés de décryptage)	Relativement acceptable	Faible (ralentissement à cause de la génération des clés)	Non assurée	Forte (les données sont cryptées)	Non assuré (toutes les données sont cryptées par la même clé)
TAMAP (une approche de protection des agents mobiles basée sur la confiance)	Prévention	Trop de communications additionnelles	Assurée (les sous-composants sensibles sont signés)	Assurée (les sous-composants sensibles sont cryptés)	Importante (plusieurs comportements pour l'agent mobile)	Relativement accepter	Non assurée	Non assurée	Non assuré

Les techniques de protection	Catégorie	Communication additionnelle	Code de l'agent mobile				Données de l'agent mobile		
			Intégrité	Confidentialité	Taille du code	Temps d'exécution	Intégrité	confidentialité	Contrôle d'accès
Matériel de confiance	Prévention	Aucune	Forte (la signature de code sensible)	Forte (le cryptage de code sensible)	Ajout d'instructions d'appel de la carte	Faible (ralentissement à cause de l'accès à la carte)	Peut assurer (signature numérique)	Peut assurer (Cryptage)	Non assuré (toutes les données sont cryptées par le même clé)
Les nœuds de confiance	Détection	Aucune	Non assurée	Forte (cryptage de l'agent entre les hôtes de confiance)	Non modifiée	Non modifié	Non assurée	Fort (les données de l'agent sont cryptées)	Assuré (l'agent visité les hôtes de confiances)
Politique de sécurité spéciale	Prévention	Trop de communications additionnelles	Non assurée	Non assurée	Non modifiée	Non modifié	Assurée	Assurée	assuré

Tableau 2. 1. Comparaison entre les approches de protection des agents mobiles

D'après le résultat obtenu (Tableau 2.1.), nous avons constaté que les approches qui ont été proposées jusqu'à présent bien qu'elles révèlent des idées intéressantes souffrent de sérieuses limitations. En effet, il n'existe pas une solution universelle au problème de l'hôte malicieux. Seulement des solutions partielles ont été proposées. De plus, la plupart de ces mécanismes de sécurité visent la détection plutôt que la prévention d'attaques émanant d'hôtes malveillants.

Notre infrastructure propose des solutions innovantes pour les questions de recherche suivantes :

a) Comment construire une ontologie dans le domaine de sécurité des agents mobiles, afin d'éliminer les différences sémantiques qui existent entre la politique de sécurité de l'agent et la plate-forme d'exécution ?

b) Comment créer des agents mobiles à base de composants et comment associer une politique de sécurité à l'agent mobile ?

c) Comment transformer l'agent à un agent mobile sécurisé de confiance ? Comment estimer le niveau de sécurité des composants de l'agent et le degré de confiance des plates-formes d'exécution ? Enfin comment définir les règles de contrôle d'accès aux données d'agent mobile ?

d) Comment assurer l'intégrité et la confidentialité de code et des données de l'agent mobile lors de son exécution ?

Les solutions spécifiques sont les suivantes :

a) La création d'une ontologie dans le domaine de sécurité des agents mobiles. Dans notre approche les ontologies peuvent être intéressantes pour exprimer la politique de sécurité d'agent et les plates-formes d'exécution. D'une part, ceci permettrait de décrire les exigences et les capacités avec des structures et des termes différents selon les connaissances propres des agents et des plates-formes. D'autre part, l'utilisation des ontologies permet de faciliter l'analyse automatique de la compatibilité sémantique entre les politiques de sécurité. Nous avons choisi de modéliser la politique de sécurité en utilisant un standard de la W3C qui est le WS-Policy pour construire les politiques de sécurité des dans le système à base d'agent mobile.

b) La création d'un agent mobile de confiance à base des composants. Les travaux de recherche relatifs à la sécurité des agents mobiles, traitent principalement le déploiement de l'agent mobile, c'est-à-dire ils décrivent comment les agents sont lancés, la façon dont ils se déplacent entre les plates-formes. Dans la plupart des travaux, le code, l'identité, les paramètres de sécurité et la

confiance entre les entités sont supposés comme étant déjà disponibles au moment de l'exécution de l'agent mobile. Notre approche répond à des questions importantes lors de la création d'agents mobiles, comme la décomposition du comportement de l'agent sous forme de composants élémentaires et spécialisés, l'évaluation et la validation par des entités de confiance et enfin la définition d'une politique de sécurité de l'agent mobile.

c) Transformer l'agent à un agent mobile sécurisé de confiance capable de s'adapter à différentes conditions de sécurité des plates-formes. Nous avons trouvé grâce à une analyse en profondeur de la littérature que toutes les approches utilisent le même mécanisme de sécurité pour protéger tous les agents mobiles contre les différentes plates-formes malveillantes, sans prendre en considération ni les exigences de sécurité de chaque agent mobile ni la politique de sécurité des plates-formes qui offrent les services aux agents. En effet, le propriétaire d'agent exige un niveau de sécurité à assurer par les plates-formes d'exécution, ainsi les plates-formes offrent des capacités de sécurité aux agents mobiles et des exigences à respecter par les agents mobiles visiteurs. Notre approche est basée sur deux stratégies d'adaptation. La première est une adaptation statique permet de transformer l'agent à un agent mobile sécurisé de confiance, cette adaptation est effectuée par une entité de confiance SGSA (créer l'itinéraire d'agent, le cryptage et la signature des composants sensibles, crée les règles de contrôle d'accès aux données d'agent, …). La deuxième est une adaptation dynamique permet à l'agent d'appliquer leur politique de sécurité dans les plates-formes d'exécution.

d) Protéger le code et les données de l'agent mobile lors de son exécution. La protection de code consiste à crypter et signer les composants sensibles de l'agent mobile. De cette manière, on garantit l'authenticité, la confidentialité et l'intégrité de code. La protection des données est une étape essentielle approche. En effet, la meilleure approche disponible pour protéger les données d'agent mobile est d'utiliser la clé publique du propriétaire d'agent. Cette approche est très restrictive, parce que le propriétaire d'agent est le seul qui peut décrypter ou vérifier l'intégrité des données transportées par l'agent. Cependant, avec notre solution de protection, toutes les plates-formes autorisées peuvent accéder aux données d'agent. En effet, dans la phase d'adaptation d'agent mobile, le SGSA estime le niveau de sécurité de chaque composant de l'agent et le degré de confiance des plates-formes qui exécutent l'agent, ensuite il définit les règles de contrôle d'accès aux données d'agent. Ces règles permettent de vérifier si une plate-forme souhaitant accéder à une donnée spécifique est bien autorisée à le faire.

2.6. Conclusion

Au niveau de ce chapitre, nous avons étudié le problème de sécurité des agents mobiles en mettant en évidence les différents types d'attaques possibles provenant des hôtes ou bien des agents mobiles. Nous avons discuté la protection des deux acteurs en mettant l'accent sur celle de l'agent mobile.

Le problème le plus important empêchant l'adoption actuelle des agents mobiles est sans nul doute la sécurité. En effet, même si la protection des hôtes est quasiment assurée, celle des agents reste un réel problème qui n'a pas encore trouvé de solution définitive. Les problèmes de sécurité liés à l'utilisation des agents mobiles constituent donc un champ d'investigation complet. L'exécution des agents mobiles sur des hôtes non fiables est un facteur qui introduit des problèmes de sécurité dont les solutions ne sont guère triviales. Ils sont relatifs, en particulier, à l'exécution correcte de l'agent et la confidentialité de ses données. Les solutions à certains problèmes ponctuels existent.

Les approches relatives à la protection des agents mobiles, constituant l'objectif de cette thèse, ont été particulièrement explorées. Nous avons souligné leurs avantages et inconvénients. Nous avons aussi mis en évidence celles qui ont inspiré notre approche de protection.

Dans le chapitre suivant, nous créons une ontologie contenant les concepts clés concernant la sécurité dans système à base d'agents mobiles, ainsi nous utilisons la spécification WS-Policy pour décrire les politiques de sécurité dans notre système à développer.

CHAPITRE 3

Ontologie dans le domaine de sécurité des agents mobiles

3.1. Introduction

L'ontologie que nous avons proposée à un double objectif : tout d'abord la mise en place d'une connaissance formelle sur la sécurité dans le cadre des systèmes à base d'agents mobiles, puis l'utilisation de l'ontologie permet de faciliter l'analyse automatique de la compatibilité sémantique entre les politiques de sécurité de l'agent et les plates-formes visitées. Nous avons choisi de modéliser les politiques de sécurité en utilisant un standard de la W3C qui est le WS-Policy. Nous ajoutons des annotations sémantiques en utilisant cette ontologie pour pouvoir décrire les exigences et les capacités de sécurité. En effet, nous avons structuré la politique de sécurité en deux parties :

- Exigences de sécurité : permet de spécifier les différents paramètres de sécurité nécessaire à l'exécution sécurisée d'un agent mobile.
- Capacité de sécurité : représente un ensemble des spécifications, des protocoles, des algorithmes…, pour satisfaire une exigence de sécurité.

Pour déterminer si une plate-forme est capable d'exécuter de façon sécurisé un agent, d'une part, les aspects fonctionnels de la plate-forme devraient satisfaire les besoins fonctionnels de l'agent pour réaliser leur tâche, d'autre part, les exigences de sécurité de l'agent doivent être satisfaites par les capacités de sécurité de la de la plate-forme, ainsi les exigences de sécurité de la plate-forme doivent être également satisfaites par les capacités de sécurité utilisées par l'agent mobile.

L'objectif de ce chapitre est de définir le concept de politique de sécurité et de présenter ses principaux fondements. Dans une deuxième étape, nous apportons un intérêt majeur à la modélisation des politiques de sécurité dans le système à base d'agents mobiles. Nous avons choisi un standard WS-policy pour exprimer les exigences et les capacités de sécurité. Enfin, nous avons construit une ontologie dans le domaine de sécurité des agents mobiles, afin d'éliminer les différences sémantiques qui existent entre ces politiques (agents et plates-formes).

3.2. Politique de sécurité

3.2.1. Définitions et Fondements

Dans la littérature, différentes définitions ont été proposées pour les politiques de sécurité. Toutefois, ces définitions partagent le même objectif qui consiste à éliminer les risques d'attaque d'un système et satisfaire ses besoins de sécurité.

Selon F.B. Schneider [Schneider, 2000], une politique de sécurité définit l'exécution (action) qui, pour une raison ou une autre, est considérée inacceptable. Une telle définition peut concerner différents types de politiques tels que : le contrôle d'accès, le contrôle de flux d'information ou la disponibilité.

Quant à A. Abou Kalem [Kalam, 2003], il définit une politique de sécurité comme un dispositif nécessaire pour renforcer la sécurité des systèmes qui implique d'empêcher la réalisation d'opérations illégitimes contribuant à mettre en défaut les propriétés de confidentialité, d'intégrité et de disponibilité, mais aussi de garantir la possibilité de réaliser les opérations légitimes dans le système.

A. Contes définit, dans [Contes, 2005] la notion de politique de sécurité par un ensemble de propriétés de sécurité qui doivent être satisfaites par le système et un schéma d'autorisations qui représente les règles permettant de modifier l'état de protection d'un système.

Sous cette variété de définitions se cache une large complexité liée à la manière d'exprimer la politique de sécurité afin d'être suffisamment compréhensible et explicite et de permettre la représentation des différents besoins de sécurité d'un système. En plus, il est essentiel de vérifier à la spécification d'une politique un certain nombre de propriétés (e.g. terminaison, complétude, cohérence, etc.) afin d'assurer convenablement son application. De même, il est fondamental de vérifier la concordance d'une politique avec les besoins de sécurité requis. Par ailleurs, il faut mettre en avant le besoin de flexibilité de la politique afin de répondre aux changements dynamiques des exigences de sécurité d'une application, au cours de son exécution. Pour tenter de répondre à ces besoins de sécurité, notre approche est basée sur un format universel XML pour modéliser les politiques de sécurité, ce qui permet d'offrir un haut niveau d'interopérabilité ainsi qu'une grande flexibilité. Nous avons choisi d'adapter la spécification *WS-policy* pour exprimer les politiques de sécurité de différentes entités de notre système.

3.2.2. Politique de sécurité dans les systèmes à base d'agents mobiles

La découverte et la sélection des plates-formes les plus adéquates pour l'exécution sécurisée des agents mobiles sont des étapes importantes dans notre approche. Nous considérons la découverte des plates-formes comme étant la localisation des plates-formes publiées qui satisfassent certaines propriétés fonctionnelles de l'agent pour réaliser leur tâche. La sélection des plates-formes correspond à l'évaluation et le classement des plates-formes déjà découverts afin d'identifier ceux qui répondent le mieux aux exigences de sécurité de l'agent mobile. En effet, chaque plate-forme doit avoir une description fonctionnelle sur les services offerts aux agents mobiles, ainsi une description non fonctionnelle concernant la sécurité de chaque service offert par la plate-forme (politique de sécurité).

Dans le but de pouvoir utiliser la politique de sécurité dans le processus de sélection des plates-formes visitées par l'agent mobile, ces derniers doivent être modélisés et attachés aux services lors de leurs publications et aux agents mobiles lors de sa création. La modélisation de la politique de sécurité des différentes entités consiste à les représenter avec un langage facile et sous un format exploitable.

3.3. Modélisation sémantique de la politique de sécurité

Nous décrivons d'ailleurs l'extension que nous avons faite à partir de la spécification WS-Policy [Vedamuthu et al., 2007]. Ensuite, nous présentons l'ontologie MASO *(Mobile Agent Security Ontology)* que nous avons construite [Razouki et al., 2016]. Cette ontologie est basée sur le standard *OWL*.

3.3.1. WS-Policy

WS-Policy (Web Service Policy) [Vedamuthu et al., 2007], est un standard du W3C basé sur le standard ouvert XML. Il permet de définir des politiques de services Web en termes d'exigences, de préférences et de caractéristiques générales. Il permet, notamment, aux fournisseurs aussi bien que consommateurs de services Web d'exprimer des exigences ou préférences en termes de sécurité ou qualité de service. WS-Policy utilise une grammaire flexible et extensible qui permet d'exprimer les possibilités, les exigences, et les caractéristiques générales des services Web en tant que « policies ». Nous notons que la dernière version de ce standard est la version *WS-Policy 1.5*.

Conformément à *WS-Policy*, une politique (*policy*) d'un service Web peut être exprimée par plusieurs politiques alternatives (*Policy Alternative*) dont chacune regroupe un ensemble d'assertions. Chaque assertion (*Assertion*) représente une

propriété décrite par le fournisseur ou une préférence souhaitée par le client concernant un sujet donné (Opération, Endpoint, …). WS-Policy utilise deux opérateurs dans l'expression des politiques : l'opérateur *All* et l'opérateur *ExactlyOne*. L'opérateur All signifie que toutes les assertions constituant une politique alternative doivent être satisfaites pour accepter cette politique alternative. Par contre l'opérateur *ExactlyOne* signifie qu'au moins une des politiques alternatives doit être satisfaite pour accepter la politique. La Figure 3.1 présente un aperçu sur la grammaire proposée par *WS-Policy*.

```
<wsp:Policy
        xmlns:wsp="http://www.w3.org/ns/ws-policy">
        <wsp:ExactlyOne>
                <wsp:All>
                        <Assertion...>....</Assertion...>
                </wsp:All>
        </wsp:ExactlyOne>
</wsp:Policy>
```

Figure 3. 1. Forme normale du WS-Policy

Finalement, *WS-Policy* permet de décrire syntaxiquement les politiques relatives à un service Web relatives à différents domaines non-fonctionnels à savoir la sécurité, la qualité de service et bien d'autres. En effet, nous avons adapté cette spécification pour exprimer les exigences et les capacités de sécurité et développer un modèle de politique de sécurité spécifique au système à base d'agents mobiles.

3.3.2. Besoin sémantique dans la correspondance des politiques

Le problème majeur de l'utilisation de WS-Policy est que la correspondance entre les politiques est basée uniquement sur une comparaison syntaxique, l'intersection des politiques peut rejeter dans plusieurs cas des partenaires potentiels, même avec des politiques compatibles. Nous démontrons l'utilité de la comparaison sémantique à travers l'exemple suivant :

Un agent mobile exige la confidentialité des données et offre une capacité d'authentification :

- Exigence de sécurité : un agent mobile qui exige une contrainte relative à la confidentialité des données produites par la plate-forme visitée et exige le cryptage de ces données avec l'algorithme *3DES*.
- Capacité de sécurité : l'agent mobile offre un mécanisme d'authentification avec un certificat numérique X.509.

Une plate-forme exige l'authentification des agents mobiles visiteurs et offre une capacité de cryptage symétrique :

- Exigence de sécurité : l'agent mobile visiteur doit être authentifié.
- Capacité de sécurité : la plate-forme dispose une spécification de cryptage *XML-Encryption*.

Dans le scénario ci-dessus, si nous utilisons le standard *WS-Policy* pour représenter et correspondre les politiques de sécurité de l'agent et la plate-forme. Le comparateur établit une comparaison syntaxique entre des chaînes de caractères afin de déterminer si la capacité de la plate-forme peut satisfaire l'exigence de l'agent, et la capacité de l'agent peut satisfaire l'exigence de la plate-forme. Le comparateur conclut forcément que ces deux politiques ne sont pas compatibles bien que les assertions sont équivalentes. En effet, l'exécution de l'agent dans cette plate-forme sera rejetée. Par conséquent, l'intégration de l'aspect sémantique et des connaissances dans le domaine de sécurité lors l'intersection entre les politiques semble être très intéressante. Pour résoudre ce problème, nous créons une ontologie dans le domaine de sécurité d'agents mobiles pour capturer les informations sémantiques suivantes :

- XML-Encryption est une spécification de cryptage symétrique, qui supporte l'algorithme de cryptage 3DES.
- Certificat X.509 est une méthode d'authentification forte pour authentifier une entité.

Lorsque ces informations supplémentaires sont rajoutées aux politiques de sécurité, et la correspondance sémantique entre les politiques est appliquée. Alors, le comparateur conclure que la capacité de la plate-forme satisfaite l'exigence de l'agent et l'exigence de la plate-forme est satisfaite par la capacité de l'agent, ce qui rend une correspondance parfaite entre ces deux politiques. Cet exemple illustre l'importance de l'information sémantique pour améliorer la qualité de correspondance entre les politiques de sécurité.

L'ontologie est considérée parmi les solutions les plus importantes pour résoudre le problème d'hétérogénéité. [Hacini et al., 2012] propose une ontologie de domaine de sécurité pour les applications et les systèmes utilisant la technologie d'agent mobile. Afin d'éliminer les différences sémantiques qui existent au niveau des objets, attributs, et structures de données de politiques de sécurité pour faciliter l'interopérabilité des agents mobiles. Les limites de cette approche tiennent dans le fait que l'ontologie est utilisée seulement dans un scénario de communication entre les agents mobiles et les plates-formes. En effet, cette ontologie ne résout pas le

problème de la spécification des politiques de sécurité dans le système à base d'agents mobiles, ainsi le problème d'hétérogénéité entre les politiques de sécurité de l'agent mobile et les plates-formes visitées. Enfin, cette ontologie ne fournit pas une solution pour décrire les besoins spécifiques de sécurité de chaque agent, ni les capacités de sécurité fournit par les plates-formes.

Dans ce qui suite nous allons montrer comment spécifier les politiques de sécurité sémantique pour les agents mobiles et les plates-formes qui offrent les services aux agents. Ainsi comment faire correspondre sémantiquement ces deux politiques de sécurité.

3.3.3. Extension WS-Policy pour modéliser les politiques de sécurité

Une plate-forme offre un ensemble de services pour les agents mobiles afin de réaliser leur tâche. Chaque service à un ensemble des propriétés fonctionnelles. Toutefois, ces propriétés ne sont pas suffisantes pour déterminer le service le plus approprié aux besoins spécifiques de l'agent parmi un ensemble de services assurant les mêmes fonctionnalités. D'où l'intérêt d'avoir une description claire de sa politique de sécurité, ce qui permet à la plate-forme d'exprimer leurs exigences et ses capacités de sécurité pour chaque service. Afin de tenir en compte les politiques de sécurité dans le processus de sélection des plates-formes pour l'exécution sécurisée d'un agent mobile, nous avons proposé une extension au WS-Policy en ajoutant de nouveaux éléments dans sa spécification initiale. Cette extension, nous a permis d'intégrer les différents concepts de sécurité, de créer les relations sémantiques entre ces concepts et d'assurer la correspondance automatique entre les politiques de sécurité. Nous avons utilisé un modèle basé sur une ontologie OWL pour représenter ces différents éléments.

La figure 3.2 présente les classes de l'ontologie MASO basée sur WS-Policy, pour bien illustrer la différence entre les relations sémantiques et la hiérarchie des classes. Nous utilisons deux lignes pour représenter les relations entre les différents concepts cette ontologie : La ligne pointillée (couleur bleu) permet de relier une classe spécifique à une classe plus générale, ce qui permet de définir les hiérarchies des classes dans l'ontologie *MASO*. La ligne continue (couleur rouge) permet de spécifier les relations sémantiques entre les différentes classes.

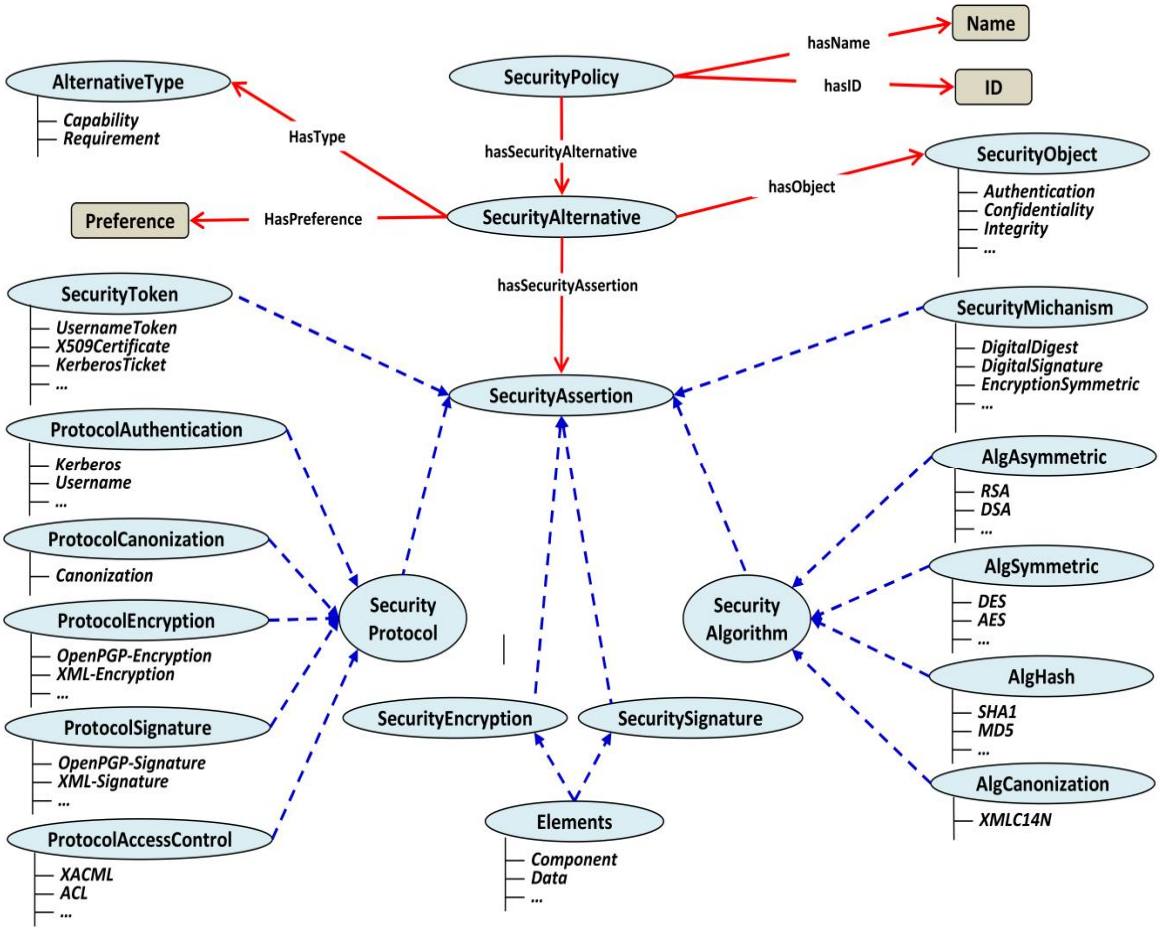

Figure 3. 2. Classes principales de l'ontologie de politique de sécurité

Nous créons trois classes *SecurityPolicy*, *SecurityAlternative* et *SecurityAssertion*, afin d'exprimer les assertions de sécurité au sein d'une politique de sécurité. En effet, le concept *SecurityPolicy* est la classe de niveau supérieur de notre ontologie. Il représente la racine de la politique de sécurité, chaque politique identifiée par un nom et un identificateur unique (*Name*, *ID*). Il se compose au moins d'une ou plusieurs alternatives de sécurité (*SecurityAlternative*).

La classe *SecurityAlternative* contient quatre propriétés sémantiques. La propriété *hasAlternativeType* permet de déterminer le type de l'alternative avec la classe *AlternativeType*. Cette classe contient deux instances *Capability* et *Requirement* pour spécifier une capacité ou une exigence de sécurité. La propriété *hasPreference* permet de spécifier la préférence d'une alternative particulière. La préférence est exprimée sous la forme *xsd:int*. Plus la valeur de la préférence est élevée, plus la préférence exprimée a de poids. Si aucune préférence n'est spécifiée, la valeur par défaut est zéro. La propriété *hasObject* permet de fixer l'objectif à assurer par l'alternative de sécurité par la classe *SecurityObject*. Enfin, la propriété *hasSecurityAssertion* permet de spécifier les différentes assertions de sécurité utilisées pour satisfaire un objectif fixé par l'alternative de sécurité, la classe

SecurityAssertion contient six sous-classes :

- *SecurityMechanism* décrit les solutions techniques et méthodes utilisées pour satisfaire un objectif de sécurité. Cette classe dispose six instances : *Authorization, DigitalSignature, DigitalDigest, EncryptionAsymmetric, EncryptionSymmetric* et *Identification*.
- *SecurityProtocol* permet de spécifier les différents protocoles et les spécifications de sécurité utilisés pour protéger les agents mobiles et les plates-formes d'exécution. Cette classe dispose cinq sous-classes (figure 3.2).
- *SecurityAlgorithm* contient les différents algorithmes de cryptage, de signature, de hachage et de la canonisation des données. Pour cela, nous avons extraire quatre sous-classes de cette classe. La classe *AlgEncryption* dispose les algorithmes de cryptage symétrique pour assurer la confidentialité des données. La classe AlgSignature contient les algorithmes de cryptage asymétrique qui permettent de garantir l'authenticité et l'intégrité de données. La classe *AlgDigest* dispose les algorithmes qui permettent de créer le condensé de données (*MD5*, *SH1*, *SH2)*. La classe *AlgCanonicalization* représente les algorithmes de canonisation, permettent de présenter de l'information XML dans une forme standard.
- *SecurityToken* permet de spécifier les différents types de jeton de sécurité utilisés par un protocole ou un algorithme de sécurité. En effet, un jeton de sécurité peut être utilisé pour l'authentification, le crypter et la signature de données. On distingue six individus pour cette classe : *AsymmetricKey*, *SymmetricKey*, *SAMLAssertion*, *KerberosTicket*, *X509Certificate* et *UsernameToken*.
- *SecurityEncryption* et *SecuritySignature* permettent de localiser les éléments à crypter/signer dans l'agent mobile. Ces deux classes utilisent la même sous-classe *Elements* pour déterminer les éléments à protéger. Cette classe contient quatre instances : *XPath*, *Data*, *Component*, *Itinerary*.

3.3.4. Contrainte de propriété MASO

Nous présentons les différentes relations sémantiques entre les concepts liés à la sécurité dans le système d'agents mobiles, comme l'objectif de sécurité, le mécanisme de sécurité, les protocoles, les algorithmes et autres. Les politiques de sécurité seront définies sur la base de l'ontologie MASO. Nous redessinons cette ontologie avec de nouvelles propriétés sémantiques (figure 3.3) :

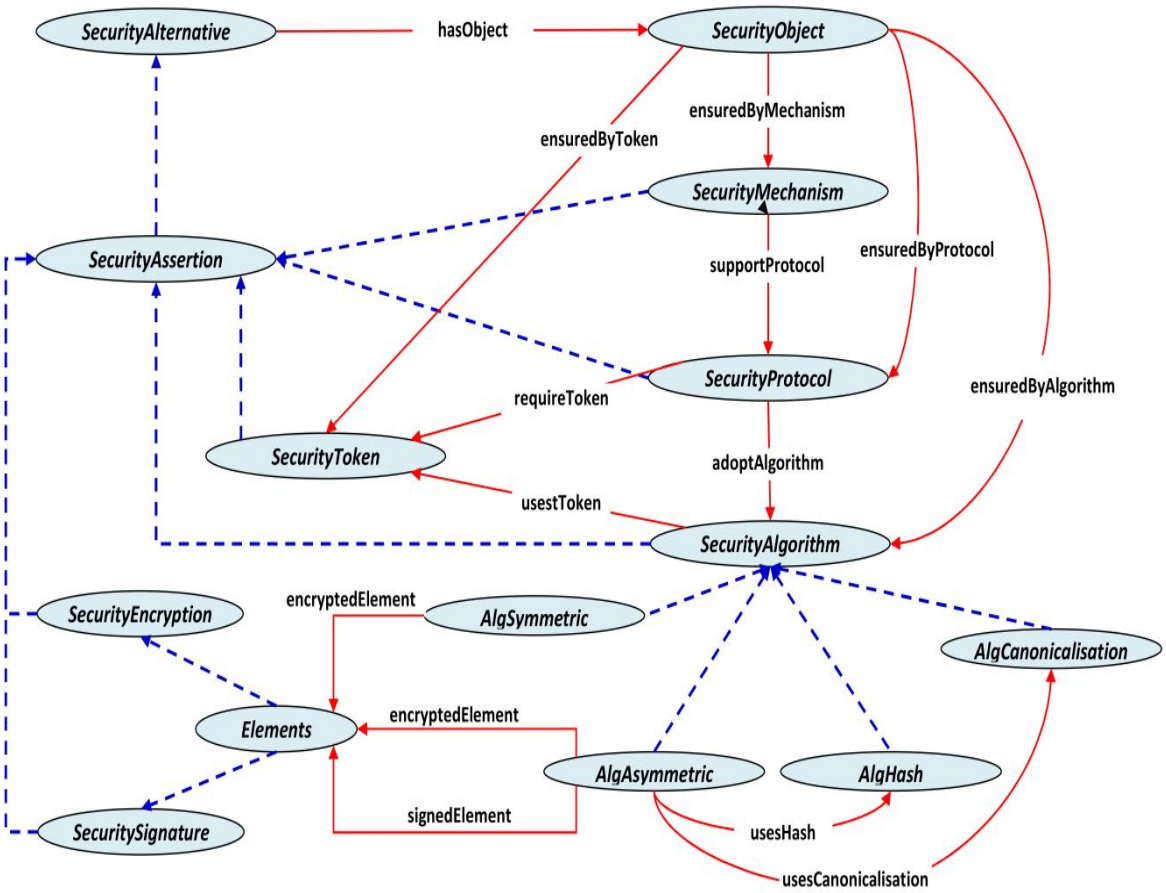

Figure 3. 3. Relations sémantiques entre les différentes classes de l'ontologie MASO

Comme le montre la figure 3.3, la classe *SecurityObject* dispose plusieurs propriétés sémantiques pour spécifier les mécanismes, les protocoles, les algorithmes et les jetons de sécurité qui assure un objectif fixé par une alternative de sécurité. La propriété *ensuredByMechanism* permet d'exprimer les mécanismes de sécurité utilisés pour satisfaire un objectif de sécurité. Par exemple, l'objectif de sécurité *Confidentiality* est assuré par deux mécanismes de sécurité *EncryptionAsymmetric* et *EncryptionSymmetric*. Les trois autres propriétés sémantiques seront traitées de la même manière.

La propriété *supportProtocol* permet de spécifier les protocoles qui satisfaits un mécanisme de sécurité. Par exemple, le protocole *XML-Signature* est utilisé pour garantir le mécanisme de sécurité *DigitalSignature*.

La propriété *adoptAlgorithm* permet à un protocole d'adopter un ou plusieurs algorithmes dans son processus d'exécution. Certain protocole exige la présence d'un jeton de sécurité avec la propriété *requisToken*. Par exemple, le protocole *XML-Encryption* adopte l'algorithme *3DES* pour crypter des données et utilise le jeton de sécurité *SymmetricKey* comme une clé de cryptage.

La propriété usesToken est utilisée par la classe *SecurityAlgorithm* pour déterminer la liste des clés utilisées dans le processus de cryptage/signature (Ex. un algorithme de signature DSA utilise un jeton de sécurité X509 pour garantir l'intégrité et l'authenticité de l'agent). La propriété *encryptedElement* et *signedElement* sont utilisées pour déterminer les éléments à crypter/signer de l'agent mobile. La propriété *usesHash* est utilisée entre la classe *AlgAsymmetric* et *AlgHash* pour fixer la fonction de hachage adoptée par l'algorithme asymétrique. Enfin, la propriété *usesCanonicalization* est utilisée par la classe *AlgAsymmetric* pour spécifier l'algorithme de canonisation afin de signer les données de l'agent mobile.

Après la définition de toutes les relations sémantiques, et les contraintes sur les propriétés, chaque sous-classe aura une relation avec une ou plusieurs autres sous-classes. La figure 3.4 (a) présente toutes les relations sémantiques entre une instance *Authentication* de la classe *SecurityObject* et les mécanismes, protocoles, algorithmes qui satisfont cet objectif.

<owl:Class rdf:ID="SecurityObject"/> <Authentication rdf:ID="SecurityObject"> <ensuredByProtocol rdf:resource="#SAML"/> <ensuredByProtocol rdf:resource="#Kerberos"/> <ensuredByProtocol rdf:resource="#UserName"/> <ensuredByProtocol rdf:resource="#X509"/> <ensuredByMechanism rdf:resource="#Identification"/> <ensuredByMechanism rdf:resource="#DigitalSignature"/> <ensuredByAlgorithm rdf:resource="#RSA"/> <ensuredByAlgorithm rdf:resource="#DSA"/> </Authentification>	<owl:Class rdf:ID="ProtocolAuthentication"> <rdfs:subClassof rdf:resource="#SecurityProtocol"/> <rdfs:subClassof> <owl:Restriction> <owl:onProperty rdf:resource="#requireToken"/> <owl:hasValue rdf:resource="#X509Certificate"/> </owl:Restriction> </rdfs:subClassof> </owl:Class>
(a)	(b)

Figure 3. 4. Exemple de classe de restriction pour les propriétés de contrainte

La figure 3.4 (b) montre un exemple d'utilisation des propriétés de contrainte de la classe *ProtocolAuthentication* et l'instance *X509Certificate* de la classe SecurityToken. Après la définition de toutes les propriétés de contrainte et la définition des relations sémantiques entre les classes. Notre ontologie pour le système d'agents mobiles devient un moyen universel pour exprimer la politique de sécurité de la plate-forme d'exécution et l'agent mobile.

3.4. Correspondance sémantique entre les politiques de sécurité

Dans cette section, nous allons présenter le processus de correspondance entre les politiques de sécurité basée sur l'ontologie MASO. En effet, le processus d'évaluation de la correspondance entre les deux politiques consiste à rechercher

une compatibilité sémantique entre les exigences et les capacités. En particulier : a) les exigences de la plate-forme sont comparées aux capacités de l'agent mobile. b) les capacités de la plate-forme sont comparées aux exigences de l'agent. Pour que cette comparaison donne un résultat positif, les deux conditions suivantes doivent être satisfaites :

- Les capacités exprimées dans la politique de sécurité de la plate-forme doivent répondre aux exigences de l'agent mobile.
- Les exigences de la plate-forme doivent être respectées par les capacités exprimées dans la politique de sécurité de l'agent mobile.

Dans la suite, nous détaillons le processus de correspondance sémantique entre ces deux politiques.

3.4.1. Création des politiques de sécurité (exigences et capacités)

Le processus de correspondance consiste à vérifier dans quelle mesure une exigence de sécurité spécifiée par l'agent est satisfaite par une capacité de sécurité spécifiée par une plate-forme d'exécution. Dans notre travail, nous avons utilisé le standard WS-Policy pour exprimer les exigences et les capacités au sein d'une politique de sécurité, en se basant sur l'ontologie MASO que nous avons créée. En effet, les politiques de sécurité sont exprimées à travers les concepts définis par cette l'ontologie. Ils peuvent être soit des instances de protocoles de sécurité tels que XACML, XML-Encryptions, ou des algorithmes de sécurité concrets tels que DES, RSA, ou collections de caractéristiques instanciées de ces protocoles comme la confidentialité, l'authentification. En d'autres termes, les exigences et les capacités sécurité peuvent être décrites en utilisant n'importe quel composant sur un niveau d'abstraction de l'ontologie de sécurité. Chaque politique de sécurité peut avoir plus d'exigence et de capacité.

Définition 1 : Nous définissons une exigence de sécurité (*Security Requirement*) comme une alternative de sécurité du type *Requirement*, permet d'atteindre un objectif spécifique de sécurité et regrouper un ensemble d'assertion de sécurité de l'ontologie MASO pour satisfaire l'objectif visé. Formellement, nous avons exprimé l'exigence de sécurité comme suite :

$$SR(object) = \sum AS_R$$

AS_R est un ensemble d'assertions de sécurité de l'ontologie *MASO* pour exprimer une exigence de sécurité. Par exemple, un agent mobile exige la confidentialité des données produites par la plate-forme d'exécution, dans ce cas, la politique de

l'agent exige un algorithme de cryptage asymétrique *RSA (AS_1)* avec un jeton de sécurité *X509Certificate (AS_2)*.

$$SR\ (\text{confidentialité}) = AS_1 + AS_2$$

Définition 2 : Nous définissons aussi une capacité de sécurité (*Security Capability)* comme une alternative de sécurité du type *Capability*, permet d'offrir un ensemble des mécanismes de sécurité, des protocoles et des algorithmes pour atteindre un objectif particulier de sécurité. Chaque alternative regroupe un ensemble d'assertions de sécurité de l'ontologie MASO pour satisfaire l'objectif visé. Formellement, nous avons exprimé la capacité de sécurité (SC) comme suite :

$$SC(object) = \sum AS_C$$

AS_C est un ensemble d'assertions de sécurité de l'ontologie MASO pour exprimer une capacité de sécurité. Par exemple, une plate-forme offre des capacités de sécurité pour assurer l'intégrité des données, la politique de la plate-forme offre une spécification *XML-Signature* comme un protocole de signature (AS_1) avec l'algorithme DSA (AS_2) et un certificat numérique X509 (AS_3) pour signer les données de l'agent mobile.

$$SC\ (\text{intégrité}) = AS_1 + AS_2 + AS_3$$

Les plates-formes précisent leurs exigences et capacités de sécurité dans une politique lisible aux agents mobiles. Aussi les agents disposent des politiques pour exprimer leurs exigences et capacités de sécurité. Les exigences de sécurité de l'agent doivent être satisfaites par les fonctionnalités de sécurité de la plate-forme, ainsi les exigences de sécurité de la plate-forme doivent également être satisfaites par les capacités de sécurité spécifiée par la politique de l'agent. Dans ce qui suit, nous présentons les règles de correspondance sémantiques entre les exigences et les capacités de sécurité.

