République Algérienne Démocratique et Populaire
Ministère de l'Enseignement Supérieur et de la Recherche Scientifique
Université Mentouri – Constantine
Faculté des Sciences de l'Ingénieur – Département d'Informatique

Sécurité des Systèmes d'Information :

Mise en œuvre de la confiance et de l'adaptabilité pour la protection de l'agent mobile

Thèse

Présentée et soutenue publiquement le 21 Avril 2008

Pour l'obtention du

Doctorat en Sciences

Spécialité : Informatique

Par

Salima Hacini

Composition du Jury:

M. Mahmoud Boufaïda	Professeur, Université de Constantine	Président
M. Med Tayeb Laskri	Professeur, Université d'Annaba	Examinateur
M. Najib Badache	Professeur, Université d'Alger	Examinateur
M. Zaïdi Sahnoun	Professeur, Université de Constantine	Examinateur
Mme Zizette Boufaïda	Professeur, Université de Constantine	Rapporteur
Mme Zahia Guessoum	HDR, IUT de Reims	Invitée

Thèse préparée au niveau du laboratoire LIRE, Université Mentouri – Constantine

2008

Résumé

L'utilisation massive d'applications mobiles, employant notamment des agents mobiles, a été engendrée par des phénomènes récents tels que le nomadisme des utilisateurs, la mobilité des terminaux ou l'activité en mode déconnectée. Ces applications évoluent dans des environnements de plus en plus dynamiques, imprévisibles et hostiles et engendrent donc un important problème de sécurité dont la résolution constitue un véritable challenge. Le problème de sécurité lié à l'utilisation des agents mobiles a effectivement freiné leur expansion et restreint de façon sévère leur champ d'application. Il a, par voie de conséquence, intéressé de nombreux chercheurs dont le but essentiel est de promouvoir le déploiement des agents mobiles en suggérant des approches et des techniques de protection. La réflexion menée dans cette thèse s'inscrit dans le cadre de cette problématique. Elle concerne particulièrement la protection des agents mobiles contre l'analyse de leur code par des intrus. La solution proposée repose sur la confiance inspirée par l'hôte visité et emploie l'adaptabilité dynamique pour déterminer le comportement adéquat de l'agent mobile selon le degré de confiance déduit. Dans cette optique, nous proposons une architecture adaptative pour l'agent mobile qui lui offre la capacité de réagir avec un comportement imprévisible. La sélection de ce comportement de l'agent mobile est essentiellement basée sur son aptitude à estimer le degré de confiance de l'hôte visité, de contrôler l'historique de son comportement et d'exploiter les diverses qualités de service fournies par l'application.

ملخص

تعتمد الأنشطة الإنسانيّة أكثر فأكثر على الاستعمال من بعيد للموارد والخدمات, وعلى التفاعل بين أطراف متباعدة التي يمكن أن تكون لا تعرف شيئا عن بعضها البعض. تعتبر تكنولوجيا الأعوان المتنقلة أكثر تناسبا مع هذه التطورات. لكن ينبغي على هذه الأعوان أن تعتد لتنفد على مضيفات مختلفة في بيئات تتميز بشروط أمن متنوعة. يقدّم هذا البحث آلية معتمدة على الثقة لتحسين أمن الأعوان المتنقلة ضدّ مضيفات خبيثة حيث يسمح تنفيذهم في بيئات مختلفة. أسّست هذه الآلية على التفاعل الديناميكي بين العون والمضيف. تستعمل المعلومات المتجمعة أثناء التفاعل في حساب مفتاح يطلق عليه أسم مفتاح البيئة. يستعمل هذا المفتاح بعد ذلك لاستنتاج درجة الثقة التي يتميز بها المضيف. يكيّف العون المتنقل تنفيذه وفقا لتلك الثقة، لتاريخ تصرفه و لنوعية الخدمة الممنوحة. اقترحت أيضا هندسة دينامكية تهب لهذا العون المتنقل القدرة على التجاوب بتصرف غير متوقّع يحميه من التحليل الديناميكي.

Abstract

Human activities are increasingly based on the use of distant resources and services, and on the interaction between remotely located parties that may know little about each other. Mobile agents are the most suited technology. They must therefore be prepared to execute on different hosts with various environmental security conditions. This paper introduces a trust-based mechanism to improve the security of mobile agents against malicious hosts and to allow their execution in various environments. It is based on the dynamic interaction between the agent and the host. Information collected during the interaction enables generation of an environment key. This key allows then to deduce the host's trust degree and permits the mobile agent to adapt its execution accordingly to the host trustworthiness, its behavior history and the provided Quality of Service. Adaptive mobile agent architecture is therefore proposed. It endows the mobile agent with the ability to react with an unexpected behavior.

A la mémoire de mon père,

A la mémoire de mon époux.

Remerciements

La réalisation d'une thèse de doctorat est une tranche de vie à part entière qui s'est nourrie de nombreuses et diverses influences. Que Dieu le Tout Puissant reçoive ma gratitude pour m'avoir donné la force morale et physique pour l'achever.

Beaucoup de personnes ont participé directement ou indirectement à sa concrétisation et méritent d'être remerciées, quel que soit, le degré de leur investissement. Malheureusement, tout le monde ne peut apparaître nommément ici et je me dois de faire une sélection subjective.

D'un point de vue professionnel, je remercie le professeur Z. Boufaïda pour le travail qu'elle a effectué à mes côtés, emprunt de conseils pour orienter mes recherches et de critiques pertinentes pour l'améliorer. Merci d'avoir dirigé ce travail et de m'avoir poussé à dépasser mes limites.

Je tiens aussi à manifester ma profonde reconnaissance envers madame Z. Guessoum, maître de conférences HDR au LIP6. Elle m'a accueilli, à maintes reprises, dans ce laboratoire (équipe SMA) où nous avions eu de nombreuses discussions très fructueuses. Elle a, de plus, eu l'amabilité de lire l'intégralité de mon manuscrit et a usé de son expérience pour apporter des suggestions très constructives et améliorer la qualité de ce travail. Ces compétences et sa rigueur scientifiques sont indiscutables et mondialement reconnues. Qu'elle trouve ici l'expression de mes remerciements les plus sincères.

Mes remerciements vont aussi, au président du jury le professeur M. Boufaïda, pour m'avoir encouragée, s'être très largement penché sur la préparation de ma thèse et pour avoir encore accepté de présider ce jury. Qu'il trouve ici l'expression de ma profonde gratitude.

Je souhaite également remercier mes examinateurs les professeurs M. T. Laskri, N. Badache et Z. Sahnoun pour avoir accepté d'évaluer mon travail et pour les commentaires qu'ils ont formulés afin de le perfectionner.

Je tiens aussi à remercier les nombreux collègues qui ont partagé mon quotidien que ce soit au département ou au sein de l'équipe CIBC du laboratoire Lire. Je leur suis reconnaissante pour le soutien moral qu'ils m'ont prodigué tout le long de la réalisation de cette thèse.

D'un côté plus personnel, je ne saurais montrer trop de reconnaissance aux membres de ma famille. A mes défunts père et époux, Ahmed Hacini et Ali EL Dunia, en premier lieu, qui auraient été tellement fiers de me voir achever cette thèse. Leur soutien m'a manqué. Je leur rends un grand hommage car je sais leur profond attachement à la connaissance. A ma mère, pour son dévouement, sa compréhension, son amour inconditionnel et son soutien permanent. Comment pourrai-je assez la remercier ?

A la mémoire de mon frère Noureddine. A mes sœurs Samira, Samia, Nacima et Fatiha, à mon frère Adlane, mes beaux frères Mohamed, Noureddine et Abdelhak et à tous mes neveux et nièces qu'ils voient dans cet ouvrage le témoignage des profonds sentiments que je leur voue.

TABLE DES MATIÈRES

TABLE DES FIGURES

LISTE DES TABLES

Chapitre 1 - Introduction générale

1.1. Contexte

Les dernières décennies ont été marquées par une révolution dans le domaine de l'informatique distribuée et des télétraitements. Les nouveaux systèmes d'information et les applications récentes sont souvent distribués et sont caractérisés par un environnement dynamique, imprévisible et hostile. Une des technologies prometteuses supportant ces caractéristiques est celle des agents logiciels mobiles. Cette technologie est reconnue comme étant la plate-forme qui accueille les services électroniques [Hohl 1997] et qui peut se libérer des contraintes d'hétérogénéité des plates-formes cibles. Par conséquent, les applications distribuées telles que, les systèmes de télécommunication, la gestion de l'information ou la vente aux enchères, sont de plus en plus basées sur le concept d'agent mobile.

Un agent mobile est une entité logicielle active et autonome pouvant suspendre son exécution, se déplacer vers un autre hôte du réseau, et continuer son activité, décidant où aller et que faire le long de son itinéraire [Rouverais 2002]. Cette définition implique le mouvement aussi bien du code que des données. Ainsi, un agent mobile peut se déplacer dans un réseau de machines offrant des services pour réaliser une tâche complexe ou rechercher une information au nom d'un utilisateur humain ou d'une application. La mobilité des agents constitue un modèle d'exécution répartie pour les réseaux de grande envergure et les stations nomades.

Les applications des agents mobiles sont nombreuses. Elles incluent les domaines du commerce électronique, de la gestion de systèmes distribués, de la gestion de système workflow, de l'assistance personnalisée, du traitement parallèle, du contrôle et de la gestion de réseaux, de l'équilibrage de charge ou de la recherche documentaire [Abdul-Rahman 1997], [Beimel 2000].

L'agent mobile migre vers l'hôte et exécute des calculs sur celui-ci, réduisant de ce fait des coûts de communication. C'est contraire au paradigme client/serveur où l'information entre les clients et le serveur s'échange par l'intermédiaire des appels de procédures à distance ou d'autres procédés de transmission de messages. L'architecture des réseaux d'information Client/Serveur, qu'elle soit mise en œuvre par une technique directe d'échange de messages, ou par des mécanismes d'appel de procédures ou d'évaluation à distance demande aux deux parties d'être connectées durant toute l'interaction. La migration d'activité dans les systèmes répartis modernes impose également cette contrainte pour l'accès aux ressources communes et le chargement à la demande du code de l'activité se déplaçant dans le réseau local (système de gestion de fichier réparti, mémoire partagée répartie). Le paradigme agent mobile propose d'utiliser la migration d'activité en supprimant cette contrainte de connexion constante qui n'est pas évidente ni dans les réseaux de grande envergure ni pour les stations nomades. Les deux parties sont connectées seulement durant la phase de migration.

Le paradigme agent mobile promet d'ouvrir des possibilités de calcul très intéressantes, particulièrement avec la popularité des réseaux sans fil où la bande passante se trouve au premier plan et où les liaisons ne sont pas toujours fiables. De plus, il offre de nombreux avantages pour la conception et le développement d'applications réparties telles que l'autonomie, l'adaptation dynamique, la distribution de données et de contrôle, la tolérance aux fautes, une meilleure utilisation des ressources de réseau ainsi qu'une réduction de

communication en ce qui concerne la latence, la largeur de la bande passante et le temps de connexion [Farmer 1996 a], [Borselius 2002], [Braynov 2002], [Cahil 2003].

Toutefois, l'emploi des agents mobiles par les systèmes d'information d'entreprise est freiné pour des raisons de sécurité : c'est la crainte d'importer du code malveillant. En effet, deux types de risques sérieux de sécurité sont engendrés: i) les attaques pouvant être perpétrées par un agent mobile malveillant, ii) les attaques émanant d'un hôte malicieux hébergeant l'agent mobile.

En effet, un agent mobile qui migre vers différents hôtes, transporte son état, son code ainsi que ses données. Pendant que l'agent s'exécute sur un hôte particulier, ce dernier a un contrôle total sur lui. D'un autre côté, l'agent a besoin de certaines ressources de l'hôte afin de s'exécuter. Si l'hôte ou l'agent est malicieux, la confidentialité et l'intégrité de l'un ou l'autre sont compromises. Un certain nombre d'attaques possibles sont citées dans [Hohl 1998], [Vitek 1999]. Elles incluent principalement, la modification non autorisée du code, l'extraction des données confidentielles, la répudiation ou l'analyse de l'exécution.

Afin de promouvoir le déploiement des systèmes à base d'agents mobiles dans les entreprises, deux aspects de la sécurité liés à l'agent mobile doivent être considérés. Le premier est lié à la sécurité de l'agent lui-même qui doit être protégé contre tout acte visant l'altération, la destruction ou la manipulation de son code, de son état et des données qu'il transporte. Le second a trait à la protection de l'hôte d'accueil contre tout accès illégal d'un agent dit malicieux à des fichiers ou à des données dans le but de les altérer ou d'espionner.

1.2. Problématique

Les travaux de recherche relatifs à la sécurité des agents mobiles suivent principalement deux axes [Sau-Koon 2000], [Bierman 2002], [Bella 2004]. Le premier axe concerne la protection de l'hôte contre des agents mobiles malicieux tandis que le second axe concerne la protection de l'agent mobile contre la malveillance des hôtes visités.

La protection de l'hôte a été le sujet d'un grand intérêt et a mis en place un certain nombre de mesures protégeant le système hôte et le système agent mobile. Des exemples de telles techniques sont la signature du code [Reiser 2000], le bac à sable (sandboxing) [Grandison 2000], [Rouverais 2002], le code avec preuve (proof carrying code) [Necula 1998], l'estimation de l'état (state appraisal) [Farmer 1996 a] et l'historique des hôtes (path histories) [Chess 1995]. Ces techniques fournissent un niveau de sécurité acceptable.

La protection de l'agent mobile contre un hôte malicieux quant à elle, demeure un problème ouvert et difficile du fait que l'environnement d'exécution a un contrôle total sur l'agent mobile (sans quoi, la protection de l'hôte ne serait pas possible). Plusieurs tentatives abordent ces menaces d'une manière totale ou partielle. Elles visent essentiellement à rendre les attaques inutiles ou détectables.

Un hôte malicieux peut essayer d'attaquer un agent mobile afin d'obtenir un service gratuitement ou accéder à la mémoire de l'agent pour y consulter ou manipuler des informations privées. D'autres exemples de telles attaques sont l'altération malicieuse du code de l'agent mobile et le contrôle de son exécution. L'agent est vulnérable pendant son exécution sur la plate-forme de l'hôte d'accueil. Par conséquent, son propriétaire exige des garanties concernant la protection de l'agent contre les menaces des hôtes malicieux. 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.

La protection des agents mobiles à l'encontre des comportements malicieux des hôtes visités représente un domaine de recherche attrayant [Karnik 1998], [Bierman 2002], [Borselius 2002], [Rouverais 2002], [Bella 2004]. Plusieurs approches ont été introduites telles que la

sécurité basée sur le matériel [Herzberg 1985], [Yee 1995], [Smith 1998], le calcul par une fonction cryptographique [Sander 1998a], [Sander 1998b], la boîte noire [Hohl 1998] ou les agents naïfs [Riordan 1998]. Ces approches aident à l'amélioration de la sécurité du code exécuté dans un environnement non crédible. Toutefois, elles présentent des inconvénients. A titre d'exemple, la sécurité basée sur le matériel a un coût prohibitif et le calcul par une fonction cryptographique n'est pas applicable à n'importe quelle tâche d'agent mobile. Il est restreint aux fonctions polynomiales et rationnelles. Par conséquent, un important défi est d'introduire une solution avec un coût raisonnable et un niveau de sécurité acceptable.

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 hôtes malveillants.

1.2. Mise en œuvre de la confiance et de l'adaptabilité pour la protection de l'agent mobile

L'objectif de cette thèse est la protection du code de l'agent mobile contre toute analyse visant sa divulgation. Notre réflexion a mis en évidence que la sécurité de l'agent mobile a un lien intrinsèque avec le degré de confiance de l'hôte visité. Nous considérons donc que la solution aux problèmes cités repose sur la confiance inspirée par l'hôte visité et la détermination de sa responsabilité en cas d'attaque. En outre, il n'est pas possible de décider à priori, au cours de son déplacement, si un hôte est digne de confiance ou pas. Par conséquent, 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.