3.4.2. Algorithme de correspondance des politiques

La découverte et la sélection des meilleures plates-formes pour exécuter les agents mobiles est une étape importante dans notre approche. En effet, le Système de Gestion des Services (SGS) extrait et analyse la politique de sécurité de chaque plate-forme pour récupérer les informations nécessaires à l'exécution sécurisée de l'agent. Ces informations permettent de spécifier la catégorie d'agents qui satisfait aux exigences de sécurité de la plate-forme, aussi les capacités de sécurité offertes par la plate-forme aux agents mobiles. L'agent mobile dispose aussi une politique

de sécurité pour déterminer ses exigences et ses capacités de sécurité. Les deux politiques sont utilisées pour filtrer les plates-formes les plus adéquates à l'exécution sécurisées des agents mobiles.

Le SGS utilise un algorithme de correspondance *(Matching-Algorithm)* pour déterminer le niveau de correspondance entre deux politiques de sécurité. Le SGS accepte les exigences et les capacités de sécurité de l'agent et celles de la plate-forme comme entrée et décide dans quelle mesure elles correspondent. L'algorithme de correspondance extrait le type le plus spécifique d'une exigence et d'une capacité qui assurent le même objectif de sécurité, ensuite il vérifie sa correspondance. Le type le plus spécifique est l'instance de la classe la plus basse dans l'ontologie de sécurité. Nous déterminons quatre résultats de correspondance possible entre une capacité et exigence de sécurité :

Correspondance parfaite *(Perfect-Match)* : une correspondance parfaite se produit lorsque l'exigence et la capacité à la fois se réfèrent au même concept ou à deux concepts équivalents. Par exemple, si une capacité et une exigence sont toutes les deux du type *SAML*, alors il y a une correspondance parfaite entre l'exigence et la capacité. Généralement, deux cas sont possibles :

- L'exigence et la capacité se réfèrent à la même notion sémantique.
- L'exigence et la capacité se réfèrent à des concepts sémantiques équivalents.

Dans les deux cas, si de plus les propriétés de l'exigence et de la capacité sont précisées alors leurs valeurs doivent être identiques.

Correspondance générale *(General-Match)* : si le type le plus spécifique de la capacité est inférieur dans la hiérarchie que le type le plus spécifique de l'exigence. On dit dans ce cas que l'exigence est plus générale que la capacité. Trois cas sont possibles :

- L'exigence précise un concept sémantique plus général que la capacité.
- L'exigence et la capacité se réfèrent au même concept sémantique, mais plus de détails sont spécifiés pour la capacité (en utilisant la construction de la propriété).
- L'exigence et la capacité se réfèrent à l'ontologie de sécurité MASO, mais l'exigence spécifie seulement l'objectif de sécurité, par contre la capacité s'exprime par des concepts de sécurité qui satisfaits l'objectif spécifié par l'exigence.

Correspondance négociable *(Negotiable-Match*) : si le type le plus spécifique de l'exigence est inférieur dans la hiérarchie que le type le plus spécifique de la capacité. On dit que la capacité est plus générale que l'exigence. Dans ce cas, la capacité ne satisfait pas adéquatement l'exigence de sécurité. Par exemple, si l'exigence est du type *X509Certiificate*, par contre la capacité est du type *Authentication*. On distingue trois cas possibles :

- L'exigence précise un concept sémantique plus spécifique que la capacité.
- L'exigence et la capacité se réfèrent au même concept sémantique, mais l'exigence précise plus en détail (en utilisant la construction de la propriété).
- L'exigence et la capacité se réfèrent à l'ontologie de sécurité MASO, mais la capacité détermine seulement un objectif de sécurité, tandis que l'exigence s'exprime par des concepts de sécurité qui satisfaits l'objectif de sécurité spécifié par la capacité.

Lorsque le résultat global de correspondance des deux politiques donne un niveau de correspondance négociable, une nouvelle étape de négociation est nécessaire pour vérifier efficacement le respect de l'exigence.

Aucune correspondance *(No-Match*) : si les types les plus spécifiques de l'exigence et de la capacité n'ont aucune relation dans l'ontologie de sécurité, alors il n'est pas de correspondance entre les deux. Deux cas sont possibles :

- L'exigence et la capacité se réfèrent à des concepts sémantiques qui n'ont aucune relation sémantique.
- L'exigence et la capacité référent au même concept sémantique, mais leurs propriétés présentes des spécifications différentes.

Le SGS utilise l'algorithme de correspondance pour déterminer le degré de correspondance entre deux politiques. Nous avons divisé le processus de correspondance sémantique entre les politiques en deux étapes. La première consiste à déterminer le résultat de correspondance entre chaque paire exigence-capacité de sécurité, l'objectif est de trouver la capacité qui correspond au mieux à une exigence. La deuxième est l'évaluation de la correspondance globale entre les deux politiques. La correspondance globale est définie comme le minimum entre les résultats de correspondance individuelle évalués dans la première étape.

Formellement nous avons représenté l'algorithme de correspondance sémantique entre deux politiques par une équation mathématique (1). On considère deux politiques de sécurité P1 et P2 (P1 pour l'agent et P2 pour la plate-forme). Nous définissons les fonctions *SR(object)* et *SC(object)* pour exprimer respectivement les

assertions de sécurité d'une exigence et d'une capacité.

$$\left.\begin{array}{l}(\forall R_i \in SR_{(object)}\ (P1)\ \exists C_j \in SC_{(object)}\ (P2) \,/\, C_j \text{ Satisfied } R_i) \\ (\forall R_i \in SR_{(object)}\ (P2)\ \exists C_j \in SC_{(object)}\ (P1) \,/\, C_j \text{ Satisfied } R_i)\end{array}\right\} \Rightarrow P1 \text{ matchs } P2 \quad (1)$$

R_i et C_j représentent l'exigence et la capacité les plus spécifiques des alternatives de sécurité de type *Requirement et* Capability pour assurer un objectif particulier de sécurité.

C_j *Satisfied* R_i signifie que le résultat de correspondance entre C_j et R_i est *Perfect-match* ou *General-match*.

3.4.3. Exemple de correspondance sémantique entre deux politiques de sécurité

Nous avons montré l'importance de la correspondance sémantique entre les politiques à travers l'exemple suivant : un agent mobile dispose une politique de sécurité définie par son créateur, cette politique exprime les exigences qui doivent être satisfaites par les plates-formes visitées, ainsi les capacités de sécurité offertes par l'agent aux plates-formes. La figure 3.5 présente une description de la politique de sécurité utilisée par l'agent mobile.

```
<wsp:Policy ID="..." name="..."
        xmlns:wsp="http://www.w3.org/ns/ws-policy"
        xmlns:MASO="http://namespace/MobileAgentSecurity">
    <wsp:ExactlyOne>
        <MASO:SecurityAlternative>
         <wsp:All>
                <!-- Spécifier le type de l'alternative de sécurité"-->
                 <MASO:hasType RDF:resource="&MASO;#Capability"/>
                <!-- Spécifier l'objectif de sécurité assuré par l'agent-->
                <MASO:hasObject RDF:resource="&MASO;#Authentication"/>
                <!-- Spécifier les assertions de sécurité pour assurer l'objectif de sécurité-->
                <MASO:SecurityAssertion>
                        <!-- L'objectif est assuré par une certificat numérique X.509-->
                        <MASO:ensuredByProtocol RDF:resource="&MASO;#X509"/>
                         <MASO:supportProtocol RDF:resource="&MASO;#X509">
                                <MASO:requireToken RDF:resource="&MASO;#X509Certificate"/>
                         </MASO:supportProtocol>
                </MASO:SecurityAssertion>
         </wsp:All>
        </MASO:SecurityAlternative>
    </wsp:ExactlyOne>
    <wsp:ExactlyOne>
        <MASO:SecurityAlternative> <wsp:All>
          <!-- Spécifier une exigence de sécurité pour assurer la confidentialité des données-->
                <MASO:hasType RDF:resource="&MASO;#Requirement"/>
                <MASO:hasObject RDF:resource="&MASO;#Confidentiality"/>
```

```
                    <MASO:SecurityAssertion>
                     <!--Les techniques demandés par l'agent pour assurer la confidentialité des données -->
                      <MASO:ensuredByAlgorithm RDF:resource="&MASO;#RSA">
                       <MASO:EncryptedElement RDF:resource="&MASO;#Data"/>
                       <MASO:usesToken RDF:resource="&MASO;#X509Certificate"/>
                      </MASO:ensuredByAlgorithm>
                    </MASO:SecurityAssertion>
          </wsp:All> </MASO:SecurityAlternative>
    </wsp:ExcactlyOne>
</wsp:Policy>
```

Figure 3. 5. Politique de sécurité de l'agent mobile

Comme le montre la figure 3.5, l'agent mobile dispose une capacité d'authentification avec un certificat numérique *X.509* et nécessite une plate-forme qui capable d'exécuté le cryptage asymétrique *RSA* pour crypter les données produites dans la plate-forme visitée avec la propriété *encryptedElement*. Le préfixe comme «*MASO : SecurityAlternative*» dans la politique représente que la composante pertinente a été référencée à partir de l'ontologie de sécurité *MASO*.

On suppose qu'une plate-forme enregistrée dans de l'annuaire UDDI satisfait l'exigence fonctionnelle de l'agent. La plate-forme exige que les agents mobiles doivent s'authentifier et capable de supporter la spécification de cryptage *XML-Encryption* pour crypter les informations sensibles dans l'agent mobile. De la même manière que précédemment, la politique de sécurité de cette plate-forme peut être décrite comme dans la figure 3.6.

```
<wsp:Policy ID="..." name="..."      xmlns:wsp="http://www.w3.org/ns/ws-policy"
                                     xmlns:MASO="http://namespace/MobileAgentSecurity">
    <wsp:ExcactlyOne>
        <MASO:SecurityAlternative> <wsp:All>
                        <!--Cette alternative exige que tous les agents doivent s'authentifier -->
                        <MASO:hasType RDF:resource="&MASO;#Requirement"/>
                        <MASO:hasObject RDF:resource="&MASO;#Authentication"/>
        <wsp:All> </MASO:SecurityAlternative>
    </wsp:ExcactlyOne>
    <wsp:ExcactlyOne>
        <MASO:SecurityAlternative>  <wsp:All>
                <MASO:hasType RDF:resource="&MASO;#Capability"/>
                <MASO:hasObject RDF:resource="&MASO;#Confidentiality"/>
                <MASO:SecurityAssertion>
                 <!--Les techniques demandés par l'agent pour assurer la confidentialité des données -->
                 <MASO:ensuredByMechanism RDF:resource="&MASO;#EncryptionAsymmetric">
                        <MASO:supportProtocol RDF:resource="&MASO;#XML-Signature"/>
                 </MASO:ensuredByMechanism>
                </MASO:SecurityAssertion>
         </wsp:All> </MASO:SecurityAlternative>
    </wsp:ExcactlyOne>
</wsp:Policy>
```

Figure 3. 6. Politique de sécurité d'une plate-forme d'exécution

Le SGS utilise le raisonneur de sécurité qui est basé sur l'ontologie de la sécurité et de l'algorithme de correspondance introduit dans la section 3.4.2. Le SGS décide si la politique de sécurité de l'agent correspond à la politique de sécurité de la plate-forme. Le SGS accepte les deux politiques de sécurité comme entrée puis fait la correspondance sémantique. Les étapes suivantes sont prises par le raisonneur de sécurité afin de décider le degré de correspondance entre les deux politiques :

Étape 1 : le raisonneur analyse les politiques de sécurité, puis extraire les informations suivantes :

- L'agent mobile offre une capacité d'authentification avec un certificat numérique *X.509*, Le certificat *X509Certificate* est le type le plus spécifique pour cette alternative d'authentification.
- L'agent mobile exige un algorithme de cryptage asymétrique, l'algorithme *RSA* est le type le plus spécifique pour l'alternative de confidentialité.
- La plate-forme offre une capacité de cryptage avec la spécification *XML-Encryption*, dans ce cas, *XML-Encryption* est le type le plus spécifique pour assurer l'objectif de confidentialité.
- La plate-forme exige l'authentification des agents mobiles qui veut utiliser leurs services. Dans ce cas, l'exigence est exprimée en tant qu'un objectif de sécurité authentification.

Étape 2 : Le raisonneur de sécurité utilise l'algorithme de correspondance sémantique pour déterminer le degré de correspondance entre l'exigence de l'agent et la capacité de la plate-forme.

Pour assurer l'objectif de confidentialité, l'agent mobile utilise l'algorithme asymétrique *RSA* comme une exigence plus spécifique, la plate-forme offre une spécification *XML- Encryption* comme une capacité plus spécifique. L'exigence et la capacité ne sont pas du même type. Dans l'ontologie *MASO,* l'instance *XML-Encryption* de la classe *SecurityProtocol* permet d'adopter d'un ou plusieurs algorithmes de cryptage avec la propriété *adoptAlgorithm*. Dans ce cas, *XML-Encryption* est un protocole qui adopte l'algorithme asymétrique *RSA* pour le cryptage asymétrique. En effet, le type le plus spécifique de l'exigence est inférieur dans la hiérarchie que le type le plus spécifique de la capacité, donc le degré de correspondance est : *Negotiable-Match*.

Étape 3 : de la même manière, le raisonneur de sécurité détermine le degré de correspondance entre l'exigence de la plate-forme et la capacité de l'agent.

Pour assurer l'objectif d'authentification, l'agent mobile offre une méthode d'authentification avec le certificat *X509Certificate* comme un type plus spécifique, par contre la plate-forme exige l'authentification des agents mobiles visiteurs. Dans l'ontologie *MASO,* l'instance *X509Certificate* est une instance de la classe *SecurityToken* qui permet d'assurer l'objectif d'authentification. Dans ce cas, le type le plus spécifique de la capacité est inférieur dans la hiérarchie que le type le plus spécifique de l'exigence, donc le degré de correspondance est : *General-Match.*

Étape 4 : Enfin, le raisonneur de sécurité détermine le degré de correspondance globale entre les deux politiques. Le raisonneur prend le minimum des degrés de correspondance entre les résultats individuels de correspondance évalués pour chaque alternative de sécurité. Dans ce cas, le degré de correspondance globale est : *Negotiable-Match.*

3.5. Conclusion

Dans ce chapitre, nous avons présenté la modélisation des politiques de sécurité en se basant sur la spécification WS-Policy. Ensuite, nous avons structurer la politique de sécurité sous forme des exigences et des capacités de sécurité. Puis, nous avons montré que la correspondance syntaxique pose un problème d'incompatibilité entre les politiques de sécurité. Pour résoudre ce problème, nous avons intégré l'aspect sémantique et des connaissances dans le domaine de sécurité lors l'intersection entre les politiques. Pour cela, on a construit une ontologie dans le domaine de sécurité d'agents mobiles «MASO».

L'ontologie MASO permet de faciliter l'analyse automatique de la compatibilité sémantique entre les politiques de sécurité, ainsi il permet d'annoter les capacités et les exigences de sécurité à l'égard de diverses notions de sécurité. Nous avons proposé un algorithme pour faire la correspondance sémantique entre la politique de sécurité d'agent et les plates-formes visitées. Le processus de correspondance consiste à vérifier dans quelle mesure une exigence de sécurité est satisfaite par une capacité de sécurité, l'évaluation de la correspondance entre les politiques est basée sur l'ontologie MASO.

Dans le chapitre suivant, nous proposons un modèle d'agents mobiles à base de composants. Ce modèle facilite la gestion, l'adaptation et la sécurisation des composants de manière unifiée. Ensuite, nous présentons l'architecture interne des plates-formes offrant des services aux agents mobiles. Chaque service est modélisé par un agent de service.

CHAPITRE 4

Modélisation des agents mobiles à base de composants et les agents de services

4.1. Introduction

Les travaux de recherche relatifs à la sécurité des agents mobiles, traitent principalement le déploiement de l'agent mobile, c'est-à-dire ils décrivent comment les agents sont lancés, la façon dont il se déplace d'une plate-forme à une autre au cours d'exécution pour accéder à des données ou des ressources distantes, de créer et d'accumuler les résultats qu'il a obtenus sur les différentes plates-formes visitées. Dans la plupart des documents, le code, l'identité, la mobilité et la sécurité de l'agent sont supposés comme étant déjà disponibles au moment où les agents sont adoptés ou lancés.

Notre approche répond à des questions importantes lors de la création d'agents mobiles, comme la modularité du code sous forme de composants, la spécification détaillée aux tâches, la mobilité et la sécurité de l'agent mobile. L'objectif principal de ces systèmes est de créer des agents mobiles à base des composants, fiables qui peuvent faire confiance et dont la confiance peut être vérifiée par une autorité de confiance.

Afin de doter l'agent mobile de la faculté d'adaptabilité, nous avons essayé d'intégrer les composants logiciels dans l'architecture interne des agents mobiles. Les agents sont alors construits par assemblage de composants. Les composants fonctionnels étant placés au niveau de base, tandis que les composants non-fonctionnels sont regroupés dans le méta-niveau. Avec cette structure, le méta-niveau contrôle et adapte le niveau de base en fonction de la politique de sécurité d'agent et la plate-forme d'exécution.

Dans la première partie de ce chapitre, nous discutons l'intérêt des agents mobiles à base de composants et les différentes approches de modélisation proposées. Ensuite, nous présentons notre approche de modélisation et le processus de création de l'agent. Dans la deuxième partie, nous exposons l'architecture interne des plates-formes offrant les services aux agents mobiles visiteurs. Nous présentons

ensuite comment modéliser ces services par des agents de services et adapter le registre UDDI pour publier les services offerts par l'agent de service.

4.2. Structure réflexive des agents mobiles

La réflexivité construit une approche attrayante pour le développement des applications adaptables. Elle est considérée comme une manière d'organisation interne d'un système pour faciliter son développement, son adaptation et sa réutilisation. Le grand intérêt de ce type d'approche est de permettre d'exprimer des traitements en termes extrêmement génériques, qui n'utilisent que les notions constitutives du système.

Un système est dit réflexif s'il est capable d'appliquer à lui-même ses propres capacités d'action (capacités de description, de calcul, de pensée…). Il possède une auto représentation décrivant la connaissance qu'il a de lui-même. En réflexivité, le programme a accès à sa propre représentation et peut donc agir sur lui-même. Autrement dit, la réflexivité représente la capacité, pour un programme, à manipuler comme données quelque chose représentant l'état de ce programme durant son exécution [Malenfant, 2004]. Le processus qui consiste à rendre accessible un encodage du programme et de son état d'exécution comme données s'appelle réflexion. Dans notre approche, la réflexivité structure un agent mobile en deux niveaux d'abstractions, un niveau de base et un méta-niveau :

- Le niveau de base désigne le comportement de l'agent mobile. C'est le code fonctionnel de l'agent organisé sous forme des composants, chaque composant effectue une tâche particulière et s'exécute dans une plate-forme.
- Le méta-niveau concerne sa description, permet de réaliser des transformations sur le niveau de base. Il comprend le code non fonctionnel de l'agent, par exemple la description non fonctionnelle de chaque composant, la sécurité, la mobilité et la communication.

Cette séparation entre les niveaux permet de rendre le niveau de base transparent à un changement structurel ou opératoire dans le méta-niveau (figure 4.1 ci-dessous).

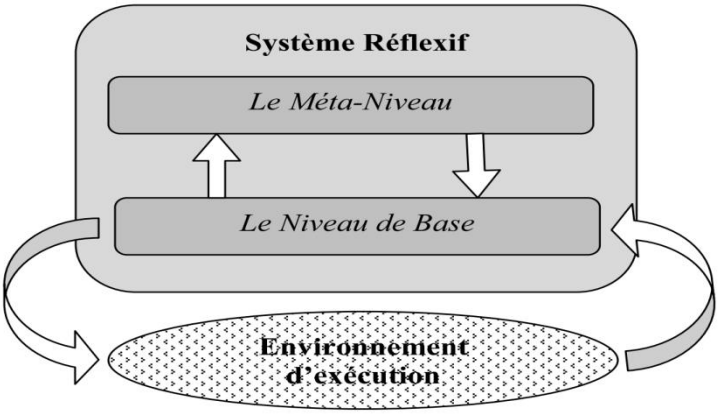

Figure 4. 1. Modèle d'exécution réflexif

Un système réflexif peut alors répondre à des questions le concernant (capacité d'introspection), mais aussi s'auto-modifier (capacité d'intercession). Les interactions entre un niveau de base et son méta-niveau forment, en effet, un protocole méta, et s'appuient sur deux démarches [David, 2002] :

- L'introspection permet, à un système, de raisonner sur lui-même pendant son exécution.
- L'intercession permet, à un système, de modifier lui-même ses propriétés pendant son exécution.

4.3. Agent à base de composants

Les concepteurs d'agents veulent modeler leur propre architecture d'agents selon les besoins d'une application ou d'un domaine applicatif. Une approche prometteuse pour ce faire consiste à concevoir les agents en composant librement différentes briques, en s'appuyant notamment sur le concept de composant logiciel [Szyperski, 1998].

Des travaux relativement récents se sont intéressés à la construction d'agents par assemblage de composants. La programmation par composants est, en effet intéressante pour ce type de développement, car les agents présentent souvent des capacités proches d'une application à une autre (communication, perception, planification, ...). Dans ce cas, la construction d'un agent consiste à assembler des composants préexistants qui matérialisent chacun une partie bien spécifique de l'architecture et du comportement de l'agent. En d'autres termes, un agent est une entité logicielle contenant un assemblage de composants qui matérialise son comportement. Le développement d'applications multi-agents bénéficierait grandement de l'utilisation d'une bibliothèque de composants implantant ces capacités partagées. Plusieurs environnements de développement ont ainsi adopté

une telle approche [Brazier et al., 2002], [Occello et al., 2002], [Ricordel et al., 2002], [Krutisch et al., 2003].

4.3.1. Intérêt des composants pour les agents

Le premier avantage des agents à base de composants est de permettre à un concepteur de décrire puis gérer de manière unifiée la structure et le comportement des agents. Un composant représente une ou plusieurs compétences que possède un agent et qui peuvent être utilisées ou déclenchées sous certaines conditions. Une compétence d'un agent s'apparente donc à un ensemble de fonctionnalités accessibles à travers une interface fournie d'un composant de cet agent. Une compétence offerte par un composant peut requérir d'autres compétences chez les composants de l'agent. Ainsi, le comportement d'un agent se matérialise par un assemblage de composants qui permet de relier entre elles, et de manière structurée, les différentes compétences de cet agent.

Le deuxième avantage des agents à base de composants est de favoriser la réutilisation et donc d'accélérer le développement et de limiter les erreurs d'implémentation. Dans les travaux sur la conception d'agents, l'approche à base de composants est avant tout utilisée comme un cadre pour la spécification du comportement de l'agent. En effet, les composants permettent de réutiliser des capacités récurrentes chez les agents (communication, perception, planification…). Construire des agents à base de composants consiste principalement à récupérer un ensemble approprié de composants et de les assembler entre eux. Donc, l'utilisation des composants tend à faciliter la construction d'agents. Le fait de réutiliser des composants maintes fois éprouvés limite les risques d'erreurs dus à une mauvaise implantation.

Le troisième avantage des agents à base de composants est de faciliter l'adaptation des agents. Il est assez facile d'ajouter, de supprimer ou de modifier les compétences d'un agent, puisque les applications à base de composants ont l'avantage de pouvoir être réassemblées (sans trop d'efforts) selon les principes d'assemblage que définit le modèle de composant utilisé. Par conséquent, les composants logiciels sont utiles non seulement pour la construction d'agents, mais aussi pour l'adaptation de ces agents. Notons que le fait de réassembler un agent à base de composant permet d'adapter à la fois la structure et le comportement de l'agent.

4.3.2. Agents auto-adaptables à base de composants

Nous nous intéressons particulièrement à une sous-catégorie de ces travaux. Nous mettons l'accent sur les travaux dans lesquels les composants logiciels sont utilisés pour construire des agents capables de s'auto-adapter. Le développement par composants est en effet intéressant pour ce type d'application, car les agents présentent souvent des capacités proches d'une application à une autre. En quelque sorte, un agent est une entité logicielle contenant un assemblage de composants qui matérialise son comportement. Dans ce qui suit, nous analysons les principaux travaux qui proposent des modèles d'agents capables de s'auto-adapter et qui sont basés sur la notion de composants logiciels.

4.3.2.1. MAST

MAST a été développé par l'équipe SMA de l'Ecole des Mines de Saint-Etienne [Vercouter, 2004]. C'est un environnement de développement et de déploiement d'applications multi-agents. Dans MAST, une application multi-agent est construite comme un assemblage de composants. La conception, qui reprend la décomposition VOYELLES [Demazeau, 2001], s'applique sur deux niveaux et le modèle de composant proposé tient compte de ces niveaux en distinguant deux types de composants. Au niveau système, un SMA est composé d'agents (vus comme des composants) et de certaines entités fonctionnelles nommées composants orientés système (COS) : par exemple, la plate-forme FIPA (Foundation for Physical Intelligent Agents) [Foundation for Intelligent Physical Agents, 2011] comme un intergiciel de communication et le kernel MadKit [Gutknecht et al., 2001] comme support organisationnel. Au niveau agent, un agent est également conçu comme un assemblage de composants au sens plus classique du terme, dits composants orientés agent (COA).

Conception d'un agent : pour construire un agent MAST, il suffit de lui ajouter un composant « noyau » et un ensemble de composants fonctionnels (les COAs). Comme le montre la figure 4.2. MAST permet de connecter automatiquement et de manière flexible les composants qui forment un agent. En effet, les événements émis par les différents composants de l'agent sont capturés par le composant noyau. Ce dernier se base sur la description sémantique de chaque composant (i.e. leur rôle) pour transmettre chaque événement reçu aux composants capables de le traiter. L'évènement est transmis aux COAs successivement selon l'attribut « priorité » qui les caractérise jusqu'à ce que l'évènement soit consommé par l'un des COAs. Pour définir un assemblage de composants, le concepteur doit définir la priorité (un réel) de chaque interface des composants présents.

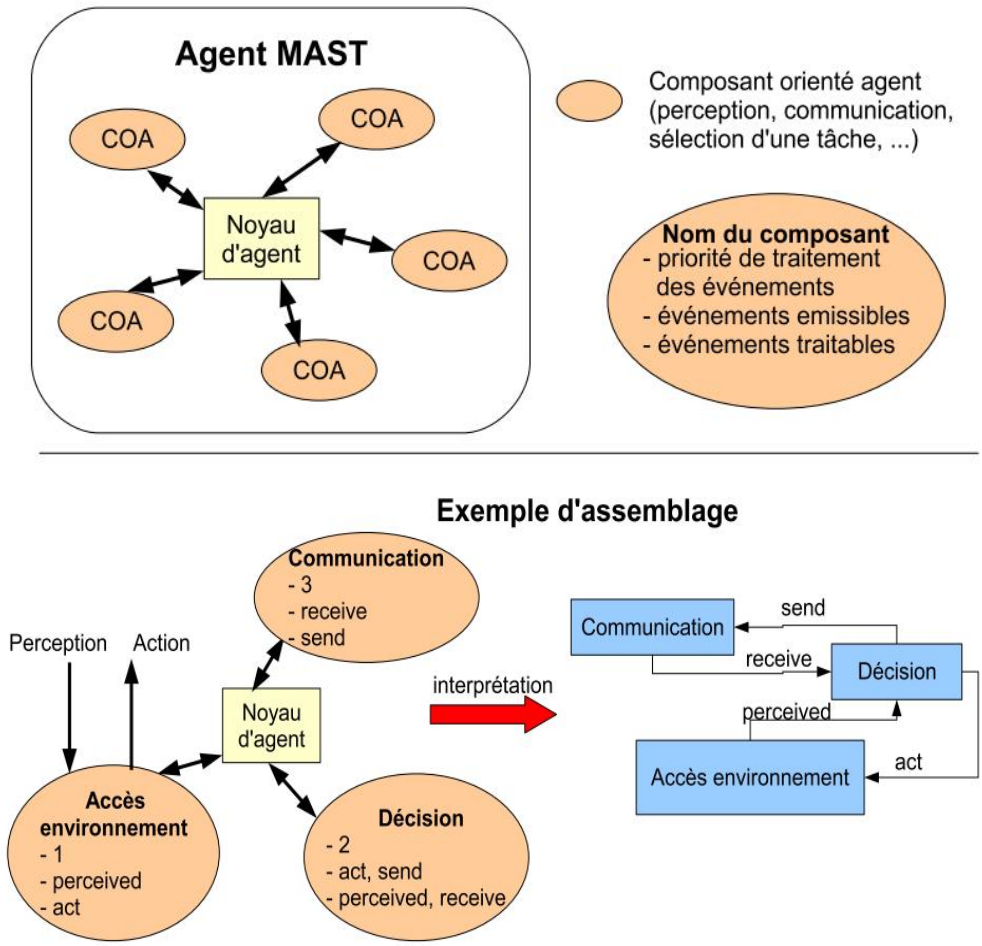

Figure 4. 2. Exemple d'architecture d'agent autonome avec MAST

Adaptation d'un agent ces connexions implicites sont intéressantes pour la flexibilité qu'elles procurent à l'assemblage de l'agent, puisqu'elles sont automatiquement déduites en fonction des interfaces des composants. Ainsi, l'adaptation des agents qui consiste à lui ajouter ou supprimer des composants est réalisée automatiquement.

4.3.2.2. MALEVA

MALEVA est un modèle d'agents proposé par le LIP6 à l'Université de Paris 6 notamment pour la conception de systèmes multi-agents [Briot et al., 2006]. Ce modèle est utilisé pour construire des agents logiciels dont le comportement et la structure peuvent être adaptés. Il prévoit deux sortes d'interaction entre composants : un flot de données et un flot de contrôle. Le principal avantage d'avoir un flot de contrôle explicite dans un assemblage de composant est de faciliter l'identification de problème de concurrence. MALEVA a notamment été utilisé dans les domaines des sociétés artificielles et de la simulation multi-agents.

Conception d'un agent : avec MALEVA, un concepteur d'agents dispose d'une bibliothèque de composants représentant des comportements élémentaires. Il peut assembler plusieurs composants existants dans un composant composite de manière à constituer des comportements complexes réutilisables dans différents contextes ou différentes familles d'agents. Les auteurs utilisent aussi une forme simplifiée de patron de conception [Gamma et al., 1995] permettant d'instancier un comportement complexe abstrait et spécialisable : un pré-câblage entre des composants fixes et des composants à insérer. Concevoir un agent revient à composer, de manière structurelle et fonctionnelle, différents composants comportementaux au sein de l'agent, comme l'illustre la figure 4.3. Par exemple, le comportement « Proie » est défini en connectant trois composants : la fuite (le composant *FuirAgent*), le mouvement aléatoire (le composant *MvtAléatoire*), et un composant de contrôle appelé *Switch* qui permet de réifier la conditionnelle perception/non-perception. La gestion du contrôle revient à définir et ajouter des composants spécifiques qui ont à charge de traiter les compositions associées aux données et celles associées aux contrôles.

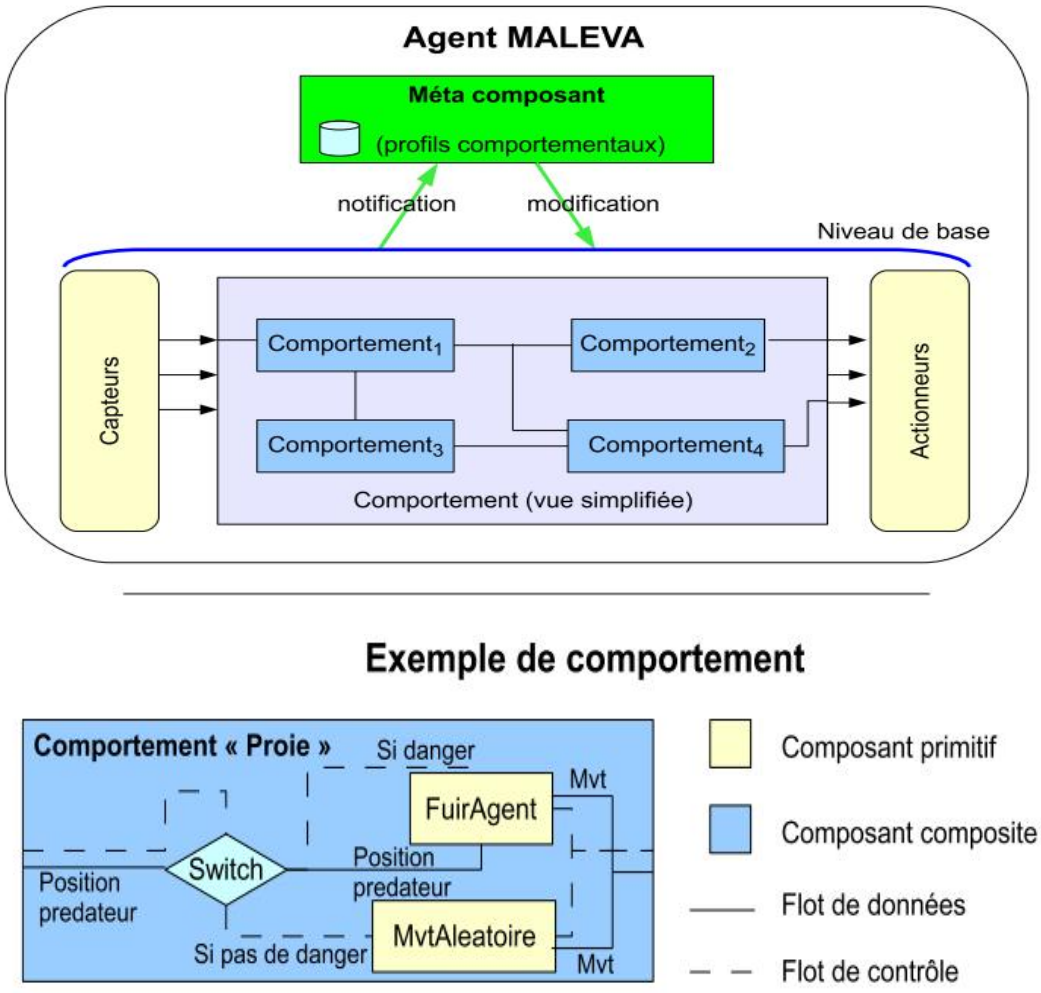

Figure 4. 3. Exemple d'architecture d'agent réactif avec MALEVA

Adaptation d'un agent : les adaptations peuvent être déclenchées par l'agent après qu'un de ses composants de comportement émette une requête particulière. Cette requête est transmise à un composant spécifique appelé « méta composant », qui se comporte comme un fournisseur de « profils comportementaux » préalablement définis par le concepteur de l'agent. Ces profils prédéfinis décrivent des assemblages spécifiques de composants. Donc, ils améliorent le contrôle du concepteur de l'agent sur les assemblages à générer en interdisant les assemblages non prédéfinis.

4.3.2.3. JavAct$^{\delta}$

JavAct$^{\delta}$ est un modèle d'agents proposé par l'équipe IRIT-LYRE de l'Université Paul Sabatier de Toulouse pour la programmation d'applications concurrentes, réparties et mobiles [Leriche et al., 2006, 2007]. Ce modèle dérive du précédent JavAct [Arcangeli et al., 2001] et de l'architecture réflexive proposée par Marcoux et al. [Marcoux et al., 2001]. Les auteurs s'attachent à permettre au concepteur de vérifier l'architecture d'un agent à partir d'une description d'assemblage de composants, afin de générer des agents flexibles, mais toujours corrects. Le modèle de composants utilisé est proche d'EJB (Entreprise Java Beans) : un composant est caractérisé par son interface, une ou plusieurs réalisations et un moyen d'exécution (son conteneur).

Conception d'un agent : chaque agent possède un niveau de base contenant du code fonctionnel (comportement déterminé par le programmeur de l'application agent) et un niveau méta décrivant les services non fonctionnels comme l'envoi asynchrone de messages, la création locale et distante d'agents, le changement de comportement, la mobilité, la gestion de la boîte aux lettres et la gestion du cycle de vie de l'agent (voir la figure 4.4). Ces derniers sont appelés les « mécanismes opératoires de l'agent » et chacun d'eux est représenté par un composant à grain fin (ou micro-composant) qui offre un unique service non fonctionnel. En outre, les agents ont une architecture de micro-composants en étoile autour d'un connecteur nommé « Contrôleur ». Ce dernier fonctionne comme un bus logiciel et permet d'implémenter le principe de délégation pour faciliter l'évolution de l'architecture. Par ailleurs, le niveau méta de l'agent correspond à un type d'agent défini selon un « style d'architecture » (niveau méta). JavAct$^{\delta}$ permet de concevoir différents styles d'agents (réactif, BDI, mobile, . . .), celui qui est explicité un style axé sur le modèle d'acteurs [Hewitt, 1977], sur l'adaptabilité et la mobilité.

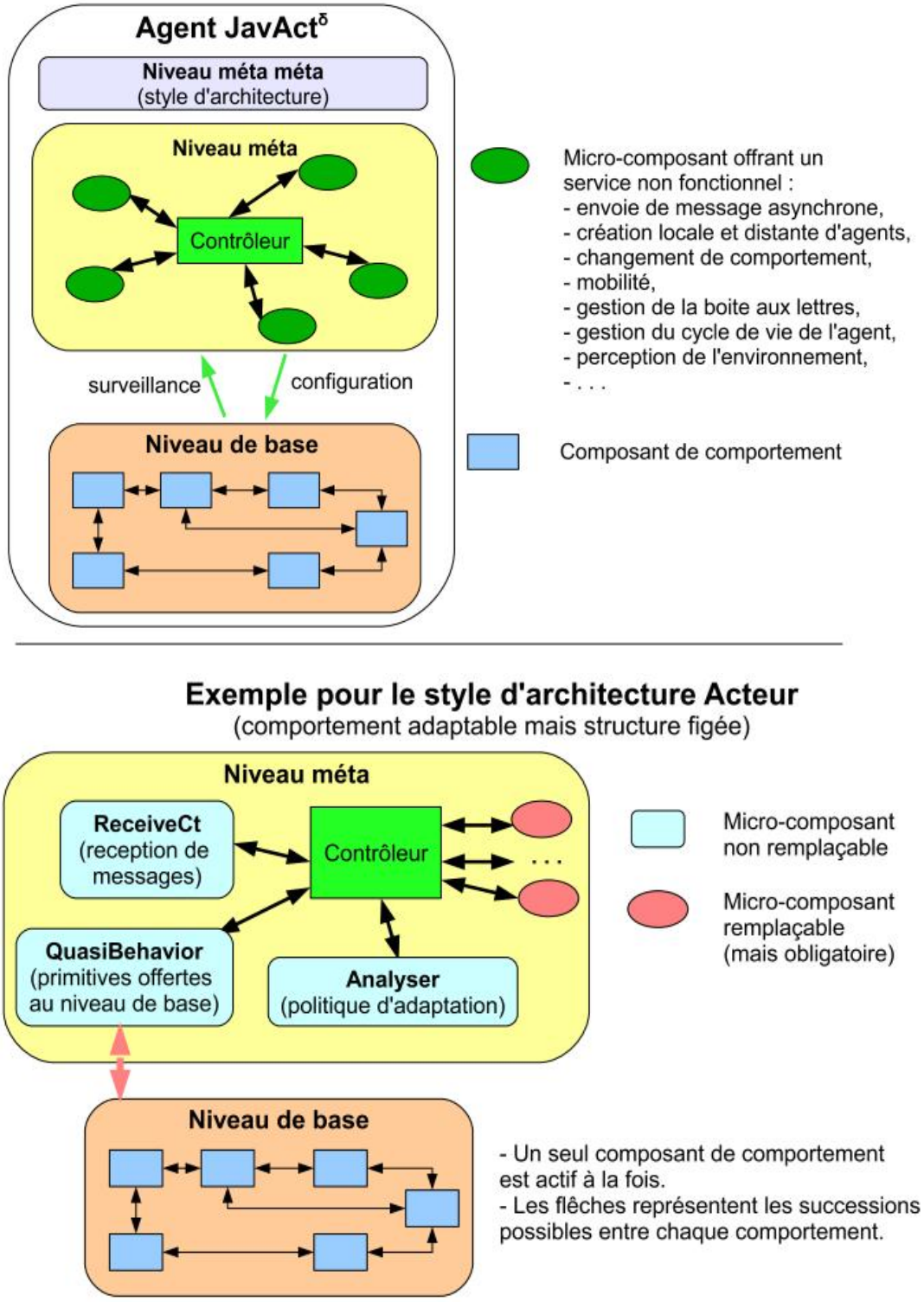

Figure 4. 4. Exemple d'architecture d'agent adaptable avec JavAct$^{\delta}$

Adaptation d'un agent : dans ce modèle, on distingue les adaptations au niveau de base et les adaptations au niveau méta. Une adaptation au niveau de base consiste à remplacer le composant de comportement courant par un autre grâce à une primitive du type *void become(QuasiBehavior b).* Une adaptation au niveau méta consiste à remplacer un micro-composant grâce à une primitive *with(Type*

newComp) pour remplacer le micro-composant de type *Type* par *newComp*. Ces primitives sont également accessibles par le niveau de base. Les adaptations sont dynamiques et pour éviter les problèmes d'incohérence, l'adaptation n'est immédiate que si l'agent est en attente de message, sinon elle est différée après la fin de l'exécution du comportement courant.

4.3.3. Synthèse

Dans cette sous-section, nous avons présenté l'intérêt de l'utilisation des composants pour les agents mobiles. Aussi, nous avons étudié plusieurs modèles d'agents auto-adaptables à base de composants : MAST, MALEVA et JavAct$^{\delta}$.

Dans chacun des travaux ci-dessus, le modèle de composants utilisé est spécifique au modèle d'agents. D'autre part, les auteurs mettent clairement en avant la forte adaptabilité des agents grâce à leur modèle d'agents. Les adaptations d'un agent sont facilitées par l'utilisation de composants.

Notre étude dans ce chapitre tend à montrer que les agents à base de composants sont plus flexibles que les agents standards et que l'emploi de composants logiciels permet d'adapter simplement le comportement et la structure des agents. Les meilleurs modèles d'agents auto-adaptables sont ceux qui offrent une infrastructure d'adaptation simple à utiliser, riche en expressivité et surtout elle doit faire partie intégrante du modèle d'agents afin que les adaptations soient non intrusives vis-à-vis de la propriété d'autonomie des agents. Ainsi, au lieu de créer une architecture d'agents figée et complexe, il est préférable que l'architecture de l'agent soit facilement adaptable et réutilisable.

Dans notre approche on utilise une structure réflexive et un modèle d'agent à base des composants pour construire un agent mobile sécurisé. En effet, l'utilisation d'une structure réflexive permet à l'agent de modifier la structure et la sécurité de ses composants fonctionnels en fonction de sa politique de sécurité. Ainsi l'utilisation de la modularité de code sous forme des composants nous semble un bon candidat pour supporter notre stratégie de protection de l'agent mobile. De plus, la conception par composant spécifiée permet de spécialiser la liaison devant être établie entre le méta-niveau de l'agent mobile et son niveau de base (la communication, la sécurité et la mobilité de l'agent etc.). Nous avons proposé de placer les composants fonctionnels au niveau de base. Tandis que les composants non-fonctionnels sont regroupés dans le méta-niveau. Cela facilite le contrôle et l'adaptation des composants fonctionnels.

4.4. Modélisation des agents mobiles à base de composants

La figure 4.5 esquisse l'architecture générale d'un modèle d'agent mobile à base de composants [Razouki et al., 2015a, 2015b]. Cette architecture définit quatre parties principales dans un agent. L'interface entre l'agent et sa plate-forme d'exécution. Le niveau de base est la partie opérationnelle de l'agent, c'est-à-dire l'ensemble de composants implémentant le comportement de l'agent. Chaque composant représente une connaissance ou compétence pour réaliser des tâches spécifiques. Le méta niveau représente les services non fonctionnels de l'agent comme la sécurité, la communication et la mobilité. La mémoire d'agent contient les informations nécessaires pour l'accomplissement du travail de l'agent mobile.

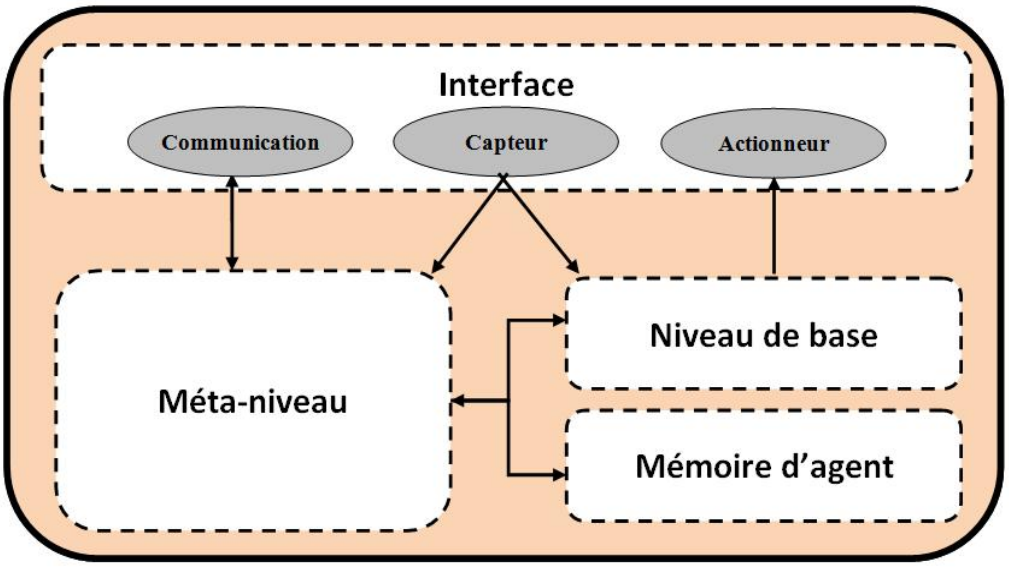

Figure 4. 5. Structure interne d'un agent mobile

La figure 4.5 illustre de manière schématique les interactions entre les différents composants internes de notre modèle d'agent mobile. La connexion entre l'actionneur et le niveau de base dépend de la plate-forme sur laquelle l'agent s'exécute et le méta-niveau de l'agent. En effet, le méta-niveau sélectionne les composants qui doivent être exécutés par l'actionneur en fonction de l'itinéraire de l'agent. Le capteur est utilisé pour l'acquisition des informations relatives à la plate-forme visitée, ces informations sont utilisées par le méta-niveau pour authentifier la plate-forme, déterminer les composants à exécuter et enfin la politique de sécurité à appliquer. Le capteur est utilisé aussi par le niveau de base pour récupérer quelques paramètres nécessaires à l'exécution d'un composant.

L'interface de communication est utilisée directement par le méta-niveau et indirectement par le niveau de base, ce qui permet à l'agent d'envoyer des messages vers d'autres agents et recevoir des messages de la part d'autres agents. Enfin, le niveau de base contient un ensemble de composants utilisables par l'agent

pour réaliser leur tâche, ces composants sont contrôlés par la partie méta-niveau et accédés à la mémoire d'agent pour ajouter ou récupérer les données d'agent.

4.4.1. Niveau de base d'un agent mobile

Le créateur d'agent crée le code de l'agent mobile sous forme des composants [Razouki et al., 2013d, 2013e]. Chaque composant à un identificateur unique dans l'agent, permet de réaliser une tâche précise, développer indépendamment et les réutiliser dans d'autres agents. En effet, un composant désigne une entité de données et d'instructions exécutée dans une plate-forme en fonction de l'itinéraire d'agent. Dans ce cas, la création de l'agent mobile consiste à assembler des composants préexistants dans une bibliothèque de composants. Comme l'illustre la figure 4.6, chaque composant chargé d'implémenter une partie bien spécifique du comportement de l'agent. En d'autres termes, un agent est une entité logicielle contenant un assemblage de composants qui matérialise son comportement.

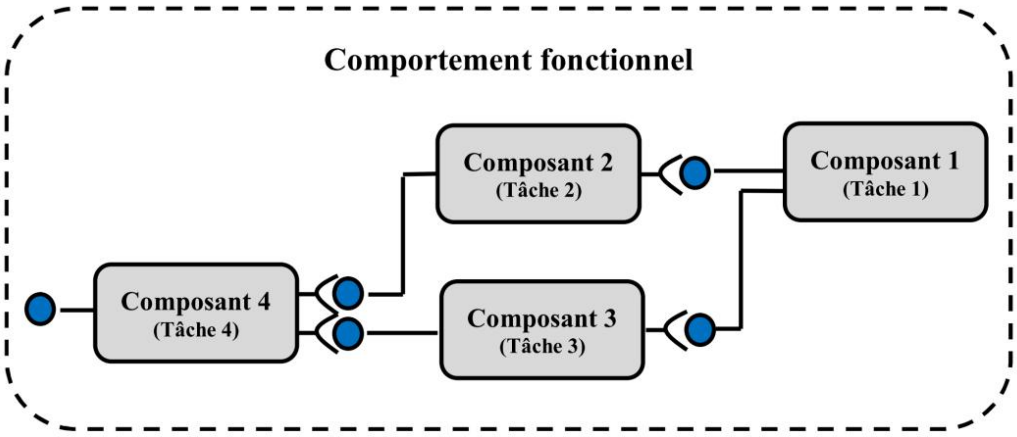

Figure 4. 6. Comportement fonctionnel d'un agent mobile

Les composants sont destinés à interopérer entre eux de manière explicite, c'est-à-dire qu'un composant peut utiliser plusieurs composants et il peut aussi être utilisé par d'autres composants. Deux composants ne peuvent interagir que s'ils ont des interfaces compatibles entre elles. Par exemple, si les fonctionnalités (généralement appelées services) qui sont nécessaires au premier composant sont décrites dans une interface requise, alors ces mêmes fonctionnalités doivent être décrites chez le second composant sous la forme d'une interface fournie. La combinaison et le séquencement entre ces composants doivent être préalablement étudiés et spécifiés par le créateur d'agent (développeur d'agent) sous forme d'une politique d'assemblage.

4.4.2. Méta niveau d'un agent mobile

Le méta niveau représente le code non-fonctionnel de l'agent mobile, il décrit les services qui contrôle, adapte et sécurise les différents composants fonctionnels du niveau de base. En effet, le méta-niveau décrit le processus d'adaptation et la sécurisation de l'agent mobile durant son exécution. Comme le montre la figure 4.7, le cœur de ce niveau nommé le contrôleur autour duquel sont connectés tous les autres éléments non-fonctionnels : le gestionnaire de communication, la politique d'assemblage, la politique de sécurité, description des composants et la mobilité de l'agent.

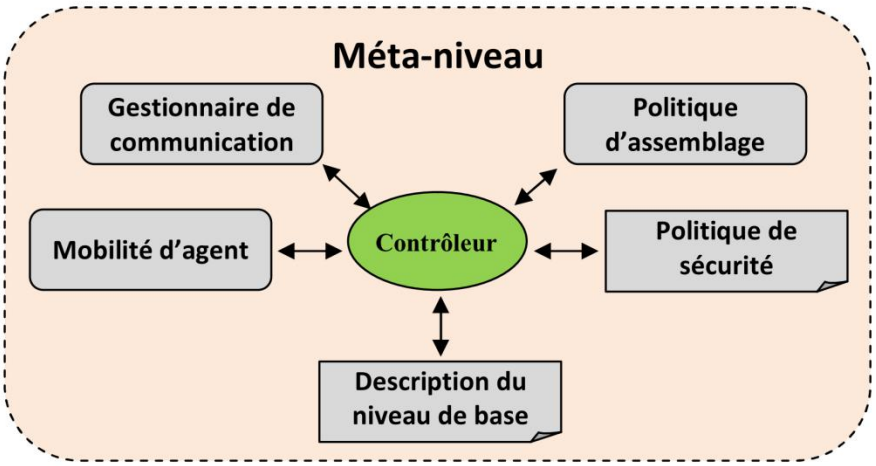

Figure 4. 7. Comportement non fonctionnel d'un agent mobile

Gestionnaire de communication : le gestionnaire de communication de l'agent permet à un agent d'utiliser l'interface de communication pour envoyer ou recevoir des messages. Le gestionnaire de communication intervient dans deux sortes de communications : la communication entre l'agent mobile et l'agent de service dans une plate-forme d'exécution. Cette communication permet à l'agent de demander un service particulier à l'agent de service pour accomplir leurs tâches. La communication entre l'agent et leur plate-forme d'origine.

Politique d'assemblage est un composant non fonctionnel permet de piloter le processus d'assemblage des composants mis en œuvre par l'agent, depuis le déclenchement d'un assemblage jusqu'à sa réalisation. Cette politique permet aussi d'initialiser les composants avec les paramètres par défaut. Le créateur d'agent doit spécifier un assemblage valide entre les différents composants fonctionnels pour construire le comportement de l'agent mobile. La politique d'assemblage est évaluée et validée par une autre entité de confiance, ce qui permet de vérifier la compatibilité entre les interfaces des différents composants.

La politique de sécurité permet de spécifier la politique de sécurité de l'agent mobile, cette politique regroupe les stratégies de sécurité utilisées par les composants fonctionnels. Chaque composant effectué une tâche précise dans l'agent mobile et exige un certain niveau de sécurité pour protéger son code et ses données produites dans une plate-forme. Dans notre approche, la définition de la politique de sécurité est basée sur l'ontologie de sécurité MASO (voir chapitre 3). La figure ci-dessus (figure 4.8) montre que l'agent mobile utilise un ensemble d'assertions de sécurité pour exiger le cryptage et la signature des données produites par les plates-formes visitées.

La politique de sécurité de l'agent mobile est signée à l'aide de la clé privée du créateur d'agent. Cela permet de s'assurer de leur intégrité.

```
<wsp:Policy ID="..." name="..."
 xmlns:wsp="http://www.w3.org/ns/ws-policy"
 xmlns:MASO="http://namespace/MobileAgentSecurity">
<wsp:ExcactlyOne>
 <MASO:SecurityAlternative>
  <wsp:All>
  <MASO:hasType RDF:resource="&MASO;#Requirement"/>
  <MASO:hasObject RDF:resource="&MASO;#Confidentiality"/>
  <MASO:SecurityAssertion>
    <MASO:ensuredByAlgorithm RDF:resource="&MASO;#3DES">
          <MASO:encryptedElement RDF:resource="&MASO;#Data"/>
    </MASO:ensuredByAlgorithm>
  </MASO:SecurityAssertion>
  </wsp:All>
 </MASO:SecurityAlternative>
 <MASO:SecurityAlternative>
 <wsp:All>
  <MASO:hasType RDF:resource="&MASO;#Requirement"/>
  <MASO:hasObject RDF:resource="&MASO;#Integrity"/>
  <MASO:SecurityAssertion>
  <MASO:ensuredByProtocol RDF:resource="&MASO;#XML-Signature">
    <MASO:adoptAlgorithm RDF:resource="&MASO;#RSA">
          <MASO:signedElement RDF:resource="&MASO;#Data"/>
          <MASO:usesCanonicalisation RDF:resource="&MASO;#XMLC14N"/>
    </MASO:adoptAlgorithm>
  </MASO:ensuredByProtocol>
  </MASO:SecurityAssertion>
 </wsp:All>
 </MASO:SecurityAlternative>
</wsp:ExcactlyOne>
</wsp:Policy>
```

Figure 4. 8. Description de sécurité d'un agent mobile

Description du niveau de base représente des informations sur les «contenus» possibles du niveau de base, en termes de structures et de fonctionnalités. Avec cette description l'agent détermine la fonctionnalité, les services fournis et requis par ses composants et enfin aide les plates-formes qui exécute l'agent d'ajouter les

données de manière structurée. Notons que nous avons utilisé le langage XML pour stocker les données collectées dans l'agent mobile.

La mobilité d'agent : un agent mobile se déplace selon un itinéraire préalablement établi. L'itinéraire est déterminé par le créateur d'agent si les plates-formes sont connues d'avance, sinon c'est le Système de Gestion de Sécurité d'Agents (SGSA) qui détermine et valide cet itinéraire. L'itinéraire de l'agent est représenté sous forme d'un ensemble des nœuds. Chaque nœud permet de lier :

- Un ou plusieurs composants qui doivent être exécutés par l'agent.
- L'identité de la plate-forme exécutant les composants.
- La politique de sécurité à appliquer lors de l'exécution d'un composant.
- La description du niveau de base des composants en cours d'exécution.

Durant l'exécution d'un composant, la plate-forme peut accéder aux données de l'agent ou ajouter des résultats à l'agent après l'exécution de ce composant. La figure 4.9 illustre la liste des plates-formes visitées par un agent mobile.

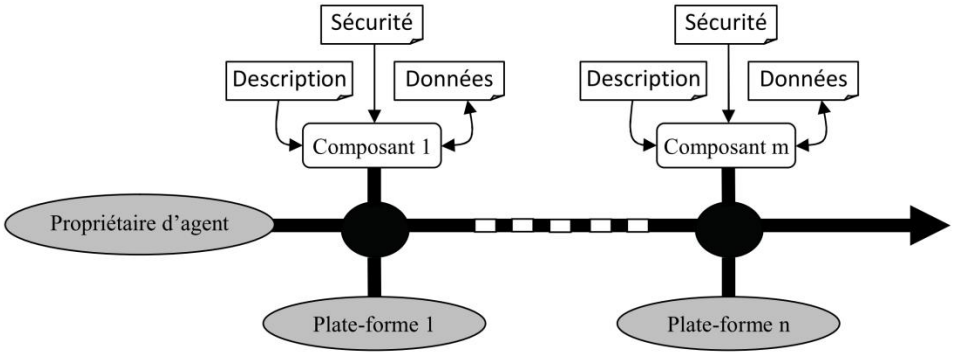

Figure 4. 9. Itinéraire d'un agent mobile

Contrôleur : joue le rôle de coordinateur entre les composants de l'agent mobile. Il permet aussi le déclenchement des opérations qui doivent être exécutées par les composants adéquats. Il est responsable également du contrôle du bon fonctionnement de tout le système de l'agent mobile. Après l'authentification d'une plate-forme par le capteur, le contrôleur identifie le composant à exécuter par la plate-forme, la politique de sécurité et la description de composant. Si le composant est crypté, le contrôleur demandé à la plate-forme de décrypter le composant (en fonction de la politique de sécurité utilisée par le composant).

Le contrôleur permet aussi de détecter les éventuelles anomalies de traitements telles qu'une tentative à une mauvaise exécution de l'agent mobile de la part de la plate-forme en question. Le contrôleur compare le temps d'exécution réel de chaque tâche avec le temps estimé. Si le contrôleur note un dépassement du temps

réglementaire, il peut supposer une utilisation malveillante et par conséquent, il doit arrêter le travail sur la plate-forme en question pour migrer vers la plate-forme suivante de son itinéraire. Le contrôleur envoie des notifications au SGSA, suite d'une attaque détectée afin de diminuer la crédibilité accordée à la plate-forme.

4.4.3. Interface d'un agent mobile

L'interface de l'agent mobile est le composant à travers lequel l'agent communique et interagit avec l'environnement extérieur. Il contient les dispositifs nécessaires lui permettant d'authentifier les plates-formes, de détecter les accès des plates-formes extérieures à ses ressources, de recevoir les requêtes et de présenter ses services sous une forme adéquate. L'interface réalise toutes ces tâches à travers les trois composants : "Communication", "Capteur" et "Actionneur" :

La communication est utilisée pour envoyer et recevoir des messages. Ce composant est contrôlé par le gestionnaire de communication.

Le capteur est utilisé pour la perception de l'environnement et de l'acquisition des données relatives à la plate-forme visitée. Ces données sont utiles à l'authentification (mot de passe, certificat, ticket, etc.). Trois types d'acquisitions peuvent être considérés : l'observation, l'inspection et l'interaction. Le type d'acquisition dépend de la donnée exigée. À titre d'exemple, le mécanisme d'interaction est employé pour l'acquisition de l'identité, le mécanisme d'inspection est utilisé pour la recherche d'un fichier spécifique et le mécanisme d'observation contrôle le nombre de tentatives de la plate-forme pour écrire son mot de passe. L'inspection, l'interaction et l'observation sont effectuées simultanément, dans le but de compliquer le comportement de l'agent mobile.

L'actionneur est utilisé pour exécuter une séquence d'actions relative à un composant sélectionné. Le choix de composant à exécuter est déterminé par le contrôleur en fonction de l'itinéraire d'agent. En effet, l'actionneur peut recevoir un arrêt d'exécution ou/et une migration par le contrôleur.

4.4.4. Mémoire d'un agent mobile

La mémoire d'agent contient les données statiques et dynamiques transportées à partir de propriétaire d'agent ou des plates-formes visitées.

Les données statiques : sont des informations transportées à partir de l'hôte d'origine. Ces informations contiennent des données qui ne changent pas telles que l'identité unique de l'agent mobile. Les données statiques contiennent aussi les données qui lui sont nécessaires pour exécuter certains composants. Afin d'assurer

l'intégrité des données statiques, le propriétaire d'agent signe numériquement ses données. Pour des raisons de confidentialité, certaines informations sensibles sont cryptées à l'aide de la clé publique de certaines plates-formes, en utilisant un algorithme de cryptage asymétrique.

Les données dynamiques : sont des informations collectées à partir de l'environnement d'exécution après chaque migration. Comme les résultats partiels de calcul de chaque plate-forme, les données obtenues au cours de l'exécution d'un composant. Ces données sont stockées sous forme d'un fichier XML en se basant sur la description du niveau de base de l'agent. La sécurité des données dynamiques est assurée par un ensemble de spécifications, de protocoles et d'algorithmes spécifiés par la politique de sécurité de l'agent.

4.5. Processus de création et de sécurisation d'un agent mobile

4.5.1. Création et émission initiale d'un agent mobile vers le SGSA

Le créateur d'agent est l'entité responsable de la création et de l'émission initiale des agents mobiles vers le SGSA (Système de Gestion de Sécurité d'Agent). Le rôle de SGSA est de déterminer l'itinéraire et de transformer l'agent à un agent mobile sécurisé par une adaptation statique, en fonction de la politique de sécurité d'agent et les plates-formes. Le créateur d'agent dispose une bibliothèque de composants représentant des comportements élémentaires. On distingue deux types de composants : les composants de niveau de base contenant le code fonctionnel de l'agent et les composants méta-niveau décrivant les services non fonctionnels.

Après la création de l'agent mobile avec l'ensemble des composants fonctionnels et non-fonctionnels nécessaires à l'exécution de l'agent (méta-niveau, niveau de base, interface), le créateur d'agent signe numériquement l'agent afin de l'authentifier durant son adaptation, et assurer l'intégrité de tous ses composants. Une signature numérique sert donc comme moyen de confirmation de l'authenticité de son créateur, de son origine et de son intégrité. Elle possède également la propriété de non-répudiation. Autrement dit, elle permet d'assurer que le créateur a bien envoyé l'agent au SGSA. Pour ce faire, nous optons deux algorithmes SHA1 avec RSA. Ces deux algorithmes sont considérés par la communauté scientifique comme sécuritaires. La figure 4.10 illustre le processus de la signature numérique d'un agent mobile par son créateur.

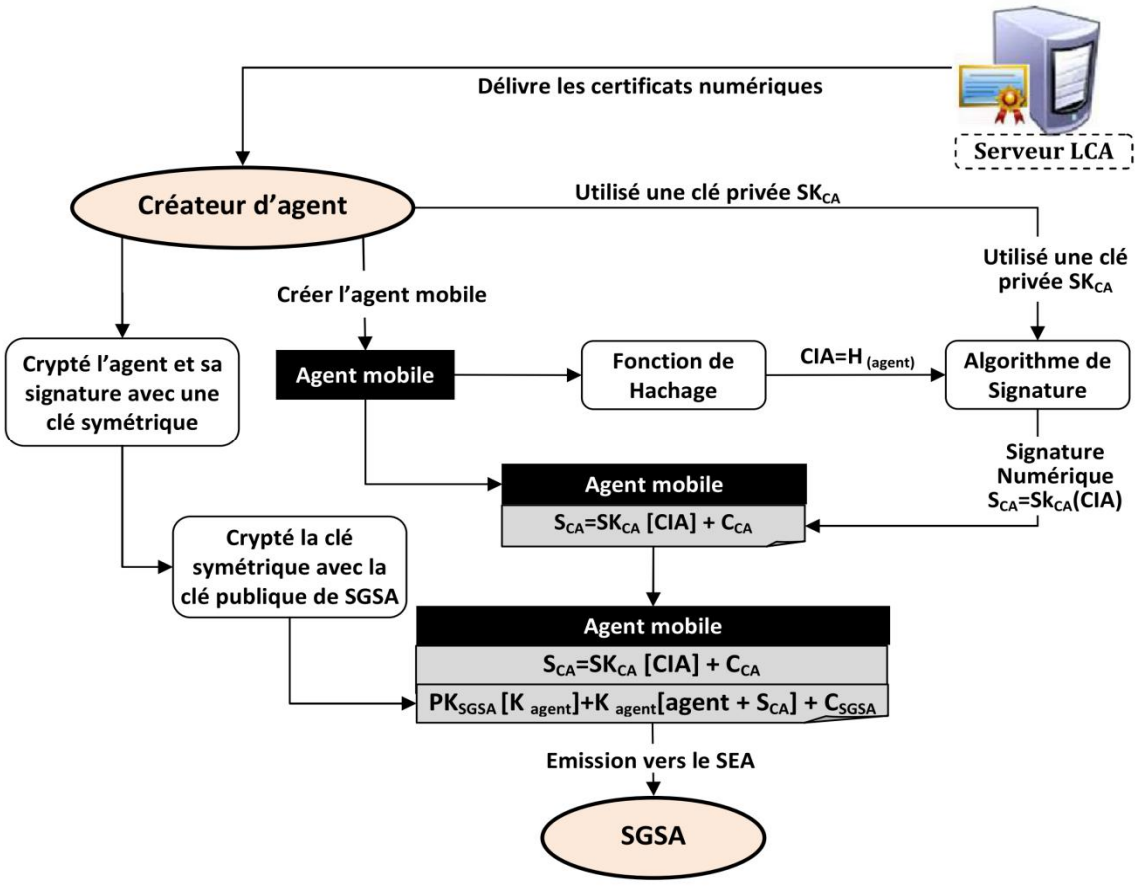

Figure 4. 10. Processus de signature numérique d'un agent mobile par son créateur

Le créateur d'agent calcule le code d'intégrité d'agent (CIA) par une fonction de hachage H.

$$CIA=H_{(agent)}$$

Nous avons intégré l'infrastructure à clé publique PKI (Public Key Infrastructure) dans notre approche, le créateur d'agent possède un certificat à clé publique délivré par l'autorité de certification locale (LCA). Il existe plusieurs formats de certificat. Le format X.509 choisi dans notre approche est le format le plus largement utilisé par la plupart des protocoles. Le créateur utilise leur certificat pour signer le code d'intégrité d'agent (CIA) avec sa clé privé SK_{CA}, puis attacher son identité CA-ID, son certificat à clé publique C_{CA}, et sa signature numérique S_{CA}. L'emballage produit par le propriétaire d'agent s'appelé $PKCS7_{CA}$ conforme au format standard PKCS7.

$$S_{CA}=SK_{CA}(H_{(agent)}) + C_{CA}$$

- SK_{CA} : clé privée du créateur d'agent.
- C_{CA} : certificat X.509 du créateur d'agent.
- $H_{(agent)}$: fonction de hachage pour calculer le code d'intégrité d'agent (CIA).
- S_{CA} : signature numérique de créateur d'agent.

La signature de l'agent par son créateur permet uniquement de garantir l'authenticité et l'intégrité de l'agent mobile. Si la signature est valide, on peut être certain de la provenance du code et de son intégrité (l'agent n'aura pas subi de modification). Enfin, pour assurer aussi la confidentialité des différents composants de l'agent entre le créateur et le SGSA, le créateur d'agent utilise un algorithme de cryptage symétrique (AES, taille de clé 256 bits) pour crypter tous les éléments de l'agent. La clé symétrique est enveloppée ensuite par la clé publique de l'entité SGSA. Notons que la cryptographie à clé symétrique joue un rôle important dans la mise en œuvre des systèmes à clé publique. Cela est dû au fait que les algorithmes de cryptage à clé asymétrique sont plus lents que ceux à clé symétrique. Après le cryptage d'agent par la clé K_{agent}, le créateur d'agent sauvegarde l'identité d'agent AM-ID et la clé K_{agent} pour l'utiliser ultérieurement.

$$\text{Agent-crypté} = PK_{SGSA}[K_{agent}] + K_{agent}[\text{agent} + S_{CA}] + C_{SGSA}$$

- PK_{SGSA} : la clé publique de l'entité SGSA.
- K_{agent} : La clé symétrique utilisée pour crypter l'agent.

Le créateur d'agent envoie l'agent après sa création vers le SGSA pour le transformer en un agent mobile sécurisé.

4.5.2. Réception el validation d'un agent mobile par son créateur

Après l'adaptation de l'agent mobile par le système SGSA (l'adaptation de l'agent mobile est expliquée dans le chapitre 5). Le créateur d'agent reçoit l'agent mobile crypté par la clé symétrique (K_{agent}) et signé par la clé privée de l'entité SGSA. Le créateur décrypte la K_{agent} par sa clé privée, puis vérifie que la clé reçue par l'agent correspond à la clé mise en cache avant l'émission de l'agent vers le SGSA. S'il y a une correspondance entre les deux clés, le créateur assure l'authenticité de leur agent. Ensuite, Le créateur décrypte l'agent avec la clé symétrique K_{agent}, et vérifie la signature de SGSA pour garantir l'intégrité de l'agent après l'adaptation et l'authenticité de l'entité qui adapte l'agent. La figure 4.11 montre l'interaction entre le créateur d'agent et le SGSA.

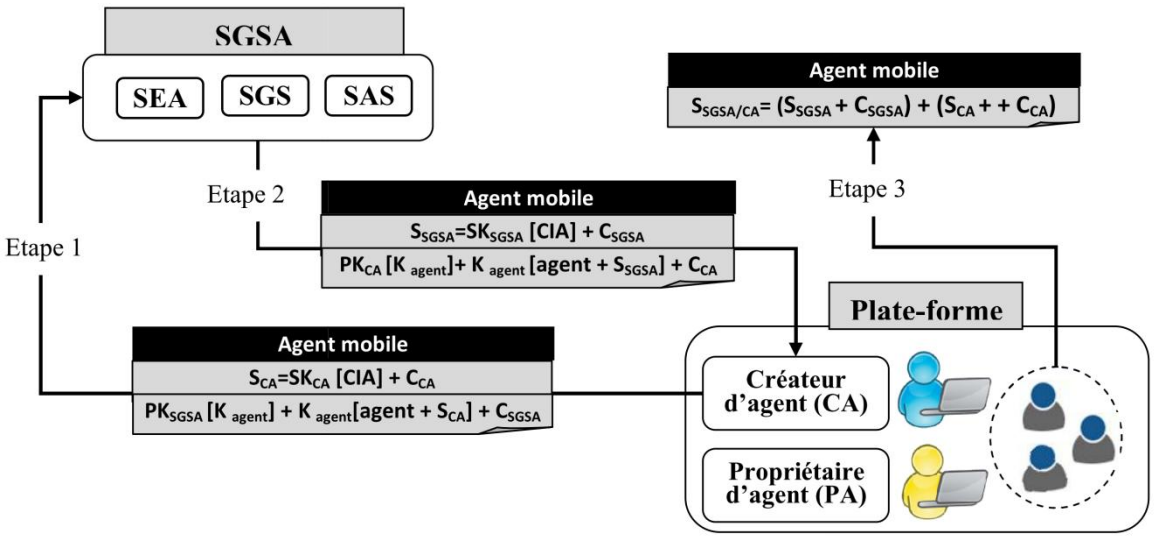

Figure 4. 11. Interaction entre le créateur d'agent et le SGSA

La dernière étape de la phase de création consiste à signer les trois parties (interface, méta-niveau et le niveau de base) par le créateur d'agent. Notons que l'agent dispose aussi une signature de l'entité SGSA. Cette signature permet d'instaurer une relation de confiance mutuelle entre les plates-formes d'exécration et les agents adaptés par cette entité. Tous les agents adaptés par le SGSA et signés par leur créateur sont stockés dans sa plate-forme d'origine. Seulement le propriétaire d'agent peut initialiser et lancer l'agent dans le réseau (l'initialisation et le lancement de l'agent mobile est détaillé dans la section 7.2).

4.6. Publication et découverte des plates-formes d'exécution

La plupart des approches qui traitent le problème de sécurité d'agents mobiles supposent que l'itinéraire de l'agent est connu d'avance, elles traitent seulement la protection des agents mobiles contre les plates-formes malveillantes. L'inconvénient majeur de ces approches est que le créateur d'agent doit avoir une connaissance préalable sur toutes les plates-formes qui offrent des services à leur agent. En outre, les plates-formes fournissant des services aux agents mobiles avec des politiques de sécurité différentes. Notre approche est fondée sur les services Web, qui permettent aux plates-formes de publier leurs services dans un registre standard UDDI [Razouki et al., 2014a].

4.6.1. Architecture interne d'une plate-forme d'exécution

Nous allons présenter dans ce qui suit, notre solution d'intégration d'un registre UDDI dans le système d'agents mobiles, pour faciliter la publication et la découverte des services offerts par les plates-formes. Nous montrons dans la figure 4.12, l'architecture interne d'une plate-forme.

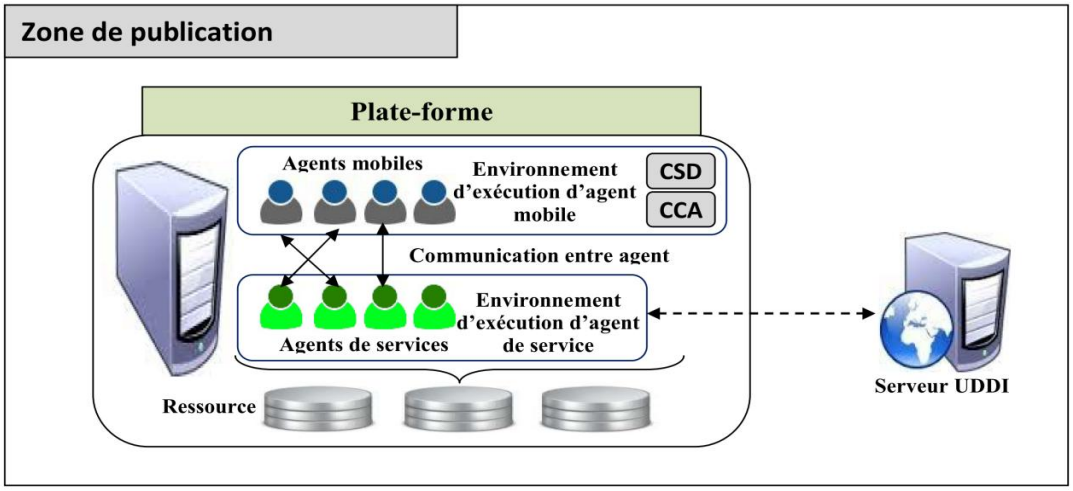

Figure 4. 12. Architecture interne d'une plate-forme d'exécution

Une plate-forme contient deux environnements d'exécutions d'agents. Le premier est un environnement d'exécution d'agents mobiles qui offrent les fonctionnalités de base pour exécuter les agents mobiles. Il assure l'accueil des agents visiteurs, la communication distante entre les agents, l'interaction avec les agents de service local, l'accès aux ressources et la migration vers d'autres plates-formes. Le deuxième est un environnement d'exécution d'agents de services qui offrent les différents services fournis par la plate-forme. Chaque service est représenté par un agent de service. Notre plate-forme contient aussi deux composants essentiels :

Composant de Stockage de Données (CSD) : permet de préparer, structurer et sauvegarder les données dans un format XML. Les données préparées sont ajoutées à l'agent mobile en cours d'exécution. Les agents sont équipés par une description du niveau de base qui aide le CSD de construire et d'ajouter les données à l'agent. Le composant CSD est basé sur plusieurs protocoles et spécifications de sécurité pour protéger ses données. Par exemple, la spécification XML-Signature est utilisée par le CSD pour signer les données ajoutées à l'agent et vérifier la signature des données signées par d'autres plates-formes. La spécification XML-Encryption permet de crypter/décrypter les données de l'agent mobile. Le CSD peut utiliser les deux spécifications en même temps pour signer et crypter les parties bien définies de données d'agent.

Composant de Contrôle d'Accès (CCA) : permet d'envoyer les demandes des clés symétriques au KDC (Key Distribution Center) pour crypter/décrypter les données d'agent mobile. Ce dernier traite la demande, identifie l'émetteur et décide s'il accepte ou non d'envoyer la clé symétrique. Les demandes générées par le CCA contiennent un ensemble d'attributs sur la plate-forme, l'agent, l'étiquette de données et l'action à réaliser. Ces attributs sont utilisés par le KDC pour accepter ou refuser la demande.

Notons que toutes les plates-formes possèdent des certificats numériques délivrés par l'autorité de certification locale (LCA). Ces certificats sont utilisés pour authentifier les plates-formes et sécuriser la communication entre les différentes entités.