Dans cette optique, nous proposons 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.

Nous avons proposé un processus d'estimation du degré de confiance. Celui-ci est basé sur le diagnostic d'une clé environnementale. La génération de cette dernière est dérivée à partir des informations collectées durant l'interaction dynamique entre l'agent mobile et l'hôte. L'agent mobile essaye de détecter les diverses conditions qui doivent être prises en compte pour l'exécution d'une tâche spécifique, estime la confiance qu'il peut mettre en l'hôte visité et choisit la réaction à entreprendre. En outre, l'utilisation de l'adaptabilité offre à l'agent mobile la propriété de flexibilité permettant la modification de son comportement et compliquant ainsi son analyse.

Nous avons opté pour une architecture réflexive. En effet, la réflexivité semble être une méthode prometteuse pour l'implantation et la réalisation de l'adaptabilité dynamique [David 2002], [Malenfant 2004]. Elle constitue un support de développement d'application et fournit des mécanismes permettant d'exprimer les traitements en des termes extrêmement génériques. Les propriétés intrinsèques de la réflexivité (l'introspection et l'intercession) permettent à l'agent mobile de raisonner et d'agir sur lui-même.

1.3. Organisation de la thèse

Le second chapitre introduit le cadre général de cette étude. Il comporte la présentation du concept d'agent mobile et sa sécurisation. Ce dernier point couvre deux aspects : la protection des hôtes et celle des agents mobiles. Une grande attention est prêtée au second volet. Nous synthétisons les différentes avancées dans ce domaine avant de situer notre approche de protection.

Le troisième chapitre introduit la présentation de l'approche en citant les trois principes sur lesquels elle est fondée à savoir: un protocole de sécurité, un mécanisme d'estimation de la confiance et un processus d'adaptation dynamique.

Le quatrième chapitre met en relief le lien intrinsèque qui relie la sécurité et la confiance dans les applications distribuées en général et celles des agents mobiles en particulier. Il définit la confiance dans le contexte de la présente thèse et passe en revue les différentes étapes du mécanisme d'estimation de la confiance élaboré par notre approche de protection.

Le cinquième chapitre présente l'architecture de l'agent mobile adaptatif et spécifie le processus d'adaptation instauré.

Le sixième chapitre décrit l'implémentation de l'agent mobile sécurisé. Les tests réalisés montrent la faisabilité de l'approche de protection élaborée ainsi que ses performances.

Enfin, la conclusion générale résume notre contribution et décrit les perspectives.

Chapitre 2 – Les agents mobiles et la sécurité

"Pour un agent, interagir avec un autre constitue à la fois la source de sa puissance et l'origine de ses problèmes."

[Jacques Ferber 1997]

2.1. Introduction

Le paradigme agent mobile est proposé comme une solution prometteuse facilitant l'exécution distribuée à travers un réseau ouvert. Il offre une panoplie d'avantages [Lange 1999] telles qu'une exécution asynchrone et autonome, une diminution du trafic sur le réseau, une tolérance aux fautes ou encore une réduction du temps d'exécution des tâches. L'agent mobile améliore les performances puisqu'il permet de transférer les unités d'exécution vers les clients plutôt que de capturer les données pour les traiter localement. Cependant, cette technologie a introduit plusieurs problèmes sérieux tels que le problème de l'hétérogénéité, celui du maintien des communications ou encore celui de la gestion des ressources partagées [Vigna 2004, Chess 1994]. De plus, elle a accentué les effets des problèmes de sécurité existants [Bella 2004, Borselius 2002, Farmer 1996 b, Karnik 1998, Reiser 2000, Roth 1999, Vigna 1998]. La raison principale du dédain suscité par l'emploi des applications basées sur les agents mobiles réside, en effet, dans le peu de confiance qu'ils inspirent.

Ce chapitre introduit le concept d'agent mobile en mettant l'accent sur le problème de la sécurité que son utilisation engendre. Il décrit les techniques les plus importantes développées pour sa protection.

2.2. Les agents mobiles

Situé aux frontières de l'intelligence artificielle, du génie logiciel et des systèmes répartis, le concept d'agent mobile a suscité un intérêt croissant ces dernières années avec l'explosion de l'utilisation d'Internet et des réseaux de communication. La communauté scientifique et plus particulièrement celle des systèmes distribués a reconnu le fort potentiel offert par le paradigme agent et spécialement par celui de l'agent mobile. Sans être une panacée, ce dernier a contribué largement à l'augmentation de la distribution des applications [Borselius 2002, Braynov 2002, Cahil 2003, Farmer 1996 a, Milojicic 1999].

La notion d'agent mobile invoque deux concepts : celui d'agent et celui de mobilité. Le concept d'agent fait référence au domaine de l'Intelligence Artificielle (IA) alors que le concept de mobilité des agents repose sur des caractéristiques empruntées à la mobilité du code. Dans cette section, nous présentons le concept d'agent. Nous exposons ensuite les modes d'exécutions qui sont à l'origine du modèle de l'agent mobile. Nous donnons une définition de l'agent mobile en tenant compte des aspects : agent, mobilité et environnement. Nous donnons un aperçu sur les propriétés fondamentales des agents mobiles. Enfin, nous

abordons le problème de sécurité, suscité par leur utilisation et qui constitue l'objet de la présente thèse.

2.2.1. Le concept d'agent

Avant de définir l'agent mobile, il est essentiel de se pencher d'abord sur la notion d'agent. Il existe, à l'heure actuelle, une pléthore de définitions de l'agent. Cela est dû principalement à la relative jeunesse du domaine. Nous avons retenu deux définitions car elles nous ont semblé correspondre le plus mieux au cadre de notre étude. La première définition stipule qu' « *un agent est une entité logicielle ou matérielle, à laquelle est attribuée une certaine mission qu'elle est capable d'accomplir de manière autonome, disposant d'une connaissance partielle de ce qui l'entoure (son environnement), et agissant par délégation pour le compte d'une personne ou d'une organisation*» [Perret 1997]. La seconde définition, énoncée par Jennings et al. [Jennings 1995], [Jennings 1998a], [Jennings 1998b], précise qu' « *un agent est un système informatique situé dans un environnement, et qui agit d'une façon autonome et flexible pour atteindre les objectifs pour lesquels il a été conçu* »

De ces deux définitions émergent les quatre caractéristiques suivantes:

1 L'agent est une entité qui agit par délégation. Il doit respecter la stratégie de son producteur vis à vis des choix qu'il est amené à faire, afin que celui-ci soit responsable des tâches effectuées par son agent.
2 L'agent est une entité autonome qui dispose de son propre environnement.
3 L'agent dispose d'une connaissance, même partielle, de son environnement courant. Ceci lui permet de prendre les décisions appropriées.
4 L'agent est caractérisé par un comportement flexible (par opposition à un comportement rigide).

Dans la littérature, existent plusieurs types d'agents :

- **Les agents réactifs** ne font que réagir d'une manière mécanique aux stimuli qu'ils perçoivent. Ils n'ont pas de représentation symbolique de leur monde.
- **Les agents cognitifs** qui disposent d'une capacité de raisonnement, d'une aptitude à traiter des informations diverses liées au domaine d'application, et des informations liées à la gestion des interactions avec les autres agents et l'environnement [Guessoum 2003].
- **Les agents hybrides** intègrent l'aspect cognitif et réactif [Guessoum 2003].

Tous ces agents peuvent être stationnaires ou mobiles. Les agents stationnaires exécutent leurs tâches au niveau du site qui les a créés et communiquent par RPC (*cf.* Sous-section 2.2.2.2). Ils sont sensés s'exécuter dans un environnement de confiance. Par conséquent, ils ne sont pas confrontés à des problèmes de sécurité.

Les agents mobiles quant à eux, peuvent se déplacer d'un site à un autre au cours de leur exécution pour accéder à des données ou à des ressources. Ils se déplacent avec leur code et leurs données propres, mais aussi avec leur état d'exécution. Ils s'exécutent sur des sites différents et peuvent donc être sujet à des attaques provenant d'hôtes malveillants.

2.2.2. Modes d'exécution

La conception d'applications distribuées fait intervenir trois modèles principaux qui étendent le paradigme client/serveur pour exploiter la mobilité du code: l'exécution à distance, le code

à la demande et les agents mobiles [Carzaniga 1997]. Ces modèles diffèrent selon l'emplacement des différents composants manipulés avant et après l'exécution d'un service. Ces composants sont [Cubat 2005] :

1 Le savoir-faire (le code)
2 Les ressources nécessaires à l'exécution du code
3 L'unité d'exécution (qui traite le code).

Les sous-sections suivantes donnent un aperçu sur ces différents modèles d'exécution.

2.2.2.1. Le Client/Serveur

Ce mode d'exécution est largement connu et son utilisation est assez bien maîtrisée. Un serveur offrant un ensemble de services, de ressources et un savoir-faire nécessaire pour exécuter ces services se trouvent sur un site A. Le client se trouvant sur un site B demande l'exécution d'un service par interaction avec le serveur A. Le serveur A exécute le code correspondant au service demandé en utilisant les ressources nécessaires se trouvant sur le site B. Le résultat est généralement délivré au client par le biais d'une autre interaction.

2.2.2.2. L'exécution à distance

Cette méthode offre la possibilité de réaliser des appels à distance grâce au mécanisme d'appel de procédure à distance ou RPC (Remote Procedure Call) [Birrell 1984, Nelson 1981]. Ce mécanisme permet aux développeurs de ne pas manipuler directement le service de communication réseau mais d'appeler une procédure distante comme si elle était locale. Ainsi, un processus utilisant le RPC ne fait aucune différence entre un appel local et un appel distant. Le principe est le suivant : A détient le savoir-faire pour fournir un service mais ne détient pas les ressources nécessaires qui se trouvent sur un site B. A envoie le code à B qui détient un composant de calcul. Le code est exécuté au niveau de B en utilisant les ressources présentes localement. B renvoie le résultat à A par une autre interaction.

Le RPC a été développé à l'origine pour des systèmes Unix et sans considération de la programmation objet. Il a été récupéré par l'OMG (Object Management Group) avec CORBA qui l'a intégré dans une approche objet. Ce concept est aussi utilisé par Java/RMI. Le service NFS (Network File System) [Lyon 1985] est construit aussi sur ce mécanisme en permettant d'accéder à des fichiers distants comme s'ils étaient sur le disque local d'une machine.

2.2.2.3. Le code à la demande

Dans le modèle code à la demande (COD : Code On Demand), le client dispose de l'unité d'exécution et des ressources mais pas du savoir-faire qui va être récupéré auprès du serveur. Ainsi, un client adresse une requête uniquement pour récupérer un code précis afin de l'exécuter localement avec les ressources présentes. Les trois éléments sont réunis sur le site client comme le montre la Figure 2.1. Cette méthode permet d'étendre les fonctionnalités d'une application directement chez le client sans avoir besoin d'effectuer une nouvelle installation. A l'aide de ce concept, le serveur peut rechercher les dernières versions d'un code à exécuter et le lier dynamiquement avec d'autres codes objets pour étendre une application. Il demande aux systèmes distants d'exécuter la tâche de collecte, et leur laisse le soin de récupérer le code correspondant à la tâche.

L'exemple le plus répandu d'utilisation de cette méthode est le téléchargement d'applet Java à partir d'un serveur Web. Ces applets sont des classes Java présentes sur le site serveur qui sont téléchargées puis interprétées par la machine virtuelle du site client.

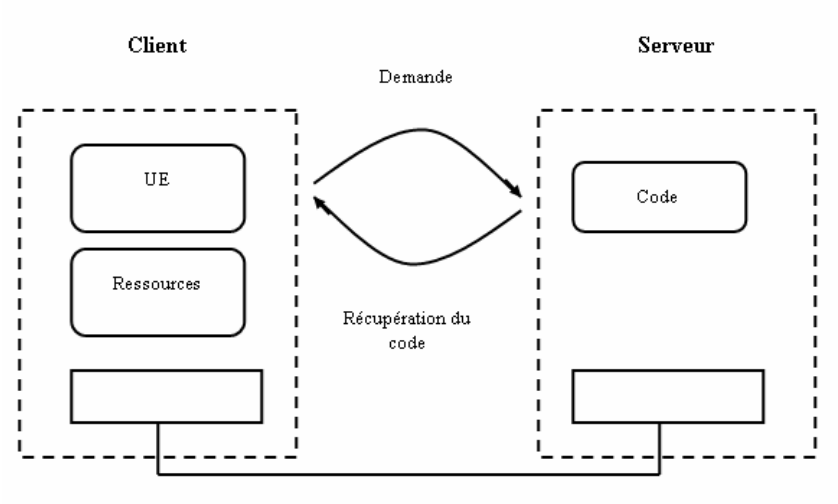

Fig.2.1. Code à la demande

2.2.2.4. L'agent mobile

Le concept d'agent mobile est différent des autres modèles de conception utilisant le code mobile. On s'intéresse à l'exécution du code et à son déplacement d'un système à un autre sous le contrôle d'une application. L'agent mobile est différent du paradigme de code mobile dans le sens où les interactions qui lui sont associées peuvent impliquer la mobilité de son état d'exécution [Picco 1998]. Par rapport donc aux notions précédentes, l'agent mobile représente une unité complète d'exécution (programme), et contient donc son propre contrôleur d'exécution.

Pour ce paradigme dont l'utilisation est relativement récente, un site A dispose d'un agent mobile qui détient le savoir-faire alors que les ressources nécessaires se trouvent sur un autre site B. Cet agent mobile migre vers le site B en emportant avec lui le savoir-faire et éventuellement des résultats intermédiaires. Arrivé sur le site B, l'agent mobile poursuit son exécution en utilisant les ressources disponibles sur ce site. Ce schéma répond à nos besoins et représente le cas de figure envisagé dans cette thèse. Le savoir-faire est en effet transporté par un agent mobile qui l'achemine vers plusieurs clients.

2.2.3. Agents mobiles : définition et propriétés

2.2.3.1. Définition

Un agent mobile est défini comme un élément autonome oeuvrant généralement au nom d'un utilisateur ou d'une application [Rothemel 1998], [Leriche 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.

La communauté scientifique attribue à l'agent mobile les capacités suivantes :

- Il doit faire preuve d'**autonomie** [Jennings 1998a], [Jennings 1998b]. Il doit pouvoir agir seul, prendre une décision en fonction d'informations lui provenant de son environnement. Il gère lui-même son état interne en fonction des informations qui lui parviennent.
- Il doit être réactif. La **réactivité** est une propriété primordiale pour un agent. Celui-ci reçoit des informations de son environnement et il est capable de réagir en conséquence.
- Il ne peut être autonome que s'il est capable de **pro-activité**. C'est le fait d'avoir un comportement piloté par des buts en prenant l'initiative.
- Il doit être efficace. L'**efficacité** est une autre caractéristique très importante. C'est la capacité à résoudre le problème, à atteindre ses buts.
- Il doit faire preuve d'**adaptation**. En fait, il doit pouvoir résoudre un certain nombre de problèmes. Maes considère que l'agent doit s'adapter aux variations de son environnement [Maes 1994]. Pour cela, il possède un jeu d'actions possibles, des capteurs qui le renseignent sur son environnement et ses objectifs sont variés.
- Il doit pouvoir établir une **communication** avec d'autres agents de son environnement.
- Il doit se prémunir de la capacité de **mobilité** lui permettant de se déplacer selon un itinéraire préalablement établi.

2.2.3.2. Propriétés de l'agent mobile

Les agents mobiles ont été introduits initialement en 1994 avec l'environnement Telescript [White 1994] qui permettait à des processus de choisir eux-mêmes de se déplacer sur les sites d'un réseau afin de travailler localement sur les ressources. Entre l'appel de procédure à distance, et l'agent mobile, en passant par le code mobile il y a eu une progression constante de l'autonomie d'exécution [Picco 1998]. L'agent mobile représente, en effet, une unité complète d'exécution (programme). Ceci lui permet de se déplacer ou d'être déplacé d'un site à un autre.

Les agents mobiles ont rapidement suscité un intérêt tout particulier dans les domaines de recherche portant sur les applications réparties. Très rapidement, cette nouvelle méthode de programmation a été évaluée afin de voir ce qu'elle pouvait apporter comme caractéristiques propres [Chess 1994] et ce qu'elle permettait d'améliorer par rapport aux méthodes de programmation plus classiques [Guy1999b]. Les améliorations apportées portent sur le gain de performances dû à une meilleure utilisation des ressources physiques mises à disposition. L'amélioration intervient à différents niveaux permettant d'optimiser la tâche globale des agents.

Baisse de la communication réseau

Le déplacement des agents mobiles permet de réduire de manière significative, voire supprimer, les communications distantes entre les clients et les serveurs. En privilégiant les interactions locales, l'utilisation du réseau se limite principalement au transfert des agents. Cette situation présente trois principaux avantages.

Le premier avantage est la diminution de la consommation de la bande passante. En effet, plusieurs études [Sahai 1998], [Gray 2002], [Guy1999a] montrent qu'en comparaison aux RPC, la mise en place des agents mobiles permet d'obtenir une réduction significative de la charge réseau en terme du nombre total de données transférées. Cette diminution est constatée dans différents types d'applications nécessitant d'intenses échanges d'informations entre le

client et le serveur. Nous pouvons citer, comme exemple, la collecte d'informations dans des bases de données réparties, l'exploration d'un Internet ou encore la gestion de réseaux.

Le second avantage notable est la diminution des temps de latence [Johansen 1998], [Kotz 1999], [Jun 2002]. Dans le contexte des réseaux à large échelle, la mise en place d'applications distribuées, nécessitant de fréquentes interactions entre client et serveur, se heurte aux temps de latence propres aux communications réseaux. Il arrive fréquemment que le temps d'attente de la réponse d'une requête soit plus long que le temps de traitement nécessaire à la réalisation du service. En rapprochant client et serveur dans un même sous-réseau, voire sur un même site, on les place dans un environnement où les temps de réponse des interactions sont limités, ce qui permet de réduire d'autant les temps de latence.

Le dernier avantage à souligner vient des brèves périodes de communication. En réduisant le plus possible les communications distantes aux seuls transferts d'agents mobiles, on diminue considérablement les périodes de connexion entre deux sites. Cette diminution de la fenêtre d'utilisation des communications réseaux permet de moins se soucier des ruptures de liens physiques qui peuvent intervenir fréquemment dans les environnements sans fil.

Exécution asynchrone

Dans le modèle classique client/serveur, les deux entités doivent rester connectés tant que le service est en cours d'exécution. En plus des problèmes liés aux fluctuations des performances réseaux, certains services, nécessitant de longues phases de traitement, ne supportent pas facilement une rupture de connexion avec le client. Dans ce cas, ils doivent souvent redémarrer entièrement leurs calculs. Mais, le maintien du lien de communication peut s'avérer difficile dans des réseaux à large échelle ou sans fil. Avec les agents mobiles, un client peut déléguer les interactions avec le service sans maintenir une connexion de bout en bout [Gray 2000]. Avec cette possibilité d'une exécution asynchrone, le client peut demander un service, se déplacer puis venir récupérer les résultats plus tard. Ce mode de fonctionnement est particulièrement intéressant lors du lancement à distance de simulations numériques.

Amélioration du temps d'exécution

L'optimisation des phases de traitement se produit à deux niveaux. Premièrement, en localisant les données et le code sur un même site, on supprime les phases de dialogue entre le client et le serveur qui sont perturbées par des temps de latence dûs aux communications réseaux [Ismael 1999]. Ensuite, le déplacement du code va permettre de déléguer les calculs sur des machines serveurs (telles que des supercalculateurs) qui sont généralement plus puissantes qu'une machine cliente. Cela est particulièrement vrai dans l'informatique nomade où la miniaturisation s'accompagne d'une perte de puissance significative.

Personnalisation

Les agents s'adaptent plus facilement aux besoins du client que des serveurs. Il n'est pas nécessaire d'installer une procédure spécifique au niveau d'un serveur pour manipuler la demande spécifique de service d'un client. Au lieu de cela des agents mobiles peuvent être adaptés aux besoins de l'utilisateur. Ceci a comme conséquence plus de dynamique.

2.2.4. Mobilité & adaptation d'un point de vue sécurité

La migration d'un agent d'un site vers un autre est de deux types. Au cours de la migration forte, l'agent se déplace avec ses données et son unité d'exécution. Ainsi, à la suite d'une

migration, l'exécution poursuit son cours comme si elle ne s'était pas arrêtée. Ce type de migration, qui se base sur une capture implicite de l'état d'exécution d'un agent, est cependant difficile et coûteux à réaliser surtout dans les environnements hétérogènes. La capacité de capture et de restauration de l'état d'exécution d'un agent constitue un point critique. La migration faible représente, quant à elle, la capacité pour un système de déplacer le code des agents accompagné seulement de données d'initialisation (et non de l'état complet). C'est donc au programmeur de gérer lui-même le mécanisme de reprise.

Il existe aussi deux grandes classes de prise de décision de la migration : la migration proactive et la migration réactive servant de base à l'autonomie de l'agent mobile. Avec la migration proactive, c'est l'agent qui initie le déplacement, ceci lui permet de garder intégralement son caractère autonome en lui offrant la possibilité de se déplacer dés qu'il termine sa tâche ou dés qu'il constate une menace. Dans la migration réactive, c'est le système qui initie les déplacements sans avoir besoin d'une demande explicite de l'agent. Cette migration réactive pose un problème du point de vue de l'autonomie. Cette dernière n'est, en effet, pas assurée puisque l'agent peut être déplacé sans son accord vers un site vers lequel il ne souhaitait pas se rendre. Cette possibilité est donc à bannir du point de vue sécuritaire.

En résumé, nous dirons qu'il existe différents types de migration de code [Fuggetta 1998] qui peuvent être classifiés selon deux grands aspects : le premier est la prise de décision de la migration (réactive/proactive) et le second est la capacité de déplacement du contexte d'exécution (forte/faible). La caractéristique principale d'un agent mobile est de disposer d'une certaine autonomie grâce à la migration proactive, qu'elle soit forte ou faible. C'est cette autonomie qui lui permettra de se protéger.

D'un autre côté, la capacité d'adaptation concédée aux agents mobiles dans les environnements dynamiques est essentiellement due à leur aptitude au déplacement en fonction de leurs besoins propres pour accomplir au mieux la tâche qui leur incombe. Elle leur permet aussi, dans le cas présent, de se protéger contre les mauvais comportements des hôtes malicieux. De plus, l'adaptation d'un agent est nécessaire lorsque l'environnement change et qu'il ne peut plus trouver les conditions de sécurité indispensables à son exécution. Ainsi, en examinant les critères de sécurité de son environnement, l'agent mobile pourrait décider de son exécution ou de sa migration vers un autre hôte plus sûr.

2.3. La Sécurité : un problème inhérent à l'emploi des 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 [Karnik 1998], [Vigna 1998], [Borselius 2002], [Alfalaleh 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.

L'adoption ainsi que le déploiement des agents mobiles reposent sur le climat de confiance qui doit régner entre [Bella 2004], [Jansen 2000a]:

1 Le propriétaire de l'agent mobile et l'agent mobile qui le représente
2 Deux agents mobiles appartenant à deux hôtes différents
3 L'agent mobile et l'hôte qui l'accueille (agent mobile éventuellement malicieux)

4 L'hôte et l'agent hébergé (hôte probablement malveillant)
5 Le réseau qu'ils utilisent.

A chacune de ces catégories correspond un ensemble de menaces susceptibles de générer des problèmes de sécurité complexes. Plusieurs de ces problèmes ne sont pas complètement résolus particulièrement au niveau de la quatrième catégorie [Sau-Koon 2000], [Bierman 2002].

FIPA[1] s'est intéressé au problème de sécurité des agents. Cependant, les normes courantes ne la traitent. FIPA a reconnu ceci et a récemment lancé un travail dans l'area3. Dans [Poslad 2000] une timide tentative [2] d'ajouter la sécurité à un système d'agent de FIPA est publiée. De son côté, NIST[3] a publié un rapport [Jansen 2000a] qui fournit une vue d'ensemble des menaces posée aux concepteurs des plates-formes d'agents et aux développeurs d'applications basées sur les agents. Ce rapport identifie également des objectifs génériques de sécurité et un éventail de mesures pour parer aux menaces identifiées et accomplir ces objectifs de sécurité.

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 [Laureiro 2001]. Les sections 2.4 et 2.5 de ce chapitre abordent en détails ces deux volets de la sécurité des agents mobiles.

2.4. Protection des hôtes accueillant les agents mobiles

Lorsqu'on désire protéger l'hôte contre un code malicieux potentiel, la mobilité du code impose les contraintes suivantes:

1 L'origine de l'agent mobile doit être authentifiée
2 L'agent mobile a été exposé à travers le réseau donc l'hôte doit vérifier l'intégrité du code qu'il vient de recevoir
3 Les actions exécutées par l'agent doivent être limitées par un contrôle d'accès.

La protection des hôtes visités contre des attaques menées pas 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 l'hôte de la malveillance des agents mobiles est de simplement limiter la fonctionnalité de l'environnement d'exécution afin de limiter les vulnérabilités.

---

[1] FIPA (Foundation for Intelligent Physical Agents, voir http://www.fipa.org) est une organisation des normes qui développe des standards pour agents logiciels permettant à des systèmes hétérogènes d'interagir. Son but est de décrire un ACL (Agents Communication Language), les ontologies et des protocoles de négociation permettant ainsi de définir parfaitement les interactions entre les agents.

[2] Il suggère l'implémentation de l'authentification des agents et des facilitateurs (entités offrant certains services aux agents) par la plate-forme d'agent ainsi que l'utilisation de canaux chiffrés. Cependant, aucun détail n'est inclus. La manière, dont la gestion de la clé et l'authentification devraient être réalisées, n'est pas spécifiée.

[3] NIST (National Institute of Standards and Technology, voir http;//www.nist.gov) est un institut de l'administration américaine qui détermine les standards employés par cette même administration.

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. Tandisque la seconde direction se consacre à la vérification de la sémantique du code mobile. Nous retrouvons ces deux directions dans les techniques présentées au sein de cette section.

2.4.1. Moniteur de référence

L'une des principales préoccupations dans l'implémentation d'un système agent est d'assurer que les agents ne sont pas capables d'interférer avec d'autres agents ou avec la plate-forme agent. Une approche communément utilisée pour éviter cette interférence est l'établissement de domaines isolés séparés pour l'agent et la plate-forme et le contrôle de tous les accès inter-domaines. Traditionnellement, ce concept appliqué à une base de calcul fiable est appelé moniteur de référence [Schneider 2001]. Un moniteur de référence est toujours invoqué. Il est incontournable et joue le rôle de médiateur pour tous les accès. Les implémentations du concept de moniteur de référence emploient un nombre de techniques de sécurité conventionnelles applicables à l'environnement de l'agent telles que :

- Des mécanismes d'isolation des processus les uns des autres et du processus de contrôle,
- Des mécanismes de contrôle d'accès des ressources de calcul,
- Des méthodes cryptographiques pour chiffrer les échanges d'informations,
- Des méthodes cryptographiques pour identifier et authentifier les utilisateurs, l'agent et les plates-formes et
- Des mécanismes d'audit de sécurité des événements pertinents survenant au niveau de la plate-forme agent.

Des exemples de moniteur de référence sont le gestionnaire de sécurité de l'environnement d'exécution de Java ainsi que l'interpréteur principal de Safe-TCL. Des domaines distincts sont établis par le moniteur de référence en se basant sur la politique de sécurité par défaut [Greenberg 1998]. Des techniques plus récemment développées visent la sécurité du code mobile et de l'agent mobile. Elles ont pour la plupart évolué le long des lignes traditionnelles préalablement citées. L'implémentation du moniteur de référence est supposée être résistante aux altérations et suffisamment petite pour être analysée et testée [Jansen 2000a].

2.4.2. Technique du Bac à Sable

La technique du bac à sable (sandbox) [Vigna 1998] fait référence aux coffres de sable utilisés par les démineurs pour faire exploser les engins en toute sécurité (*cf.* Figure 2.2). Il s'agit d'exécuter le code de l'agent mobile dans un environnement restreint (la sandbox) qui apparaît comme étant le système dans sa globalité [Gong 1998b]. Elle interdit, par exemple, l'accès au système de fichiers, l'ouverture d'une connexion réseau, accèder à des propriétés ou à des programmes du système local [McGrath 2001]. Ce contexte est construit par le système de contrôle d'accès aux ressources et dépend de la politique choisie sur le système. Les agents ont des privilèges limités et sont interprétés de manière sécurisée. Ainsi, un hôte peut exécuter un agent suspect dans le bac à sable sans trop se soucier des problèmes de sécurité. Cette approche peut facilement se mettre en place en utilisant des interpréteurs de code. Pour illustrer cette technique nous pouvons citer les applet Java exécutés à l'intérieur d'un navigateur Web. Java [Hauswirth 2000, Gong 1997] utilise, en effet, cette technique et fournit donc une solution partielle au problème posé par les agents malicieux.

L'interpréteur de Java comporte trois principaux composants : ClassLoader, Verifier et Security Manager [Rubin 1998, Hauswirth 2000, Gong 1997, McGraw 1996]. ClassLoader convertit le code distant en structures de données qui peuvent être rajoutées à la hiérarchie de la classe locale. Avant que le code distant soit chargé, Verifier réalise un ensemble de vérifications afin d'assurer que seulement le code légitime est exécuté [Hauswirth 2000, Gong 1997]. Enfin, les opérations effectuées par les classes distantes sont contrôlées par le Security Manager.

Le problème posé par cette technique réside dans le fait que l'échec, au niveau de l'une des trois étapes mentionnées plus haut, peut conduire à une violation de la sécurité. En effet, une classification incorrecte d'une classe distante en tant que classe locale permet à celle-ci de jouir de tous les privilèges accordés à une classe locale. Par conséquent, la politique de sécurité peut être violée [Rubin 1998]. Un autre inconvénient de cette technique est qu'elle augmente le temps d'exécution d'un code distant légitime [Wahbe 1993]. Par ailleurs, les applications qui sont exécutées dans cet environnement restreint sont elles mêmes rarement utiles.

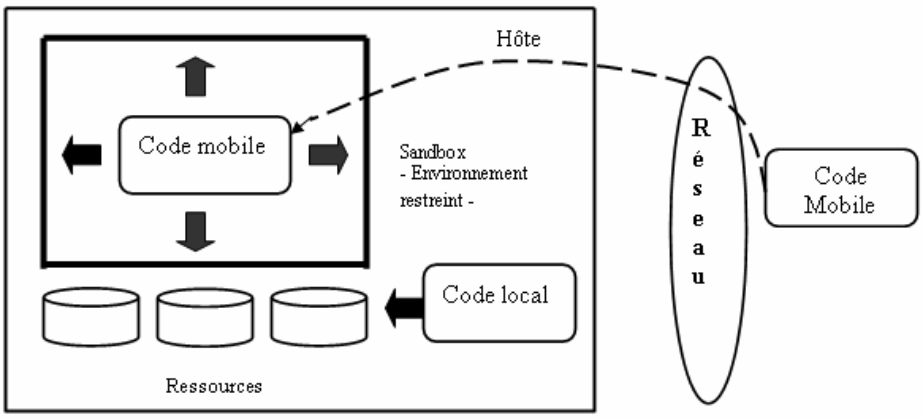

Fig.2.2. Le bac à sable (sandbox)

2.4.3. Signature du code

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 (*cf.* Figure 2.3). Cette technique permet d'obtenir une authentification de haut niveau pour les hôtes. Elle assure aussi l'intégrité du code pour l'hôte visité [Reiser 2000]. 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. Etant donné qu'un agent opère à la place d'un usager ou d'une organisation, les systèmes d'agents mobiles utilisent la signature comme une indication de l'autorité sous laquelle l'agent opère.