4.6.2. Modélisation des agents de services

Un agent de service est un agent stationnaire capable de raisonner sur les services qu'ils offrent. Il coopère avec d'autres agents afin de satisfaire le service demandé par un agent mobile (visiteur). Un service peut être accompli par un seul ou par la coopération de plusieurs agents de service dans la même plate-forme. Un agent de service est encapsulé dans un service déployable et publiable par le registre UDDI. La figure 4.13 montre qu'un agent de service se compose de trois principales couches :

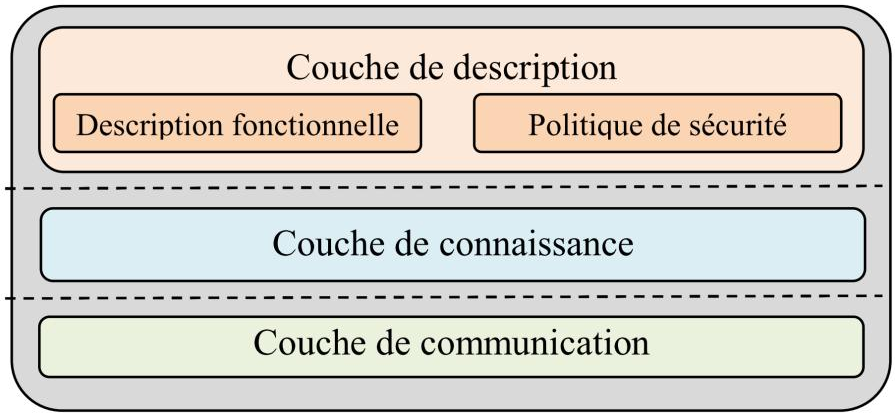

Figure 4. 13. Structure d'un agent de service

- Couche de description est une couche qui permet de décrire la fonctionnalité et la politique de sécurité pour chaque agent de service. On distingue deux catégories de description. La première est une description fonctionnelle qui permet de décrire la fonctionnalité et le service fournis par la plate-forme aux agents mobiles. La deuxième est une politique de sécurité définie par le créateur du service. Cette politique est basée sur les concepts définis dans l'ontologie de sécurité MASO. Elle exprime les exigences de sécurité pour utiliser le service et les capacités de sécurité offertes par ce service. Les services offerts par les agents de services sont publiés dans le registre UDDI.
- Couche de connaissance contient le code fonctionnel de l'agent développé par le créateur du service et inclut un ensemble de données et d'actions à exécuter.

- Couche de communication assure la communication entre l'agent de service et les agents mobiles visiteurs. Ainsi l'interaction avec les autres agents de services pour réaliser des tâches complexes. La communication constitue l'un des moyens fondamentaux pour assurer la répartition des tâches et la coordination des actions.

Pour faciliter la publication et la découverte des services offerts par les plates-formes, nous avons adapté le registre UDDI à notre approche. En effet, chaque service offert par une plate-forme est représenté par un agent de service. Ensuite, la description fonctionnelle et la politique de sécurité de cet agent sont publiées dans le registre UDDI que nous avons adapté.

4.6.3. Nouveau registre UDDI pour le système d'agents mobiles

Nous proposons d'utiliser un nouveau registre UDDI qui diffère du registre standard par son contenu. L'extension que nous avons proposée permet de publier les informations sur les entreprises, les plates-formes hébergées par l'entreprise et les services offerts de façon structurée. Ces informations peuvent donc être recherchées par catégorie.

4.6.3.1. Extension du registre UDDI

Afin de réaliser l'extension proposée du registre UDDI, nous proposons d'ajouter à sa structure actuelle des informations concernant la sécurité des services publiés dans le registre. Dans ce cas, ces informations sont présentées par l'entité *SecurityPolicy* qui décrit la politique de sécurité d'un service particulier. Cette entité est ajoutée au niveau de *BusinessService* (figure 4.14).

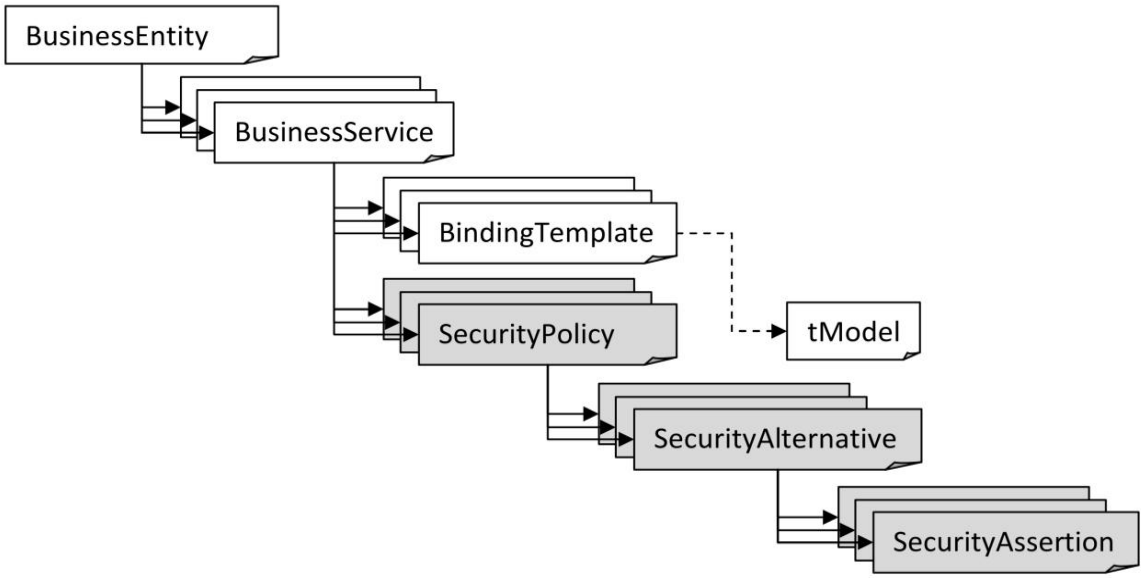

Figure 4. 14. Structure d'un enregistrement dans le registre UDDI.

Le modèle proposé comporte quatre types principaux de structures de données XML qui aident à la localisation et la compréhension rapide des informations qui constituent un enregistrement UDDI :

- *BusinessEntity* est une structure de donnée de haut niveau qui donne les informations sur l'organisation ou l'entreprise offrant le service.
- *BusinessService* ce sont en quelque sorte les pages jaunes d'un annuaire UDDI, qui décrit de manière non technique les services proposés par les différentes plates-formes. Cette structure donne les informations descriptives (nom, description textuelle, classification ...) sur le service offert.
- *BindingTemplate* définit les coordonnées des services, les informations requises pour localiser le service ainsi la plate-forme qui héberge ce service.
- *SecurityPolicy* donne la politique de sécurité de service. Cette politique est utilisée par le Système de Gestion des Services (SGS) pour sélectionner les services qui satisfaits les besoins de sécurité de l'agent mobile.

La structure nouvelle du registre UDDI est définie dans un schéma XML qui étend le schéma XML d'origine proposé par la spécification UDDI. L'entité *SecurityPolicy* a été ajoutée dans la structure d'enregistrement, ainsi dans les requêtes de recherche et de publication de services dans ce registre.

4.6.3.2. Publication et recherche des services

La plate-forme envoie au nouveau registre UDDI une requête de publication qui contient la description fonctionnelle et la politique de sécurité associées à un agent de service. La figure 4.15 montre un exemple de requête de publication sur ce registre qui enregistre un service en ajoutant les informations de sécurité concernant l'utilisation de ce service. Cette requête enregistre le service « *Réservation* ». Elle précise le nom et une description de ce service. Cette requête enregistre aussi la politique de sécurité définie par le créateur du service. À titre d'exemple, l'utilisation de ce service exige que les agents mobiles visiteurs utilisent les certificats X.509 comme un mécanisme d'authentification, ainsi le service dispose un protocole XML-Encryption comme une capacité de sécurité pour crypter/décrypter les données dynamique d'un agent mobile.

```
<save_service xmlns="urn:uddi-org:api_v2" generic="2.0">
    <authInfo>88727f7e88727f7</authInfo>
    <businessService businessKey="f58ad65d-53f5-8ad6-c196-edeb4" serviceKey="">
        <name xml:lang="fr">réservation</name>
        <description xml:lang="fr">
            Un service de réservation d'hôtel assurer par un agent de service
        </description>
        <wsp:SecurityPolicy
            xmlns:wsp="http://www.w3.org/ns/ws-policy"
            xmlns:MASO="http://namespace/MobileAgentSecurity">
            <MASO:SecurityAlternative>
                <MASO:hasType RDF:resource="&MASO;#Requirement"/>
                <MASO:hasObject RDF:resource="&MASO;#Authentication"/>
                <MASO:SecurityAssertion>
                    <MASO:ensuredByToken RDF:resource="&MASO;#X509Certificate"/>
                </MASO:SecurityAssertion>
            </MASO:SecurityAlternative>
            <MASO:SecurityAlternative>
                <MASO:hasType RDF:resource="&MASO;#Capability"/>
                <MASO:hasObject RDF:resource="&MASO;#Confidentiality"/>
                <MASO:SecurityAssertion>
                    <MASO:ensuredByProtocol RDF:resource="&MASO;#XML-Encryption"/>
                </MASO:SecurityAssertion>
            </MASO:SecurityAlternative>
        </wsp:SecurityPolicy>
    </businessService>
</save_service>
```

Figure 4. 15. Politique de sécurité d'un service (requête de publication)

La découverte de la plate-forme qui offre un service consiste à chercher et trouver dans le registre UDDI le service qui répond à l'ensemble de caractéristiques souhaitées pour exécuter l'agent mobile. Nous donnons l'exemple suivant pour expliquer l'utilisation du registre UDDI. Au moment de l'adaptation de l'agent mobile, le Système de Gestion de Service (SGS) est utilisé pour créer l'itinéraire de l'agent mobile (liste des plates-formes qui offrent des services à l'agent mobile). Dans ces cas, le SGS envoie une requête à l'UDDI constituée par un ensemble d'attributs sur le service demandé par l'agent (par exemple le nom de service réservation). Cette requête est ensuite traitée et découpée puis comparée avec les attributs stockés dans le registre UDDI. Un ensemble de descriptions des services est ensuite donné comme résultat de recherche. Chaque description contient l'adresse de la plate-forme, l'identité de service (agent de service) et la politique de sécurité associée à ce service. Le SGS utilise les politiques de sécurité service/agent pour déterminer le degré de correspondance entre ces politiques (plus de détaille dans la section 5.3.2).

4.7. Conclusion

Dans la première section, nous avons présenté un modèle d'agent à base des composants. Ce modèle permet au concepteur d'agent de spécifier à travers des concepts de haut-niveau (composants, politique de sécurité, la mobilité, la communication, etc.) quand et comment adapter la sécurité de l'agent mobile à l'environnement d'exécution.

Notre modèle se compose de quatre parties principales : l'interface servant l'interaction avec le monde extérieur. Le niveau de base qui correspond à la partie applicative de l'agent, c'est-à-dire l'ensemble de composants implémentant le comportement de l'agent. Les données statiques et dynamiques de l'agent. Le méta niveau qui contrôle le niveau de base. Le méta niveau intègre notamment le gestionnaire de communication, la politique d'assemblage pour gérer le processus d'assemblage des composants fonctionnels, une description du niveau de base identifie l'activité des composants, la mobilité d'agent qui contrôle la liste des plates-formes visitées, et enfin la politique de sécurité de l'agent. La distinction entre le méta-niveau et le niveau de base permet une séparation explicite entre le comportement fonctionnel et non fonctionnel de l'agent. Cette modularité simplifie à la fois la conception, l'adaptation et la sécurisation des agents mobiles.

Dans la deuxième section, nous avons modélisé les agents de services pour représenter les services offerts par les plates-formes d'exécution. Ensuite, nous avons proposé une extension du registre UDDI pour la publication et la découverte de ces services. Notre registre permet de publier deux types de descriptions de service : la première est une description fonctionnelle sur les services offerts par la plate-forme, la deuxième est une politique de sécurité basée sur l'ontologie MASO.

Nous avons vu comment intégrer la politique de sécurité dans la structure interne de l'agent mobile, ce qui permet aux créateurs d'agents de spécifier leurs exigences/capacités de sécurité. Aussi, comment associer les politiques de sécurité aux services offerts par les plates-formes. Ces politiques sont utilisées plus tard pour sélectionner les plates-formes les plus adéquates à l'exécution sécurité de l'agent mobile.

Le chapitre suivant montre de façon détaillée comment sélectionner ces plates-formes par le SGS pour créer l'itinéraire de l'agent. Ensuite, nous présentons comment le SAS transforme l'agent à un agent mobile sécurisé, puis estime le niveau de sécurité de chaque composant et le degré de confiance des plates-formes visitées pour créer les politiques de contrôle d'accès aux données d'agent.

CHAPITRE 5

Evaluation, Validation et Adaptation des agents mobiles par le SGSA

5.1. Introduction

L'étude effectuée sur les approches de protection des agents mobiles montre que toutes les approches utilisent le même mécanisme de sécurité pour protéger l'agent mobile contre les différentes plates-formes malveillantes. Ces approches ne prennent pas en considération ni les besoins spécifiques de sécurité de chaque agent (la sensibilité de ses tâches) ni le changement des politiques de sécurité entre les plates-formes d'exécution. Dans notre travail, le modèle d'agent que nous avons proposé contient plusieurs composants fonctionnels. Ces composants disposent des stratégies de sécurité différentes (la sensibilité des composants) regroupées sous forme d'une politique de sécurité. De même manière, les plates-formes disposent des politiques de sécurité pour utiliser ses services. Au moment de l'adaptation de l'agent mobile, le SGSA sélectionne les plates-formes qui répondent mieux aux exigences de sécurité de l'agent mobile (la compatibilité entre les politiques de sécurité), puis sécurisé l'agent en fonction de sa politique de sécurité.

L'objectif de cette phase est de transformer l'agent à un agent mobile sécurisé. Cette adaptation repose sur trois systèmes de confiance. Le premier est le SEA lui permet d'évaluer et de valider : l'identité, l'intégrité et l'assemblage des composants fonctionnels de l'agent. Le deuxième est le SGS son rôle est d'analyser la correspondance sémantique entre les politiques de sécurité (plate-forme/agent) pour créer l'itinéraire de l'agent mobile. Le troisième est le SAS joue un rôle important dans le processus d'adaptation, il estime le niveau de sécurité des composants de l'agent et le degré de confiance des plates-formes d'exécution. Ensuite, il sécurise les composants sensibles et définie les règles de contrôle d'accès aux données d'agent.

Dans la section suivante, nous présentons l'architecture générale et les fonctionnalités des différentes entités de notre système d'adaptation. Ensuite nous proposons d'utiliser l'authentification mutuelle pour sécuriser la communication et

l'interaction entre ces différentes entités. Après, nous détaillons le processus d'adaptation de l'agent mobile.

5.2. Architecture générale du système d'adaptation

Dans notre infrastructure, l'objectif de l'interaction entre les différentes entités du système est de transformer l'agent à un agent mobile sécurisé [Razouki et al., 2014b, 2014c]. En effet, le créateur d'agents fait confiance au SGSA pour adapter ses agents, et les plates-formes ne faites confiance qu'aux agents adaptés et signés par le SGSA. L'adaptation de l'agent est liée à deux politiques de sécurité (agent mobile et plate-forme d'exécution).

Dans notre démarche de sécurisation, il existe huit étapes principales pour adapter la sécurité d'un agent mobile :

1. Évaluer et valider les différents composants de l'agent.
2. Analyser le rôle et les besoins fonctionnels de l'agent.
3. Découvrir les plates-formes et analyser la correspondance sémantique entre les politiques de sécurité.
4. Sélectionner les plates-formes et créer l'itinéraire de l'agent mobile.
5. Estimer les niveaux de sécurité des composants fonctionnels et les degrés de confiance des plates-formes d'exécutions.
6. Sécuriser les composants sensibles de l'agent mobile.
7. Définir les règles de contrôle d'accès aux données d'agent.
8. Le SGSA signe l'agent pour assurer l'intégrité et mettre en place une relation de confiance entre l'agent et les plates-formes d'exécution.

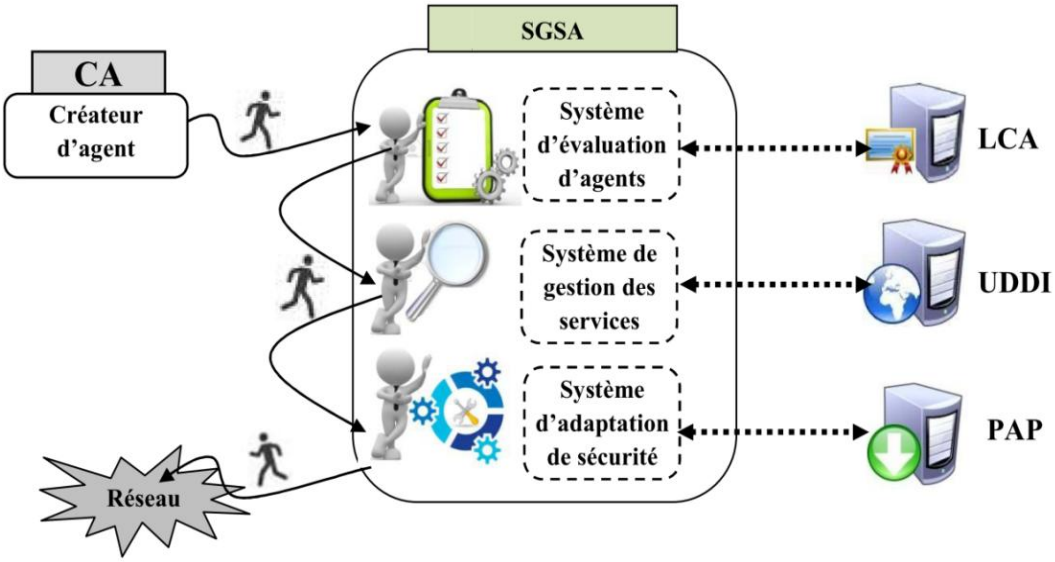

Figure 5. 1. Architecture de la phase d'adaptation

La figure 5.1 montre l'architecture interne de notre système d'adaptation, il contient trois sous-systèmes principaux : le système d'évaluation d'agents, le système de gestion des services et le système d'adaptation de sécurité. Nous détaillons la fonctionnalité de chaque sous-système dans la section 5.3.

5.3. Authentification mutuelle entre les entités

Dans notre infrastructure de sécurité, toutes les interactions entre les différentes entités (SES, SGS, SAS, KDC, PDP…) sont protégées par l'utilisation d'un protocole d'authentification forte basée sur la norme FIPS 196 [FIPS, 1996]. Pour expliquer le processus d'authentification dans notre approche, nous prenons l'exemple d'interaction entre les entités SAS et PAP (voir la section 6.3.1). La figure 5.2 montre les étapes d'authentification entre le SAS et PAP. Nous avons utilisé l'Autorité de Certification Locale (LCA) pour vérifier la validité des certificats X.509 délivrés aux différentes entités.

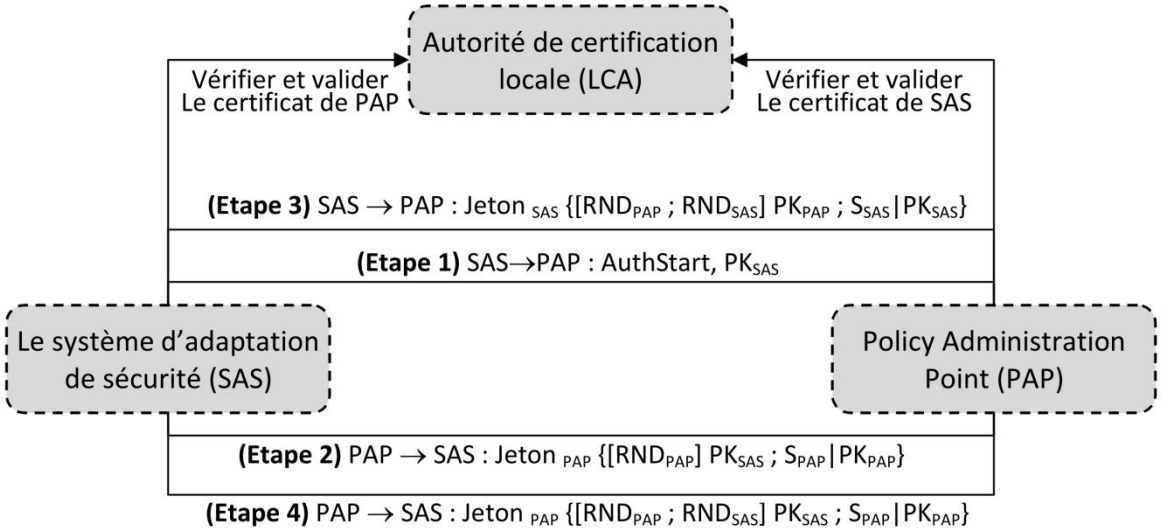

Figure 5. 2. Protocole d'authentification mutuelle

Dans l'approche proposée, nous avons utilisé les certificats X509 comme un mécanisme d'authentification pour protéger la communication entre les différentes entités du système. Pour cela, le SAS doit vérifier la validité du certificat présenté par le PAP et inversement le PAP doit vérifier que le certificat du SAS est validé et que les autres paramètres sont corrects. L'authentification mutuelle entre deux entités avec des certificats X.509 comprend les étapes suivantes :

Étape 1 : L'entité (SAS) initie le processus d'authentification par le message AutStart, le message contient le certificat à clé publique de SAS (PK_{SAS}).

$$SAS \rightarrow PAP : AuthStart, PK_{SAS}$$

Étape 2 : Le serveur PAP doit décider si elle va continuer ou arrêter le processus d'authentification. Le serveur vérifie si le certificat présenté par SAS est stocké dans sa liste d'entités autorisées de publier les politiques de contrôle d'accès. Il vérifie ensuite la validité de PK_{SAS} en contactant l'autorité de certification (LCA). Si les deux vérifications sont réussies, le PAP crée et envoie un jeton d'authentification contenant un nombre aléatoire RND_{PAP} crypté avec la clé publique de SAS. Ce nombre est stocké par le PAP pour l'utiliser ultérieurement.

$$PAP \rightarrow SAS : Jeton_{PAP} \{[RND_{PAP}]\ PK_{SAS} ; S_{PAP} | PK_{PAP}\}$$

Étape 3 : L'entité SAS reçoit le jeton d'authentification, puis vérifie la validité de PK_{PAP} et l'intégrité de jeton d'authentification par S_{PAP}. Le SAS décrypte ensuite le RND_{PAP} avec sa clé privée et génère un nombre aléatoire RND_{SAS}. Enfin, le SAS crypte [RND_{SAS} ; RND_{PAP}] avec la clé publique de PAP puis signe l'ensemble avec sa clé privée. Le RND_{SAS} est stocké aussi par le SAS pour l'utilisation ultérieurement.

$$SAS \rightarrow PAP : Jeton_{SAS} \{[RND_{PAP} ; RND_{SAS}]\ PK_{PAP} ; S_{SAS} | PK_{SAS}\}$$

Étape 4 : le PAP vérifie l'intégrité et l'authenticité de jeton par la signature S_{SAS}, puis décrypte [RND_{PAP} ; RND_{SAS}] avec sa clé privée. Le PAP compare le RND_{PAP} reçu avec le RND_{PAP} mémorisé. Si les deux nombres sont égaux, le PAP crypte l'ensemble [RND_{PAP} ; RND_{SAS}] avec la clé publique de SAS puis signe avec sa clé privée.

$$PAP \rightarrow SAS : Jeton_{PAP} \{[RND_{PAP} ; RND_{SAS}]\ PK_{SAS} ; S_{PAP} | PK_{PAP}\}$$

Étape 5 : de la même façon que l'étape précédente, le SAS vérifie la signature de jeton, puis décrypte [RND_{PAP} ; RND_{SAS}] avec sa clé privée. Le SAS compare le RND_{SAS} reçu avec le RND_{SAS} mémorisé. Si les deux nombres sont égaux, le processus d'authentification est terminé avec succès.

5.4. Fonctionnalités des différentes entités

5.4.1. Système d'évaluation d'agents (SEA)

Il est important d'évaluer et de valider l'identité, l'intégrité, l'assemblage et la spécification des différents composants de l'agent mobile par une entité de confiance indépendante. Dans notre méthodologie, cette fonction est assurée par le Système d'Evaluation d'Agent (SEA) [Razouki et al., 2013a, 2013b, 2013c]. En effet, la figure 5.3 montre que le créateur d'agent envoie l'agent après sa création vers le SGSA, le SEA reçoit et décrypte la clé symétrique K_{agent} avec la clé privée SK_{SGSA}, puis décrypte l'agent avec cette clé. Ensuite, le SEA utilise la signature S_{CA} du

créateur d'agent pour vérifier l'authenticité et l'intégrité de l'agent. En effet, le SEA décrypte le code d'intégrité d'agent CIA1 avec la clé publique du créateur PK_{CA}. Pour assurer l'intégrité de l'agent, le SEA recalcule le code d'intégrité d'agent CIA2 et le compare avec le CIA1. Si les deux codes sont égaux, alors l'agent n'est pas altéré par des plates-formes malveillantes.

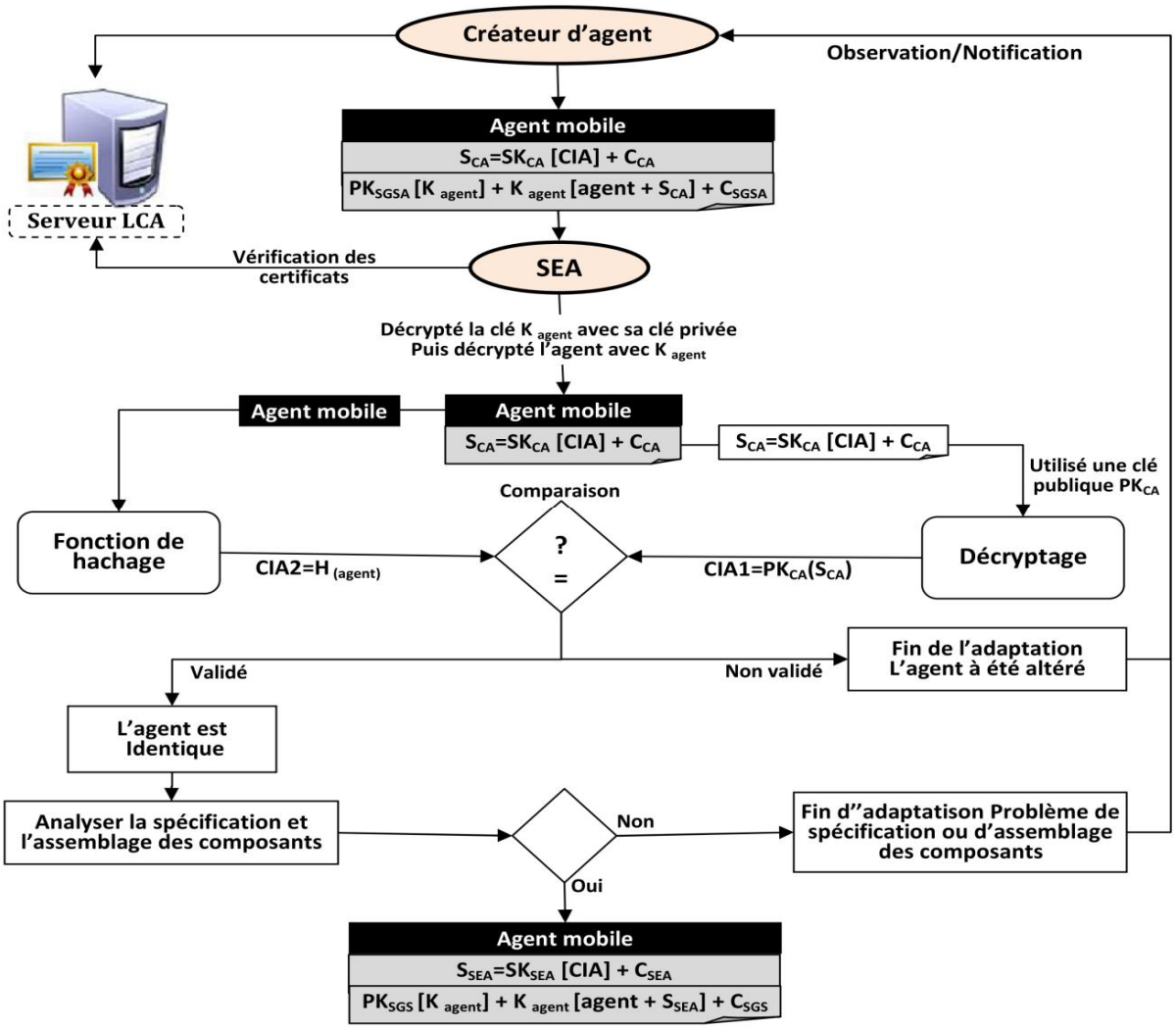

Figure 5. 3. Processus d'évaluation et de validation d'un agent mobile par le SEA

Après l'authentification et la vérification de l'intégrité de l'agent, le SEA analyse la spécification de chaque composant et l'assemblage des différents composants de l'agent. Les composants sont assemblés via leurs interfaces. Une interface fournie par un composant peut-être connectée avec une interface requise d'un autre composant, si la première offre toutes les fonctionnalités permettant d'implanter la seconde. Les composants doivent être connectés de manière appropriée. Afin de garantir cette interopérabilité entre les composants, le créateur d'agent associé une *« description du niveau de base »* et une *« politique d'assemblage »*, ce qui permet de décrire la fonctionnalité, les interfaces et l'assemblage entre les différents composants de l'agent mobile.

Le SEA vérifie que les fonctionnalités et les interfaces des composants sont correctes. Pour valider le bon fonctionnement de l'agent, le SEA vérifie l'assemblage des différents composants, si toutes les vérifications se terminent avec succès. Le SEA utilise leur certificat pour signer l'agent mobile avec sa clé privée SK_{SEA}, Le SEA utilise ensuite la même clé reçue par l'agent K_{agent} pour crypter à nouveau l'agent. Enfin, il utilise la clé publique de SGS pour crypter la clé d'agent. L'emballage produit s'appeler $PKCS7_{SEA}$. Si les vérifications ne sont pas réussies, le SEA arrête cette adaptation et envoie une observation au créateur d'agent explique la cause de l'arrêt.

5.4.2. Système de gestion des services (SGS)

La découverte et la sélection des plates-formes candidates à l'exécution sécurisées des agents mobiles sont deux étapes importantes dans notre démarche de sécurisation. Il y a toujours des confusions autour de la définition de ces deux termes. Dans notre travail, nous considérons la découverte de plate-forme comme étant la localisation des plates-formes qui offrent des services à l'agent mobile (assurent les besoins fonctionnels de l'agent). La sélection des plates-formes correspond à l'évaluation et le classement des plates-formes déjà découverts afin d'identifier ceux qui correspondent le mieux à la politique de sécurité de l'agent.

Nous avons intégré le serveur UDDI pour récupérer la liste des services offerts par les plates-formes d'exécution (la création d'un nouveau registre UDDI est expliquée dans la section 4.6). Dans notre démarche de découverte/sélection, notre registre dispose deux types de descriptions : Une description fonctionnelle représente le nom et la description textuelle des services ainsi qu'une référence à la plate-forme proposant le service. Cette description est utilisée par le SGS pour découvrir les plates-formes qui offrent les services à l'agent mobile. Une description non fonctionnelle représente un ensemble de paramètres de sécurité associés à un service sous forme d'une politique de sécurité. Cette description est utilisée par le SGS pour sélectionner les services les plus appropriés à l'exécution sécurisée de cet agent. Notre solution de découverte et de sélection des plates-formes d'exécution est présentée dans la figure 5.4.

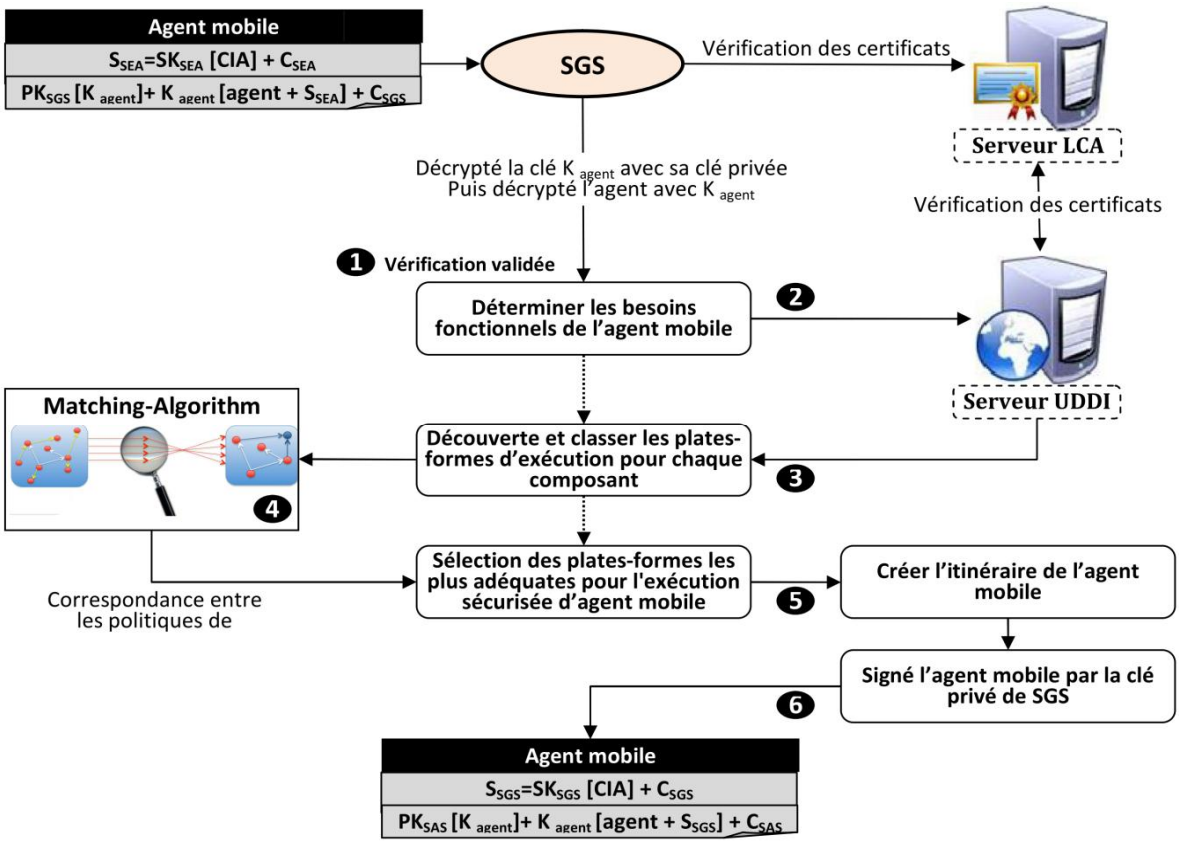

Figure 5. 4. Sélectionner les plates-formes pour créer l'itinéraire d'un agent mobile

Nous donnons une grande importance dans notre processus de découverte et de sélection aux politiques de sécurité. La description fonctionnelle des services ne sera pas traitée en détail dans le processus découverte vu le nombre des travaux qui ont été faits dans ce domaine.

Le SGS reçoit l'agent mobile validé par le SEA, puis il décrypte la clé K_{agent} avec sa clé privée. Ensuite, il utilise la signature S_{SEA} pour garantir l'intégrité et la validité de l'agent par le SEA (étape 1). Si les deux vérifications réussissent, le SGS analyse la description du niveau de base de chaque composant fonctionnel afin d'extraire les différents besoins fonctionnels nécessaires pour réaliser les tâches attendues de l'agent. Ensuite, le SGS interroge le serveur UDDI pour découvrir les plates-formes les plus appropriées avec les fonctionnalités requises (étape 2). Pour cela, le SGS génère et envoie une requête SOAP avec les différents paramètres selon la norme UDDI. Le SGS reçoit la liste des plates-formes candidates à l'aide d'un service web sécurisé (étape 3).

Plusieurs plates-formes peuvent offrir le même service à l'agent mobile, le rôle de SGS est de sélectionner les plates-formes qui répondent le mieux à ses besoins spécifiques de sécurité. Pour cela, le SGS récupère les politiques de sécurité des plates-formes candidates, puis il utilise l'algorithme de correspondance sémantique décrit dans la section 3.4.2 pour évaluer la correspondance entre ces politiques et

la politique de sécurité de l'agent. En effet, cet algorithme doit vérifier que les exigences de sécurité de l'agent sont satisfaites par les capacités des plates-formes et l'inverse, les exigences des plates-formes sont satisfaites par les capacités de l'agent. A la fin de cette étape, l'algorithme retourne comme résultat le degré de correspondance entre ces politiques « Perfect-Match », « General-Match », « Negotiable-Match », « No-Match » (étape 4).

Le SGS sélectionne seulement les plates-formes ayant un résultat « Perfect-Match », « General-Match » pour créer l'itinéraire de l'agent mobile (étape 5). Enfin, le SGS utilise le même mécanisme de sécurité que SEA. Il signe et crypte l'agent, puis envoie l'agent vers le SAS. L'emballage produit s'appeler $PKCS7_{SGS}$ (étape 6).

5.4.3. Système d'adaptation de sécurité (SAS)

Les systèmes à base d'agents mobiles sont caractérisés par un aspect très dynamique. Ceci est dû principalement à la migration des agents mobiles vers plusieurs plates-formes avec des politiques de sécurité différentes. En effet, à la visite d'une nouvelle plate-forme, l'agent doit s'adapter dynamiquement à l'exigence de sécurité de son environnement d'exécution. Le but de SAS est de protéger l'agent mobile via une adaptation statique. Cette adaptation consiste à transformer l'agent à un agent mobile sécurisé. En effet, après la découverte et la sélection des plates-formes les plus appropriées à l'exécution sécurisée de l'agent, le SAS utilise la politique de sécurité de l'agent pour déterminer le niveau de sécurité de chaque composant fonctionnel et le degré de confiance des plates-formes visitées par l'agent. Ensuite, le SAS sécurise les composants sensibles et crée les règles de contrôle d'accès aux données d'agent (figure 5.5).