Ce modèle a été introduit par Microsoft dans le framework ActiveX. Java JDK 1.1 utilise aussi le modèle de signature du code: un applet avec une signature valide est exécuté comme un code fiable et donc autorisé à accéder à toutes les propriétés disponibles dans Java [ht4]. Un applet non signé ou dont la signature n'est pas valide est exécuté dans une sandbox.

Les signatures digitales profitent grandement de l'infrastructure à clé publique car les certificats contenant l'identité d'une entité et sa clé publique peuvent aisément être localisés et vérifiés.

L'inconvénient de cette méthode est qu'en fait un code signé est soit une garantie pour l'utilisation de toutes les ressources comme il peut ne pas du tout être exécuté. De plus, cette solution ne résout pas le risque fondamental lié au comportement de l'agent mobile et qui laisse le consommateur vulnérable aux dommages dus au code défectueux ou malicieux provenant d'une source fiable. Autrement dit, la signature du code ne peut révéler ce qu'il peut faire ni garantir que son exécution soit sûre [ht4, ht5]. Un agent mobile malicieux, peut employer ces privilèges pour causer non seulement des dommages directs à la plate-forme d'exécution mais d'ouvrir également une porte dérobée à d'autres agents malveillants en modifiant la politique d'acceptation sur la plate-forme. D'ailleurs, les effets d'une attaque d'un agent malveillant peuvent être ressentis seulement beaucoup plus tard, ce qui rend l'établissement du lien entre l'attaque et l'attaquant difficile [Rubin 1998]. De plus, cette technique est restrictive vis à vis des agents appartenant à des entités inconnues, puisqu'ils ne sont pas du tout exécutés. Plutôt que de compter uniquement sur la réputation du producteur du code, il serait plus prudent d'avoir une vérification du code réalisée par une partie fiable ou par un service d'évaluation.

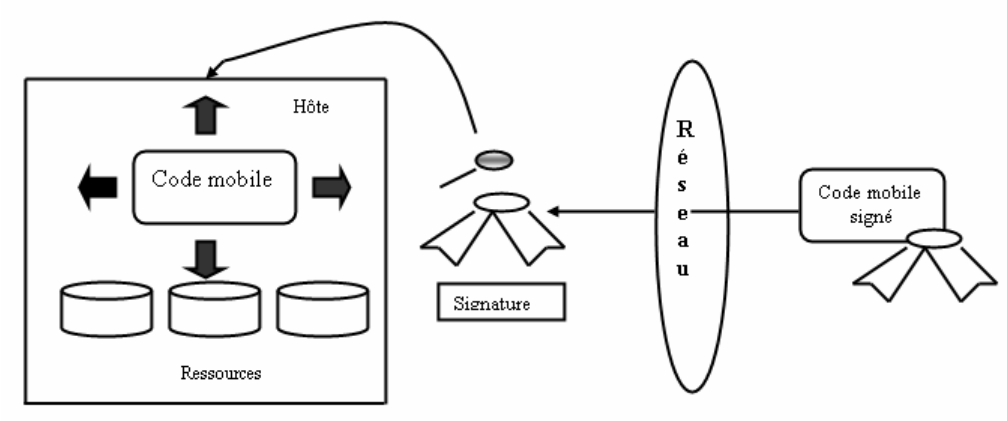

Fig.2.3. Signature du code mobile

2.4.4. Le contrôle d'accès

Tout programme qui s'exécute sur un système a besoin d'accéder à des ressources pour réaliser son travail. D'autre part, un agent mobile doit voir son environnement d'exécution restreint pour des soucis de sécurité. Les ressources dont il a besoin pour accomplir sa tâche doivent donc être rigoureusement contrôlées.

Pour améliorer les deux techniques précédentes, une politique de contrôle d'accès plus complexe est mise en place. En effet, la gestion des droits d'accès aux ressources par un agent doit passer par l'élaboration d'une politique de sécurité. Il est possible de choisir entre les politiques suivantes:

- L'hôte autorise l'accès à toutes les ressources (dangereuse)
- Tous les codes sont soumis à la même politique (restrictive mais simple)
- Une négociation a lieu pour chaque agent mobile (difficile à mettre en œuvre, mais c'est la meilleure).

Il est essentiel pour un système de gestion sécurisé d'être capable d'identifier les sujets par une authentification. L'identité du signataire du code, aide aussi à raffiner la définition de la politique de l'exécution à condition qu'elle puisse être établie par certains moyens tels que la technique de la clé publique. Par ailleurs, l'autorisation est nécessaire pour relier les droits aux sujets. Pour cela les droits et les permissions doivent être bien décrits. Le contrôle d'accès

doit alors appliquer les droits et les restrictions durant l'exécution. Il prévient l'accès illégal aux objets. De plus, tout canal d'information représentant une relation entre des entités a besoin d'être protégé. La confidentialité est satisfaite si un canal n'est accessible que par les participants autorisés. On peut donc voir cette politique de contrôle d'accès comme un raffinement d'une politique de bac à sable générale vers une politique spécifique à chaque application ou classe d'agents. En fonction des agents, l'hôte pourra autoriser l'accès à un ensemble précis de fonctionnalités. Le contrôle d'accès permet de mixer les deux premières techniques en offrant aux agents signés plus de fonctionnalité qu'un simple bac à sable sans pour autant accéder à toutes les fonctionnalités. Néanmoins, l'application d'un schéma d'accès a un coût puisqu'il est appliqué dynamiquement à l'exécution.

2.4.5. Les langages de programmation sécurisés

L'utilisation de langages dits "sécurisés" est aussi un moyen permettant de sécuriser l'exécution d'un agent mobile [Schneider 2001], [Vigna 1998]. Ces langages se basent sur les typages forts et sûrs pour garantir la non utilisation d'actions illégales telles que le fait de dépasser les limites d'un tableau ou de placer des instructions dans une chaîne de caractères. De plus, un certain nombre de précautions de programmation doivent être prises telles que proscrire l'appel de fonctions présentant des problèmes de sécurité[4].

L'approche basée sur les langages sécurisés donne des solutions d'exécution de la politique de sécurité qui peuvent être facilement étendues ou modifiées pour répondre aux exigences d'applications spécifiques. Concevoir un langage pouvant être interprété de façon sécurisée en accord avec une large variété de politiques de sécurité constitue, à l'heure actuelle, un défi. Le problème a été approché de quatre manières différentes : la distribution du code, la distribution du code intermédiaire, la distribution du code spécifique et la compilation just-in-time [Vigna 1998].

2.4.5.1. Distribution du code

Le code téléchargé est découpé, examiné[5] par un interpréteur sur le système local afin de fournir un contexte d'exécution sécurisé. Parmi les langages utilisant ce concept, on trouve :

- Safe-TCL [Tclet 1996] dispose de deux interpréteurs : un pour le code provenant d'hôtes de confiance et un second pour les autres hôtes. L'interpréteur « sûr » peut recevoir à tout moment une partie du code authentifié et étend de manière simple et temporaire le contexte d'exécution.
- Javascript est un langage script intégré au document html. Il souffre d'un manque évident de sécurité : le code est soit autorisé à s'exécuter pleinement ou ne l'est pas du tout.

La sécurité, dans ce cas, dépend de la capacité de l'interpréteur à analyser le code (ses spécifications et son implémentation).

2.4.5.2. Distribution de code intermédiaire

Cette approche est basée sur un code précompilé. Il en résulte que le code est plus proche du langage machine et n'a plus besoin d'être découpé pour être syntaxiquement analysé.

[4] A titre d'exemple, une fonction d'accès au système de fichiers pourrait être remplacée par une fonction qui ne pourrait écrire que dans un répertoire temporaire.

[5] Vérifier qu'il obéit aux restrictions syntaxiques et sémantiques du langage.

L'interpréteur fournit la sandbox sur laquelle repose la sécurité du système. Des exemples d'utilisation de cette approche sont :

- Java qui est un langage interprété, dynamique, typé et orienté objet. Il fournit une vérification du code exécutable, un chargement retardé des modules, une gestion automatique de la mémoire et la gestion d'exceptions. Des erreurs dans la spécification et dans l'implémentation de l'outil de vérification, de l'interpréteur et de la librairie standard ont mené à des failles de sécurité. Java s'oriente, actuellement, vers le concept d'agents mobiles (RMI, Voyager) ;
- Telescript est une technologie qui utilise le concept d'agent mobile en natif et fournit les mêmes services que Java. Un système d'authentification et de protection est mis en place.

Ce concept souffre d'une vitesse d'exécution moyenne du fait du recours à un interpréteur ainsi que d'un manque d'informations sémantiques susceptibles d'aider à la construction d'un contexte d'exécution.

2.4.5.3. Distribution de code spécifique (Byte code)

Cette méthode s'appuie sur les mécanismes de protection traditionnels, lourds en terme de performance, sur l'utilisation d'un compilateur connu qui garantit la sûreté du code délivré ainsi que sur l'emploi de technologies « d'isolation des fautes logicielles ». Ces dernières augmentent la quantité de code à traiter en insérant de multiples points de vérification de l'exécution correcte du code visant à limiter les effets de fautes.

2.4.5.4. Compilation just-in-time

Cette approche combine la distribution des codes source et intermédiaire qui sont compilés par un compilateur choisi par l'utilisateur. Ceci permet de garantir, selon le cas, un niveau de vérification élevé ou une vitesse d'exécution élevée. De plus, l'utilisateur peut choisir son mécanisme d'isolation des fautes.

Java utilise ce concept dans son compilateur JIT. C'est le cas aussi de la technologie Omniware [Lucco 1995] où chaque module jouit de son propre environnement sécurisé et où l'accès aux ressources est défini par le contexte de sa machine virtuelle OmniVM. Cette technologie a l'avantage d'utiliser le langage standart C++ qui est connu et rapide, et de fournir une sûreté d'exécution efficace.

2.4.6. Vérification du code

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"). Les bacs à sable exercent une certaine forme de vérification rudimentaire du code pour s'assurer que les types des opérandes d'une instruction sont corrects (vérification statique) ou encore pour localiser une tentative d'accès à une ressource protégée (vérification dynamique). Cependant, cela est fortement coûteux. Une autre approche est de vérifier automatiquement le code, avant son lancement, en s'appuyant sur une preuve de conformité (*cf.* Figure 2.4). Pour cela on peut utiliser l'approche PCC (Proof Carrying Code) [Necula 1998, Lee 1997]. 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 2000a, Lee 1997,

Loureiro 2001, Lee 2003]. 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].

En effet, PCC est considérée comme « auto- certifiante », car aucune cryptographie ou tierce partie de confiance n'est exigée. Elle requiert un contrôle statique peu coûteux du code à la suite duquel ce dernier pourra être exécuté sans aucun contrôle d'exécution. En outre, PCC est considérée en tant que « tamper-proof » puisque n'importe quelle modification du code ou de la preuve peut être détectée. Ces avantages rendent cette technique utile non seulement pour les agents mobiles mais aussi pour d'autres applications telles que les réseaux actifs ou les systèmes d'exploitation extensibles [Lee 1997, Lee 2003].

L'approche PCC permet donc d'exprimer des propriétés complexes de sûreté et semble prometteuse pour les propriétés de sécurité. Cependant, automatiser la génération de preuve reste encore un problème non trivial, et jusqu'à l'heure actuelle, les preuves doivent généralement être produites de façon manuelle [Loureiro 2001].

Les inconvénients de cette approche résident essentiellement dans les faits que :

- Actuellement, la génération de preuves n'est possible que pour des propriétés simples telles que la sûreté de la mémoire. La génération automatique de preuves pour des propriétés complexes est un problème très difficile
- Des limitations de la technique PCC incluent la taille potentielle de la preuve ainsi que le temps consommé par le processus de validation de la preuve [Lee 1997]

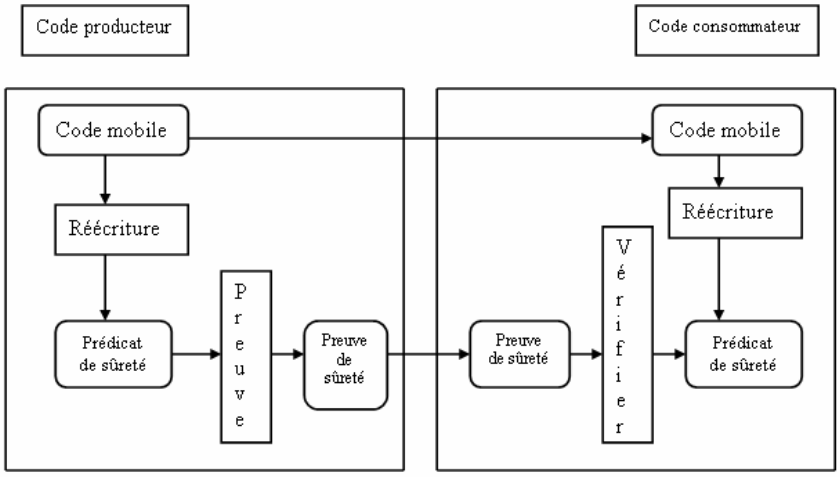

Fig.2.4. L'approche Proof Carrying Code

- PCC suppose que le producteur du code connaît toutes les politiques de sécurité appliquées sur l'hôte consommateur. Cette hypothèse n'est pas réaliste car les politiques de sécurité varient considérablement avec les hôtes et leurs environnements d'exploitation

- PCC place le fardeau sur le producteur du code qui doit identifier et prouver la sûreté des propriétés intéressant le consommateur.

Une alternative de l'approche PCC pourrait être l'approche MCC (Model Carrying Code) [Sekar 2001]. L'idée clé de cette approche est l'introduction de modèles de comportement de programmes. Le modèle du code est envoyé avec le code mobile au consommateur. Puisque ces modèles sont beaucoup moins complexes que les codes, le consommateur peut déterminer machinalement si un modèle est conforme aux politiques de sécurité en question. En se basant sut le résultat de compatibilité obtenu et sur la fonctionnalité du code projeté, le consommateur peut affiner ses politiques de sécurité et réessayer.

Afin d'établir la validité du modèle, le consommateur peut gérer l'exécution du code mobile et affirmer que son comportement est conforme au modèle. Une vérification d'exécution efficace (runtime-checking) est possible lorsque les politiques sont spécifiées en termes d'événements observables externes tels que les appels système réalisés par un programme afin d'accéder aux ressources du système d'exploitation.

Cette approche peut être combinée avec d'autres approches telles que la signature cryptographique pour l'authentification et l'intégrité du code et de son modèle, ainsi que PCC pour permettre au producteur de fournir une preuve formelle automatiquement vérifiable prouvant que le modèle est valide. Cette preuve est vérifiée par le consommateur du code avant que le modèle ne soit jugé correct.

2.4.7. Evaluation d'état

Durant le cheminement de l'agent mobile entre les différents hôtes, il transporte du code, des données statiques, des données collectées et un état d'exécution. L'état d'exécution constitue des données dynamiques créées pendant l'exécution de l'agent au niveau de chaque hôte. Il est utilisé comme entrée des calculs exécutés au niveau du prochain hôte. L'état de l'agent mobile change durant son exécution sur un hôte. Farmer et al. [Farmer 1996 a] ont introduit la technique d'évaluation de l'état (state appraisal) 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.

L'objectif de l'approche d'évaluation d'état est d'assurer qu'un agent mobile n'a pas été modifié. Cette technique est employée autant pour la protection de l'hôte que celle de l'agent mobile lui-même (*cf.* Section 2.4). Elle utilise des fonctions d'évaluation (appraisal functions) [Farmer 1996 a] 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.

Le producteur et/ou l'émetteur de l'agent mobile fournissent les fonctions d'évaluation qui constituent une partie du code de l'agent ; Ils appliquent des contraintes d'état afin de réduire la responsabilité et/ou les coûts de contrôle. Lorsque le producteur et l'émetteur de l'agent mobile le signent, les fonctions d'évaluation sont protégées contre la modification non détectable. Une plate-forme agent utilise ces fonctions pour vérifier l'état d'un agent arrivé et déterminer les privilèges que cet agent peut obtenir pour s'exécuter. Ces privilèges se basent sur les résultats des fonctions d'évaluation et sur la politique de sécurité qui est instaurée. En plus de s'assurer qu'un agent n'est pas devenu malveillant pendant son itinéraire, l'évaluation d'état peut également être employée pour désarmer un agent sciemment modifié [Farmer 1996 a]. Par ailleurs, cette technique fournit une manière flexible, pour un agent, de demander des permissions selon son état courrant et la tâche qu'il a besoin d'exécuter sur un hôte particulier [Swarup 1997, Farmer 1996 a].

L’application de cette théorie n’est pas clairement définie puisque l’espace état d’un agent peut être assez large. En effet, le problème principal de cette technique est qu'il n'est pas facile de formuler les propriétés appropriées de sécurité pour l'agent mobile et d'obtenir une fonction d'évaluation d'état qui garantit ces propriétés [Swarup 1997]. En outre, les fonctions d’évaluation peuvent être facilement formulées ; toutefois, plusieurs attaques subtiles demeurent difficiles à détecter car le producteur et l'émetteur de l’agent mobile ne peuvent pas prévoir certaines attaques et ne peuvent donc pas les inclure dans les fonctions d'évaluation et assurer la protection nécessaire [Jansen 2000a, Swarup 1997, Farmer 1996 a].

2.4.8. Historique de l’itinéraire

L’idée de cette approche est de maintenir un enregistrement authentifiable des plates-formes déjà visitées par un agent [Reiser 2000]. Constituer un historique de l’itinéraire « path history » exige de chaque plate-forme d’y rajouter une entrée signée indiquant son identité et l’identité de la prochaine plate-forme à visiter. La nouvelle plate-forme se base sur les informations contenues dans l’historique de l’itinéraire pour décider de l’exécution ou non de l’agent et des privilèges qu’elle doit lui accorder. Afin de prévenir la modification, la signature de la prochaine plate-forme peut déterminer si elle peut se fier aux précédentes plates-formes visitées soit par une révision de la liste des identités fournies ou par une authentification individuelle des signatures de chaque entrée de l’itinéraire pour confirmer l’identité.

Cette technique ne prévient pas la plate-forme des comportements malicieux mais exerce un effet de dissuasion puisque l’entrée signée de la plate-forme au niveau de l’itinéraire est non répudiable. L’inconvénient majeur de cette approche est que le contrôle de l’itinéraire devient de plus en plus coûteux au fur et à mesure que celui-ci grandit. [Jansen 2000a, Chess 1995, Ordille 1996]. Cette technique dépend aussi de la capacité des plates-formes à juger correctement la fiabilité de celles qui ont été déjà visitées. Il est, en effet, difficile de maintenir la confiance dans les agents qui ont visité un grand nombre de plates-formes. De même, qu’il n’est pas aisé de faire confiance à un agent dont l’itinéraire est inconnu à l'avance, par exemple un agent de recherche qui crée son chemin de manière dynamique. Par conséquent, construire des algorithmes pour l’évaluation de l’historique de l’itinéraire constitue un sujet de recherche intéressant [Ordille 1996].

2.4.9. Synthèse

De nombreux travaux ont souligné l’importance de la protection des hôtes contres des agents mobiles malicieux [Bella 2004], [Bierman 2002], [Borselius 2002], [Karnik 1998]. 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. Protection des agents mobiles

Les agents mobiles sont exposés à diverses menaces de la part des hôtes visités [Bierman 2002]. Ce problème est difficile car l'environnement visité a un contrôle total sur l'exécution de l'agent mobile. Pour comprendre ce que risque un agent mobile lors de son exécution sur un site malveillant, nous pouvons référencer les éléments transportés pouvant être cible d'attaque [Loureiro 2001] :

- Le Code : ensemble des instructions composant la tâche de l'agent.
- Les données statiques : données ne changeant pas durant les déplacements (telles que la signature)
- Les données collectées : ensemble des résultats obtenus au cours des déplacements réalisés par l'agent depuis son lancement.
- L'état courant : ensemble de données servant à l'exécution courante de l'agent.

Du point de vue des données, il est évident qu'un agent ne souhaite pas divulguer des informations critiques à n'importe quel site. Par exemple, un site malveillant pourrait récupérer la signature d'un code et l'utiliser pour créer un nouvel agent afin de s'introduire dans des environnements auxquels il n'a normalement pas accès. Pour le code, un agent transporte un savoir-faire propre à son concepteur qui pourrait tomber aux mains de ses concurrents.

Il est possible de distinguer trois grandes catégories d'attaques que des hôtes malicieux pourraient mener : l'inspection (espionnage de données et du code), la modification (altération des données et du code) et le rejeu (la réexécution du code) [Zachary 2003]. L'inspection consiste à examiner le contenu de l'agent, ou le flot d'exécution afin de récupérer des informations sensibles transportées par l'agent mobile. La modification est réalisée en remplaçant certains éléments (donnée ou partie de code) de l'agent mobile dans le but de conduire une attaque. Le remplacement du code incitera l'agent à effectuer des opérations malveillantes sur les futurs hôtes à visiter. Tandis que le remplacement des données permet à l'hôte malicieux de manipuler l'agent mobile à son avantage. Le rejeu, quant à lui, s'obtient en clonant l'agent puis en exécutant le clone dans plusieurs configurations pour retrouver le savoir de l'agent.

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 [Hohl 1999], [Karjo 1998], [Biehl 1998], [Mir 2002]. Enfin, protéger la confidentialité revient à cacher le code et l'état de l'agent mobile vis-à-vis du site qui l'exécute [hohl 1997], [Riordan 1998], [Sander 1998b], [Bennet 1999], [Beimel 2000], [Algesheimer 2001], [Yunguick 2002], [Dadon-Elichai 2004].

Plusieurs approches de protection de l'agent mobile ont été proposées. Elles tentent de garantir l'accès de l'agent mobile à des hôtes dans lesquels il peut avoir toute confiance ou de déceler ceux qui sont malveillants. Elles visent principalement à détecter les attaques ou à les rendre inefficaces. Ces différentes approches sont détaillées au sein des sections suivantes.

2.5.1. Approches cryptographiques

Les approches les plus connues sont la trace d'exécution (execution tracing) ou l'encapsulation des résultats partiels (Partial Result Authentification Codes) ou encore la fonction cryptographique (hiding function).

2.5.1.1. Trace d'exécution

L'approche de la trace d'exécution [Vigna 1997], [Vigna 1998] a été proposé 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. La technique se présente comme suit :

Quand une plate-forme A reçoit un agent de la plate-forme d'origine, et veut l'émettre vers une plate-forme B, après l'avoir exécuté, elle envoie un message signé contenant son code p, son état SA, et sa trace d'exécution TA. B sauvegarde ce message. Ainsi, A ne peut pas par la suite nier avoir envoyé l'agent. B envoie ensuite un reçu signé consistant en un hachage du message reçu, accusant réception de p, SA et TA. Ce reçu est stocké par A. La plate-forme B exécute ensuite l'agent, ce qui résulte en un nouvel état SB et une nouvelle trace TB. B signe ensuite le tout (p, SB et TB), puis l'envoie à la prochaine plate-forme C. De la même façon que B, C sauvegarde ce message et envoie un accusé de réception signé à B, que ce dernier sauvegarde. A possède donc la preuve que B a bien reçu de sa part p, SA et TA, et B la preuve que C a reçu de sa part p, SB et TB. Ce processus continue jusqu'à ce que l'agent revienne à sa plate-forme d'origine.

Si le producteur de l'agent a des soupçons quant à l'exécution de son agent et pense qu'il a été altéré, il demande les traces cryptographiques à toutes les plates-formes du parcours de l'agent. La vérification commence à la première plate-forme *A*, qui a la preuve d'avoir envoyé *p*, S_A et T_A à *B*. Le propriétaire exécute donc l'agent et vérifie s'il résulte effectivement en l'état S_A et la trace T_A, en s'exécutant normalement. Si ce n'est pas le cas, cela veut dire que l'agent a été altéré par *A*. Sinon, *B* doit donner au propriétaire les preuves d'exécution de l'agent comme décrit précédemment. Le processus de vérification continue de cette façon pour toutes les plates-formes du parcours de l'agent, jusqu'à l'état atteint au niveau de la plate-forme d'origine.

Cette approche souffre de 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. De plus, cette technique est difficile à utiliser dans le cas d'agents à plusieurs fils d'exécution (multi-threaded) [Vigna 1998], [Tan 2001]. Par ailleurs, des problèmes de confidentialité peuvent surgir lorsque ce type d'information est enregistré. Des améliorations ont été apportées à cette technique:

1°) Amélioration de Hohl:

Cette méthode a été adaptée par Hohl [Hohl 2000] pour pallier certains de ces problèmes. Ainsi, il propose qu'après son exécution sur une plate-forme, l'agent soit émis avec les variables d'état et les traces précédant et suivant l'exécution. De cette manière, la vérification de la bonne exécution sur la plate-forme précédente peut se faire directement dans la plate-forme suivante et non à la fin du parcours. Par conséquent, la détection d'une attaque se fait juste après l'attaque. La taille des traces se trouve réduite car on n'échange que les traces d'une exécution d'un hôte et non pas la somme des traces de tous les hôtes précédemment visités. Des optimisations de la taille des variables d'états (donnée et variables d'exécution) sont possibles. En effet, après son exécution sur une plate-forme, I'agent va être envoyé sur la prochaine plate-forme avec les variables d'états avant exécution et après exécution, ainsi que les traces. Cela permet à la plate-forme suivante de vérifier la bonne exécution de l'agent sur la plate-forme précédente. L'optimisation de la taille des variables d'états peut se faire avec une méthode qui permettrait de choisir la manière d'envoyer ces deux états (avant et après exécution) : soit deux états distincts, soit un état et une différence par rapport à cet état. De plus, une méthode de sélection entre une entrée entière fournie par la plate-forme ou le résultat de la première opération sur cette entrée est souhaitable. En effet, dans la trace, on conserve normalement le résultat de la première ligne de code qui utilise l'entrée. Mais, si l'entrée est de taille inférieure à ce résultat, il est préférable de la conserver. Le problème des entrées qui peuvent être manipulées par l'hôte sans protection n'est pas résolu. De plus, cette solution n'est pas efficace dans le cas de deux plates-formes hostiles qui collaborent et qui sont consécutives dans l'itinéraire de l'agent.

2°) Amélioration de Tan et Moreau:

Une nouvelle version de la technique de la trace d'exécution, proposée par Tan et Moreau [Tan 2001, Tan 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 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 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.

2.5.1.2. Encapsulation des résultats partiels

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, tandisque 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 l'encryptage, 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. un résultat partiel peut être, à titre d'exemple, l'offre la moins chère, dans le cas d'un agent mobile cherchant le meilleur prix d'un certain produit. Si ces résultats ne sont pas protégés, une plate-forme hostile peut effacer ou modifier toutes les offres qui sont meilleures que la sienne. L'encapsulation des résultats partiels permet d'aider à détecter la modification ou l'élimination de ces résultats. Si l'itinéraire de l'agent est connu d'avance, il suffit que chaque plate-forme appartenant à cet itinéraire signe numériquement son résultat partiel et/ou l'encrypte avec la clé publique du propriétaire de l'agent mobile. En général, cet itinéraire n'est pas connu d'avance. De plus, quelques hôtes peuvent être inaccessibles et engendrer des lacunes dans la liste des résultats partiels, provoquant ainsi des soupçons.

Les PRACs peuvent être générés en utilisant des algorithmes de cryptage symétriques. L'agent est équipé d'un certain nombre de clés de cryptage. A chaque fois que l'agent mobile quitte un hôte pour migrer vers un autre, son état ou bien certains résultats sont traités en utilisant une des clés, produisant un MAC (Message Autentication Code) sur le message. La clé utilisée est alors détruite avant que l'agent ne migre vers une nouvelle destination. Le PRAC peut être vérifié au dernier point pour identifier certains types d'altérations.

Dans ce mécanisme basé sur le code d'authentification de message, MAC (Message Authentification Code), proposé par Yee dans [Yee 1999], une clé k1 est remise à l'agent mobile avant qu'il ne quitte sa plate-forme d'origine. Celle-ci est utilisée par l'agent pour calculer le code d'authentification des résultats partiels, PRAC1 (Partiel Result Authentification Code), de l'offre de la première plate-forme sur laquelle il s'est exécuté. Une fois ce code calculé, il génère une autre clé k2 et détruit k1 :

$$k2 = f(k1) \qquad \text{où f est une fonction non inversible}$$

La clé k2 est utilisée pour calculer le PRAC2 de l'offre de la seconde plate-forme, et ainsi de suite. Toutes les valeurs du PRAC sont rajoutées aux données de l'agent mobile. Lorsque ce dernier retourne à sa plate-forme d'origine, son propriétaire peut vérifier la validité des PRAC puisqu'il connaît k1 et il peut ainsi détecter si une offre a été modifiée ou effacée. À noter qu'il est dans l'intérêt de la plate-forme courante que k1 soit détruite, car elle ne désire pas qu'une autre plate-forme puisse modifier son offre. De plus, détruire k1 assure l'intégrité future (forward integrity) des résultats d'encapsulation [Yee 1999] puisqu'elle garantit qu'aucune plate-forme à visiter à l'avenir ne peut modifier aucun des résultats des plates-formes précédemment visitées, car il n'y a aucune clé secrète pour calculer le PRAC pour ces résultats. Il est vrai donc qu'avec cette méthode les résultats partiels calculés dans des plates-formes de 1 à i ne peuvent pas être modifiés par une plate-forme malicieuse j ($i<j$). Toutefois, l'inconvénient majeur de cette approche est que la plate-forme malveillante j peut accéder à la valeur de la clé kj contenue dans l'agent, et calculer ainsi les valeurs subséquentes des clés

kl (l>j). Si l'agent retourne par la suite à cette plate-forme, celle-ci peut modifier son offre ainsi que toutes les offres des plates-formes suivantes. Ceci est dû au fait que les résultats partiels ne sont pas protégés. Yee [Yee 1999] a suggéré que les résultats pourraient également être chiffrés en utilisant la cryptographie asymétrique en laissant l'hôte produire une signature à la place de l'information afin de garantir l'intimité et l'intégrité.

Karjoth et al [Karjo 1998] ont proposé une intégrité future robuste (strong forward integrity), qui, garantit que la plate-forme visitée ne peut pas plus tard modifier ses propres résultats. L'approche de Karjoth dépend de la plate-forme visitée qui réalise le processus d'encapsulation à la place de l'agent. Celle-ci chiffre les résultats de l'agent en employant la clé publique du producteur de l'agent pour assurer la confidentialité des résultats. Elle emploie sa clé privée pour signer numériquement les résultats chiffrés ainsi que la chaîne hachée. Cette dernière relie les résultats de la plate-forme précédente avec l'identité de la prochaine plate-forme à visiter. Ceci empêche la plate-forme de changer ses résultats plus tard et assure ainsi une intégrité future robuste [Karjo 1998].

Young et Yung [Young 1997] ont suggéré une autre implémentation de cette technique où c'est l'agent lui-même qui réalise le processus d'encapsulation. Il utilise une implémentation spéciale du cryptage appelée « Sliding Encryption ». Cette technique chiffre uniquement une partie des données et obtient ainsi de petits morceaux de texte chiffrés à l'aide de la clé publique de son producteur/émetteur et qui seront décryptés plus tard au niveau de la plate-forme d'origine en utilisant sa clé privée. Ce procédé est particulièrement approprié à certaines applications où l'espace mémoire est coûteux comme le cas des smartcards [Karjo 2000].