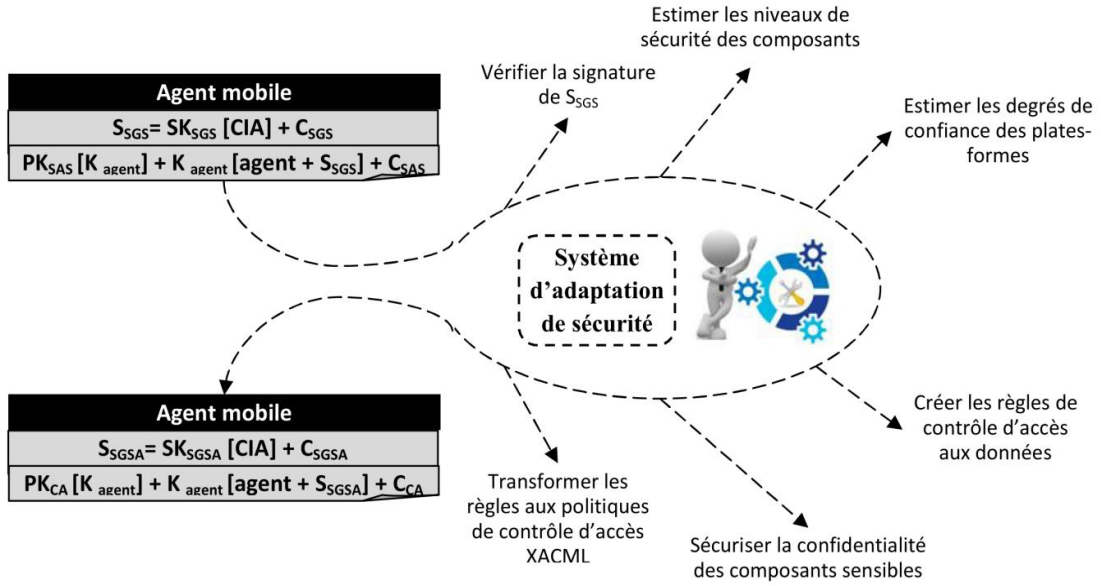

Figure 5. 5. Processus de sécurisation d'un agent mobile par le SAS

Enfin, le SAS signe numériquement les agents adaptés avec le certificat du système C_{SGSA} pour mettre la confiance entre cet agent et les plates-formes qui exécutent l'agent. De cette façon, chaque plate-forme peut vérifier l'identité de l'entité qui a adapté l'agent, ainsi assurer l'intégrité des différents composants de l'agent. Pour assurer la confidentialité de l'agent entre le SGSA et le créateur d'agent, le SAS crypte l'agent avec la clé K_{agent}, puis crypte cette clé avec la clé publique de créateur d'agent PK_{CA}. Enfin, le SAS envoie l'agent vers son créateur.

5.4.3.1. Estimer le niveau de sécurité des composants d'un agent mobile

Nous avons utilisé la politique d'autorisation multi-niveaux pour protéger la confidentialité des données d'agent mobile. Les politiques d'autorisation basées sur une approche multi-niveaux reposent sur une partition de l'ensemble des sujets et de l'ensemble des objets présents dans le système. Un niveau est associé à chaque partition. Les niveaux de sécurité sont généralement partiellement ou totalement ordonnés. Les objectifs de sécurité, qui peuvent être relatifs à la confidentialité ou l'intégrité des objets, sont formulés vis-à-vis de ces différents niveaux, et les règles du schéma d'autorisation qui régissent les contrôles obligatoires prescrits par la politique de sécurité s'appuient également sur ces niveaux.

Nous avons adapté le modèle multi-niveaux de Bell et LaPudula [Lapadula, 1976] afin de créer les politiques de contrôle d'accès aux données d'agents. Ce modèle a été développé dans la société Mitre, dans le cadre de la réalisation d'un système informatique de sécurité pour les forces aériennes américaines du département de défense. L'objectif est d'assurer la confidentialité des données militaires. Ce modèle de sécurité est basé sur la notion de treillis qui formalise les objectifs de sécurité et le schéma d'autorisation de leur politique, sur lequel on peut s'appuyer pour en démontrer la cohérence.

Dans le cadre d'un règlement multi-niveaux, nous avons associé des niveaux de sécurité aux différents composants de l'agent en fonction sa politique de sécurité, et des degrés de confiance aux plates-formes qui exécute l'agent. Nous définissions le niveau de sécurité d'un composant comme suite :

> *«Le Niveau de Sécurité d'un Composant (NSC) est le quadruplet (ES1, ES2, ES3, ES4) d'exigences de sécurité définies par le créateur d'agent sous forme d'une politique de sécurité. Ces exigences sont classées selon quatre niveaux de sécurité ordonnés (top secret, secret, confidentiel, faible). Le NSC reflète la sensibilité de code et les données produites par ce composant dans une plate-forme, à savoir, les dommages potentiels qui pourraient survenir en cas de divulgation non autorisée»*

Le principe notre approche consiste à attribuer une classification à chaque exigence de sécurité (ES). Ces exigences sont obtenues à partir de la politique de sécurité d'agent pour protéger un composant particulier. On distingue quatre niveaux de sécurité hiérarchique ordonnés : top secret (TS=3), secret (S=2), confidentiel (C=1) et faible (F=0). La valeur le plus élevée d'une exigence est considérée comme la plus sécurisée.

Quatre exigences de sécurité sont prises en charge dans le processus d'estimation du niveau de sécurité d'un composant fonctionnel de l'agent mobile : la première indique les services, la fonctionnalité et la structure de données à ajouter par les plates-formes après l'exécution de ce composant. La deuxième permet de spécifier la méthode d'authentification utilisée par les plates-formes pour exécuter ce composant (mot de passe, certificat, ticket). La troisième permet préciser la méthode et le type de cryptage utilisé pour assurer la confidentialité de ce composant. La quatrième permet de spécifier le mécanisme de sécurité pour protéger les données produites par ce composant dans une plate-forme d'exécution. Le tableau 5.1 ci-dessous donne la classification des exigences de sécurité pour estimer le niveau de sécurité d'un composant.

Formellement, nous présentons le niveau de sécurité d'un composant comme l'ensemble des exigences de sécurité décrits dans le tableau 5.1. Ces exigences sont classées selon la politique de sécurité définie par le créateur d'agent. On suppose un agent mobile en phase d'adaptation contient n composants fonctionnels. Le quadruplet suivant permet d'estimer le niveau de sécurité des composants :

$$\forall\, i \in n \qquad NSC_i = (ES1_i, ES2_i, ES3_i, ES4_i)$$

- NSC_i : représente le niveau de sécurité d'un composant i.
- $ES1_i$, $ES2_i$, $ES3_i$ et $ES4_i$: représentent la classification des exigences de sécurité d'un composant i.

Après l'estimation des niveaux de sécurité des composants par le SAS, l'étape suivante consiste à estimer le degré de confiance des plates-formes visitées par l'agent.

	La classification des exigences de sécurité			
	Faible F=0	Confidentiel C=1	Secret S=2	Top Secret TS=3
ES1 : Exigence 1	Le créateur d'agent ne spécifie aucune description du niveau de base pour ce composant.	La description du niveau de base spécifie par le créateur d'agent décrit seulement les services fournis et requis par ce composant.	La description contient aussi des informations concernent la fonctionnalité de ce composant.	La description décrit aussi la structure de données à ajouter dans l'agent après l'exécution de ce composant par des plates-formes.
ES2 : Exigence 2	La politique de sécurité de l'agent n'exige aucune authentification pour exécuter le composant.	La politique de sécurité de l'agent exige une authentification simple pour authentifier les plates-formes d'exécution (Nom d'utilisateur et mot de passe)	La politique de l'agent exige une authentification forte avec un certificat numérique.	La politique de l'agent exige une authentification forte avec un certificat numérique délivré ou validé par l'autorité de certification locale (LCA)
ES3 : Exigence 3	La politique de sécurité n'exige aucun cryptage pour le composant.	La politique de sécurité de l'agent exige le cryptage symétrique de ce composant (3DES, AES). La clé est partagée entre les plates-formes qui exécutent le composant.	Le composant est crypté avec une clé symétrique (3DES, AES). La clé symétrique est enveloppée avec la clé publique des plates-formes qui exécutent le composant.	Le composant est crypté avec la clé publique de la plate-forme qui exécute le composant, dans ce cas, une seule plate-forme peut décrypter et exécuter ce composant.
ES4 : Exigence 4	La politique de sécurité n'exige aucun cryptage de données collectées par le composant.	La politique de sécurité exige le cryptage symétrique de données collectées par le composant. La clé symétrique est partagée entre les plates-formes.	La politique de sécurité exige le cryptage symétrique, puis la signature des données collectées par le composant.	La politique de sécurité exige le cryptage de données collectées avec la clé publique de propriétaire d'agent, seulement le propriétaire d'agent peut décrypter ces données avec sa clé privée.

Tableau 5. 1. Classification des exigences de sécurité

5.4.3.2. Estimer le degré de confiance des plates-formes d'exécution

Notre agent mobile se compose d'un ensemble de composants fonctionnels et une politique de sécurité à suivre durant son exécution. Chaque composant s'exécute dans une ou plusieurs plates-formes en fonction de l'itinéraire d'agent. Pour gérer le contrôle d'accès aux données d'agent entre les plates-formes, le SAS défini un ensemble de règles de contrôle d'accès basées sur le niveau de sécurité des composants fonctionnels et le degré de confiance des plates-formes visitées. Nous définissions le degré de confiance d'une plate-forme comme suite :

«Le Degré de Confiance d'une Plate-forme (DCP)) est le quadruplet (DH1, DH2, DH3, DH4) de Degrés d'Habilitation (DHi) estimés à partir d'un ou plusieurs composants exécutés par cette plate-forme. Chaque degré d'habilitation (DHi) prend la valeur maximale de la classification des exigences de sécurité des composants exécutés par cette plate-forme».

Pour maintenir la confidentialité de données collectées à partir des plates-formes d'exécution. On considère qu'un agent contient *n* composants (*C)* et exécuté par *m* plates-formes *(P)*. Le SAS analyse la politique de sécurité de l'agent pour estimer le niveau de sécurité de chaque composant (NSC). Ensuite, il estime le degré de confiance de chaque plate-forme à partir des composants exécutés par cette plate-forme.

$\forall$ j $\in$ [1, m] $\exists$ $A_j \subset [1, n]$ / $\forall i \in A_j$ le composant C_i s'exécute dans P_j

$$NSC_i = (ES1_i, ES2_i, ES3_i, ES4_i)$$

$$DH1_j = \max_{i \in A_j}(ES1_i) \; ; DH2_j = \max_{i \in A_j}(ES2_i)$$

$$DH3_j = \max_{i \in A_j}(ES3_i) \; ; DH4_j = \max_{i \in A_j}(ES4_i)$$

$$DCP_j = (DH1_j, DH2_j, DH3_j, DH4_j)$$

À la fin de cette étape, le SAS estime le degré de confiance des différentes plates-formes d'exécution, puis il utilise le modèle multi-niveau pour créer les règles de contrôle d'accès aux données d'agent. En effet, les niveaux de sécurité et les degrés de confiance constituent un treillis partiellement ordonné par une relation de dominance notée " ≤ ". La protection de la confidentialité des données s'exprime par le fait qu'une plate-forme P à un degré de confiance DCP. Cette plate-forme ne peut pas accéder aux données collectées par le composant C dans une autre plate-forme, que si le degré de confiance de la plate-forme (DCP) domine le niveau de sécurité de ce composant (NSC). Trois contraintes pour créer les règles de contrôle d'accès à ses données :

Contrainte 1 : Si une plate-forme P_j est autorisée d'exécuter un composant C_i alors la plate-forme P_j a le droit d'ajouter des données cryptées à l'agent après l'exécution de ce composant.

Contrainte 2 : Une plate-forme P_j veut lire les données collectées par un composant C_k que si les deux conditions suivantes sont vérifiées :

- La somme de toutes les exigences de sécurité du composant C_k doit être inférieure ou égale à la somme de tous les degrés d'habilitation de la plate-forme P_j.

$$\sum_{i=1}^{4} \text{ESi}\,(C_k) \leq \sum_{i=1}^{4} \text{DHi}\,(P_j)$$

- L'exigence de sécurité concerne la confidentialité des données produite par le composant C_k ne doit pas être classé dans la catégorie « Top secret = 3 ».

$$ES_4\,(C_k) < 3$$

Contrainte 3 : Une plate-forme P_j veut modifier les données collectées par un composant C_k que si les trois conditions suivantes sont vérifiées :

- La somme de toutes les exigences de sécurité du composant C_k doit être strictement inférieure à la somme de tous les degrés d'habilitation de la plate-forme P_j.

$$\sum_{i=1}^{4} \text{ESi}\,(C_k) < \sum_{i=1}^{4} \text{DHi}\,(P_j)$$

- Le degré de confiance de la plate-forme P_j domine le niveau de sécurité du composant C_k (Chaque exigence de sécurité ESi du composant doit être inférieure ou égale au degré d'habilitation de la plate-forme qui correspond).

$$\forall\, i \in \{1,2,3,4\}\ \text{ESi}\,(C_k) \leq \text{DHi}\,(P_j)$$

- L'exigence de sécurité concerne la confidentialité des données produite par le composant C_k doit être classé dans la catégorie « faible = 0» ou « confidentiel = 1 ».

$$ES_4\,(C_k) < 2$$

Pour illustrer notre approche, nous présentons un exemple qui permet d'expliquer les démarches de création des règles de contrôle d'accès entre les plates-formes d'exécution. On suppose dans cet exemple que l'agent mobile dispose dix composants exécutés par quatre plates-formes :

- P1 exécute les composants (C1, C3).
- P2 exécute les composants (C2, C4, C5, C9).
- P3 exécute les composants (C6, C7, C8).
- P4 composants (C6, C10).

Le SAS estime les niveaux de sécurité des composants fonctionnels en fonction de la politique de sécurité définie par le créateur d'agent. Ensuite, il estime les degrés de confiance des différentes plates-formes qui exécutent l'agent. La figure 5.6 présente le treillis construit par le SAS.

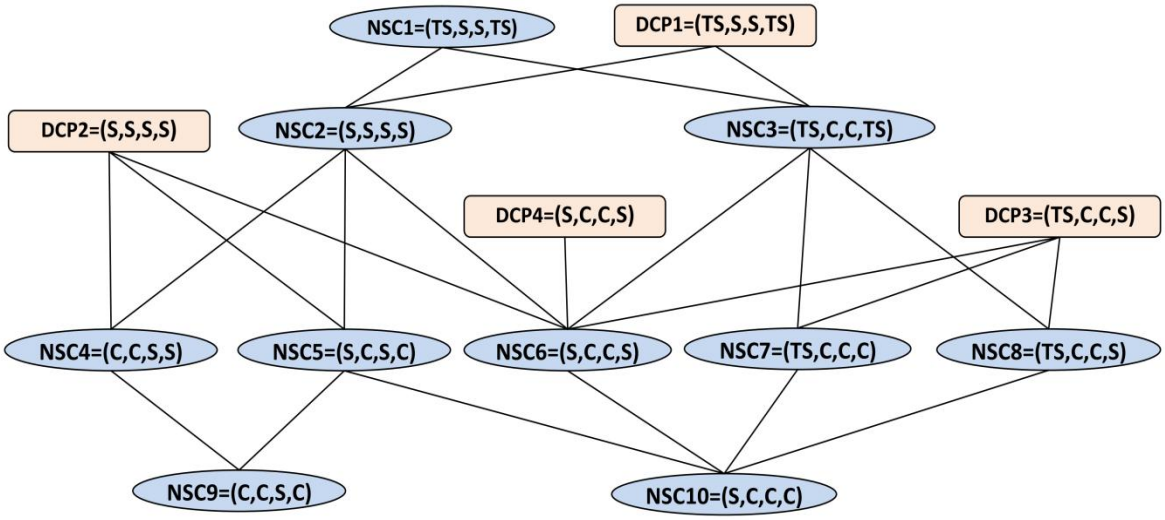

Figure 5. 6. Treillis construit par le SAS

Le SAS utilise le treillis pour classer les niveaux de sécurité des composants (NSC) et les degrés de confiance des plates-formes (DCP). Ensuite, il applique les trois contraintes pour définir les règles de contrôle d'accès. Le tableau 5.2 présente les droits d'accès aux données d'agent entre les plates-formes d'exécution.

	C1	C2	C3	C4	C5	C6	C7	C8	C9	C10
P1	C	R	C	R	R/W	R	R/W	R	R/W	R/W
P2	-	C	-	C/R	C/R/W	R	R	R	C/R/W	R/W
P3	-	-	-	R	R	C/R	C/R/W	C/R	R	R/W
P4	-	-	-	R	R	R	C/R	R	R	C/R/W

Tableau 5. 2. Règles de contrôle d'accès entre les plates-formes d'exécution

Ces règles sont transformées par le SAS en politiques de contrôle d'accès XACML via PAP (voir chapitre 6).

5.4.3.3. Sécuriser la confidentialité des composants sensibles

Assurer la confidentialité de code revient à cacher les composants sensibles dans les environnements d'exécutions non sûrs. En effet, Les plates-formes malveillantes tentent de décompiler le code et le ré-exécuter le composant avec des paramètres différents, sans la permission de leur propriétaire. Le cryptage de tous les composants fonctionnels est la meilleure protection. Cependant, les opérations de cryptage/décryptage sont généralement coûteuses en temps d'exécution. Dans notre approche, seuls les composants sensibles sont cryptés.

Le créateur d'agent exige dans sa politique de sécurité de garantir la confidentialité de certains composants sensibles. Le rôle de SAS est d'analyser la politique de sécurité de l'agent, puis récupérer les exigences de sécurité concernent la confidentialité de chaque composant. On distingue quatre situations différentes selon les exigences de sécurité de chaque composant :

- Le créateur d'agent n'exige aucune exigence concernant la confidentialité d'un ou plusieurs composants. Toutes les plates-formes visitées par l'agent peuvent manipuler le code de ces composants.
- Le créateur d'agent exige le cryptage d'un ou plusieurs composants sensibles avec une clé symétrique. La clé est partagée entre les plates-formes qui exécutent le même composant sensible. Seulement les plates-formes qui possèdent la clé peuvent accéder au code de ce composant.
- Le créateur d'agent exige le cryptage d'un ou plusieurs composants sensibles avec une clé symétrique. La clé est enveloppée ensuite par la clé publique des plates-formes qui peuvent exécuter ces composants. Dans ce cas, seulement les plates-formes qui possèdent un certificat à clé publique valide peuvent accéder à la clé symétrique pour décrypter le composant sensible.
- Le créateur d'agent exige qu'un composant très sensible ne s'exécute que dans une seule plate-forme. Pour cela, le créateur exige le cryptage asymétrique d'un composant avec la clé publique de la plate-forme. De cette manière, une seule plate-forme authentifiée peut exécuter le composant.

Nous présentons un exemple d'un agent avec trois composants fonctionnels. La figure ci-dessous (figure 5.7) illustre la politique de sécurité utilisée par l'agent pour sécuriser la confidentialité de ses composants. On suppose que le composant «ID-CMP1» s'exécute par les plates-formes P1 et P2. Le composant «ID-CMP2» s'exécute par la plate-forme P3. Le composant «ID-CMP3» s'exécute par la plate-forme P1.

```
<!-- Exigence de sécurité pour la confidentialité de composant 1-->
<MASO:SecurityAlternative Component="ID-CMP1">
    <MASO:hasType RDF:resource="&MASO;#Requirement"/>
    <MASO:hasObject RDF:resource="&MASO;#Confidentiality"/>
    <MASO:SecurityAssertion>
        <MASO:ensuredByAlgorithm RDF:resource="&MASO;#3DES">
            <MASO:encryptedElement RDF:resource="&MASO;#Component"/>
            <MASO:usesToken RDF:resource="&MASO;#symmetricKey">
                <MASO:envelopedBy RDF:resource="&MASO;#RSA"/>
            </MASO:usesToken>
        </MASO:ensuredByAlgorithm>
    </MASO:SecurityAssertion>
</MASO:SecurityAlternative>

<MASO:SecurityAlternative Component="ID-CMP2">
    <!-- Aucune politique pour le composant 2-->
</MASO:SecurityAlternative>

<!-- Exigence de sécurité pour la confidentialité de composant 3-->
<MASO:SecurityAlternative Component="ID-CMP3">
    <MASO:hasType RDF:resource="&MASO;#Requirement"/>
    <MASO:hasObject RDF:resource="&MASO;#Confidentiality"/>
    <MASO:SecurityAssertion>
        <MASO:ensuredByAlgorithm RDF:resource="&MASO;#RSA">
            <MASO:encryptedElement RDF:resource="&MASO;#Component"/>
            <MASO:usesToken RDF:resource="&MASO;#X509Certificate"/>
        </MASO:ensuredByAlgorithm>
</MASO:SecurityAlternative>
```

Figure 5. 7. Politique de sécurité concernant la confidentialité d'un composant

La politique de sécurité présentée dans la figure 5.7 est divisée en trois parties :

- La première est une exigence de sécurité de type Requirement pour assurer la confidentialité de composant «ID-CMP1». Le SAS génère une clé secret aléatoire SK et crypte le composant «ID-CMP1» avec l'algorithme symétrique 3DES spécifie dans la politique. Elle crypte ensuite la clé secret SK par la clé publique PK_{P1} et PK_{P2} des plates-formes qui exécutent ce composant. Le créateur exige l'algorithme asymétrique RSA pour envelopper la clé secret SK.
- La deuxième n'exige aucune contrainte de sécurité concernant la confidentialité de composant «ID-CMP2».
- La troisième exige le cryptage asymétrique du composant très sensible «ID-CMP3». Dans ce cas, le SAS extrait la clé publique du certificat X.509 de la plate-forme P1, puis il utilise l'algorithme asymétrique RSA pour crypter ce composant avec cette clé.

La figure 5.8 montre les composants fonctionnels adaptés de l'agent par le SAS. Avec cette adaptation, seulement les plates-formes P1 et P2 peuvent décrypter et exécuter le composant «ID-CMP1». Elles décryptent la clé secret SK avec ses clés privées PK_{P1} et PK_{P2}, puis elles décryptent le composant avec cette clé SK. Le composant «ID-CMP3» est accessible seulement par la plate-forme P1 qui utilise sa clé privée pour décrypter le composant.

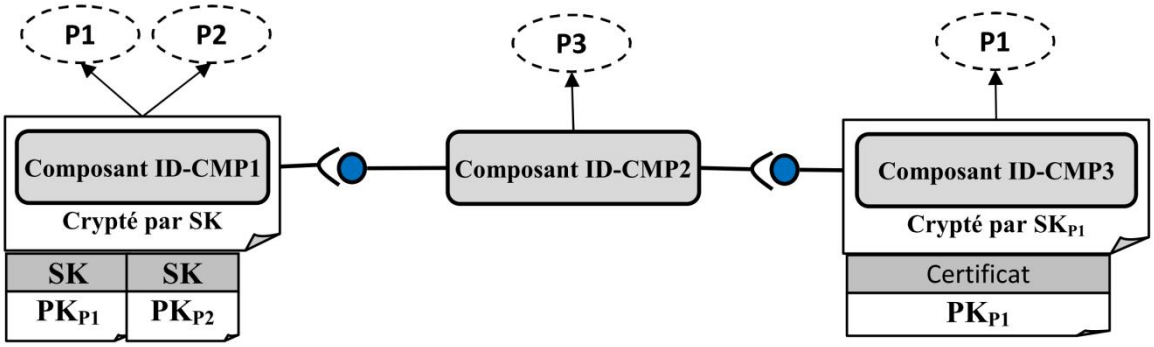

Figure 5. 8. Composants fonctionnels adaptés de l'agent mobile par le SAS

Le tableau 5.3 explique comment le SAS garantit la confidentialité et le contrôle d'accès aux différents composants de l'agent.

	Composant : ID-CMP1	Composant : ID-CMP2	Composant : ID-CMP2
Plate-forme : P1	Autorisée	Autorisée	Autorisée
Plate-forme : P2	Autorisée	Autorisée	Non-autorisée
Plate-forme : P3	Non-autorisée	Autorisée	Non-autorisée

Tableau 5. 3. Contrôle d'accès aux composants sensibles cryptés

Le cryptage symétrique/asymétrique des composants sensibles permet d'une part, d'empêcher les plates-formes non autorisées d'accéder aux composants sensibles. D'autre part, d'assurer l'authenticité des plates-formes qui exécutent ces composants.

5.4.3.4. Sécuriser la confidentialité des données d'agent

L'agent mobile se déplace d'une plate-forme à une autre pour exécuter ses composants et accéder à des données, l'agent peut recueille des données après l'exécution d'un composant sensible, ces données peuvent modifier par des plates-formes malveillantes. Dans ce cas, il est nécessaire de crypter les données sensibles de telle sorte que seules les plates-formes autorisées peuvent lire/modifier ces données. Pour atteindre cet objectif, nous déterminons quatre exigence de sécurité pour assurer la confidentialité des données en fonction de la politique de sécurité d'agent (faible, confidentiel, secret, top secret) :

- Niveau faible si l'agent n'exige aucun cryptage de données.
- Niveau confidentiel si l'agent exige le cryptage symétrique de données collectées dans une plate-forme, seules les plates-formes qui ont le droit de récupérer la clé secrète peuvent lire ou modifier les données cryptées.
- Niveau secret si l'agent exige le cryptage et la signature de données, les plates-formes doivent crypter et signer leurs données pour empêcher la modification de ses données.
- Niveau top secret est le niveau de sécurité le plus élevé qui demande aux plates-formes de crypter les données avec la clé publique du propriétaire d'agent. Dans ce cas, aucune plate-forme n'a le droit de lire ou modifier ces données.

Nous donnons un exemple illustrant la politique de sécurité d'un agent mobile. Cette politique exige les plates-formes d'exécution de crypter ses données avec l'algorithme 3DES.

```
<MASO:SecurityAlternative Component="…">
    <MASO:hasType RDF:resource="&MASO;#Requirement"/>
    <MASO:hasObject RDF:resource="&MASO;#Confidentiality"/>
        <MASO:SecurityAssertion>
            <MASO:ensuredByAlgorithm RDF:resource="&MASO;#3DES">
                <MASO:encryptedElement RDF:resource="&MASO;#Data"/>
                <MASO:usesToken RDF:resource="&MASO;#SymmetricKey"/>
            </MASO:ensuredByAlgorithm>
        </MASO:SecurityAssertion>
</MASO:SecurityAlternative>
```

Figure 5. 9. Politique de sécurité concernant la confidentialité des données

Dans ce cas, le SAS conclu que le niveau de sécurité, pour assurer la confidentialité de données, est confidentiel. Ces données sont cryptées par les plates-formes avec une clé secrète générée par le KDS. Pour gérer l'accès à ces données, le SAS crée des règles de contrôle d'accès en se basant sur les contraintes définies dans la section 5.3.3.2 pour permettre uniquement aux plates-formes autorisées d'accéder à ces données cryptées. Après la création des règles, le SAS demande au PAP de créer des politiques de contrôle d'accès aux données de cet agent.

Si une plate-forme a besoin d'utiliser les données cryptées par d'autres plates-formes, elle demande la clé symétrique de KDC. Le KDC extrait les informations nécessaires pour créer une demande d'autorisation SAML au PDP. Le PDP envoie la réponse permit/rejet en fonction de la politique de contrôle d'accès aux données d'agent définie par le SAS. En fonction de la réponse de PDP, le KDC décide d'envoyer ou non la clé de décryptage.

5.5. Conclusion

L'adaptation de l'agent mobile est une étape cruciale dans le cycle de vie des agents mobiles. Nous avons développé un système de confiance (SGSA) pour transformer l'agent à un agent mobile sécurisé. Ce système contient trois sous-systèmes. Le système d'évaluation d'agent (SEA) qui reçoit l'agent de la part de son créateur pour valider l'authentification et la structure interne de l'agent. Le système de gestion de services (SGS) qui reçoit les agents validés par le SEA pour créer l'itinéraire d'agent, ce sous-système est caractérisé par l'algorithme de correspondance sémantique afin de déterminer la compatibilité entre les politiques de sécurité. Enfin, un système d'adaptation d'agent (SAS) qui doit satisfaire les besoins de sécurité de l'agent et créer les politiques de contrôle d'accès aux données d'agent.

Dans le chapitre suivant, nous présentons les politiques de contrôle d'accès aux données dynamiques des agents mobiles. La solution proposée est basée sur le modèle ABAC/XACML pour protéger les données collectées par l'agent dans les plates-formes d'exécution.

CHAPITRE 6

Protection des données dynamiques des agents mobiles : Politiques XACML/ABAC

6.1. Introduction

Un agent mobile peut se déplacer d'une plate-forme à une autre au cours d'exécution pour accéder à des données ou à des ressources. Il se déplace avec son code et ses données propres. Les agents prennent deux rôles dans l'environnement d'exécution. Le premier, les agents sont traités comme des sujets, parce que les agents accèdent aux différentes ressources d'une plate-forme. Le second, ils sont considérés comme des ressources actives dans l'environnement d'exécution. La protection des ressources des plates-formes contre l'attaque des agents mobiles malveillants est déjà traitée dans la littérature. Cependant, la protection des données dynamiques de l'agent est un problème ouvert et difficile à résoudre.

La mise en place d'une politique de contrôle d'accès aux agents mobiles est une étape importante dans notre système. En effet, le système d'adaptation de sécurité (SAS) définit un ensemble de règles de contrôle d'accès aux données d'agent en fonction des niveaux de sécurité des composants de l'agent et les degrés de confiance des plates-formes d'exécution. Ensuite, il utilise le modèle de contrôle d'accès ABAC pour créer les politiques de contrôle d'accès.

Dans ce chapitre, nous présentons une solution basée sur ABAC et XACML pour protéger les données dynamiques de l'agent et nous montrons ensuite comment intégrer ces deux techniques au sein de notre système d'agents mobiles.

6.2. Politique de contrôle d'accès XACML/ABAC

6.2.1. Modèle ABAC pour le système d'agents mobiles

Le modèle ABAC a été développé par Eric Yuan et Jin Tong [Eric et al., 2005], dans le but de pallier aux difficultés que rencontrent les architectures web services en

termes de sécurité. En effet, les accès à l'information au niveau de ces architectures web services se font non seulement sur les systèmes distribués mais très dynamiquement. Les modèles classiques sont généralement destinés à un fonctionnement statique, ils ne permettent guère une évolution dynamique. De part sa définition, le modèle ABAC peut être le plus adapté pour les architectures fonctionnant dans un environnement ouvert où différentes organisations peuvent assurer à la fois les accès aux informations et la protection de leurs ressources. Parmi les avantages de ce modèle par rapport à nos travaux :

- Introduction de la notion de gestion de contexte via les entités d'environnement. Ceci permet d'obtenir des modèles plus souples, qui peuvent s'adapter à différentes situations et de dynamiser ainsi la prise de décision pour le contrôle d'accès. Cela est très appréciable pour le système multi-agent.
- L'utilisation des attributs offre une granularité très fine pour définir les règles d'autorisation : il suffit de définir un attribut pour prendre en compte un nouveau paramètre entrant en jeu dans la définition des autorisations, qu'il s'applique aux agents, aux ressources ou aux environnements.
- Par rapport à la mobilité, ABAC correspond à notre approche, car il présente un modèle dynamique basé sur les attributs.
- Il définit un formalisme basé sur les attributs qui offre plus de flexibilité en termes de spécifications des conditions de contrôle d'accès.
- Une architecture qui s'adapte au système multi-agent.

L'approche proposée est basée sur le modèle de contrôle d'accès ABAC, ce qui permet d'écrire des politiques de contrôle d'accès aux données d'agent en se basant sur les règles définies par le SAS. Notre solution est basée sur trois catégories d'attributs nécessaires à la description de la politique : catégorie *Subjects* contient un ensemble d'attributs pour identifier les plates-formes qui peuvent accéder aux données d'agent. Catégorie *Resources* permet de spécifier les données dans l'agent mobile, nous utilisons le langage XPath pour localiser ces données avec l'identité de la plate-forme et du composant. Catégorie *Actions* contient trois attributs :

- L'attribut « Create » permet de spécifier la liste des plates-formes qu'ont le droit de demander la clé symétrique de KDS pour créer, crypter et ajouter les données à l'agent.
- L'attribut « Read » permet de spécifier la liste des plates-formes qu'ont le droit de lire les données cryptées par d'autres plates-formes.

- L'attribut « Update » permet de spécifier la liste des plates-formes qu'ont le droit de modifier les données cryptées par d'autres plates-formes.

6.2.2. Aperçu du langage XACML

XACML (eXtensible Access control MarkupLanguage) est un langage de contrôle d'accès basé sur XML et qui a été standardisé par OASIS [XACML, 2011]. XACML fournit un langage de politiques de contrôle d'accès basés sur les attributs (ABAC). De plus, ce langage fournit une architecture pour la mise en œuvre du contrôle d'accès : un protocole de type requête/réponse donne les moyens d'exprimer des requêtes d'accès et les réponses appropriées.

Le langage de politique XACML décrit les exigences de contrôle d'accès en termes de contraintes sur les attributs des sujets, des ressources, des actions et de l'environnement. Plus précisément, les attributs peuvent être des caractéristiques du sujet, de la ressource, de l'action, ou de l'environnement dans lequel la requête est faite. Les attributs ont un identifiant, l'Uniform Resource Name (URN), un type de données et une valeur. Le langage XACML vise à atteindre plusieurs objectifs comme :

- Assurer une protection efficace pour les ressources du système.
- Permettre de concevoir un système indépendant de la plate-forme utilisée.
- Permettre d'intégrer les politiques XACML dans des applications déjà existantes.

6.2.3. Modèle de contrôle d'accès adapté aux systèmes d'agents mobiles

La figure *6.1* présente le diagramme de classes qui décrit la structure de la politique de contrôle d'accès aux données d'agent basée sur le langage *XACML*. La racine de la politique est modélisée l'aide d'un élément « *Policyset »*. Cet élément doit spécifier une cible et un algorithme de combinaison qui permet de combiner les décisions issues des politiques. Une politique « *Policy* » doit aussi avoir une cible ainsi qu'un algorithme de combinaison, qui permet de choisir une décision parmi les règles de la politique. Enfin, une règle spécifie sa cible, sa condition et son effet.

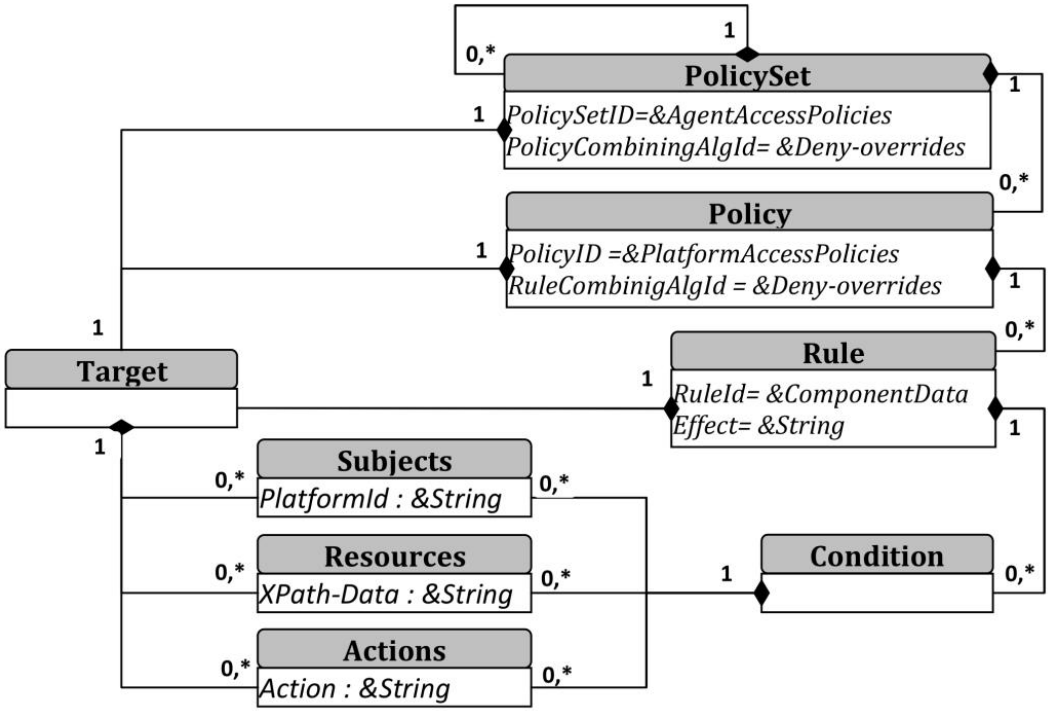

Figure 6. 1. Modèle de contrôle d'accès adapté aux systèmes d'agents mobiles

L'élément « *Policyset* » est exprimé par une cible et un ensemble de politiques. La « *Policyset* » est une agrégation de plusieurs politiques ou des ensembles de politiques. Il contient deux attributs : le premier est l'identificateur unique de la politique « *PolicySetID* » et la deuxième, est un algorithme de combinaison pour combiner la décision de politiques « *PolicyCombiningAlgID* ».

L'élément « *Policy* » est exprimé par une cible et un ensemble de règles. Chaque politique contient deux attributs un identificateur unique de la politique « *PolicyID* » et un algorithme de combinaison entre les règles « *RuleCombiningAlgID* ». Pour identifier les politiques appropriées à l'évaluation de la requête, il faut d'abord comparer la cible de la requête avec la cible de la politique, et par la suite, vérifier les conditions des règles de la politique afin de déterminer la décision « *permit* » ou « *deny* ». Il est possible que les règles d'une politique retournent des décisions différentes par rapport à une requête donnée. L'algorithme de combinaison des règles permet de spécifier comment déterminer la décision de la politique.

La cible « *Traget* » est utilisée pour identifier les règles et les politiques qui concernent une requête. La cible est composée d'attributs qui décrivent le sujet, les ressources et les actions. Le sujet décrit les attributs de la plate-forme qui a fait une demande d'accès aux données d'agent (*PlatformID*). La ressource décrit les attributs de l'objet auquel l'accès est demandée. L'action représente les attributs qui décrivent les mesures que le sujet veut prendre sur la ressource demandée (*Read*, *Update*, *Create*).

Les éléments de type « *Rule* » contiennent deux attributs, l'attribut « *Ruleid* » représente l'identifiant de cette règle et l'attribut « Effect » dont la valeur peut être « *Deny* » ou « *Permit* ». Ils peuvent aussi contenir un élément de type « *Target* », celui-ci hérite des valeurs de l'élément englobant s'il n'est pas présent (c'est-à-dire de l'élément « *Policy* » ou « *Policyset* » contenant l'élément « *Rule* »). Un élément de type « *Rule* » peut aussi contenir un élément de type condition correspondant à une fonction booléenne définie sur les sujets, les ressources et les actions ou sur les différents attributs présents dans la politique.

6.3. Architecture générale de la phase de déploiement

Dans cette section, nous montrons l'interaction entre les différentes entités de la phase de déploiement, puis nous décrirons ensuite en détail le rôle de chaque entité

6.3.1. Interaction entre les différentes entités

L'architecture générale de la phase de déploiement est basée sur trois modes d'utilisation : création, publication et utilisation des politiques de contrôle d'accès pour protéger les données d'agent mobile. Le modèle d'interaction entre les différentes entités de notre architecture est illustré au moyen d'un modèle de flots de données (figure 6.2).