La technique du *PRAC* a été améliorée par Karnik [Karnik 1998] pour assurer l'intégrité des résultats partiels. Dans le *PRAC* amélioré, aucune plate-forme n'est capable de modifier les résultats d'aucune autre plate-forme dans le cas d'une visite ultérieure, même pas les siens. Ceci est effectué en réalisant une chaîne avec les valeurs de hachage des résultats partiels, de façon à ce que le résultat partiel obtenu au niveau de chaque plate-forme est relié à l'identité de la plate-forme suivante et est signé numériquement. La chaîne de hachage est obtenue en appliquant une fonction de hachage cryptographique à sens unique, telle que *SHA-1*. Plus précisément, la plate-forme d'origine P_0 choisit un nombre aléatoire r_0 et calcule O_0, qui est envoyé à la plate-forme P_1 :

$$O_0 = D_{p0}(E_{p0}(o_0,r_0),h(r_0,P_1))$$

Où o_0 représente l'instance de l'agent dans la plate-forme d'origine, $D()$ une opération de signature, $E()$ une opération d'encryptage à clé publique et $h()$ une opération de hachage. Ensuite, chaque plate-forme suivante P_i, choisit un nombre aléatoire r_i et protège ses résultats partiels o_i en les encapsulant de la façon suivante :

$$O_i = D_{p0}(E_{p0}(o_i,r_i),h(O_{i-1},P_{i+1}))$$

Le fait d'inclure $h(O_{i-1},P_{i+1})$, permet au résultat partiel d'être chaîné au résultat partiel précédent, et à l'identité de la plate-forme suivante. Une plate-forme P_i envoie tous les résultats partiels précédents O_k, $(k=0,\ldots, i)$ à la plate-forme suivante P_{i+1}.

Cette méthode ainsi modifiée assure l'intégrité, la confidentialité et la non répudiation des résultats partiels. De plus, le propriétaire de l'agent peut vérifier, une fois son agent de retour, sa bonne exécution et désigner la plate-forme coupable en cas d'attaque grâce aux O_k. Toutefois, 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.

2.5.1.3. Fonction cryptographique

Cette technique est basée sur l'exécution, sur une plate-forme d'agent, d'un programme englobant une fonction encryptée. Elle assure que cette plate-forme ne connaîtra rien de substantiel sur la fonction encryptée. Abadi et Feigenbaum [Abadi 1990] ont suggéré la version initiale de cette technique. Leur solution est interactive et exige plusieurs rounds d'échange de message avec la plate-forme d'origine. Cependant, la solution interactive ne convient pas au scénario d'agent mobile puisque celui-ci opère de façon autonome sans beaucoup d'interactions avec sa plate-forme d'origine.

De leur côté, Sanders et Tschudin [Sander 1998a], [Sander 1998b] ont décrit une approche non interactive pour codifier une protection basée sur l'exécution de fonctions encryptées. 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. Supposons que l'agent veuille signer numériquement des données fournies par la plate-forme qui l'accueille. Même si la clé privée de l'agent est encryptée, il faut qu'il l'a décrypte avant d'effectuer la signature, ce qui permettrait alors à la plate-forme de l'intercepter. L'approche décrite par cette méthode pour éviter ce problème est d'encrypter la fonction qui permet d'effectuer la signature. Le résultat de l'encryptage de la fonction f est la fonction E(f), qui permet d'avoir un résultat encrypté. Le programme qui permet d'effectuer ces calculs se trouve dans l'agent mobile. Ainsi, pour un résultat x produit dans la plate-forme courante, le résultat signé est E(f)(x). Ce résultat ne peut être décrypté que par le propriétaire de l'agent, puisqu'il est le seul à connaître $E^{-1}(f)$. Le problème est énoncé par Sanders et Tschudin comme suit : Alice possède un algorithme qui calcule une fonction f. Bob possède une donnée x et désire calculer f(x) pour elle, mais Alice ne veut pas que Bob connaisse f. De plus, Bob n'a pas besoin d'interagir avec Alice durant le calcul de f(x).

Il est primordial de distinguer entre une fonction et le programme qui l'implémente. Les fonctions peuvent être cryptées de façon à ce que leur transformation peut encore être implémentée comme un programme. Le programme résultant consiste en des instructions en texte clair que le processeur comprend. Cependant, ce dernier ne comprend pas la « fonction programmée ».

Supposons qu'il soit possible de transformer la fonction f en une autre fonction E(f). P(f) désigne le programme qui implémente la fonction f. Un protocole pour le calcul non interactif à l'aide de fonctions encryptées est donné ci-dessous (*cf.* Figure 2.5) :

1 Alice encrypte f.
2 Alice crée un programme P(E(f)) qui implémente E(f).
3 Alice envoie P(E(f)) à Bob.
4 Bob exécute P(E(f)) sur la donnée x.
5 Bob envoie P(E(f))(x) à Alice.
6 Alice décrypte P(E(f))(x) et obtient f(x).

Un schéma général basé sur cette idée est décrit comme suit :

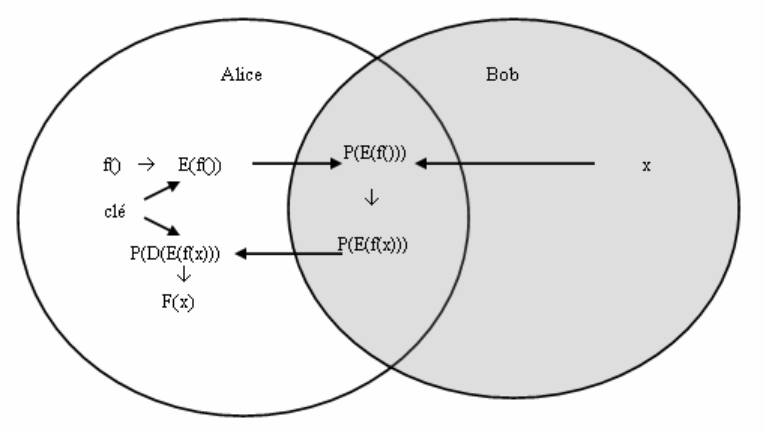

Fig. 2.5. Alice ne connaîtra jamais x et Bob ne connaîtra jamais f

F est supposée être une fonction rationnelle qui est facilement inversible. Posons E (f)= s o f. Pour décrypter, Alice inverse et calcule s^{-1}o E (f). Cette approche offre la confidentialité de l'exécution puisqu'il devient possible d'exécuter des agents sur des plates-formes non fiables. Autrement dit, en supposant que la fonction est un algorithme d'encryptage ou de signature qui contient une clé, l'agent a la capacité d'encrypter une information ou la signer sans révéler la valeur de la clé utilisée. Toutefois, cette technique demeure théorique car le codage est appliqué uniquement à un ensemble restreint de fonctions. Ce sont les fonctions polynomiales et rationnelles [Sander 1998b, abadi]. Cette restriction rend l'approche peu réaliste car il est très rare de rencontrer des programmes comportant uniquement des fonctions polynomiales ou rationnelles. Il existe, de surcroît, un problème de décomposition car on ne connaît pas d'algorithme polynomial pour décomposer des fonctions rationnelles à plusieurs variables. Des méthodes de génération de fonctions rationnelles, pour encrypter f, qui sont facilement inversibles ont été proposées par Shamir [Shamir 1994].

Dans le cas où ces restrictions sont surmontées, cette approche offre une protection de l'agent facilement prouvable et des coûts de protection vraisemblablement bas. L'idée est donc élégante et le défi est de pouvoir trouver les arrangements appropriés de chiffrage qui peuvent transformer des fonctions arbitraires. Sanders et Tschudin proposent d'étudier des arrangements homomorphes algébriques de chiffrage (algebraic homomorphic encryption scheme) en tant que candidats possibles. Leurs résultats préliminaires semblent prometteurs, et pourraient constituer une base pour découvrir d'autres classes de fonctions. Bien que très puissante, cette technique n'empêche ni la re-exécution, ni l'extraction de code ou le déni de service [6][Sander 1998b]. Récemment, Barak et al. [Barak 2001] ont démontré que l'obtention d'une justification théorique pour la capacité d'un programme à cacher totalement son information semble utopique.

2.5.2. Approche de la clé environnementale

Cette approche, telle que proposée dans [Riordan 1998], permet à l'agent d'exécuter des actions prédéfinies, quand certaines conditions sont vérifiées au niveau de sa plate-forme d'exécution. Le procédé de la clé environnementale est quelque peu apparenté à la manière dont sont maintenus les mots de passe dans les logiciels d'exploitation modernes (tels que le

[6] Le déni de service est une attaque menée par l'hôte d'accueil qui empêche l'agent mobile de s'exécuter normalent en lui interdisant, par exemple, de communiquer avec d'autres agents. Son objectif est d'arrêter ou de retarder l'exécution de l'agent mobile.

stockage du hachage d'un mot de passe). Ce procédé est employé pour déterminer si les tentatives d'identification sont valides.

La technique de la génération de la clé environnementale peut être utilisée lorsqu'une plate-forme désire communiquer avec une autre plate-forme en lui envoyant un message, et cette dernière ne peut lire ce message que si certaines conditions sont réunies. Ceci peut être réalisé en émettant un agent mobile transportant un message crypté. Le message chiffré peut inclure des données ou/et du code exécutable. Ni l'agent mobile ne peut prévoir avec précision sa propre exécution, ni la plate-forme qui l'accueille ne peut deviner la tâche que va exécuter l'agent. Celui-ci attendra au niveau de la plate-forme de réception jusqu'à ce qu'un certain état environnemental se produise. Dès que la condition environnementale est atteinte, une clé d'activation est générée afin de déchiffrer le message transporté par l'agent. Cette technique assure ainsi que, ni la plate-forme, ni un observateur extérieur ne peut découvrir le message déclencheur simplement en lisant le code.

La condition qui déclenche le mécanisme est généralement cachée soit par une fonction de hachage, soit à travers un mécanise d'encryptage. Supposons que l'agent transporte une chaîne de caractère N et une valeur de hachage $H=h(N \oplus S)$. Il recherche alors la plate-forme qui contient la chaîne de caractère s qui vérifie $h(N \oplus s)= H$. Le propriétaire de l'agent ne veut pas dévoiler ce que l'agent recherche ou accomplit, car ceci peut révéler ses intentions à un concurrent ou à un hôte malicieux. L'agent calcule alors $h(N \oplus s)$ pour chaque chaîne de caractères et compare le résultat à H. Quand la bonne valeur est trouvée, l'agent calcule la clé de décryptage $h(s)=h(S)$, décrypte et exécute un code, puis envoie, par exemple, un mail au propriétaire lui indiquant que s vérifiant H a été trouvée dans la plate-forme courante P.

La clé d'activation pourrait être dissimulée dans un canal fixe de données (serveur de nouvelles, pages web, etc.). Si ce dernier est, par exemple, un système de fichiers, alors la clé pourrait être dissimulée dans un fichier ou être un résumé d'un nom de fichier. D'autre part, si le canal de données est un message de courrier électronique, la clé d'activation pourrait être une chaîne de caractères à l'intérieur de ce message ou bien un résumé de ce message. L'utilité de cette méthode est fonction des entités qui ont accès directement ou indirectement au canal, de celles qui peuvent manipuler les données et de la manière dont varie l'observation du canal en fonction des observateurs. Il est aussi possible de générer une clé à partir du temps : certaines méthodes permettent de générer la clé seulement avant une certaine date, alors que d'autres ne le permettent qu'après une date. En utilisant les deux, on peut arriver à générer une clé seulement pendant un intervalle de temps fixé.

Cette approche accule le propriétaire de l'agent à spécifier un ensemble de contraintes sur l'environnement sur lequel l'agent mobile sera exécuté. Filiol, dans [Filiol 2005], utilise la clé environnementale comme une technique de base pour interdire l'analyse du code d'un virus. Tandis que Tschudin [Tschudin] a exploité l'idée de la génération de la clé environnementale pour la mort programmée d'un service mobile, i.e., le suicide du service mobile quand on n'en a plus besoin. Riordan et al. [Riordan 1998] ont proposé, quant à eux, l'utilisation d'un ″agent naïf″ (clueless agent). C'est un agent qui possède du code encrypté et une fonction qui permet de générer la clé de décryptage en fonction de l'environnement. Si l'environnement n'a pas les conditions pour générer la clé, il est impossible de deviner la fonction de I'agent.

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. Il est possible,

cependant, d'appliquer cette technique en même temps que d'autres mécanismes de protection pour des applications spécifiques sur les plates-formes appropriées.

Cette technique peut être utilisée lorsque le récepteur ne connaît pas les conditions exigées pour l'exécution du service requis. Ce cas correspond à notre problème d'analyse du comportement de l'agent mobile. La clé environnementale est utilisée, dans notre approche, pour estimer la confiance que l'agent mobile peut avoir en l'hôte visité.

2.5.3. Approches basées sur un matériel de confiance

Il existe quelques études utilisant du matériel pour fournir une base de calcul de confiance dans les systèmes d'agents mobiles [Funfrocken 1999, Yee 1999]. Il s'agit d'installer des dispositifs câblés sur les hôtes exécutant les agents mobiles. Puisque le matériel ne peut pas être modifié par une attaque logicielle, l'exécution sur le matériel de confiance est sûre même si les hôtes sont malveillants. Etant donné que les agents sont exécutés sur des dispositifs qui ne sont pas contrôlés par les hôtes, ces derniers ne peuvent pas lire le code et les données des agents s'ils sont chiffrés.

L'idée de ce type d'approches est d'introduire un composant matériel résistant aux altérations (tamper-resistant hardware) au niveau de chaque plate-forme, qui assure que l'agent s'exécute sans qu'il soit accédé ou modifié illégalement par cette dernière. Ce composant constituerait alors l'environnement d'exécution de l'agent sur lequel la plate-forme n'a pas de contrôle, et qui interagirait avec lui en client/serveur. La résistance aux altérations est telle que, la plate-forme devrait perdre plus de temps/argent à essayer de rompre le mécanisme de protection qu'elle n'en gagnerait en le rompant effectivement.

Ces approches, basées sur du matériel de confiance, sont utilisées par les chercheurs pour garantir un certain comportement d'un système. Une conception détaillée d'un système d'agents mobiles avec un composant matériel résistant aux altérations, permettant un échange d'agents mobiles sécurisé, et basé sur un protocole à clé publique est décrite dans [Wilhelm 1999].

Par ailleurs, Herzberg et pinter [Herzberg 1985] décrivent un dispositif qui peut être utilisé pour protéger un logiciel contre la piraterie. L'approche de Yee et tygar [Yee 1995], plus récente, assure que le système fonctionne de manière sécurisée.

Ce type d'approche est effectivement utilisé dans la protection des logiciels contre le piratage. L'exécution du logiciel est reliée à l'existence d'un périphérique spécial (dongle) que le logiciel teste au début ou en cours d'exécution [hohl 1997]. Cette protection peut toutefois être contournée par l'élimination (ou saut) de la partie test du code de l'agent [Mana 2002] (*cf.* Figure 2.6). Une autre méthode de sécurisation consiste à utiliser un coprocesseur dédié à l'exécution de l'agent. Ce dernier s'exécute exclusivement à l'intérieur de ce périphérique et ne dialogue avec le site visité qu'à travers une interface sécurisée. Une approche de ce type a été proposée par Wilhelm [Wilhelm 1997] et est basée sur un environnement d'exécution de confiance TPE (Trusted Processing Environment). Le périphérique TPE est fabriqué sous le contrôle d'une autorité de confiance et dispose d'un certificat et d'une politique de sécurité. Dans cette approche, les agents sont encryptés par la machine de leur propriétaire et ne sont plus exécutés par le site visité mais par le périphérique TPE. Les agents restent encryptés durant tout leur itinéraire. Lorsqu'ils arrivent à leurs sites de destination, ils accèdent au périphérique TPE où ils seront décryptés puis exécutés. De telles approches offrent une sécurisation très puissante des agents mobiles puisqu'elles les protègent aussi bien sur le site d'exécution que durant leur cheminement à travers le réseau. Néanmoins, le périphérique TPE

n'est pas facile à fabriquer ce qui explique son coût élevé. De plus, l'environnement d'exécution de confiance TPE ne présente pas les performances d'un ordinateur, ce qui réduit l'efficacité d'exécution de l'agent. Par ailleurs, le problème d'exécution de plusieurs agents en parallèle reste posé.