Création : au moment de l'adaptation de la sécurité d'agent, le système d'adaptation de sécurité (SAS) définit les règles de contrôle d'accès aux données d'agent, en fonction des degrés de confiance des plates-formes et les niveaux de sécurité des composants de l'agent (étape 1). Ensuite le SAS demande au PAP de créer les politiques correspondants à ces règles. Le PAP assure que les règles d'accès sont conformes à la syntaxe XACML, puis créer et stocker les politiques de contrôle d'accès à son niveau (étape 2).

Publication : les politiques spécifiées avec le PAP sont accédées par le PDP, cet accès peut être réalisé au moyen d'un dépôt sécurisé. Le PDP calcule des décisions d'autorisation pour les politiques déployées par le PAP. La décision d'autorisation est calculée à la suite d'une requête d'autorisation. Les politiques qui correspondent à la cible de la requête d'autorisation sont utilisées pour faire le calcul (étape 5).

Utilisation : lorsqu'une plate-forme désire accéder à une donnée cryptée dans l'agent mobile pour laquelle une politique de contrôle d'accès est définie, il envoie sa demande d'autorisation SAML au KDS pour avoir la clé symétrique, cette demande contient l'identité de la plate-forme qui fait la demande, la localisation XPath des données et l'action à réaliser (étape 3). Le KDS vérifie l'intégrité et

l'authenticité de la demande puis utilise le composant PEP pour envoyer une demande de décision au PDP avec les attributs nécessaires à la prise de décision (étape 4). Le PDP détermine les règles et les politiques applicables à la requête de PEP. Après l'évaluation des cibles et des conditions de l'ensemble des règles, le PDP choisit la réponse à envoyer au PEP (étape 5). Ensuite, le PDP retourne la décision d'autorisation au PEP (étape 6). Si la réponse est favorable « Permit » le PEP donne l'autorisation au KDS de chercher dans sa base de données la clé correspond à la requête, puis envoie la clé symétrique à la plate-forme pour décrypter les données, sinon il refuse la demande de la plate-forme (étape 7).

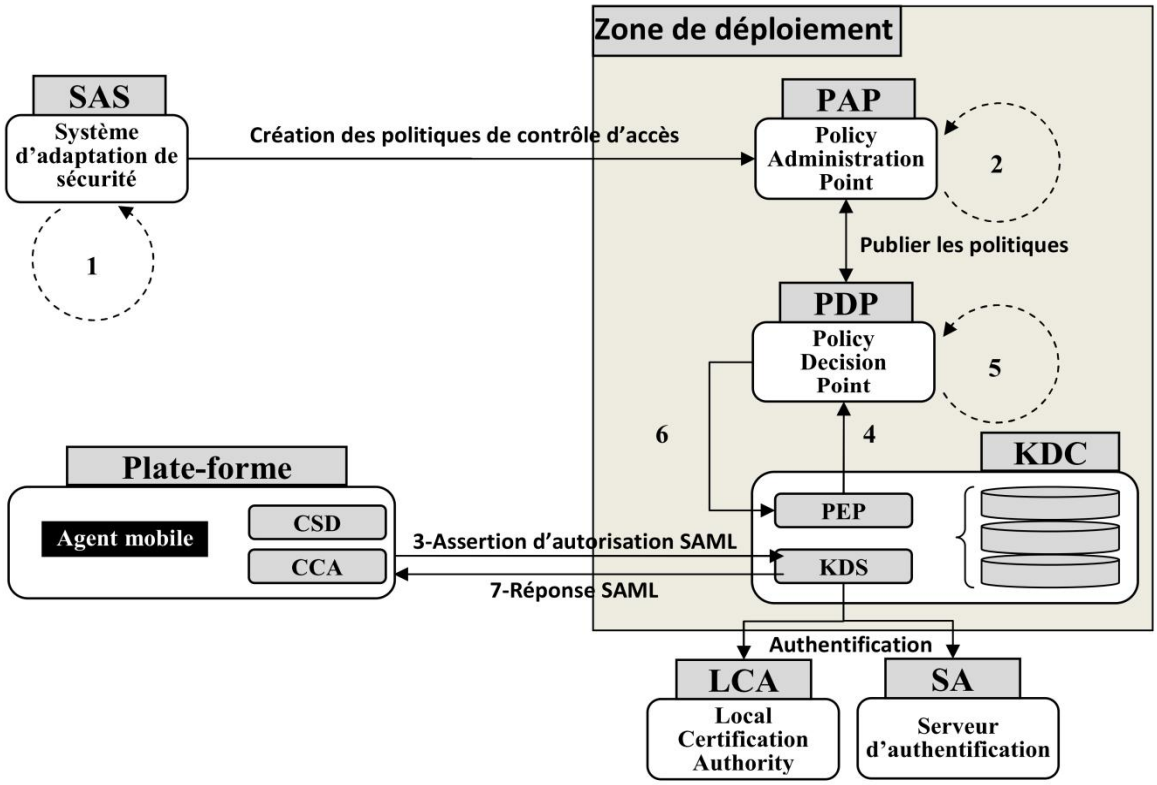

Figure 6. 2. Interaction entre les différentes entités dans la phase de déploiement

6.3.2. Composants de la phase de déploiement

L'architecture de notre système dans la phase de déploiement est constituée par quatre composants :

Serveur PAP (Policy Administration Point) permet de créer les différentes politiques XACML pour un agent mobile. Au moment de l'adaptation de la sécurité d'agent, le SAS définit un ensemble de règles de contrôle d'accès aux données d'agent (expliqué dans la section 5.3.3), puis demande au PAP de créer les politiques de contrôle d'accès correspondent à ces règles. Le PAP stocke les politiques et les rend disponibles au PDP. Ces politiques représentent la politique de gestion complète qui contrôle les décisions prises par le PDP.

Serveur PDP (Point de décision des politiques) est l'entité qui gère les demandes d'autorisation en provenance de serveur KDC, il prend en charge de déterminer les règles et les politiques applicables à une requête donnée. Après l'évaluation des cibles et des conditions de l'ensemble des règles, le PDP renvoie la réponse au PEP. Le serveur PDP repose sur l'autorité de certification (LCA) pour vérifier la validité des certificats des différentes entités.

Serveur KDC (Key Distribution Center) est utilisé par notre système pour protéger les données d'agent durant son exécution. Par conséquent, seules les plates-formes autorisées par le serveur PDP peuvent accéder aux données cryptées d'agent mobile. Le KDC contient deux composants :

- Policy Enforcement Point (PEP) est un composant qui effectue des requêtes d'autorisation au PDP, il applique la décision d'autorisation prise par le PDP. C'est le PEP qui réalise techniquement l'attribution ou le refus des clés symétriques aux plates-formes pour crypter ses données ou décrypter les données des autres plates-formes. Il est logiquement situé entre les demandeurs des clés (plate-forme) et le distributeur des clés (KDS). Le PEP permet aussi de traiter les conditions envoyées par le PDP pour que la politique s'applique à une requête donnée.
- Key Distribution Service (KDS) permet d'intercepter les requêtes et vérifier l'identité des demandeurs des clés. Le KDS utilise une base de données pour gérer les clés symétriques distribuées aux plates-formes. La base de données contient des informations sur l'identité de la plate-forme propriétaire de la clé. L'identité de l'agent qui emporte les données cryptées et le chemin XPath des données à protéger par la clé. Le composant KDS effectue trois fonctionnalités : la création des clés, la distribution des clés et le contrôle d'accès aux clés. Il utilise le PEP pour récupérer la décision d'autorisation à partir de PDP.

Serveur LCA (Local Certificate Authority) est un composant de PKI (Public Key Infrastructure) permet de créer et publier les certificats numériques signés comportant la clé publique et l'identité de chaque entité dans notre infrastructure. Les certificats numériques sont utilisés pour authentifier des différentes entités du système. Pour avoir la confiance quant à la distribution des clés publiques. Seule l'autorité de certification locale peut créer et distribuer des certificats signés, chaque entité possède la clé publique de LCA et s'en sert pour vérifier la validité des certificats des autres entités.

6.4. Création des politiques de contrôle d'accès aux données d'agent

Quand un agent arrive sur une plate-forme d'exécution, il expose son code et ses données. De ce fait, cette plate-forme aura l'habilité de les modifier. Elle peut modifier les données propres de l'agent ou bien les résultats obtenus par l'agent mobile à l'intérieur des plates-formes précédentes. Le propriétaire d'un agent ne veut pas qu'une plate-forme non fiable puisse modifier l'agent ou les résultats qu'il a obtenus sur d'autres plates-formes. Dans ce cas, il est nécessaire de crypter les données sensibles d'une manière que seules les plates-formes autorisées peuvent lire ou modifier ces données.

Le modèle d'agent que nous avons construit dispose une politique de sécurité définie par le créateur d'agent. Cette politique contient un ensemble d'exigences de sécurité qui doivent être satisfaites par les plates-formes visitées comme le cryptage et la signature des données. Pour contrôler l'accès à ces données cryptées, le système d'adaptation de sécurité SAS définit un ensemble de règles de contrôle d'accès aux données d'agent, ces règles déterminent quelle plate-forme peut accéder aux données produites par d'autre plate-forme. Ces règles sont transformées ensuite à des politiques de contrôle d'accès XACML.

La figure 6.3 présente une politique de contrôle d'accès aux données d'agent basé sur le langage XACML. Les politiques sont des ensembles de politiques (*policySet*) éventuellement réduits à une seule, comme on peut le voir ci-dessous.

```
<PolicySet
   xsi:schemaLocation="urn:oasis:names:tc:xacml:2.0:policy:schema:os
   http://docs.oasis-open.org/xacml/access_control-xacml-2.0-policy-schema-os.xsd"
   PolicySetId="urn:oasis:names:tc:xacml:2.0:example:policysetid:1"
   PolicyCombiningAlgId="urn:oasis:names:tc:xacml:1.0:policy-combining-algorithm:deny-
   overrides">
   <Description>
        Potilique de sécurité globale d'un agent mobile
   </Description>
 <Target/>
 <Policy
   PolicyId="urn:oasis:names:tc:xacml:2.0:example:policyid:2"
   RuleCombiningAlgId="urn:oasis:names:tc:xacml:1.0:rule-combining-algorithm:deny-
   overrides">
   <Description>
        Politique de sécurité pour une plate-forme spécifique.
   </Description>
   <Target/>
   <Rule RuleId="urn:oasis:names:tc:xacml:2.0:example:ruleid:1" Effect="Permit">
        <Target>....</Target>
        <Condition FunctionId="urn:oasis:names:tc:xacml:1.0:function:and">

                        ……

        </Condition>
   </Rule>
 </Policy>
</PolicySet>
```

Figure 6. 3. Exemple de politique de contrôle d'accès aux données d'agent en XACML

La figure 6.3 représente une politique (*PolicySet*) de contrôle d'accès. La politique contient un identificateur unique (*PolicySetId*) et des politiques (*Policy*) pour les plates-formes d'exécution.

Une politique est exprimée par une cible, un ensemble de règles et un algorithme de combinaison. Pour identifier les politiques appropriées à l'évaluation de la requête, il faut d'abord comparer la cible de la requête avec la cible de la politique, et par la suite, vérifier les conditions des règles de la politique afin de déterminer la décision «*permit*» ou «*deny*». Il est possible que les règles d'une politique retournent des décisions différentes par rapport à une requête donnée. L'algorithme de combinaison des règles permet de spécifier comment déterminer la décision de la politique.

6.5. Conclusion

Le principal avantage de l'approche proposée dans ce chapitre est la définition de la politique de contrôle d'accès aux données dynamiques des agents mobiles. Le système d'adaptation de sécurité SAS définit les règles de contrôle d'accès aux données d'agent en fonction du niveau de sécurité des composants fonctionnels et le degré de confiance des plates-formes visitées par l'agent. Nous avons utilisé ensuite le modèle ABAC et le langage XACML pour créer les politiques de contrôle d'accès.

Dans le chapitre suivant, on présente les différents processus de protection des données dynamiques des agents mobiles. Nous présentons d'abord l'initialisation de l'agent mobile par son propriétaire et l'intégration de son authentification unique dans notre approche. Ensuite nous discuterons les différents scénarios de protection des données.

CHAPITRE 7

Protection des agents mobiles et Etude des performances

7.1. Introduction

Nous avons montré qu'il y'a deux techniques disponibles pour sécuriser les agents mobiles (technique de prévention et de détection). Cependant, notre contribution majeure est une approche de protection de code et des données d'agent mobile. La protection de code consiste à crypté et signé les composants sensibles (code critique) pour garantir l'intégrité, la confidentialité et l'authenticité de l'agent. Cette approche consiste à proposer une politique de sécurité définie par le créateur d'agent qui détermine comment sécuriser son code et ses données sensibles.

La protection des données de l'agent consiste à assurer l'intégrité, la confidentialité et le contrôle d'accès aux données de l'agent. Notre approche est basée sur le langage XML pour stocker les données dans l'agent mobile. Chaque composant est équipé par une description du niveau de base qui aide les plates-formes d'ajouter leurs données dans l'agent, ainsi une politique de sécurité pour protéger ces données. En effet, la meilleure approche disponible pour protéger les données de l'agent est d'utiliser la clé publique de leur propriétaire. Cette approche est très restrictive, parce que le propriétaire d'agent est le seul qui peut décrypter les données transportées par l'agent. Cependant, notre solution est basée sur plusieurs méthodes de protection en fonction de la sensibilité des données transportées par l'agent.

Ce chapitre décrit la fiabilité, la performance et la robustesse de notre approche de protection proposée vis-à-vis des différents types attaques. Nous dressons un tableau montrant les mécanismes de sécurité proposés contre les différents type d'attaque discutés dans le chapitre 2.

7.2. Initialiser et authentifier l'agent mobile par son propriétaire

7.2.1. Initialiser l'agent mobile par son propriétaire

Après la création et l'adaptation de l'agent mobile, le rôle du propriétaire d'agent (PA) est d'initialiser et de lancer l'agent dans le réseau pour effectuer ces tâches. Le résultat obtenu par l'agent durant son exécution est retourné vers son propriétaire. Avant le lancement de l'agent, le PA crée les données statiques de l'agent (les données qui ne changent pas durant l'exécution de l'agent). Comme le montre la figure 7.1, les trois composants de l'agent (interface, méta-niveau et le niveau de base) sont signés par le créateur d'agent (CA) et l'entité qui sécurise l'agent (SGSA). Afin d'assurer l'intégrité des données statiques, le propriétaire d'agent (PA) signe numériquement ses données avec sa clé privée SK_{PA}, et il attache ensuite son identité PA-ID, son certificat à clé publique C_{PA} et sa signature S_{PA} à l'agent mobile sous la forme signedData de PKCS7.

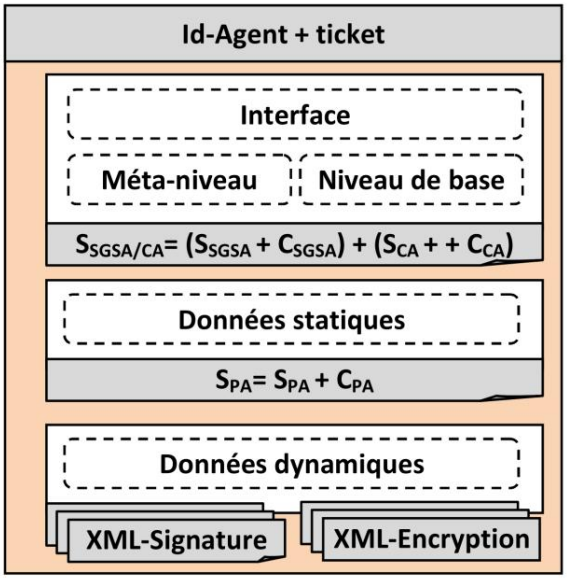

Figure 7. 1. Structure interne d'un agent mobile sécurisé

Notons que l'approche proposée utilise le langage XML pour créer et structurer les données dynamiques de l'agent mobile. Ces données sont ajoutées par les plates-formes d'exécutions durant l'exécution de l'agent. Dans ce cas, c'est la politique de sécurité de l'agent qui exige la protection de la confidentialité et l'intégrité des données collectées dans les plates-formes visitées.

Notre solution est basée sur deux recommandations de W3C pour signer et crypter les données dynamiques. La première est XML-Signature utilisée pour signer une partie de données dynamique de l'agent. Les plates-formes peuvent modifier le reste des données. Avec cette fonctionnalité, deux plates-formes peuvent donc signer des parties différentes de données. Dans ce cas, l'agent mobile peut visiter

plusieurs plates-formes et chacune d'eux peut signer sa partie et en assurer la responsabilité. La deuxième XML Encryption permet de crypter certaines parties des données dynamiques de l'agent, laissant ainsi d'autres parties lisibles pour d'autres plates-formes. Aussi, cela permettra aux différentes plates-formes de lire les données de manières différentes, chacune pouvant lire certaines parties de celui-ci sans avoir accès à la totalité des données d'agent.

7.2.2. Authentification unique d'un agent mobile

Nous avons utilisé un système d'authentification unique SSO (Single Sign On) pour faciliter les procédures d'authentification des agents mobiles. Alors qu'il devait s'identifier successivement auprès chaque migration. Le SSO lui permet de ne s'authentifier qu'une seule fois par un Serveur d'Authentification (SA). L'utilisation d'un serveur commun d'authentification devrait également faciliter l'évolution des méthodes d'authentification ou la prise en compte de plusieurs niveaux d'authentification en fonction de la politique de sécurité de l'agent et les plates-formes. Notre serveur d'authentification prend en charge les différentes techniques d'authentification, comme le nom d'utilisateur/mot de passe et les certificats numériques X509.

Le Propriétaire d'Agent (PA) envoie au SA une demande de ticket d'authentification contenant les informations d'identités de l'agent sous forme d'un jeton d'authentification. Les entités SA et PA s'authentifient mutuellement en utilisant le protocole décrit dans la section 5.3.

$$\textit{Demande de ticket} = [\{\textit{Id-Agent} ; H_{(CA)} ; S_{CA} ; JA\} S_{PA}] PK_{SA}$$

La requête contient quatre informations, l'identité unique de l'agent mobile, le condensé de l'agent $H_{(CA)}$ est utilisé pour associer un ticket à un agent et empêcher la vole des tickets, la signature de créateur d'agent S_{CA} est utilisé pour valider le condensé de l'agent et authentifier son créateur et le jeton d'authentification (JA) permet de spécifier la méthode et les paramètres d'authentification utilisés par le propriétaire d'agent. Les quatre informations sont signées par la clé privée de S_{PA} pour empêcher l'altération de la demande. Enfin, pour assurer la confidentialité de la demande, le PA utilise la clé publique PK_{SA} de SA pour crypter la demande.

Le serveur d'authentification SA est l'élément central du système puisqu'il assure l'authentification des agents mobiles en fonction de son jeton d'authentification. Il génère les tickets et transmis au PA, ainsi que de récupérer des informations additionnelles associées à son identité. En effet, le serveur d'authentification reçoit et décrypte la demande de PA avec sa clé privée, puis vérifie la signature de

différentes entités (PA, CA). Enfin le SA récupère le JA pour déterminer la méthode et les paramètres d'authentification utilisé par l'agent. Par exemple, si la méthode d'authentification est «UsernameToken», la phase d'authentification implique la vérification du nom d'utilisateur et le mot de passe auprès d'une base de référence.

Lorsque l'agent mobile a été authentifié, le serveur d'authentification mis en cache le ticket pour authentifier l'agent plus tard. Le ticket est signé par la clé privée de SA pour garantir l'intégrité et l'authenticité de ticket.

$$\textit{Ticket d'agent} = \{\textit{DN-Agent, Id-Agent ; MA ; IP}_{SA} \textit{ ; } T_{exp} \textit{ ; } H_{(CA)} \textit{ ; } S_{CA}\}\ S_{SA}$$

Le ticket attribué au propriétaire d'agent contient un identifiant de l'entité qui utilise :

- DN-Agent : un identifiant de l'entité qui utilise le ticket.
- Id-Agent : une identité unique de l'agent mobile.
- MA : la méthode d'authentification utilisée pour authentifier l'agent.
- IP $_{SA}$: L'adresse IP du serveur d'authentification.
- T_{exp} : Déterminé le temps d'expiration du ticket.
- $H_{(CA)}$: le condensé de l'agent mobile.
- S_{CA} : la signature du créateur d'agent.
- S_{SA} : la signature du serveur d'authentification.

Le PA reçoit le ticket, vérifie la signature de serveur SA, puis attache le ticket à l'agent mobile. La figure 7.2 présente les composants et les messages échangés entre le PA et SA lors de l'authentification de l'agent mobile.

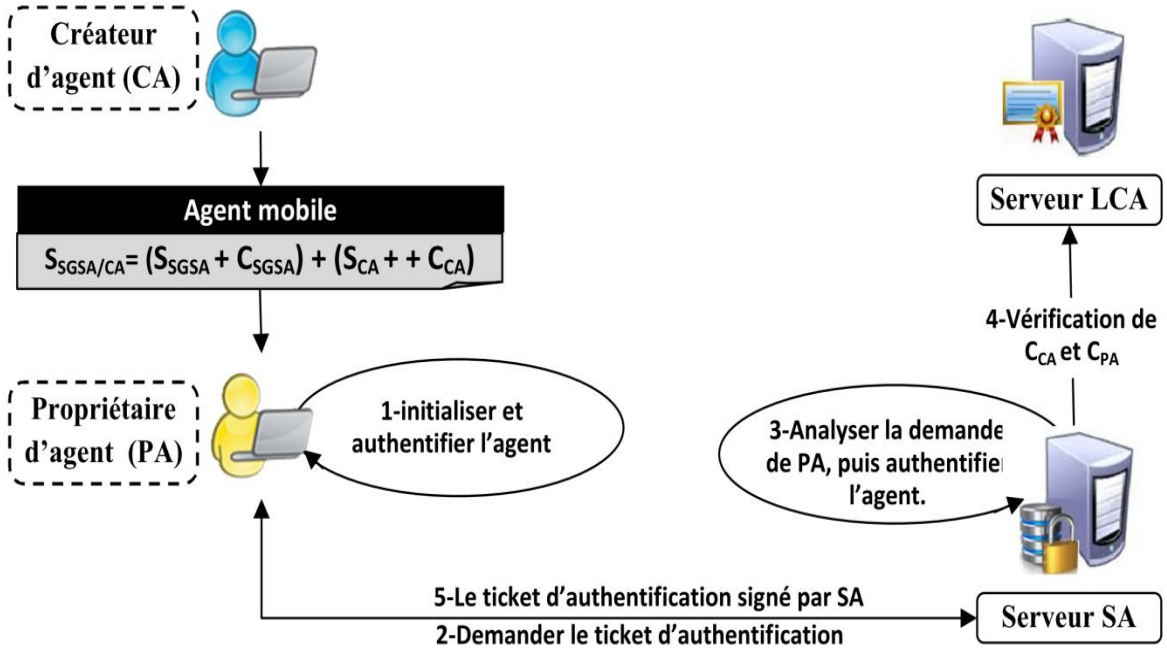

Figure 7. 2. Authentification d'un agent mobile par le SA

Le ticket est le seul moyen technique fiable pour que l'agent mobile soit reconnu comme authentifié lors de son exécution dans une plate-forme. Il est important de noter que notre solution est plus puissante que les autres solutions. En effet, les données sensibles d'authentification sont stockées de manière sécurisée dans le serveur d'authentification. Les agents utilisent des tickets délivrés par le SA pour s'authentifier durant son exécution et les plates-formes fait une confiance absolue aux tickets fournis par le SA.

7.3. Protection d'un agent mobile dans les plates-formes d'exécution

7.3.1. Valider, authentifier et décrypter les composants sensibles à exécuter

Lorsqu'un agent a été lancé dans le réseau, l'agent est reçu par les plates-formes d'exécution. Ces plates-formes offrent les fonctionnalités de base pour exécuter l'agent mobile (l'accueil, la communication, les ressources, la migration…). Avant l'exécution de l'agent, les plates-formes vérifient la validité des signatures associées à l'agent (S_{CA}, S_{SGSA}, S_{PA}). La première signature S_{CA} est utilisée pour vérifier l'authenticité et l'intégrité de code, la deuxième signature S_{SGSA} permet d'authentifier l'entité ayant adapté l'agent et le troisième S_{PA} authentifie l'entité qui utilise l'agent ainsi l'intégrité des données statique de l'agent. Si toutes les vérifications sont réussies, les plates-formes analysent ensuite le ticket de l'agent délivré par le serveur d'authentification (SA), la validé de la signature de SA, le temps d'expiration de ticket et la méthode d'authentification utilisée pour authentifier l'agent. Pour empêcher l'utilisation de ticket par un autre agent, les plates-formes peuvent calculer le condensé de l'agent et le comparer avec le condensé dans le ticket de l'agent, s'il y a une correspondance entre les deux condensés alors le ticket n'est pas volé.

Dans notre approche, l'agent mobile contient plusieurs composants sensibles cryptés avec des clés symétriques. Chaque clé est enveloppée par la clé publique de la plate-forme qui exécute ce composant. De cette manière, seules les plates-formes autorisées peuvent décrypter et exécuter ces composants.

7.3.2. Contrôle d'accès aux données d'agent mobile

L'agent mobile se déplace à travers le réseau d'une plate-forme vers une autre dans le but de fournir un service précis à son propriétaire. Au cours de son exécution, l'agent peut recueillir ou revoir des données. Dans ce cas, il est nécessaire de crypter les données sensibles de sorte que seules les plates-formes autorisées peuvent accéder à ces données. En effet, chaque agent mobile est équipé par une politique

de sécurité qui détermine les données sensibles à crypter par les plates-formes, ainsi une description du niveau de base qui aide les plates-formes à construire leurs données dans un format uniforme et standard.

Les données dynamiques sont ajoutées sous forme des balises XML, ces derniers représentent les valeurs des attributs dans le code de l'agent, de cette façon une plate-forme peut contribuer plusieurs variables dans les données d'agent. Ces variables peuvent être d'un type simple ou complexe. En cas de type complexe, nous sérialisons l'objet et le stocker sous forme d'octets. Les plates-formes vérifient la description du niveau de base de l'agent pour savoir comment ajouter leurs données dans l'agent. La figure 7.3 montre un exemple de données ajoutées par la plate-forme PL-1 après l'exécution de composant CM-1.

```
<?xml version="1.0" encoding="ISO-8859-1"?>
      <Platform Id="PL-1">
            <Component ID="CM-1">
                  <Variable1> Valeur 1 </Variable1>
                        ....
                        ....
                  <VariableN> Valeur N </VariableN>
            </Component>
      </ServivePlateForme>
```

Figure 7. 3. Données ajoutées par une plate-forme d'exécution

Notons que les données sont ajoutées à l'agent soient claires ou cryptées en fonction de la politique de sécurité d'agent. Dans la suite, nous présentons deux scénarios pour protéger les données d'agent mobile. La première est comment les plates-formes ajoutent ses données cryptées à l'agent mobile, la deuxième est comment les plates-formes lient ou modifient les données cryptées par d'autres plates-formes.

7.3.2.1. Protection de données collectées sur une plate-forme

Notre agent mobile se déplace selon un itinéraire préalablement établi par l'entité SGSA. Supposons que la première plate-forme visitée par l'agent est PL-1, l'itinéraire d'agent détermine le composant qui doit être exécuté par cette plate-forme, ainsi la politique de sécurité utilisée par le composant. Après l'authentification de l'agent, si le composant est crypté, la plate-forme décrypte la clé symétrique avec sa clé privée, puis décrypte le composant avec cette clé. Supposons que la plate-forme veut ajouter des données à l'agent. Dans ce cas, la plate-forme utilise son Composant de Stockage de Données (CSD) pour ajouter ses données (format XML) à l'agent en cours d'exécution.

Supposons que le créateur d'agent exige dans sa politique de sécurité le cryptage des données produites par ce composant, la plate-forme PL-1 utilise le Composant

de Contrôle d'Accès (CCA) pour envoyer une demande d'une clé symétrique au KDS (Le KDS est un composant interne de serveur KDC voir la section 6.3.1). Cette demande contient l'identité de la plate-forme, l'identité de l'agent et le composant en cours d'exécution, l'étiquette de données à crypter et l'action « *Create* ». Dans notre approche, on utilise trois valeurs pour l'attribut action, la valeur « *Create* » pour demander la création d'une nouvelle clé symétrique au KDS (le KDS ajoute la clé dans sa base de données), les valeurs « *Update* » et « *Read* » sont utilisées pour récupérer les clés symétriques de KDS.

La demande de la plate-forme est signée par sa clé privée. Le KDS reçoit la demande, puis authentifie le demandeur de la clé. Notre solution est basée sur les règles de contrôle d'accès définies par le SGSA au moment de l'adaptation d'agent (les règles sont définies par le SAS dans la section 5.3.3.2). Dans ce cas, seules les plates-formes autorisées peuvent ajouter les données à l'agent. Le KDS utilise le composant PEP pour envoyer une demande d'autorisation SAML au PDP, cette demande contient les attributs nécessaires à la prise de décision. Le PDP détermine les règles et les politiques applicables à la demande de PEP. Après l'évaluation des cibles et des conditions de l'ensemble des règles, le PDP retourne la décision d'autorisation au PEP. Si la réponse est favorable «Permit», le PEP donne l'autorisation au KDS pour générer une nouvelle clé symétrique spécifique à la plate-forme PL-1. Le KDS enregistre la clé dans un tableau indexé par l'identité de la plate-forme. Le tableau 7.1 contient les informations enregistrées par le KDS.

Id-plate-forme	Id-Agent	Id-composant	Etiquette de données	Date de création	Date d'expiration	Clé symétrique
PL-1	AM-1	CM-1	Data1	08/08/2016	09/09/2016	Key1
PL-2	AM-1	CM-2	Data2	07/02/2016	06/05/2016	Key2
PL-3	AM-1	CM-3	Data3	19/07/2016	20/07/2016	Key3

Tableau 7. 1. Les informations enregistrées par le KDS

Après l'enregistrement de la clé, le KDS envoie la clé générer au CCA, le composant de stockage de données (CSD) utilise la clé pour crypter ces données. La clé utilisée est alors détruite avant que l'agent ne migre vers une nouvelle plate-forme. Notons que le tableau des clés est utilisé plus tard par le KDS pour fournir les clés aux plates-formes autorisées pour décrypter ces données.

7.3.2.2. Accès aux données cryptées par les autres plates-formes

Quand l'agent migre d'une plate-forme à une autre, il transporte avec lui les données de son exécution dans les plates-formes précédentes, on prend l'exemple d'une plate-forme (PL-2), qui souhaite de lire les données créées et cryptées par la plate-forme (PL-1). De même manière que l'ajout de données, le composant (CCA) de la plate-forme PL-2 demande la clé symétrique de KDS pour décrypter les données cryptées par PL-1. La demande contient l'identité de la plate-forme, de l'agent et le composant qui produit les données, l'étiquette de données et l'action à réaliser « Update » ou « Read ». Le KDS vérifie l'authenticité et l'intégrité de la demande puis redirige la demande vers le PEP, le PEP envoie ensuite une demande d'autorisation SAML au PDP avec les attributs nécessaires à la prise de décision. Le PDP évalue les règles et les politiques applicables à la demande puis envoie sa décision au PEP. Si la décision est favorable, le PEP demande au KDS de chercher et d'envoyer la clé symétrique à la plate-forme. Le CCA récupère la clé symétrique puis décrypte les données cryptées par la plate-forme PL-1. La figure 7.4 montre comment une plate-forme décrypte les données cryptées par d'autres plates-formes, ainsi l'interaction entre les différentes entités pour récupérer la clé symétrique.

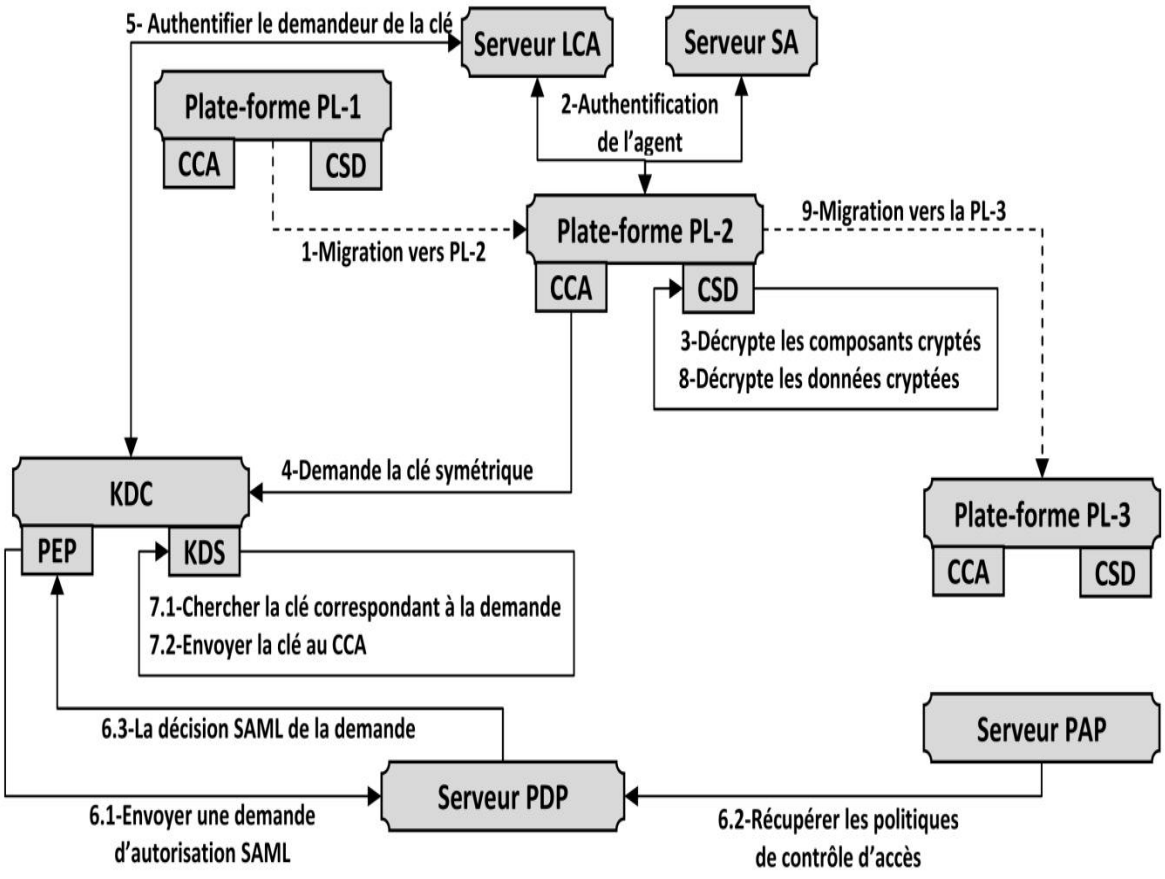

Figure 7. 4. Processus de cryptage/décryptage des données d'un agent mobile

7.3.3. Récupérer les résultats d'exécution de l'agent par son propriétaire

L'agent mobile se déplace entre les plates-formes en fonction de leur itinéraire pour accéder à des ressources et collecter des données. Lorsque l'agent retourne vers sa plate-forme d'origine, le propriétaire d'agent (PA) commence le processus de vérification si nécessaire. Tout d'abord, il authentifie l'agent puis vérifie que le code n'est pas altéré par les plates-formes malveillantes. Pour cela, il vérifie les signatures S_{CA}, S_{SGSA}, S_{PA}. Si les vérifications sont réussies, le PA récupère les données dynamiques collectées par l'agent durant son exécution. Ces données peuvent contenir des données signées ou cryptées par les plates-formes d'exécution. Premièrement, le PA vérifie la signature de données par la clé publique des plates-formes, et il utilise aussi l'Autorité de Certification Locale (LCA) pour vérifier la validité des certificats. Deuxièmement, si les données dynamiques de l'agent contiennent des parties cryptées, le PA envoie une demande de toutes les clés symétriques utilisées par les plates-formes au KDC. Cette demande contient l'identité de créateur d'agent (sa signature), l'identité de l'agent et les composants qui créent les données cryptées. Le serveur KDC utilise les mêmes procédures que celles décrites dans la section précédente pour envoyer les clés symétriques au PA.

Dans notre approche, les règles de contrôle d'accès aux données d'agent autorisant le créateur d'agent d'accéder à toutes les données cryptées de son agent.

7.4. Expérimentation et étude des performances

Après avoir détaillé les concepts de notre approche protégeant les composants et les données de l'agent mobile contre les attaques des plates-formes malveillantes et avoir discuté l'architecture à base de composants de l'agent, nous décrivons au sein de cette patrie l'implémentation de l'ontologie MASO avec le langage OWL sous Protégé 4 [Musen, 2015]. Nous utilisons ensuite le système Racer pour tester cette ontologie [Haarslev et al., 2012]. Le développement de l'algorithme *Matching-algorithm* est réalisé par le langage Java et deux API pour manipuler l'ontologie MASO et les politiques de sécurité (API Jena et API Jdom). Nous choisissons la plate-forme JADE sous l'environnement Eclipse [Eclipse, 2016] pour développer un prototype de notre système de protection. Cette partie est divisée en trois :

- Implémentation de l'ontologie MASO et l'algorithme Matching-algorithm.
- Différentes technologies de sécurité existantes qu'on a intégrées dans notre système.
- Protocole de protection vis-à-vis des différentes attaques des plates-formes malveillantes.

7.4.1. Ontologie MASO

7.4.1.1. Implémentation de l'ontologie MASO

Après la conception, nous allons implémenter notre ontologie. Notre choix porte sur OWL qui représente un langage de codification utilisé pour implémenter l'ontologie en OWL, et cela pour toutes les fonctionnalités sémantiques que permet OWL et qui sont plus riches que celles des langages RDFS & DAML+OIL.

Pour construire nos ontologies, nous avons utilisé Protégé OWL 4, un éditeur open-source Java permettant de créer et de manipuler des ontologies au format OWL. Protégé OWL 4 est le plus connu et le plus utilisé des éditeurs d'ontologie. Il offre notamment la possibilité d'obtenir une représentation graphique de l'ontologie développée et permet d'exécuter des raisonneurs sur cette ontologie (les descripteurs logiques). Les ontologies sont ensuite utilisées avec la librairie OWL API [Horridge et al., 2011]. OWL API est une librairie Java open-source sous licence LGPL qui permet de créer et de manipuler des ontologies. C'est cette librairie qui nous permet de faire correspondre les politiques de sécurité de l'agent et les plates-formes visitées. La Figure 7.5 présente l'interface de Protégé 4, avec les différents classes, instances et propriétés de l'ontologie MASO.

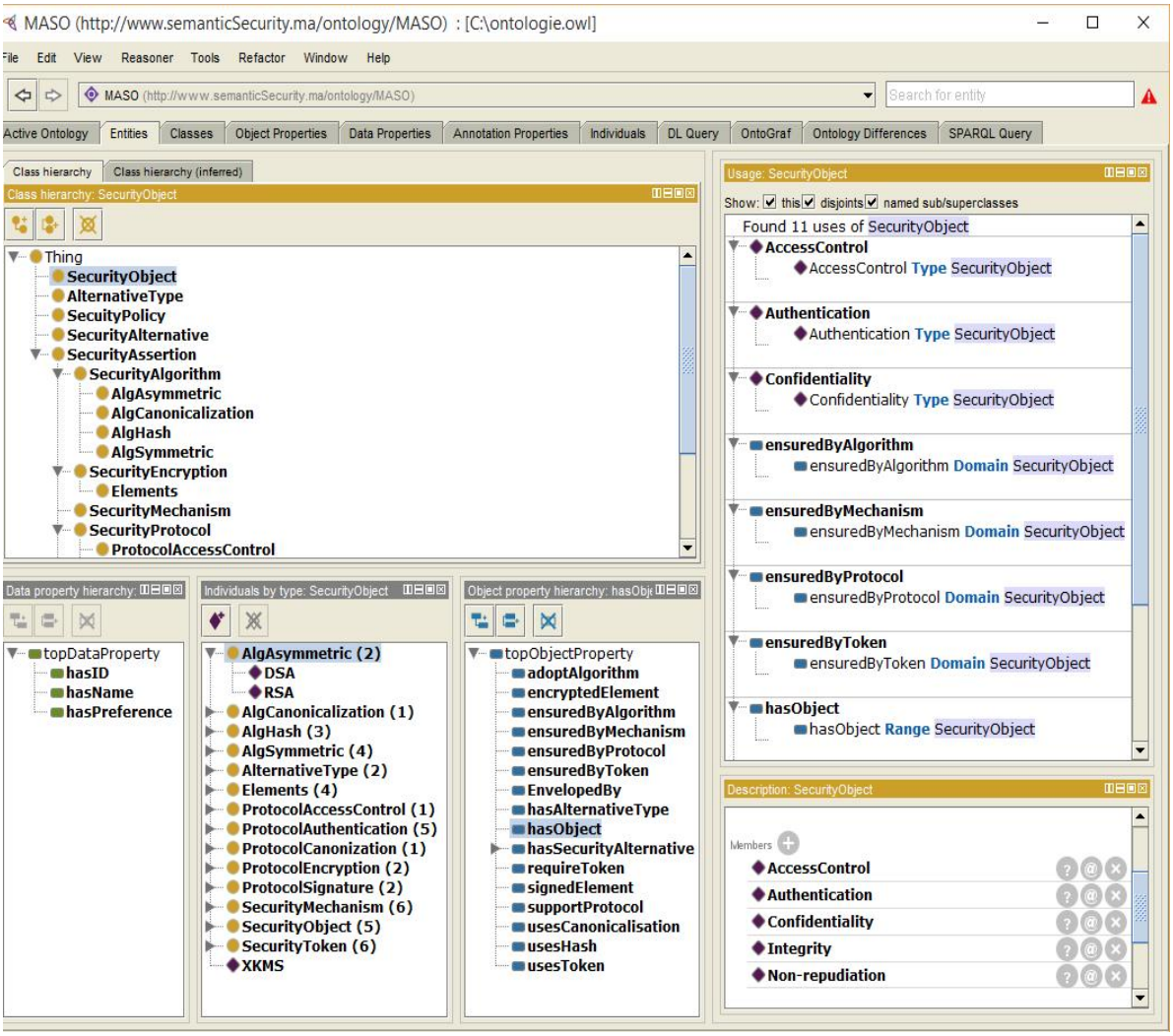

Figure 7. 5. Interface de Protégé 4 (classes, instances et propriétés de l'ontologie MASO)

L'interface, très bien conçue, et l'architecture logicielle permettant l'insertion de plug-ins pouvant apporter de nouvelles fonctionnalités (par exemple, la possibilité d'importer et d'exporter les ontologies construites dans divers langages opérationnels de représentation tels que OWL ou encore la spécification d'axiomes) ont participé au succès de Protégé 4, qui regroupe une communauté d'utilisateurs très importantes et constitue une référence pour beaucoup d'autres outils.

- Définition de la hiérarchie des classes : nous commencerons tout d'abord par la création des concepts spécifiés dans l'étape de conceptualisation. Protégé 4 nous offre également un moyen de construire la hiérarchie de concepts.
- Définition des instances : la dernière étape consiste à créer les instances des classes dans la hiérarchie.
- Définition des propriétés : après avoir construit les classes, nous allons maintenant créer les propriétés pour chacun d'eux, les attributs vont être

créés sous Protégé 4 par « dataTypeProperty » et les relations par « objectProperty ».

- Définitions des restrictions : la dernière étape consiste à créer des restrictions sur les classes et les propriétés. Protégé 4 nous offre un moyen pour créer des restrictions sur les classes et les propriétés.

7.4.1.2. Test de l'ontologie

Il est possible de raisonner sur les ontologies en utilisant un moteur d'inférence général tel que FACT++ [Tsarkov et al., 2006], ou des outils d'inférence spécifiques au Web sémantique basés sur des DL tels que RACER [Haarslev et al., 2012]. Ces deux outils peuvent être facilement intégrés à Protégé 4. Les logiques de description permettent de définir les bases logiques des différents formalismes de représentation de la connaissance tant sur le plan de la représentation que sur le raisonnement.

Dans notre travail, notre choix est porté sur le système RACER (Renamed Abox and Concept Expression Reasoner). Ce dernier se présente sous la forme d'un serveur qui peut être accédé par le protocole TCP ou HTTP. C'est un système de représentation de connaissances. Il implémente des tableaux de calcul hautement optimisé pour und DL très expressive. Il interprète les documents OWL et offre des services de raisonnement aussi bien pour le niveau terminologique que pour le niveau assertionnel de l'ontologie. Il permet aussi de vérifier la consistance et la relation de subsomption entre les concepts et d'autres tests plus élaborés sur les instances et les rôles.

Nous avons testé l'ontologie MASO par le biais du raisonneur RACER, et les résultats ont été très concluants. Nous distinguons deux types de test : la consistance c'est le test de satisfiabilité et de cohérence, il permet de s'assurer qu'aucune définition d'une classe est contradictoire avec une autre (l'inexistence des classes contradictoires). La classification permet de vérifier si une classe est une sous classe d'une autre classe ou non.

La vérification de la consistance et la classification de l'ontologie sont réalisées automatiquement à l'aide de Racer dans l'environnement Protégé 4. Comme montre la figure 7.6. A partir du résultat obtenu par ce test nous remarquons qu'aucune suggestion n'est produite par le raisonneur et que « Class hierarchy » et « Class hierarchy inferred » sont identiques.

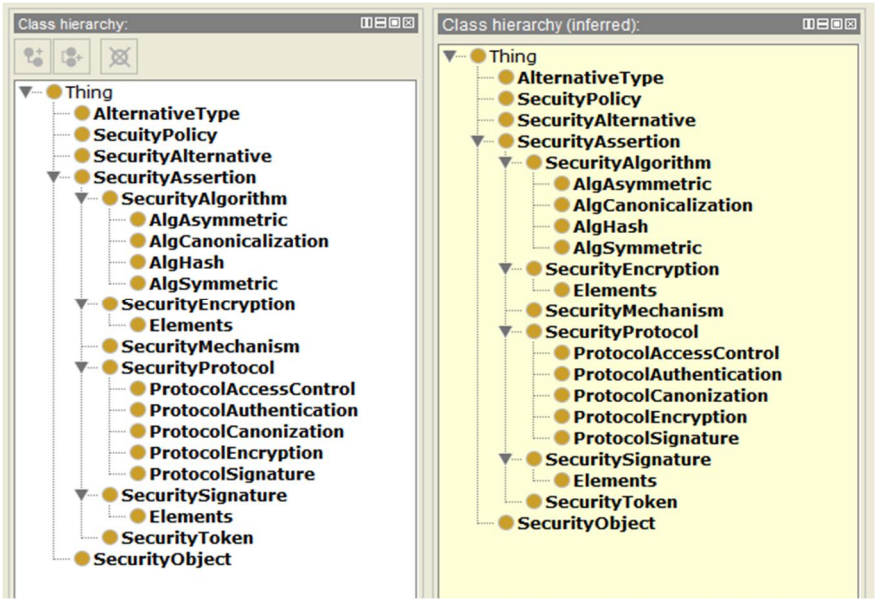

Figure 7. 6. Vérification de la consistance et classification de l'ontologie MASO

7.4.2. Algorithme de correspondance sémantique (Matching-Algorithm)

7.4.2.1. Langage de programmation

Nous avons choisi JAVA pour implémenter les agents mobiles et l'algorithme de correspondance sémantique (Matching-algorithm). C'est un langage multi plates-formes qui possède de nombreuses caractéristiques qui en font un des langages de choix pour la programmation d'un code mobile. En effet, JAVA a été conçu pour mettre en œuvre des applications susceptibles de s'exécuter sur n'importe qu'elle plate-forme. C'est un langage fortement typé, donc très sûr. Il offre en même temps une souplesse relative grâce aux opérations de coercition de type (casting). En plus du typage fort, Java met à notre disposition plusieurs niveaux de protection des données, ainsi qu'un mécanisme de traitement des exceptions très sophistiqué.

JAVA est extensible sans aucune limitation car pour étendre le langage il suffit de définir de nouvelles classes. De plus, tous les composants écrits pour traiter un problème particulier peuvent être ajoutés au langage et utilisés pour résoudre des nouveaux problèmes comme s'il s'agissait d'objets standard.

Contrairement à de nombreux compilateurs, celui de JAVA traduit le code source dans le langage d'une machine virtuelle (JVM). Ce code produit appelé bytecode est confié à un interpréteur qui le lit et l'exécute. C'est le principe capital qui permet à l'agent mobile de s'adapter au sein de tous les environnements d'exécution en se basant, dans la tâche d'implémentation, sur un langage interprété tel que JAVA. De plus, les agents peuvent migrer entre des plates-formes hétérogènes, parce que c'est JAVA qui accorde la mobilité du code et des objets grâce aux primitives et

méthodes déjà définies dans les diverses classes et au mécanisme de sérialisation qui capture et restaure l'état des objets avant et après le déplacement entre sites.

L'argument le plus important qui nous a poussé à choisir ce langage est qu'il a été développé dans un souci de sécurité maximal. L'idée maîtresse et qu'un programme comportant des erreurs ne doit pas pouvoir être compilé. Ainsi, les erreurs ne risquent pas d'échapper au programmeur. De même, en détectant les erreurs à la source, on évite qu'elles se propagent en s'amplifiant. JAVA permet la gestion dynamique de tableaux, c'est-à-dire le changement de leur taille au cours de l'exécution du programme. De plus, Java permet aussi la possibilité d'extension à travers l'usage de Class-Loader. Ceci signifie que Java peut charger dynamiquement les classes et les utiliser à l'exécution. En effet, le programme Java n'a pas besoin de connaître toutes les classes au moment de la compilation. Cette possibilité est très utile pour le chargement et l'exécution de certain composant de l'agent mobile. Par ailleurs, Java possède une bibliothèque riche qui comporte de nombreuses bibliothèques de sécurité qui assure l'authentification, le cryptage et la signature des composants fonctionnels et les données de l'agent mobile, parmi ces bibliothèques nous mentionnons javax.security, javax.CryptoPackage ou javax.crypto.

7.4.2.2. Implémentation de Matching-algorithm

L'environnement de développement intégré (EDI) que nous devons utiliser doit être extensible, universel, polyvalent, libre et compatible à la plate-forme JADE choisie. Notre choix s'est porté sur ECLIPSE parce qu'il répond à tous les critères énumérés. Eclipse est un environnement de développement intégré libre (open source) lancé par IBM. Il est extensible, universel et polyvalent, permettant potentiellement de créer des projets de développement. La spécificité d'Eclipse vient du fait de son architecture totalement développée autour de la notion de plug-in : toutes les fonctionnalités de cet atelier logiciel sont développées en tant que plug-in.

Pour implémenter l'algorithme de correspondance sémantique, nous avons utilisé deux API pour exploiter et manipuler l'ontologie MASO et les politique de sécurité d'agent mobile et les plates-formes :

- L'API Jena pour manipuler l'ontologie MASO. Cette API fournit l'interface du niveau basique pour des fichiers RDF, RDFS et OWL. Elle est une logicielle libre développée par le laboratoire de recherche de HP à Bristol. Jena offre un ensemble des parseurs et de moteurs de requête SPARQL sous forme de classes de Java.

- L'API JDOM pour manipuler les politiques de sécurité de l'agent et les plates-formes. Cette API permet d'analyser les politiques de sécurité, extraire ses différents concepts de sécurité et représenter ces concepts sous forme d'arbre.

Dans ce qui suit, nous présentons le diagramme de classes de l'algorithme de correspondance sémantique :

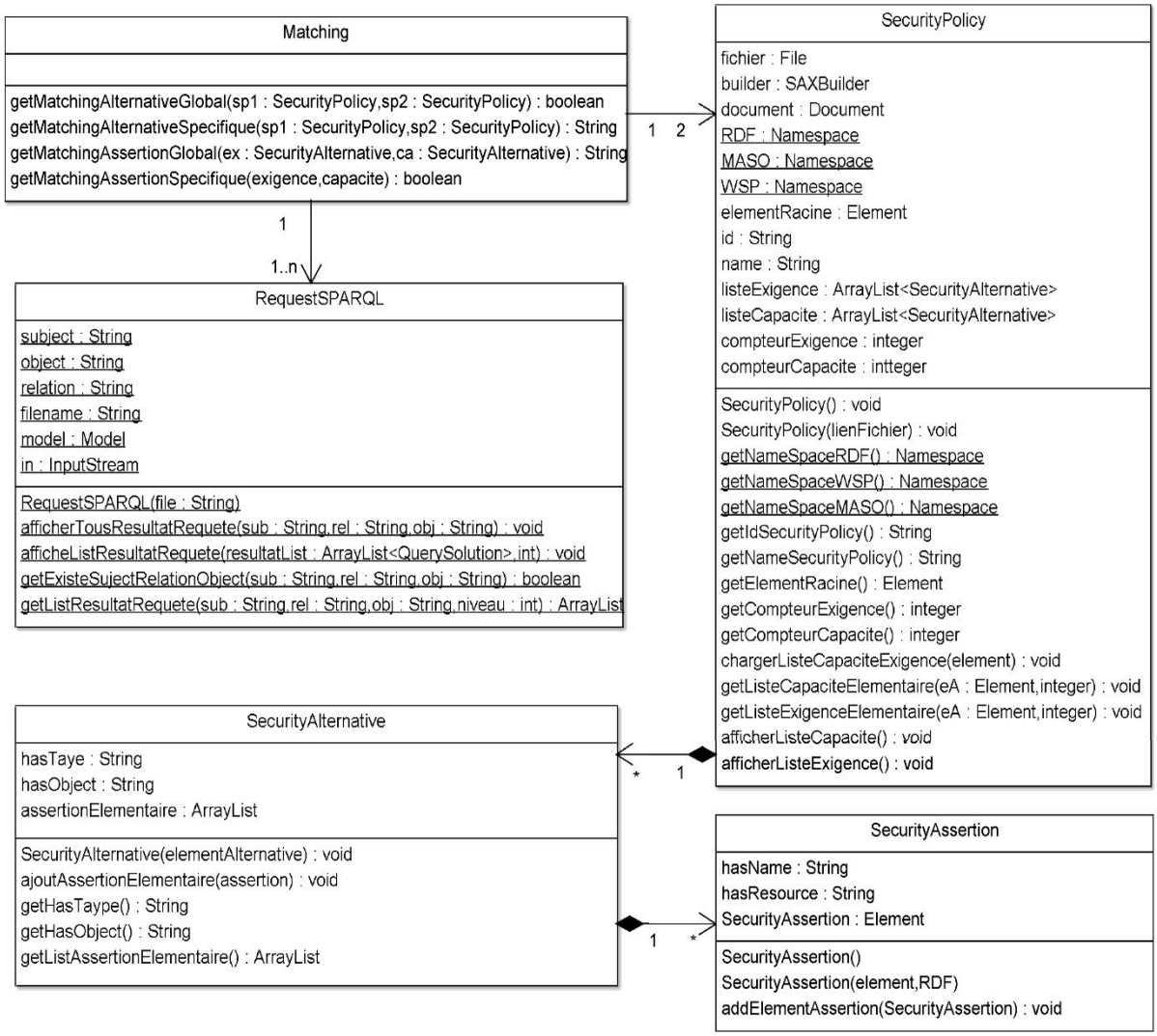

Figure 7. 7. Diagramme de classes de l'algorithme de correspondance sémantique

- La classe RequestSPARQL permet d'interroger l'ontologie MASO avec le langage de requêtes SPARQL pour récupérer les différentes relations entre une exigence et une capacité de sécurité. Les exigences et les capacités sont récupérées à partir des politiques de sécurité (agent et plate-forme) avec la classe SecurityPolicy.
- La classe Matching permet de vérifier la correspondance sémantique entre deux politiques de sécurité (SecurityPolicy). Il utilise deux type de

vérification. La première, il vérifie que chaque exigence dans une politique à une capacité de sécurité dans l'autre politique assurent le même objectif de sécurité. La deuxième, il vérifie la correspondance individuelle entre chaque paire exigence-capacité de sécurité, puis la correspondance globale entre les deux politiques.

- La classe SecurityPolicy permet d'analyser la politique de sécurité d'agent et la plate-forme. Il récupère la liste des exigences et des capacités de sécurité pour chaque politique, puis les assertions de sécurité les plus spécifiques pour chaque exigence/capacité.
- La classe SecurityAlternative permet manipuler les alternatives de sécurité, il récupère le type, l'objectif et la liste des assertions de sécurité utilisées par une alternative.
- La classe SecurityAssertion permet de manipuler les assertions de sécurité de chaque alternative de sécurité.

L'algorithme de correspondance accepte deux politiques de sécurité (agent et plate-forme), ensuite il donne le résultat de correspondance globale entre ces deux politiques. La figure 7.8 présente le résultat de correspondance sémantique entre les deux politiques de sécurité, le résultat est General-Match.

```
<arrêté> Program [Application Java] C:\Program Files (x86)\Java\jre6\bin\javaw.exe (accès direct par 5 juin 2017 15:06:43)
log4j:WARN No appenders could be found for logger (com.hp.hpl.jena.util.FileManager).
log4j:WARN Please initialize the log4j system properly.
log4j:WARN See http://logging.apache.org/log4j/1.2/faq.html#noconfig for more info.
______________________________Liste des exigences et des capacités de l'agent mobile______________________________
        Le nombre d'alternative de sécurité de type exigence est = 1
                Exigence de type = MASO:Requirement Object = MASO:Confidentiality
                        Exigence = ensuredByAlgorithm RDF:resource : MASO:RSA
        Le nombre d'alternative de sécurité de type capacite est = 1
                Capacité de type = MASO:Capability Object = MASO:Authentication
                        Capacité = ensuredByProtocol RDF:resourceMASO:X509
______________________________Liste des exigences et des capacités de la plate-forme______________________________
        Le nombre d'alternative de sécurité de type exigence est = 1
                Exigence de type = MASO:Requirement Object = MASO:Authentication
                        Exigence = ensuredByProtocol RDF:resource : MASO:X509
        Le nombre d'alternative de sécurité de type capacite est = 1
                Capacité de type = MASO:Capability Object = MASO:Confidentiality
                        Capacité = encryptedElement RDF:resourceMASO:Data
                        Capacité = usesToken RDF:resourceMASO:X509Certificate
Matchiiiiiiiiiiiiiiiiiiiiiiiiiiiiiiiiiiiiiiiiiiiiiiiiiiiiiiiiiiiiiiing....
La correspondance globale Exigence<-->Capacité entre les politiques est testée avec succée
                        ....Vérifier que les exigences de l'agent sont satisfaites par les capacités de la plate-forme...
        Matching Exigence : MASO:Requirement Object : MASO:Confidentiality <--> Capacite : MASO:Capability Object : MASO:Confidentiality est ...: General-Match
                        ....Vérifier que les exigences de la plate-forme sont satisfaites par les capacités de l'agent...
        Matching Exigence : MASO:Requirement Object : MASO:Authentication <--> Capacite : MASO:Capability Object : MASO:Authentication est ...: Perfect-Match
Résultat Algorithm-Matching ...:General-Match
```

Figure 7. 8. Résultat de correspondance sémantique entre les deux politiques de sécurité

7.4.3. Plate-forme JADE

Pour développer des agents, il est pratique de choisir une plate-forme. A l'heure actuelle, plusieurs outils ont été développés pour la réalisation d'agents. Si ces plates-formes abondent en quantité, elles ne répondent pas toutes aux critères que nous nous sommes fixés.

Les critères de choix d'une plate-forme agent sont naturellement liés avec les besoins de notre approche de protection et ses caractéristiques. Ainsi, la plate-forme devra répondre aux contraintes suivantes :

- La migration de l'agent doit être souple.
- La plate-forme doit répondre à plusieurs fonctionnalités et offrir une large gamme de bibliothèques.
- Le langage de programmation sous-jacent doit fournir des fonctionnalités de sécurité assez suffisantes telles que les fonctions cryptographiques, de signature et de hachage.
- Une plate-forme libre.

JADE étant la plate-forme la plus proche de nos critères, nous l'avons utilisée pour implémenter l'architecture de l'agent mobile. C'est une plate-forme d'agents, développée par Gruppo Telecom Italia [JADE, 2010]. Elle a pour but de simplifier le développement des systèmes multi-agents tout en fournissant un ensemble complet de services et d'agents conformes aux spécifications FIPA, dans un environnement Java. C'est pour ces raisons que notre choix s'est porté sur cette plate-forme. Elle peut être répartie sur plusieurs serveurs. Une application Java utilise une seule machine virtuelle de Java (JVM), est exécutée sur chaque serveur. Chaque JVM est un conteneur d'agents qui fournit un environnement complet pour l'exécution d'un agent et permet à plusieurs agents de s'exécuter en parallèle sur le même serveur. La figue 7.9 monte le conteneur principal de la plate-forme JADE.

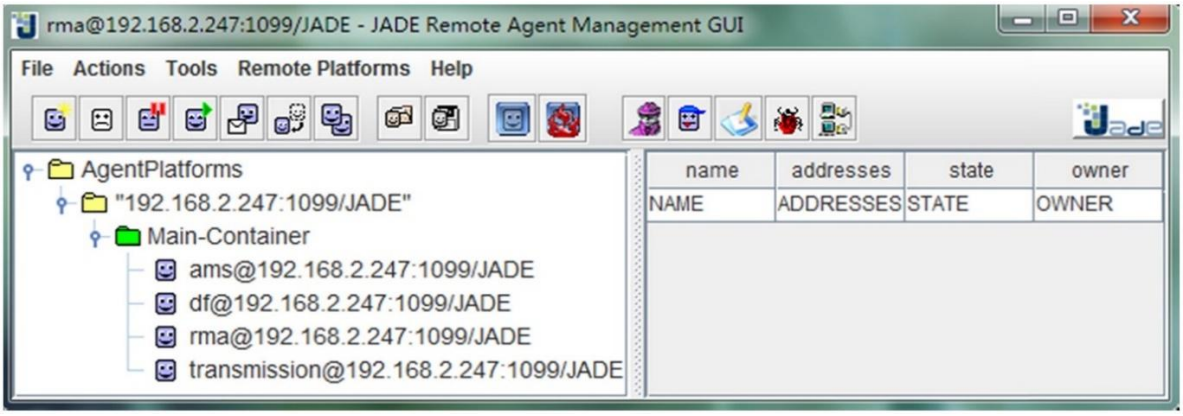

Figure 7. 9. Conteneur principal de la plate-forme JADE

Dans notre approche, les différentes tâches d'un agent doivent obéir à certaines règles et doivent être écrite sous une forme compréhensible par la plate-forme utilisée. Pour qu'un agent JADE exécute une tâche, nous avons tout d'abord besoin de définir ces tâches. Chaque tâche est représentée par un composant et implémentée par un behaviour dans JADE (une instance de la classe

jade.core.behaviours). Pour qu'un agent exécute une tâche on doit lui l'attribuer par la méthode addBehaviour(Behaviour b) de la classe jade.core.Agent.

7.4.4. Technologies de sécurité intégrées

Tous les mécanismes de sécurité proposés par notre système sont basés sur des technologies de sécurité avancées. Nous avons intégré dans notre approche deux mécanises d'authentification robuste et nous avons adapté un ensemble de spécifications, protocoles et d'algorithmes pour protéger le code et les données des agents mobiles. Par exemple, le modèle XACML/ABAC pour créer les politiques de contrôle d'accès aux données d'agent, les spécifications XML-Encryption et XML-Signature pour crypter et signer les données dynamiques des agents et le protocole SAML pour échanger les informations d'authentification et d'autorisation. L'utilisation de ces technologies éprouvées permet de renforcer la sécurité de notre système. Nous discutons dans cette section, les performances de ces mécanismes de protection :

Authentification mutuelle : tout modèle de sécurité fiable doit aussi garantir le bon fonctionnement du concept d'authentification mutuelle afin d'établir une confiance entre les parties communicantes. La notion de confiance s'établit après la phase d'authentification. Toute entité autorisée aura confiance pour accéder aux services disponibles en fonction de ses droits d'accès, au contraire aux entités qui ne sont pas autorisées n'auront pas l'accès. Pour cela, nous avons utilisé une authentification mutuelle par certificat entre les différentes entités du système (SEA, SGS, SAS, PAP, PDP…). Cette authentification permet à l'entité qui offre les services de s'assurer que les entités qui utilisent leurs services ne sont pas les imposteurs, et permettent aux utilisateurs de services de s'assurer qu'ils utilisent les services de l'entité auxquels ils font confiance avant de commencer toute communication. Avec ce mécanisme de sécurité on peut mettre des relations de confiance entre les différentes entités du système, ainsi la protection de l'interaction et l'échange de données entre ces entités.

Authentification unique SSO : nous avons intégré dans notre système le mécanisme d'authentification unique SSO (Single Sign-On) pour authentifier les agents mobiles durant son exécution. Le serveur d'authentification authentifie une seul fois les agents mobiles, puis il délivre des tickets d'authentification aux agents. Ces tickets sont le seul moyen technique fiable pour que l'agent mobile soit reconnu lors de son exécution. Avec ce mécanisme de sécurité on peut empêcher le vol de l'identité d'agents.

Modélisation des politiques de sécurité : le standard WS-Policy ne permet pas de résoudre le problème d'interopérabilité sémantique entre les politiques de sécurité hétérogènes. Pour cela, nous avons développé une ontologie MASO qui permet de définir les principaux concepts de sécurité utilisés dans notre système. L'utilisation de l'ontologie permet de faciliter l'analyse automatique de la compatibilité sémantique entre les politiques de sécurité d'agent et les plates-formes visitées. La modélisation de ces politiques est réalisée par le standard WS-Policy avec des annotations sémantiques de l'ontologie MASO.

Politiques de contrôle d'accès aux données d'agents mobiles : le modèle ABAC et sa mise en œuvre avec le langage standardisé XACML permet de protéger les données dynamiques des agents mobiles. Le modèle ABAC offre un contrôle d'accès flexible et adaptable pour notre système. Avec ce modèle on peut évaluer les demandes d'accès aux données d'agents à travers des politiques contenant trois catégories d'attributs (*Subjects, Resources, Actions*). Ensuite, nous avons adapté le langage XACML pour créer ces politiques en se basant sur les règles définies par le SAS. Ces mécanises sont considérés comme efficace pour gérer les droits d'accès d'une manière distribuée et sécurisée.

7.4.5. Protection d'un agent mobile contre les différents types d'attaque

Dans cette section, nous allons discuter la fiabilité, la performance et la robustesse de notre protection vis-à-vis des attaques présentées antérieurement. Les attaques seront présentées sous forme de scénarii, afin de refléter le raisonnement tenu par un pirate et de discuter chaque attaque pas à pas.

Scénario 1 : Vol de l'identité de l'agent mobile

Un agent mobile malveillant peut voler l'identité et le ticket de l'agent légitime. Dans ce cas, l'agent malveillant peut avoir pour but d'obtenir des ressources auxquelles l'attaquant n'a normalement pas accès ou de faire endosser la responsabilité de certaines actions à un autre agent.

Discussion 1

Nous avons utilisé trois modes d'authentification pour chaque agent mobile : l'agent mobile est signé par son créateur. Dans ce cas, les plates-formes peuvent vérifier cette signature et identifier l'origine de l'agent. Les données statiques de l'agent sont signées par le propriétaire d'agent. Pour cela, les plates-formes peuvent authentifier l'entité qui utilise l'agent. Enfin, nous avons utilisé un serveur d'authentification pour authentifier une seule fois les agents mobiles (voir la section 7.2.2). En effet, chaque agent dispose d'un ticket délivré par le serveur

d'authentification contenant l'identité de l'agent, la méthode d'authentification, le temps d'expiration du ticket, le condensé de l'agent, la signature du créateur d'agent et le serveur d'authentification. Avec cette méthode, il est impossible de voler l'identité ou le ticket de l'agent. Notons aussi que les données sensibles d'authentification sont stockées de manière sécurisée dans le serveur d'authentification. L'agent utilise seulement le ticket pour s'authentifier durant son exécution.

Scénario 2 : Espionnage de code d'agent mobile

Lorsque l'agent mobile est envoyé vers les plates-formes cibles, il passe par plusieurs plates-formes avant d'atteindre sa cible. Certaines plates-formes sur le chemin, sont équipés d'analyseurs de paquets qui leurs permettent d'intercepter des objets spécifiques. Notre agent mobile peut en être victime. La plate-forme malicieuse va analyser le code de l'agent pour en tirer quelques informations sur le comportement.

Discussion 2

Nous avons utilisé la notion de composants logiciels pour structurer le comportement de l'agent en plusieurs composants fonctionnels. Pour assurer la confidentialité des composants sensibles, le créateur d'agent exige dans sa politique de sécurité le cryptage de ces derniers. Dans la phase d'adaptation, le SAS crypte les différents composants sensibles avec des algorithmes spécifiés dans la politique de sécurité d'agent. Avec ce mécanisme de sécurité seule les plates-formes autorisées peuvent décrypter et exécuter les composants sensibles. Ce cryptage permet de contrecarrer l'analyse pour que l'agent mobile puisse passer inaperçu à travers les plates-formes et éviter une attaque sérieuse. L'analyse statique est donc non efficace contre cette protection.

Scénario 3 : Altération du code ou des données d'agent mobile

Quand un agent arrive sur une plate-forme d'exécution, il expose son code et ses données. Cette plate-forme contrôle tous les éléments de l'agent mobile, et il aura l'habilité de les modifier. En effet, elle peut modifier les données statiques, les composants sensibles ou bien les résultats obtenus par l'agent à l'intérieur des plates-formes précédentes. Le propriétaire d'agent ne veut pas qu'une plate-forme non fiable puisse modifier l'agent ou les résultats qu'il a obtenus sur d'autres plates-formes.

Discussion 3

Le modèle d'agent que nous avons proposé contient trois éléments fondamentaux : l'interface, le niveau de base et le méta-niveau sont signés par la clé publique du créateur d'agent. Cette signature permet de garantir l'intégrité de ces éléments. La mémoire d'agent contient deux types de données : les données statiques et dynamiques. Les données statiques sont ajoutées et signées par la clé publique du propriétaire d'agent, ce qui permet d'assurer l'intégrité de ces données et authentifier l'entité qui utilise l'agent. Les données dynamiques sont collectées par l'agent durant son exécution. L'agent peut exiger dans sa politique de sécurité la signature des données ajoutées par les plates-formes. Dans ce cas, il est impossible de modifier ces données par des plates-formes malveillantes.

Scénario 4 : Accès non autorisé aux données d'agent

Un agent mobile se déplace sur plusieurs plates-formes, dans des domaines de sécurité différents. La lecture des données statiques ou dynamiques de cet agent par des plates-formes malveillantes est très critique vu que cette attaque ne laisse pas de trace contrairement à l'attaque de modification des données.

Discussion 4

Nous avons utilisé des politiques de contrôle d'accès XACML/ABAC pour protéger l'accès aux données d'agent mobile. Dans la phase d'adaptation, le SAS estime les niveaux de sécurité des composants et les degrés de confiance des plates-formes visitées. Ensuite, il utilise les contraintes de contrôle d'accès pour créer les règles de contrôle d'accès aux données d'agent. Ces règles sont transformées ensuite aux politiques XACML gérer par les entités PAP et PDP (La section 7.3.2 montre comment contrôler l'accès aux données d'agent mobile). Ces politiques gérer l'accès aux données entre les plates-formes d'exécution.

Afin d'évaluer la performance de notre approche, nous avons dressé le tableau 7.2 qui montre les mécanismes de sécurité utilisés contre les différents types d'attaque.

Les menaces	Réaliser	Mécanise de sécurité	Les entités participant
Mascarade entre les entités du système	Oui	L'authentification mutuelle par certificat entre les différentes entités du système (SEA, SGS, SAS, PAP, PDP…).	LCA permet de gérer les certificats et de vérifier la validité de ces certificats.
Mascarade (vol d'identité)	Oui	- La signature du créateur d'agent. - La signature du propriétaire d'agent. - Les tickets délivrés par le serveur d'authentification.	- Le créateur d'agent. - Le propriétaire d'agent. - Le serveur d'authentification.
Altération du code	Oui	Le code de l'agent est signé par deux entités : - Le créateur d'agent pour assurer l'intégrité de son code. - Le système SGSA assure aussi l'intégrité du code et mettre une relation de confiance entre l'agent el les plates-formes d'exécution.	- Le créateur d'agent. - Le système SGSA.
Espionnage du code	Oui	Le système SAS utilise la politique de sécurité de l'agent pour protéger la confidentialité de ses composants sensibles. On distingue trois mécanismes de protection : - Les composants sont cryptés par une clé symétrique, la clé est partagée entre les plates-formes d'exécution. - Les composants sont cryptés par une clé symétrique. Ensuite, la clé est enveloppée par les clés publiques des plates-formes d'exécution. - Les composants sont cryptés par la clé publique de la plate-forme d'exécution.	- Le créateur d'agent. - Le système SAS. - L'autorité de certification locale (LCA)

Déni de service	Oui	Le créateur d'agent associe à un certain composant une estimation du temps d'exécution. Cette estimation est utilisée par le contrôleur d'agent pour le comparer avec le temps d'exécution réel de ce composant dans une plate-forme. Si le contrôleur détecte un dépassement du temps réglementaire, il arrête l'exécution de ce composant sur cette plate-forme pour migrer vers la plate-forme suivante.	- Le créateur d'agent.
Contrôle d'accès au code	Oui	Le contrôle d'accès au code est assuré par le cryptage. Les composants sensibles sont cryptés par des clés symétriques ou asymétriques. Quelle que soit l'algorithme de cryptage, seules les plates-formes autorisées peuvent décrypter et exécuter ces composants.	- Le créateur d'agent. - Le système SAS. - L'autorité de certification locale (LCA)
Espionnage des données statiques	Oui	Le propriétaire d'agent initialise l'agent avec des données statiques. Les informations confidentielles sont cryptées par le propriétaire d'agent.	- Le propriétaire d'agent
Altération des données statiques	Oui	Les données statiques sont signées par le propriétaire d'agent. Dans ce cas, il impossible de modifier les données statiques par les plates-formes malveillantes.	- Le propriétaire d'agent

Espionnage des données dynamiques	Oui	La politique de sécurité d'agent exige le cryptage des données collectées par un composant durant son exécution. Dans ce cas, la plate-forme d'exécution envoie une demande de clé symétrique au KDC. Si la plate-forme est autorisée d'ajouter les données cryptées à l'agent, le KDC génère une nouvelle clé spécifique pour cette plate-forme. La plate-forme utilise cette clé pour crypter ses données.	- Le serveur KDC - Le serveur PDP - Le serveur PAP - L'autorité de certification locale (LCA)
Altération des données dynamiques	Oui	L'agent mobile exige dans sa politique de sécurité la signature des données produites par certain composant. Dans ce cas, les plates-formes utilisent leurs certificats à clé publique pour signer ces données.	- Les plates-formes d'exécution - L'autorité de certification locale (LCA)
Contrôle d'accès aux données	Oui	L'agent mobile migre d'une plate-forme à une autre avec des données cryptées. Pour gérer le contrôle d'accès à ces données entre les plates-formes d'exécution, nous avons créé des politiques de contrôle d'accès XACML/ABAC. Ces politiques sont basées sur les règles définies par le SAS au moment de l'adaptation d'agent. Avec ce mécanisme de sécurité seules les plates-formes autorisées par le PDP peuvent récupérer la clé symétrique à partir de KDC pour accéder à ces données cryptées.	- Le serveur KDC - Le serveur PDP - Le serveur PAP - L'autorité de certification locale (LCA)

Tableau 7. 2. Mécanismes de sécurité contre les différents types d'attaques

7.5. Conclusion

Dans ce chapitre, nous avons présenté un modèle de protection complet du code et des données des agents mobiles. Ce modèle est basé sur des technologies de sécurité éprouvées. Dans la première partie, nous avons présenté comment le propriétaire d'agent initialise et authentifie ses agents. Initialiser l'agent est le fait d'ajouter et signer les données statiques dans l'agent mobile. Ensuite, le propriétaire d'agent utilise un serveur d'authentification pour authentifier une seule fois ses agents.

La deuxième partie présente la phase d'exécution des agents mobiles. Cette phase montre comment les agents mobiles sont-ils authentifiés et validés par les plates-formes, et comment ces plates-formes décryptent les composants sensibles à exécuter. Ensuite, il décrit les étapes pour protéger les données dynamiques de l'agent mobile.

Dans la troisième partie, nous avons présenté les performances de certaines technologies de sécurité existantes qu'on a intégrées dans notre système. La discussion de la fiabilité, la performance et la robustesse de notre protection vis-à-vis des différents types d'attaques clore ce chapitre.

Conclusion générale et perspectives

Le besoin de sécurité s'est considérablement accru avec l'émergence du modèle d'agent mobile. Ainsi, dans le cadre de cette thèse, notre vision de la mobilité a mis l'accent non pas sur les problèmes de migration des agents en cours d'exécution, mais sur les problèmes liés à la sécurité des agents mobiles et des systèmes d'accueil d'agents. Dans notre travail, nous avons étudié les approches liées à la sécurité des systèmes à base d'agents mobiles pour exhiber leurs limites en regard de leurs avantages. Cette étude a permis d'évaluer l'importance de l'utilisation des agents mobiles dans les applications distribuées et de mesurer l'intérêt de résoudre le problème de sécurité qu'ils ont engendré pour pouvoir pleinement profiter de leurs atouts. De même, elle a montré l'inexistence de solution de protection universelle au problème de la plate-forme malicieuse.

Afin de lever la complexité liée à la sécurité des systèmes d'agents mobiles, nous avons proposé un nouveau modèle d'agent mobile à base de composants. Ce modèle permet au concepteur d'agents de spécifier à travers des concepts de haut niveau les composants fonctionnels, la politique de sécurité, la mobilité, la description du niveau de base et les données d'un agent mobile. Avec cette nouvelle structure, on a facilité l'adaptation et la protection de cet agent. Ensuite, pour gérer les services offerts par les plates-formes, nous avons modélisé ces services par des agents de services. Un nouveau registre UDDI a été utilisé pour publier les politiques de sécurité et les descriptions fonctionnelles de ces services.