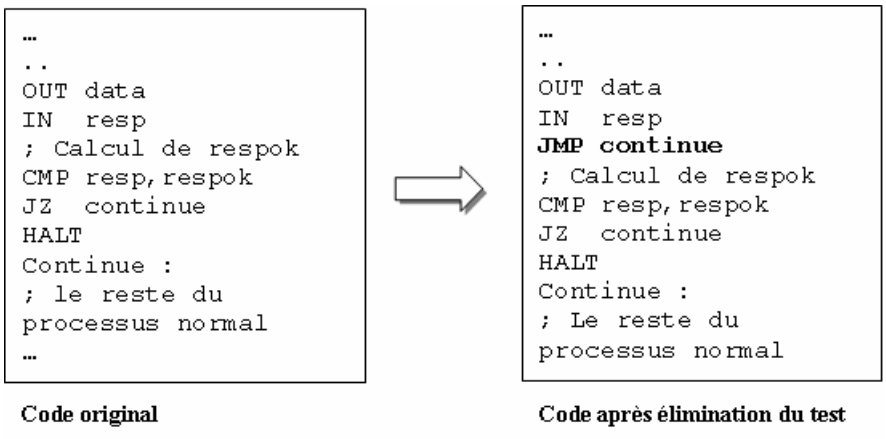

Fig.2.6. Attaque d'un logiciel protégé par un périphérique

Ces solutions basées sur un tel composant matériel sont très coûteuses et ne sont pas viables à grande échelle, car elles nécessiteraient que toutes les plates-formes en soient dotées.

Dans le but de généraliser l'utilisation du concept du périphérique spécial pour l'exécution d'un agent, Mana [Mana 2002] a proposé l'utilisation de cartes à puces. Son 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 (*cf.* Figure 2.7).

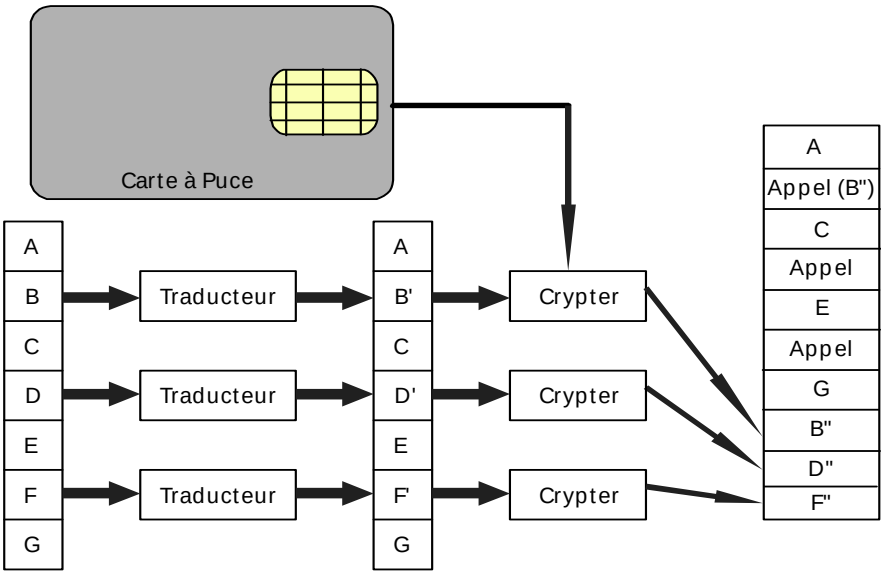

Fig.2.7. Sécurisation à base de cartes à puces

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 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. La déduction des fonctionnalités des sections cachées permet une attaque à la boîte noire [Boneh 1996]. Les sections cachées peuvent être appelées par un code écrit par celui qui attaque afin de réaliser des tâches au nom de l'agent et sur le compte de son propriétaire. A titre d'exemple, la fonction signature peut être exploitée pour signer un document autre que celui choisi par l'agent.

Cette solution est limitée par le fait que la plate-forme peut contrôler la communication entre la partie de l'agent s'exécutant dans son environnement d'exécution, potentiellement hostile, et celle, sécuritaire, s'exécutant dans la carte à puce. Ainsi, lors des interactions avec la partie du code s'exécutant dans la carte à puce, la plate-forme peut rajouter des informations en lui faisant croire que c'est le résultat effectivement trouvé par l'agent. La plate-forme peut rajouter par exemple des articles plus chers et les faire signer par la partie s'exécutant dans la carte à puce. Ce problème peut être partiellement résolu si cette dernière rajoute des limitations pour chaque commande avant de la signer (par exemple, valide pour seulement des achats inférieurs à 1000 DA). De plus, le matériel de confiance doit être utilisé avec précaution et peut sembler offrir plus de sécurité qu'il ne le fait réellement. L'agent doit être encore capable de communiquer avec les ressources de la plate-forme locale (la partie de l'agent qui est sous le contrôle de l'hôte non fiable) pour, à titre d'exemple, interagir avec la base de données locales. De telles interactions peuvent encore être interceptées par l'hôte non fiable. Dans le cas où des interactions avec des bases de données sont exigées, il est possible d'avoir recours à des techniques permettant de retrouver l'information sans la révéler (telles que la technique de recherche d'information privée: Private Information Retrieval- PIR [FIPS 2000]).

Deux efforts continus, le TCG (Trusted Computing Group [TCG 2001]) et le NGSCB (Next-Generation Secure Computing Base [Security 2003] se sont fixés pour but de produire des sous-systèmes de calcul (d'ordinateur) de confiance. Ces technologies semblent avoir le potentiel de fournir un environnement de confiance pour les agents mobiles. La fonctionnalité fournie inclut une exécution protégée, un stockage protégé ainsi qu'une authentification de la plate-forme.

Ces approches basées sur un matériel de confiance sont jugées être parmi les plus puissantes. Néanmoins, elles présentent un coût trop élevé.

2.5.4. Techniques d'obscurcissement

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 [Collberg 1997]. En se basant sur la génération de nombres aléatoires, un obscurcisseur peut

réaliser habituellement plusieurs transformations différentes. Collberg [Collberg 1997] fournit un aperçu détaillé sur cette question.

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 (obfuscating, mess-up algorithms). Hohl décrit les caractéristiques que devront avoir ces algorithmes : l'algorithme doit être paramétrable avec un grand nombre de valeurs possibles et il ne doit pas être possible de casser la protection avant l'exécution. Il existe divers algorithmes de confusion tels que la recomposition de variable ou la conversion des éléments de flux de contrôle dans des branchements dépendants de valeurs [Hohl 1998]. Le calcul avec des fonctions cryptographiques pourrait aussi être utilisé avec leur restriction (*cf.* Section 2.4.3.1). L'agent masqué n'est pas difficile à réaliser et les capacités de protection peuvent être ajustées par le choix de différents algorithmes de masquage. Cependant, des attaques existent contre ces algorithmes. Elles peuvent être efficaces si l'algorithme de masquage est connu.

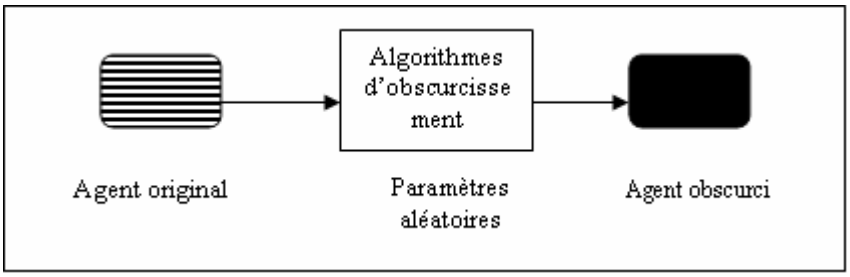

Fig.2.8. Approche Black box à temps limité

Puisque cette technique ne permet pas la protection absolue du code [Barak 2001, D'Anna 2003], on suppose l'existence d'un intervalle de temps durant lequel l'agent est protégé, i.e. 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, à titre d'exemple, dans le cas de monnaie électronique. Elle ne peut cependant pas être utilisée pour des données qui ont une longue durée de vie, comme les numéros de cartes de crédit. De plus, cette technique est difficile à appliquer en pratique, car il n'y a pas d'approche générale garantie par un algorithme pour quantifier l'intervalle de protection.

Il s'agit, en fait, de produire à partir d'un agent A, un autre agent B (cf. Figure 2.8) analogue au premier sur le plan fonctionnalités mais ayant un code difficile à analyser [Wilhelm 1997]. Ceci peut être assuré par le fait d'introduire un désordre au niveau du code de l'agent. Hohl propose la violation totale des règles du génie logiciel afin de produire un code non lisible. Parmi ces règles nous citons :

- Attribuer des noms significatifs aux variables.
- Ecrire un code modulaire.
- Utiliser des structures de contrôle qui simplifient le programme.

Ces règles seront respectées au niveau de l'agent source A. Pour passer à l'agent sécurisé B, on crée tout d'abord à partir des variables originelles un nombre différent de nouvelles variables portant des noms non significatifs. Comme indiqué dans la Figure 2.9, chaque nouvelle variable est composée de fragments de quelques variables originelles.

Après l'étape de génération des variables, on procède à la déstructuration du code. Pour cela on élimine les variables locales et on les remplace par des variables globales. Ensuite, on remplace l'appel procédural par le corps des procédures et on utilise la structure de contrôle "GOTO" pour remplacer les autres structures [hohl 1997]. Enfin, on pourra insérer des fragments de code mort pour améliorer la protection. Seulement il faut se méfier des mécanismes de détection de codes morts.

Les techniques d'obscurcissement assurent la protection de la confidentialité des agents. Pour assurer l'intégrité de ces derniers, Hohl propose l'utilisation de la signature électronique de l'agent en totalité et en association avec sa date de validité après laquelle l'agent ne sera plus accepté par aucun site et ses actions ne seront plus valides [hohl 1997].

Cette approche prévient toute tentative d'analyse sophistiquée du code [Beaucamps 2007] et toute re-exécution de l'agent mobile. A l'inverse de l'approche des fonctions cachées, celle-ci permet l'utilisation de fonctions compliquées. Toutefois, on peut lui reprocher le manque de fondement théorique. De plus, elle ne garantit la protection de l'agent mobile que durant un certain intervalle de temps. Ce qui concerne uniquement les agents mobiles qui transportent du code ou des données à courte durée de validité. Autrement, l'agent sera enregistré et analysé lentement afin de déduire les données qu'il transporte. En outre, un inconvénient sérieux à cette technique réside dans l'absence d'une approche pour la mesure de l'intervalle de protection fourni par l'algorithme d'obscurcissement. Cette lacune constitue un frein à son application dans la pratique. Il manque, en effet, un modèle qui donne le degré de masquage d'une variable et la complexité pour la retrouver. De plus, il faut déterminer à partir de quel intervalle cette protection est utile pour l'agent : les agents ont besoin de plus ou moins de temps pour effectuer leur tâche. Donc, si l'intervalle est trop faible, le nombre d'agents où cette protection pourra être utilisée sera réduit. Par ailleurs, aucune technique n'est actuellement connue pour établir les limites inférieures sur la complexité pour un attaquant pour arriver à réaliser la décompilation du code d'un agent. De plus, comparer la date d'expiration avec la date courante, nécessite des horloges synchronisées. Par ailleurs, l'agent doit subir encore un obscurcissement après une certaine période parce qu'un hôte peut obtenir les spécifications originales de l'agent blackbox après le temps d'expiration, et envoyer des informations à d'autres hôtes complices. Enfin, Luo [Luo 2002] note que l'élimination de l'appel procédural entraîne la perte de l'utilisation des bibliothèques sur site d'exécution.

Barak et al. [Barak 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 [D'Anna 2003] indique 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.

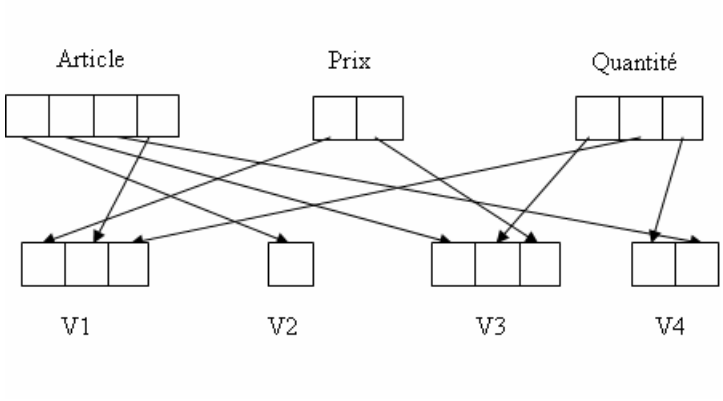

Fig.2.9. Génération de variables à noms non significatifs

Dans le même ordre d'idée, Bazzi [Bazzi 1998] a proposé une définition pour le masquage du code et pour la résistance aux modifications. Il décrit sa méthode qui consiste à rajouter du code de manière systématique et automatique en gardant la même fonction. Une proposition de solution pour les fonctions polynomiales est faite, le masquage et la résistance aux modifications sont prouvés. Il reste à mieux spécifier le problème du masquage du code en tenant compte notamment de la quantité de connaissances que l'hôte détient sur le code. Il est aussi important de définir une évaluation des algorithmes de masquage.

2.5.5. Approches basées sur les nœuds de confiance

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 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[7] 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.

[7] L'hôte qui initie la migration de l'agent mobile.

2.5.6. Approches basées sur la distribution du secret

Il existe des approches basées sur la subdivision du secret transmis en plusieurs petits secrets. C'est le cas de l'approche 'k-out-of-n threshold scheme' [Beimel 2000]. Cette idée apparaît également dans l'approche du code à la demande ou celle de la coopération d'agents.

2.5.6.1. Copies d'agent

Dans l'approche **« k-out-of-n threshold scheme »** [Beimel 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, i.e. n'est pas affectée par le nombre de fois qu'elle est effectuée, plusieurs copies de l'agent peuvent être lancées dans le réseau [Schneider 1997]. Par exemple, la recherche du meilleur prix pour un article est idempotente, mais le paiement du produit ne l'est pas. De cette façon, même si quelques copies sont corrompues, les autres vont effectuer la tâche correctement. Pour vérifier quelles copies sont corrompues et lesquelles ne le sont pas, des schémas de vote sont utilisés. Cette technique de tolérance aux fautes fonctionne avec la supposition que la majorité des plates-formes de migration ne sont pas hostiles.

Une des méthodes proposées dans [Schneider 1997] est basée aussi sur le schéma de partage de secret. Dans un tel schéma, un secret est partagé en n morceaux de façon à ce que au moins k d'entre eux sont nécessaires pour reconstituer le secret, mais (k-1) ou moins ne révèlent aucune information sur le secret. On suppose qu'il y a l plates-formes sur l'itinéraire, incluant la plate-forme d'origine qui est présente au début et à la fin du parcours. Le protocole fonctionne selon les étapes suivantes, comme illustré à la Figure 2.10.

- *Étape 1 :* La plate-forme d'origine connaît le secret *s* et le divise en utilisant un schéma de partage *(2k-1, k)*. *2k-1* morceaux, $p_1,\ldots, p_{2k-1}$, sont envoyés à *2k-1* plates-formes, et une copie de l'agent est envoyée avec chaque morceau.
- *Étape 2 :* Chaque plate-forme divise le morceau qu'elle a reçu en appliquant à son tour un schéma de partage *(2k-1, k)*. La plate-forme *i* envoie chacun des *2k-1* morceaux, $p_{2,i,1},\ldots, p_{2,i,2k-1}$, avec une copie de l'agent à une plate-forme de l'étape 3. Les indices du triplet indiquent respectivement le numéro de l'étape, la plate-forme générant les morceaux, et le numéro du morceau.
- *Étape 3 :* Jusqu'à l'étape *l-2*, chaque plate-forme à l'étape *j* concatène tous les morceaux qu'elle reçoit des plates-formes du niveau *j-1*. La concaténation effectuée par une plate-forme *i* est un nouveau secret $(p_{j-1,1,i} ||\ldots|| p_{j-1,2k-1,i})$, qui est lui-même divisé en *2k-1* morceaux, $p_{j,i,1},\ldots, p_{j,i,2k-1}$, en appliquant un schéma de partage *(2k-1,k)*. Chaque morceau est envoyé à l'étape suivante avec une copie de l'agent, à chacune des *2k-1* plates-formes de l'étape suivante.