Les politiques de sécurité ont été modélisées pour exprimer les exigences et les capacités de sécurité de chaque entité dans notre système. Les exigences de sécurité permettent de spécifier les différents paramètres de sécurité nécessaire à l'exécution sécurisée d'un agent mobile. Les capacités de sécurité représentent un ensemble de spécifications, de protocoles, d'algorithmes, etc. pour satisfaire une exigence de sécurité. Pour éviter la différence sémantique entre les politiques de sécurité, nous avons proposé une ontologie dans le domaine de sécurité des agents mobiles pour faciliter l'analyse automatique de la compatibilité sémantique entre ces politiques.

Nous avons développé un système d'adaptation complet pour transformer l'agent à un agent mobile sécurisé de confiance. Ce système est appelé SGSA (Système de Gestion de la Sécurité d'Agents) repose sur trois sous-systèmes. Le système

d'évaluation d'agents (SEA) qui permet d'évaluer et de valider l'identité, l'intégrité et l'assemblage des composants fonctionnels de l'agent. Le système de gestion de sécurité (SGS) permettant la découverte et la sélection des plates-formes d'exécution pour créer l'itinéraire de l'agent. Ce système est basé sur l'algorithme « *Matching-Algorithm* » qui permet d'analyser la correspondance sémantique entre les politiques de sécurité (plate-forme/agent). Enfin, le système d'adaptation de sécurité (SAS) permettant d'estimer le niveau de sécurité des composants fonctionnels de l'agent et le degré de confiance des plates-formes d'exécution. Ces deux estimations sont utilisées plus tard pour créer les règles de contrôle d'accès aux données d'agent. Ce système utilise ensuite la politique de sécurité d'agent pour sécuriser les composants sensibles de cet agent.

Le modèle de contrôle d'accès ABAC et le langage XACML ont été adaptés pour créer les politiques de contrôle d'accès aux données d'agent. Ces politiques sont basées sur les règles définies par le SAS au moment de l'adaptation de l'agent. Aucune approche, à notre connaissance, n'a utilisé les politiques de contrôle d'accès ABAC/XACML pour protéger les données dynamiques de l'agent mobile.

Le résultat de ces travaux de thèse constitue un référentiel dans les différentes phases de développement des systèmes à base d'agents mobiles. Une suite logique de notre travail consiste à développer, sur la base de ce référentiel, une démarche de développement des systèmes à base d'agents mobiles sécurisés.

Bibliographie

[Alfalayleh et al., 2004]
M. Alfalayleh and L. Brankovic, “An Overview Of Security Issues And Techniques In Mobile Agents". Conference on Communications and Multimedia Security. University of Newcastle, Australia, 2004.

[Al-Jaljouli et al., 2007]
R. Al-Jaljouli and J. Abawajy, ”Secure Mobile Agent-based E-Negotiation for On-Line Trading”. IEEE International Symposium on Signal Processing and Information Technology, pages 610–615, 2007.

[All'ee et al., 2005]
G. All´ee, S. Pierre, R. Glitho and A. El Rhazi, ”An improved itinerary recording protocol for securing distributed architectures based on mobile agents”. Mobile Information Systems, 1(2), pages 129–147, 2005.

[Appel, 2001]
A. Appel, “Foundational proof-carrying code”. In Proceedings of the 16th Annual Symposium on Logic in Computer Science. IEEE Computer Society Press, pages 247-256, 2001.

[Arcangeli et al., 2001]
J. Arcangeli, C.Maurel and F. Migeon, “An API for highlevel software engineering of distributed and mobile applications”. In FTDCS’01 : Proceedings of the 8th IEEE Workshop on Future Trends of Distributed Computing Systems, IEEE Computer Society, Washington, USA, page 155, 2001.

[Arcangeli et al., 2001]
J. Arcangeli, C. Maurel and F. Migeon, "An API for High Level Software Engineering of Distributed and Mobil Applications”. In Proceedings of the 8th IEEE Workshop on Future Trends of Distributed Computing Systems, FTDCS ’01, 2001.

[Barak et al., 2001]
B. Barak, O. Goldreich, R. Impagliazzo, S. Rudich, A. Sahai, S. Vadhan and K. Yang, “On the (Im)possibility of Obfuscating Programs”. Advances in Cryptology, Proceedings of Crypto'01, Lecture Notes in Computer Science, Vol. 2139, pages 1-18, 2001.

[Baumer et al., 1999]
C. Baumer, M. Breugst, S. Choy and T. Magedanz, "Grasshopper – A universal agent platform based on OMG MASIF and FIPA standards". In Proceedings of the First International Workshop on Mobile Agents for Telecommunication Applications (MATA’99), pages 1-18, Ottawa, Canada, October 1999.

[Beimel et al., 2000]
A. Beimel and M. Burmester, “Computing Functions of a Shared Secret”. SIAM J. Discrete Math., Vol. 13, No.3, pages 324-345, 2000.

[Belkhelladi, 2010]
K. Belkhelladi, "Stratégies d’échange d’informations dans un système de calcul distribué pour l’optimisation des problèmes combinatoires". Thèse de doctorat, Université d’Angers, 2010.

[Bella vista et al., 2004]
P. Bella vista, A. Corradi, C. Frederici, R. Montanari and D. Tibaldi, “Security for Mobile Agents: Issues and Challenges”. Invited Chapter in the Book Handbook of Mobile Computing, I. Mahgoub, M. Ilyas(eds.), CRC Press, 2004.

[Bellifemine et al., 1999]
F. Bellifemine, A. Poggi and G. Rimassa, “JADE : A FIPA-compliant agent framework”. In Proceedings of 4th International Conference and Exhibition on the Practical Applications of Intelligent Agents and Multi-Agent Systems (PAAM ’99), pages 97–108, London, UK, April 1999.

[Bellifemine et al., 2007]
F. Bellifemine, G. Caire and D. Greenwood, ”Developing Multi-Agent Systems with JADE”. Chichester, England: John Wiley & Sons Ltd, 2007.

[Bettini et al., 2002]
L. Bettini, R. De Nicola and M. Loreti, “Formalizing properties of mobile agent systems”. in Coordination Models and Languages, pages 72–87, January 2002.

[Bierman et al., 2002]
E. Bierman and E. Cloete, “Classification of Malicious Host Threats in Mobile Agent Computing”. Proceedings of SACICSIT2002, pages 141-148, 2002.

[Bob, 2000]
T. Bob, N. Danko and F. Sterling, ”Introduction To Mobile Agent Systems and Applications”. Tools USA 2000 Mobile Agents. Department of Defense, 2000.

[Borselius, 2002]
N. Borseliu, “Mobile Agent Security”. Electronics Communication Engineering Journal, vol. 14, No 5, IEEE. London, pages 211-218, 2002.

[Brazier et al., 2002]
F.M. T. Brazier, J.M. Catholijn and T. Jan, “Principles of component-based design of intelligent agents”. Data and Knowledge Engineering, 41(1), pages 1-27, 2002.

[Briot et al., 2006]
J.P. Briot, T. Meurisse and F. Peschanski. “Une expérience de conception et de composition de comportements d’agents à l’aide de composants. L’OBJET, 11(3), 2006.

[Chhetri et al., 2006]
M. Chhetri, B. Price, R. Krishnaswamy and S.W. Loke, ”Ontology-based agent mobility modeling”. In Proceedings of the 39th Annual Hawaii International Conference on System Sciences (HICSS’06), Kauai, IEEE Computer Society, 2006.

[Contes,2005]
A. Contes, “Une architecture de sécurité hiérarchique, adaptable et dynamique pour la grille”. PhD thesis, Universite de Nice-Sophia Antipolis, Faculté des Sciences, September 2005.

[D’Anna et al., 2003]
L. D'Anna, B. Matt, A. Reisse, T. Van-Vleck, S. Schwab and P. LeBlanc, “Self-Protecting Mobile Agents Obfuscation Report”. Report #03-015, Network Associates Laboratories Report, 2003.

[David, 2002]
P.C. David and T. Ledoux, "An Infrastructure for Adaptable Middleware ". DOA’02, Springer Verlag, LNCS 2519, Irvine, California, 2002.

[Demazeau, 2001]
Y.Demazeau, “VOYELLES: Rapport d’Habilitation à Dirige des Recherches”, Institut National Polytechnique de Grenoble, Laboratoire Leibniz, Avril 2001.

[El Falou et al., 2005]
S. El Falou and F. Bourdon, "A MDP Solution for Mobile Agents Displacement". In Proceedings of the 17th IEEE International Conference on Tools with Artificial Intelligence, ICTAI ’05, pages 29-33,2005.

[El Falou, 2006]
S. El Falou, "Programmation répartie, optimisation par agent mobile". Thèse de doctorat, Université de Caen Basse-Normandie, 2006.

[Eric et al., 2005]
Y. Eric and T. Jin, “Attribute Based Access Control, A New Access Control Approach for Service Oriented Architectures (SOA)”, New Challenges for Access Control Workshop, Ottawa, ON, Canada, 2005.

[Eclipse, 2016]
Eclipse documentation – Current Release, https://help.eclipse.org/oxygen/index.jsp, 2016.

[Farmer et al., 1996a]
W.M. Farmer, J.D. Guttman and V. Swarup, “Security for Mobile Agents: Authentication and State Appraisal”. Proceedings of the European Symposium on Research in Computer Security (ESORICS), pages 118-130, 1996.

[Filiol, 2005]
E. Filiol, ”Strong Cryptography Armored Computer Viruses Forbidding Code Analysis”. Proceedings of the 14th EICAR Conference, pages 216-227, 2005.

[FIPS, 1996]
National Institute of Standards and Technology: “Entity authentication using public key cryptography” [Specification], Federal Information Processing Standards Publication, US Department of Commerce, 1997, February 18, Gaithersburg, Maryland, web: http://csrc.nist.gov/publications/fips/.

[Foundation for Intelligent Physical Agents, 2011]
Foundation for Intelligent Physical Agents (2011). Online http://www.fipa.org/.

[Gamma et al., 1995]
E. Gamma, R. Helm, R. Johnson and J. Vlissides, ”Design patterns: elements of reusable object-oriented software”. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA, 1995.

[Garrigues et al., 2008]
C. Garrigues, S. Robles and J. Borrell, ”Securing dynamic itineraries for mobile agent applications”. Journal of Network and Computer Applications, 31 (4), pages 487-508, 2008.

[Gray et al., 2002]
R. S. Gray, G. Cybenko, D. Kotz, R.A. Peterson and D. Rus,”D’agents : applications and performance of a mobile agent system”. Software-Practice & Experience Special issue : Mobile agent systems, 32(6), pages 543–573, 2002.

[Greenberg et al., 1998]
M.S. Greenberg and J.C. Byington, “Mobile Agents and Security”, IEEE Communications, July 1998.

[Gutknecht et al., 2001]
O. Gutknecht and J. Ferber. “The madkit agent platform architecture”. In Revised Papers from the International Workshop on Infrastructure for Multi-Agent Systems, Springer-Verlag, pages 48-55, London, UK, 2001.

[Haarslev et al., 2012]
V. Haarslev, K. Hidde, R. Möller and M. Wessel. The RacerPro knowledge representation and reasoning system. Semantic Web Journal, 3(3):267–277, 2012.

[Hacini et al., 2007]
S. Hacini, Z. Guessoum and Z. Boufaïda, "TAMAP: A New Trust-based Approach for Mobile Agent Protection". Journal in computer virology. Springer Paris. ISSN 1772-9890 (print) 1772-9904 (online). Vol. 3, No. 4. November 2007, pages 267-283, 2007.

[Hacini et al., 2012]
S. Hacini and R. Lekhchine, “Security Ontology for Mobile Agents Protection”. International Journal of Computer Theory and Engineering, Vol. 4, No. 3, June 2012.

[Hadingham et al., 2000]
R. Hadingham, S. Poslad and P. Buckle, “The FIPA-OS agent platform : Open Source for Open Standards”. In Proceedings of the 5th International Conference and Exhibition on the Practical Application of Intelligent Agents and Multi-Agents, pages 355–368, Manchester, UK, April 2000.

[He, 1998]
Q. He, K.P. Sycara and T.W. Finin, “Personal security agent: KQML-Based PKI”. In K. P. Sycara and M. Wooldridge, editors, Proceedings of the 2nd International Conference on Autonomous Agents, pages 377-384, ACM Press, New York, USA, 1998.

[Hewitt, 1977]
C. Hewitt. “Viewing control structures as patterns of passing messages”. Artificial Intelligence, 8(3), pages 323-364, 1977.

[Hohl, 1998]
F. Hohl, “Time Limited Blackbox Security: Protecting Mobile Agents from Malicious Hosts”. G. Vigna (Ed.), Mobile Agents and Security. Lecture Notes in Computer Science, Vol. 1419, Springer-Verlag, pages 52-59, 1998.

[Horridge et al., 2011]
M. Horridge and S. Bechhofer, “The OWL API: A Java API for OWL ontologies”, Semantic Web, vol. 2, no 1, p. 11-21, 2011.

[Islam et al., 2010]
N. Islam, G.A. Mallah and Z.A. Shaikh, ”FIPA and MASIF Standards: A Comparative Study and Strategies for Integration”. In Proceeding NSEC '10 Proceedings of the 2010 National Software Engineering Conference, Article No. 7, ACM New York, NY, USA, 2010.

[JADE, 2010]
JADE programmer’s guide, http://jade.tilab.com/doc/programmersguide.pdf, update, 08 April 2010.

[Jansen et al., 2000a]
W. Jansen and T. Karygiannis, “Mobile Agent Security”. NIST Special Publication 800-19, National Institute of Standard and Technology, 2000.

[Jansen, 2000b]
W. Jansen, “Countermeasures for mobile Agent security”. Computer Communications, Special issue on Advance Security Techniques for Network Protection. Elsevier Science, 2000.

[Jansen, 2001a]
W. Jansen, "Determining Privileges of Mobile Agents". In Proceedings of the Computer Security Applications Conference, December 2001.

[Jansen, 2001b]
W. Jansen, "A Privilege Management Scheme for Mobile Agent Systems". First International Workshop on Security of Mobile Multi-agent Systems, Autonomous Agents Conference, May 2001.

[Jansen, 2001c]
W. Jansen, "Countermeasures for Mobile Agent Security". National Institute of Standards and Technology, Gaithersburg, MD 20899, USA, October 2001.

[Johansen, 1998]
D. Johansen, “Mobile agent applicability“. In Mobile Agents, pages 80-98, 1998.

[Kaffille et al., 2003]
S. Kaffille and G. Wirtz, “Integrating MASIF and FIPA Standards for Agent System Interoperability”. In Proceedings of the 7th International Conference on Software Engineering and Applications (SEA’03), pages 798-806, Marina del Rey, USA, 03-05 November 2003.

[Kalam, 2003]
A.A. Kalam, “Modèles et politiques de securité pour les domaines de la santé et des affaires socials”. PhD thesis, Institut National Polytechnique de Toulouse, Decembre 2003.

[Krutisch et al., 2003]
R. Krutisch, P. Meier and M. Wirsing, “The Agent Component Approach, Combining Agents and Components”. In Michael Schillo, Matthias Klusch, Jörg Müller, and Huaglory Tian field, editors, Multi-agent System Technologies, Lecture Notes in Computer Science 2831, pages 1-12. Springer-Verlag, Heidelberg, Germany, 2003.

[Kusek et al., 2005]
M. Kusek and G. Jezic, ”Modeling agent mobility with UML sequence diagram”. In Agent-Oriented Software Engineering TFG, AL3, février 2005.

[Lange et al., 1999]
D.B Lange and M. Oshima, "Seven Good Reasons for Mobile Agents”. In Communications of the ACM, Vol. 42, No. 3, mars 1999.

[Lapadula, 1976]
B.E. Lapadula, "secure computer systems: Unified Exposition and Multics interpretation". The MITRE Corporation, Technical report, 1976.

[Lee et al., 2004]
H. Lee, J. Alves-Foss and S. Harrison, ”The use of encrypted functions for mobile agent security”. In Proceedings of the Proceedings of the 37th Annual Hawaii International Conference on System Sciences, Vol. 9, pages 05-08, 2004.

[Leriche et al., 2006]
S. Leriche and J. Arcangeli, “Vers un modèle d’agent flexible”. In JMAC’06 : Journées Multi-Agent et Composant, École des Minesd’Alès, Nimes, France, pages 49-59, Mars 2006.

[Leriche et al., 2007]
S. Leriche and J. Arcangeli. ”Adaptive Autonomous Agent Models for Open Distributed Systems”. In International Multi-Conference on Computing in the Global Information Technology (ICCGI), Guadeloupe, pages 19–24, mars 2007.

[Loukil et al., 2006]
A. Loukil, H. Hachicha and K. Ghedira, ”A proposed approach to model and to implement mobile agents”. In International Journal of Computer Science and Network Security (IJCSNS), 6(3B), pages 125-129, mars 2006.

[Loulou, 2010]
M. Loulou, M.Jmaiel and M.Mosbah. “Dynamic security framework for mobile agent systems: specification, verification and enforcement”. International Journal of Information and Computer Security, 3(3/4), pages 321-336, 2009.

[Malenfant, 2004]
J. Malenfant, F. Peschanski, L. Seinturier and J-P. Briot, "ALADYN: Architectures logicielles pour l’auto-adaptabilité dynamique". Université Pierre et Marie Curie UFR d’informatique, 2004.

[Mana et al., 2002]
A. Mana and E. Pimentel, “An efficient software protection scheme”. University of Malaga, Spain, 2002.

[Marcoux et al., 2001]
A. Marcoux, C. Maurel, F. Migeon, and P. Sallé, “Generic operational decomposition for concurrent systems: semantics and reflection”. Progress in computer research, pages 225-242, 2001.

[McCann et al., 1998]
P. J. McCann and G. C. Roman, ”Compositional programming abstractions for mobile computing”. In IEEE Transactions on Software Engineering, 24(2), pages 97-110, 1998.

[Michel et al., 2010]
O. Michel and F. Brazier, “Security in Large-Scale Open Distributed Multi-Agent Systems”. Book chapter in Autonomous Agents, INTECH, 2010.

[Milojicic et al., 1998]
D. Milojicic, M. Breugst, I. Busse, J. Campbell, S. Covaci, B. Friedman, K. Kosaka, D. Lange, K. Ono, M. Oshima, C. Tham, S. Virdhagriswaran and J. White, “MASIF-The OMG mobile agent system interoperability facility”. Proceedings of the 2nd International Workshop of Mobile Agents (MA ’98) (Lecture Notes in Computer Science, Vol. 1477), K. Rothermel, F. Hohl. Springer-Stuttgart, pages 50–67, 1998.

[Muscutariu et al., 2001]
F. Muscutariu and M.P. Gervais. ”On the modeling of mobile agent based systems”. In Proceedings of the 3th International Workshop on Mobile Agents for Telecommunication Applications (MATA’01), vol. 2164, pages 219-233, Montreal, Canada, 2001.

[Musen, 2015]
Musen, M.A. The Protégé project: A look back and a look forward. AI Matters. Association of Computing Machinery Specific Interest Group in Artificial Intelligence, 1(4), June 2015. DOI: 10.1145/2557001.25757003.

[Necula et al., 1998]
G.C. Necula and P. Lee. “Safe, Entrusted Agent using Proof-Carrying Code”. In Mobile Agent Security, ed. Par Vigna (Giovanni), pages 61-91, Lecture Notes in Computer Science, 1998.

[Nwana et al., 1999]
H.S. Nwana, D.T. Ndumu, L.C. Lee and J.C. Collis Zeus, “A toolkit for building distributed multi-agent systems”. Applied Artificial Intelligence Journal, 13(1), pages 129-185, January 1999.

[Object Management Group, 2000]
Object Management Group, “Mobile Agent Facility Specification”. Online http://www.omg.org/cgi-bin/doc?formal/2000-01-02.

[Occello et al., 2002]
M. Occello, C. Baeijs, Y. Demazeau and J-L. Koning. “Mask: An aeio toolbox to develop multi-agent systems”. In Knowledge Engineering and Agent Technology, IOS Series on Frontiers in AI and Applications, Amsterdam, The Netherlands, 2002.

[Ordille, 1996]
J.J. Ordille, “When Agents Roam, who Can You Trust?”. In Proceedings of the First Conference on Emerging Technologies and Applications in Communications, Portland, Oregon, May 1996.

[Ouardani et al., 2007]
A. Ouardani, S. Pierre and H. Boucheneb. “A security protocol for mobile agents based upon the cooperation of sedentary agents”. Journal of Network and Computer Applications, 30(3), 1228-1243, 2007.

[Pillou, 2005]
J-F. Pillou, “Tout sur la sécurité informatique”, -1 vol. (XI-202 p.) – EAN. 9782100496549, Dunod, Paris, 2005.

[Razouki et al., 2013a]
H. Razouki, A. Hair, "Protéger des agents mobiles via deux politiques d'adaptation dynamique". Rencontre des Sciences Géomantiques (RSG'13), 8-9 avril 2013, Faculté des sciences, Rabat, Maroc, 2013.

[Razouki et al., 2013b]
H. Razouki, A. Hair, "Components-based software architecture for secure mobile agents via two strategies of adaptation". Proceedings of the 8th International Conference on Intelligent Systems: Theories and Applications (SITA'13), May 08-09, 2013, EMI, Rabat, Morocco, pp 327-334. (DOI: 10.1109/SITA.2013.6560821)

[Razouki et al., 2013c]
H. Razouki, A. Qostal, A. Hair, "Protéger les agents mobiles via deux stratégies d'adaptation". 1er Workshop sur l'Imagerie, Systèmes et Applications (ISA'2013), 23-24 mai 2013, Faculté Polydisciplinaire, Taza, Maroc.

[Razouki et al., 2013d]
H. Razouki, A. Hair, "Vers un modèle d'agent mobile sécurisé auto-adaptable". Deuxième Rencontre des Jeunes Chercheurs de l'UAE (2RJC-UAE'13), 25-26 mai 2013, Faculté des sciences, Tétouan, Maroc.

[Razouki et al., 2013e]
H. Razouki, , A Qostal, A. Hair, "L'adaptation statique et dynamique à base de composants pour la conception d'un agent mobile sécurisé". 2ème Edition des Journées Doctorales en Systèmes d'Information, Réseaux et Télécommunications (JDSIRT'13), 27-28 juin 2013, ENSIAS, Rabat, Maroc, pp 53-57.

[Razouki et al., 2014a]
H. Razouki, A. Hair, "Towards a new security architecture of mobile agents". International Journal of Soft Computing and Engineering (IJSCE) ISSN: 2231-2307, Vol. 3, Issue-6, January 2014.

[Razouki et al., 2014b]
H. Razouki, A. Hair, "Self-adaptive security for mobiles agents: Using the sensitivity of the services requested by mobile agent and the host's trust degree". First International Conference on Business Intelligence (CBI'14), April 29-30, 2014, Beni Mellal, Morocco.

[Razouki et al., 2014c]
H. Razouki, A. Hair, "Self-Adaptive Security for Mobiles Agents". International Journal of Computer Applications (IJCA) ISSN: 0975-8887, Vol. 94, No. 13, pages 24-29, May 2014.

[Razouki et al., 2015a]
H. Razouki, A. Hair, "Security for Mobile Agents: Trust estimate for platforms". Second International Conference on Business Intelligence (CBI'15), April 23-25, 2015, Beni Mellal, Morocco.

[Razouki et al., 2015b]
H. Razouki, A. Hair, "Security for Mobile Agents: Trust Estimate for Platforms". TELKOMNIKA Indonesian Journal of Electrical Engineering, ISSN: 2302-4046, Vol. 15, No. 2, August 2015, pages 381-389 (DOI: 10.11591/telkomnika.v15i2.8375)

[Razouki et al., 2016]
H. Razouki, A. Hair, "Ontologie pour le domaine de sécurité des agents mobiles : la correspondance sémantique entre les politiques de sécurité". La première édition des Journées Scientifiques des doctorants de la FSTG (1erJSD 2016), 21-25 avril, 2016, Marrakech, Maroc.

[Recursion Software, 2011]
Recursion Software (2011). Voyager Documentation. Online http://www. recursions-w.com/ voyager-documentation/. (page consultée le 10 août 2016).

[Ricordel et al., 2002]
P-M. Ricordel and Y. Demazeau, “La plate-forme volcano: modularité et réutilisabilité pour les systèmes multi-agents”. Numéro spécial sur les plates-formes de développement SMA. Revue Technique et Science Informatiques (TSI), 2002.

[Riordan et al., 1998]
J. Riordan and B. Schneier, “Environment key Generation towards Clueless Agents”. Lecture Notes in Computer Science, Vol. 1419, pages 15-24, 1998.

[Romito, 2012]
B. Romito, ”Stockage décentralisé adaptatif: autonomie et mobilité des données dans les réseaux pair-à-pair”. Dans thèse de doctorat Université de Caen Basse-Normandie, France, 2012.

[Roth, 1999]
V. Roth, “Mutual Protection of Cooperating Agents”. Secure Internet Programming: Security Issues for Mobile and Distributed Objects. J. Vitek and C. Jensen (Eds.), Springer-Verlag, 1999.

[Roth, 2004]
V. Roth, “Obstacles to the adoption of mobile agents”. In 5th IEEE International Conference on Mobile Data Management (MDM 2004). IEEE Computer Society, pages 19-22, January 2004.

[Sahai et al., 1998]
A. Sahai et C. Morin, "Mobile agents for enabling mobile user aware applications". Proceedings of the second international conference on Autonomous agents, AGENTS ’98, pages 205-211, 1998.

[Sayeb et al., 2002]
M. Sayeb, M. Hamza and A. Soliman, "Protecting Mobile Agents against Malicious Host Attacks Using Treat Diagnostic AND/OR Tree", Arab Academy for Science, Technology & Maritime Transport Computer Engineering Department, Alexandria, Egypt, 2002.

[Schneider et al., 2001]
F.B. Schneider, G. Morrisett and R. Harper, “A Language-Based Approach to Security”, in R. Wilhelm (Ed.): Informatics, 10 Years Ahead, LNCS 2000, Springer-Verlag Berlin Heidelberg, Pages 86-101, 2001.

[Schneider, 2000]
F.B. Schneider, “Enforceable security policies”. ACM Transactions on Information and System Security, 3(1), 30-50, February 2000.

[Sewell et al., 1999]
P. Sewell, P.T. Wojciechowski, and B.C. Pierce, ”Location independent communication for mobile agents: A two-level architecture”. Dans Lecture Notes in Computer Science Volume 1686, 1999, pages 1-31, 1999.

[Szyperski, 1998]
C. Szyperski. “Component software : beyond object-oriented programming”. ACM Press/Addison-Wesley Publishing Co., New York, NY, USA, 1998.

[Tan et al., 2001]
H.K. Tan and L. Moreau, “Trust Relationships in a Mobile Agent System“. In G. P. Picco, editor, Fifth IEEE International Conference on Mobile Agents, Lecture Notes in Computer Science, vol. 2240, Springer-Verlag, Atlanta, Georgia, 2001.

[Tan et al., 2002a]
H.K. Tan and L. Moreau, “Extending execution tracing for mobile code security”, In K. Fischer and D. Hutter, Eds. Proc. of 2nd Intl Workshop on Security of Mobile Multi-Agent Systems (SEMAS'2002), Bologna, Italy, pages 51-59, July 2002.

[Tan et al., 2002b]
H.K. Tan, L. Moreau, D. Cruickshank and D.De Roure, “Certificates for Mobile Code Security,” In Proceedings of The 17th ACM Symposium on Applied Computing (SAC'2002) - Track on Agents, Interactions, Mobility and Systems, page 76. 2002.

[Tsarkov et al., 2006]
Tsarkov, D. and I. Horrocks: 2006, ‘FaCT++ Description Logic Reasoner: System Description’. In: Proc. of the Int. Joint Conf. on Automated Reasoning (IJCAR 2006), Vol. 4130 of Lecture Notes in Artificial Intelligence. pp. 292–297, Springer Verlag.

[Vedamuthu et al., 2007]
A. Vedamuthu, D. Orchard, F. Hirsch, M. Hondo, P. Yendluri, T. Boubez, and U. Yalcinalp, “Web Services policy 1.5 – Framework”, W3C recommendation, 2007a. Available at: http://www.w3.org/TR/ws-policy/.

[Vercouter, 2004]
L. Vercouter, “MAST: Un modèle de composants pour la conception de SMA”. Journées Multi-Agents et Composants (JMAC’04), École des Mines de Paris, pages 18-31, Paris, France, Novembre 2004.

[Vigna, 1998]
G. Vigna, “Mobile Code Security”. Lecture Notes in Computer Science, Vol.1419, Springer-Verlag, Berlin Heidelberg, Berlin, 1998.

[Vigna, 2004]
G. Vigna, “Mobile Agents: Ten Reasons for failure”. In 5th IEEE International Conference on Mobile Data Management (MDM 2004), IEEE Computer Society, pages 19-22, January 2004.

[Woodcock et al., 1996]
J. Woodcock and J. Davies, ”Using Z: Specification, Refinement and Proof”. In International Thomson Computer Press, Upper Saddle River, NJ, USA, 1996.

[XACML, 2011]
OASIS eXtensible Access Control Markup Language (XACML) Site web: http://www.oasis-open.org/committees/tc_home.php?wg_abbrev=xacml, 2011.

[Yee, 1999]
B. Yee, “A sanctuary for mobile agents”, In Jan Vitek and Christian Jensen, editors, Secure Internet Programming: Security Issues for Mobile and Distributed Objects, number 1603 in LNCS. Springer-Verlag, Berlin, pages 261-274, 1999.

[Zhang et al., 2001]
M. Zhang, A. Karmouch and R. Impey, “Towards a secure agent platform based on FIPA”. In Proceedings of the 3th International Workshop on Mobile Agents for Telecommunication Applications (MATA ’01), volume 2164/2001 of LNCS, Springer Berlin/Heidelberg, pages 277-289, Montreal, Canada, 2001.

Annexe A : Acronymes et abréviations

Abréviations	Explication
3DES	Triple Data Encryption Standard
ABAC	Attribute-Based Access Control
AES	Advanced Encryption Standard
AS_C	Assertions de Sécurité de type Capability
AS_R	Assertions de Sécurité de type Requirement
CA	Créateur d'Agent
CA-ID	Identité du Créateur d'Agent
CCA	Composant de Contrôle d'Accès
CIA	Code d'Intégrité d'Agent
COA	Composants Orientés Agent
COS	Composants Orientés Système
CSD	Composant de Stockage de Données
C_X	Certificat X.509 d'une entité X
DCP	Degré de Confiance d'une Plate-forme
DES	Data Encryption Standard
DH	Degrés d'Habilitation
DN	Distinguished Name
DSA	Digital Signature Algorithm
ES	Exigence de Sécurité
FIPA	Foundation for Intelligent Physical Agents
$ID\text{-}CMP_X$	Identité d'un Composant X
IP	Internet Protocol
JA	Jeton d'Authentification
K_{agent}	Key agent
KDC	Key Distribution Center
KDS	Key Distribution Service
LCA	Local Certificate Authority
MA	Méthode d'Authentification
MASO	Mobile Agent Security Ontology
NSC	Niveau de Sécurité d'un Composant
OASIS	Organization for the Advancement of Structured Information Standards
OWL	Ontology Web Language
PA	Propriétaire d'Agent
PA-ID	Identité d'une plate-forme
PAP	Policy Administration Point
PDP	Point de Décision des Politiques
PEP	Policy Enforcement Point
PK_x	Public Key Certificate – Entity X

PKCS7	Public-Key Cryptography Standards
PKI	Public Key Infrastructure
RND	Random Number
RSA	RSA stands for Ron Rivest, Adi Shamir and Leonard Adleman
S_X	Signature de l'entité X
SA	Serveur d'Authentification
SAML	Security Assertion Markup Language
SAS	Système d'Adaptation de Sécurité
SC	Security Capability
SEA	Système d'Evaluation d'Agents
SGS	Système de Gestion des Services
SGSA	Système de Gestion de Sécurité d'Agent
SHA	Secure Hash Algorithm
SK_X	Secret Key Certificate – Entity X
SMA	Système Multi-Agents
SR	Security Requirement
SSO	Single Sign On
T_{exp}	Temps d'expiration du ticket
UDDI	Universal Description Discovery and Integration
URN	Uniform Resource Name
W3C	World Wide Web Consortium
WS-policy	Web Services Policy
XACML	eXtensible Access control MarkupLanguage
XML	Extensible Markup Language

Tableau A. 1. Acronymes et les abréviations utilisés dans cette thèse

Annexe B : Politique de contrôle d'accès en XACML

Exemple de politique contrôle d'accès aux données dynamique d'un agent sous le format original de XACML.

```
<Policy PolicyId="VacationPolicy" RuleCombiningAlgId="Permit-overrides">
 <Description> Une politique de contrôle d'accès aux données d'agent mobile </Description>
 <Target>
 <Subjects>
  <Subject>
  <SubjectMatch MatchId="string-equal">
   <AttributeValue DataType="string">ID-PL1</AttributeValue>
   <SubjectAttributeDesignator AttributeId="IdentityPlatform" DataType="string"/>
  </SubjectMatch>
  </Subject>
  <Subject>
  <SubjectMatch MatchId="string-equal">
   <AttributeValue DataType="string">ID-PL2</AttributeValue>
   <SubjectAttributeDesignator AttributeId="IdentityPlatform" DataType="string"/>
  </SubjectMatch>
  </Subject>
  <Subject>
  <SubjectMatch MatchId="string-equal">
   <AttributeValue DataType="string">ID-PL3</AttributeValue>
   <SubjectAttributeDesignator AttributeId="IdentityPlatform" DataType="string"/>
  </Subject>
 </Subjects>
 <Resources>
  <Resource>
  <ResourceMatch MatchId="string-equal">
   <AttributeValue DataType="string">ID-Agent</AttributeValue>
   <ResourceAttributeDesignator AttributeId="IdentityAgent" DataType="string"/>
  </ResourceMatch>
  </Resource>
 </Resources>
 <Actions>
  <AnyAction/>
 </Actions>
 </Target>
 <Rule RuleId="Rule11" Effect="Permit">
     <Description>
     Seulement la plates-formes PL1 peut accéder aux données produites par la plate-forme PL3
     </Description>
 <Target>
  <Subjects>
  <Subject>
   <SubjectMatch MatchId="string-equal">
   <AttributeValue DataType="string">ID-PL1</AttributeValue>
   <SubjectAttributeDesignator AttributeId="IdentityPlatform" DataType="string"/>
   </SubjectMatch>
  </Subject>
  </Subjects>
  <Resources>
  <Resource>
```

```
 <ResourceMatch MatchId="string-equal">
 <AttributeValue DataType="string">ID-Agent</AttributeValue>
 <ResourceAttributeDesignator AttributeId="IdentityAgent" DataType="string"/>
 </ResourceMatch>
 </Resource>
        <Resource>
 <ResourceMatch MatchId="string-equal">
 <AttributeValue DataType="string">ID-Component</AttributeValue>
 <ResourceAttributeDesignator AttributeId="IdentityComponent" DataType="string"/>
 </ResourceMatch>
 </Resource>
        <Resource>
 <ResourceMatch MatchId="string-equal">
 <AttributeValue DataType="string">Data-Platform3</AttributeValue>
 <ResourceAttributeDesignator AttributeId="Label-Data" DataType="string"/>
 </ResourceMatch>
 </Resource>
 </Resources>
 <Actions>
 <AnyAction/>
 </Actions>
 </Target>
 </Rule>
<Rule RuleId="Rule12" Effect="Permit">
 <Description>
    Seule la plate-forme PL3 peut accéder en lecture à les données produites par PL1
    </Description>
 <Target>
 <Subjects>
 <Subject>
 <SubjectMatch MatchId="string-equal">
 <AttributeValue DataType="string">ID-PL3</AttributeValue>
 <SubjectAttributeDesignator AttributeId="IdentityPlatform" DataType="string"/>
 </SubjectMatch>
 </Subject>
 </Subjects>
        <Resources>
 <Resource>
 <ResourceMatch MatchId="string-equal">
 <AttributeValue DataType="string">ID-Agent</AttributeValue>
 <ResourceAttributeDesignator AttributeId="IdentityAgent" DataType="string"/>
 </ResourceMatch>
 </Resource>
        <Resource>
 <ResourceMatch MatchId="string-equal">
 <AttributeValue DataType="string">ID-Component</AttributeValue>
 <ResourceAttributeDesignator AttributeId="IdentityComponent" DataType="string"/>
 </ResourceMatch>
 </Resource>
        <Resource>
 <ResourceMatch MatchId="string-equal">
 <AttributeValue DataType="string">Data-Platform1</AttributeValue>
 <ResourceAttributeDesignator AttributeId="Label-Data" DataType="string"/>
 </ResourceMatch>
 </Resource>
 </Resources>
 <Actions>
 <Action>
 <ActionMatch MatchId="string-equal">
 <AttributeValue DataType="string">Read</AttributeValue>
 <ActionAttributeDesignator AttributeId="ActionId" DataType="string"/>
 </ActionMatch>
 </Action>
```

```
 </Actions>
</Target>
</Rule>
<Rule RuleId="Rule13" Effect="Permit">
<Description>
        Toutes les plates-formes peuvent modifier les données de la plate-forms PL2
   </Description>
<Target>
 <Subjects>
        <AnySubject/>
 </Subjects>
 <Resources>
 <Resource>
  <ResourceMatch MatchId="string-equal">
  <AttributeValue DataType="string">ID-Agent</AttributeValue>
  <ResourceAttributeDesignator AttributeId="IdentityAgent" DataType="string"/>
  </ResourceMatch>
 </Resource>
        <Resource>
  <ResourceMatch MatchId="string-equal">
  <AttributeValue DataType="string">ID-Component</AttributeValue>
  <ResourceAttributeDesignator AttributeId="IdentityComponent" DataType="string"/>
  </ResourceMatch>
 </Resource>
        <Resource>
  <ResourceMatch MatchId="string-equal">
  <AttributeValue DataType="string">Data-Platform2</AttributeValue>
  <ResourceAttributeDesignator AttributeId="Label-Data" DataType="string"/>
  </ResourceMatch>
 </Resource>
 </Resources>
 <Actions>
 <Action>
  <ActionMatch MatchId="string-equal">
  <AttributeValue DataType="string">Read</AttributeValue>
  <ActionAttributeDesignator AttributeId="ActionId" DataType="string"/>
  </ActionMatch>
 </Action>
   <Action>
  <ActionMatch MatchId="string-equal">
  <AttributeValue DataType="string">Update</AttributeValue>
  <ActionAttributeDesignator AttributeId="ActionId" DataType="string"/>
  </ActionMatch>
 </Action>
 </Actions>
</Target>
</Rule>
</Policy>
```

Figure B. 1. Politique de contrôle d'accès aux données d'agent mobile en XACML