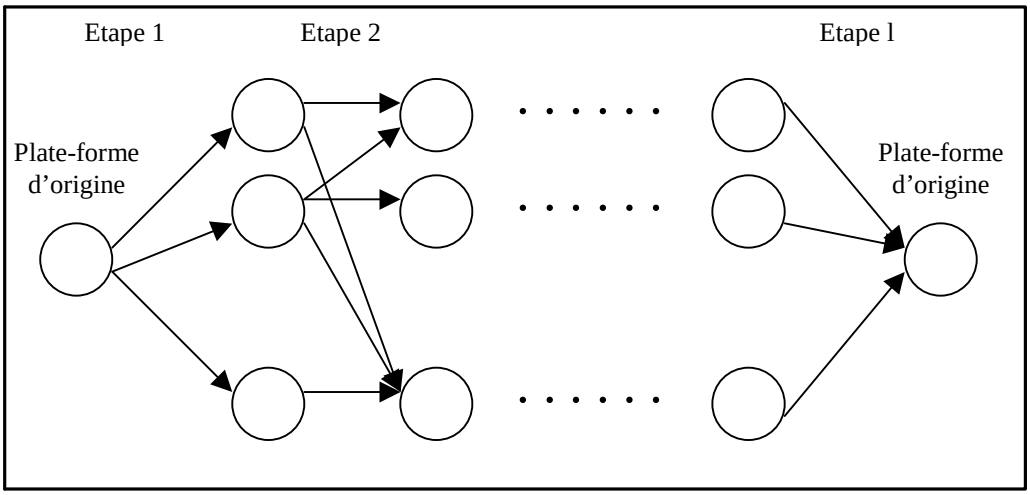

Fig.2.10. Représentation de la technique de copie d'agents

- *Étape l-1 :* Chaque plate-forme à l'étape *l-1* concatène tous les morceaux reçus des plates-formes de l'étape *l-2*. La concaténation composée par la plate-forme *i* est un nouveau secret $(p_{l-2,1,i} ||\ldots|| p_{l-2,2k-1,i})$, qui est envoyé à la plate-forme d'origine.
- *Étape l :* La plate-forme d'origine reçoit les *2k-1* concaténations $(p_{l-2,1,i} ||\ldots|| p_{l-2,2k-1,i})$, $i=1,\ldots,2k-1$, de toutes les plates-formes de l'étape *l-2*. Elle applique d'abord le schéma de partage pour récupérer la concaténation de l'étape *l-2*, en enlevant les morceaux de la première position dans chaque concaténation. Ensuite, elle continue à appliquer le schéma inverse pour récupérer le secret d'origine *s*.

Ce protocole fonctionne même si k-1 plates-formes hostiles ou moins collaborent à chaque étape. Si le secret d'origine ne peut pas être retrouvé, cela veut dire que la majorité des plates-formes ont collaboré, ce qui n'a pas beaucoup de chance d'arriver en général. Pour prévenir la destruction des morceaux du message par une plate-forme hostile, un schéma de partage de secret vérifiable devrait être utilisé. Un autre problème est que le nombre de morceaux augmente à chaque étape de calcul. Ceci peut être évité en utilisant un partage de secret proactif, comme suggéré dans [Schneider 1997].

Le problème de la confidentialité est résolu puisque 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.

L'idée du partage du secret est exploitée dans notre approche pour subdiviser un service (secret) en plusieurs tâches.

2.5.6.2. Approche du code à la demande

L'approche du code à la demande [Wang 2000] protège l'intégrité de l'agent mobile en utilisant un code de l'agent dynamiquement extensible. Au niveau de cette approche, de nouveaux modules de la fonction de l'agent peuvent être ajoutés et les modules superflus peuvent être supprimés durant l'exécution de l'agent. Cette approche améliore la confidentialité, réduit le coût de transport et aide à la récupération des agents à la suite d'attaques malicieuses. Néanmoins, cette approche exige une communication additionnelle avec l'hôte d'origine.

2.5.6.3. Agents coopératifs

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 (*cf.* Figure 2.11 extraite de [Allée 2001]). 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.

Roth [Roth 1998, Roth 1999] fait plusieurs hypothèses. La première est qu'aucun hôte sur l'itinéraire d'un agent ne coopère avec un hôte de l'itinéraire de l'autre agent. Il présume aussi l'existence d'un canal de communications. Enfin, il suppose que les identités sont données sans altération à l'agent.

Cette méthode est intéressante. Toutefois, elle est entachée d'un certain nombre d'inconvénients. En effet, mettre en place le canal de communications à chaque migration est une opération coûteuse. 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'interchanger les agents qui enregistrent 1'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.

Roth [Roth 1998] adapte un protocole de paiement basé sur le principe de deux agents coopérants. Chaque agent contient une partie de la monnaie électronique et les deux parties sont nécessaires pour effectuer un paiement valide. Par exemple, le premier agent peut obtenir une offre de la part d'une plate-forme et l'envoie au deuxième via un canal sécuritaire. Ce dernier peut vérifier si l'offre satisfait tous les critères. Si c'est le cas, il envoie sa partie de la

monnaie électronique de sorte que le premier agent puisse payer pour les articles indiqués dans l'offre. Ce protocole peut être généralisé à plus de deux agents. Pour certaines applications, il serait possible qu'un agent soit statique sur la plate-forme de l'utilisateur.

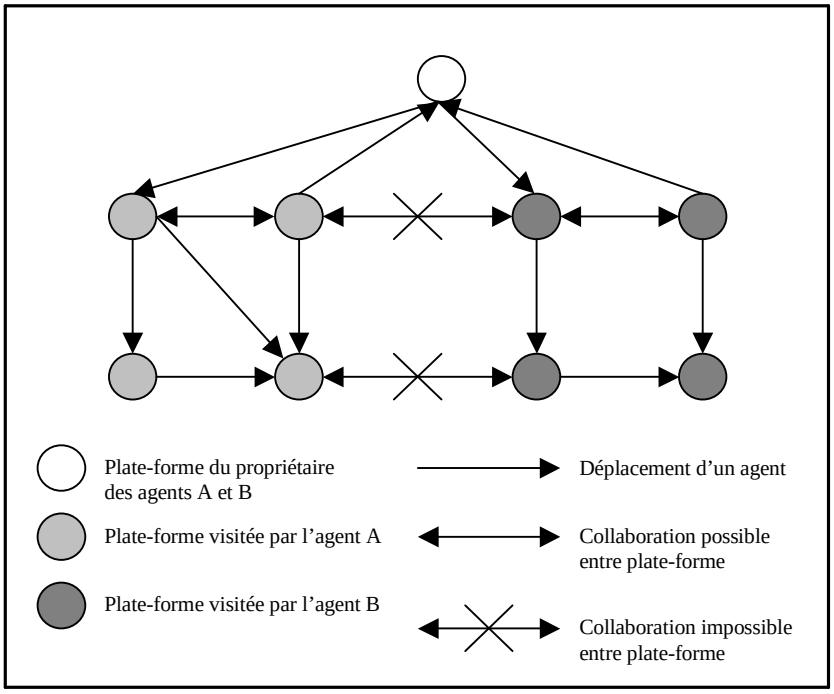

Fig.2.11. Hypothèses de Roth sur les collaborations entre plates-formes

Dans son protocole d'enregistrement d'itinéraire, Roth n'a pas précisé comment l'itinéraire de l'agent coopérant était défini. De plus, il a évoqué le problème d'interversion possible de deux agents coopérants qui effectueraient leurs vérifications sur le même hôte. Allée [Allée 2001] a proposé une amélioration de ces deux points en déplaçant l'agent coopérant B à chaque déplacement de l'agent mobile A. En fait, avant de se déplacer, l'agent A envoie les informations concernant l'identité de la plate-forme courante et suivante. Quand l'agent B les reçoit, il vérifie la validité de l'itinéraire, l'enregistre et se déplace à son tour, et attend que l'agent A communique avec lui à nouveau. Pour ce faire, il suppose que l'agent mobile A possède une liste ordonnée des plates-formes que son agent coopérant B doit visiter.

Dans la même lignée d'idées, Hohl [Hohl 1999] suggère l'idée d'employer des états de référence pour détecter des attaques d'interférence. Cette approche est connue sous le nom d'*États de référence*. L'agent mobile est exécuté sur un hôte référencé ainsi qu'un hôte potentiellement malicieux. La différence des parties variables de l'agent est mesurée par la suite. De cette manière, les attaques de modification de l'état peuvent être détectées. Ce mécanisme détecte les attaques d'intégrité mais pas ceux liés à la confidentialité.

La technique de la coopération d'agents [Jansen 2000a, Roth 1998, Roth 1999] peut être aussi vue comme une technique distribuant les tâches critiques d'un seul agent mobile entre des agents coopératifs. En employant les techniques de tolérance aux fautes, le comportement malveillant de quelques hôtes malicieux peut être contré. Malgré que la plate-forme malicieuse puisse engendrer un mauvais comportement de l'agent, l'existence de répliques garantit un résultat correct. Cette technique réduit la possibilité de vol des données partagées par un seul hôte.

2.5.7. Évaluation d'état

Cette approche, définie par Farmer et al. [Farmer 1996 a], identifie la notion de l'évaluation d'état, qui est un mécanisme permettant à un agent mobile de décider de quels privilèges il aura besoin au niveau d'une plate-forme d'agent particulière. Ceci permet au propriétaire de l'agent de limiter l'autorité de ses agents d'une façon flexible. L'approche repose sur le fait que la fonction d'évaluation d'état appartient à la partie immuable de l'agent mobile, dont l'intégrité peut être protégée par le propriétaire et/ou l'utilisateur de l'agent.

Cette fonction d'évaluation d'état est estimée en utilisant l'état de l'agent comme paramètre lorsque l'agent mobile arrive au niveau d'une nouvelle plate-forme d'agent et vérifie si l'état de l'agent rencontre les invariants importants. Ainsi, elle peut protéger le propriétaire d'agent contre l'abus par l'intermédiaire de modifications d'état dangereuses mais détectables qui peuvent générer un agent nocif bien que le code n'ait pas été altéré et qu'il est présumé être bénin. De tels états nocifs peuvent être définis par l'intermédiaire des relations arbitrairement complexes entre les variables d'état dynamiques de l'agent mobile.

Dans cette approche Farmer et al. [Farmer 1996 b] définissent un mécanisme qui permet à un agent d'évaluer les privilèges dont il dispose sur un site particulier. Ce qui permet au propriétaire de l'agent de limiter les actions que l'agent peut effectuer. L'approche repose sur une fonction protégée qui permet l'évaluation de l'état de l'agent sur chaque site visité. La fonction sera exécutée une fois l'agent arrivé sur un site donné et permettra de vérifier la stabilité de son état. L'existence d'une telle fonction protège l'agent en détectant les manipulations de son état. La fonction repose sur un calcul complexe à partir d'un ensemble de variables d'état. Les auteurs distinguent le privilège spécial "exécuter" qui définit si une plate-forme d'agent exécutera réellement un agent mobile. Ainsi, si la fonction d'évaluation d'état ne demande pas le privilège "exécuter" (puisque l'agent mobile est dans un état peu sûr), la plate-forme d'agent n'exécutera pas l'agent, et limite ainsi les dommages qui peuvent être causés par un agent altéré.

Une amélioration a été proposée par Jansen [Jansen 2001a][Jansen 2001b] [Jansen 2001c] et qui consiste à 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 X509 [ISO9594-8]. 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 n'offre aucune protection des données que l'agent transporte. Une autre approche proposée par Sayeb [Sayeb 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.

2.5.8. Fonction de validation

Les approches à base de fonctions de validation ont été proposées par Blum [Blum 1988] afin de vérifier la fiabilité d'un code. Ces dernières peuvent être appliquées dans le but de vérifier l'intégrité d'exécution d'un agent mobile sur un site distant. L'approche se base sur la propriété de réductibilité de la fonction qu'implémente l'agent. Une approche plus récente proposée par Laurero [Laureiro 2001] définit une fonction de vérification V comme suit : Soient un programme P qui implémente une fonction f, Y l'ensemble des résultats possibles de f(xi), avec xi appartient à $\{0,1\}^n$ et D(y') le résultat reçu après exécution à distance. La fonction V satisfait la condition suivante si (y= D(y')) n'appartient pas à Y alors P(V(y')= accept) < δ, où P représente la probabilité et δ la probabilité d'erreur.

2.5.9. Synthèse

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. Il n'existe pas, en fait, de 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.

Une comparaison entre les approches précédemment présentées (*cf.* Table 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. Comme critères relatifs à la sécurité, nous examinons les apports de chaque approche en terme de confidentialité d'exécution du code mobile sur chaque site visité ainsi que sa capacité à assurer l'intégrité de l'exécution. Cette intégrité est-elle garantie tout le long de l'itinéraire de l'agent mobile ou est-elle vérifiée seulement après son retour ? Les coûts de la solution de sécurité sont quant à eux examinés selon la taille du code engendrée, le temps d'exécution généré ainsi que les communications additionnelles à travers le réseau en raison des vulnérabilités que ces dernières peuvent impliquer.

Les approches présentées ont été implicitement subdivisées en deux groupes selon qu'elles soient basées sur le matériel ou sur le 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 refus de disponibilité, de confidentialité et les risques d'authentification. Selon le classement des menaces donné par Bierman [Bierman 2002], nous avons élaboré un tableau comportant les différentes contre-mesures pouvant pallier à ces menaces.

Critères / Approches	Catégorie	Confidentialité d'exécution	Intégrité d'exécution	Taille du code	Temps d'exécution	Communication additionnelle
Trace d'exécution	détection	Non assurée	Vérifiable	Augmentée de la taille de la trace associée	Acceptable	Aucune
Encapsulation des résultats partiels	Détection	Forte	Assurée	Augmentée des résultats partiels	Relativement acceptable	Aucune
Fonction cryptographique	Prévention	Forte	Forte	Relativement acceptable	Relativement acceptable	L'agent doit retourner à son site après chaque visite
Clé environnementale	Prévention	Forte	Non assurée	Relativement acceptable	Relativement acceptable	Aucune
Matériel de confiance - Cartes à puces	Prévention	Forte	Forte	Ajout d'instructions d'appel de la carte	Faible ralentissement à cause de l'accès à la carte	Aucune
Obscurcissement	Prévention	Forte pendant une courte durée de vie de l'agent	Forte pendant une courte durée de vie de l'agent	Relativement acceptable en cas d'insertion de code mort	Non modifié	Aucune
Nœuds de confiance	Détection	Forte	Non assurée	Non modifié	Non modifié	Aucune car les nœuds de confiance font partie de l'itinéraire établi
Copie d'agent	Détection	Forte	Non assurée	Non modifié	Non modifié	Aucune
Code à la demande	Prévention	Améliorée	Forte	Non modifié	Non modifié	Nombre de communications additionnelles restreint
Agents coopératifs	Détection	Non assurée	Forte	Relativement acceptable	Fortement ralenti	Trop de communications additionnelles
Evaluation d'état	Détection	Non assurée	Partiellement assurée	Relativement acceptable	Relativement acceptable	Aucune
Fonctions de validation	Détection	Non assurée	Vérifiable	Relativement acceptable	Relativement acceptable	Aucune

Table 2.1. Comparaison des approches de protection des agents mobiles

D'après le résultat obtenu (*cf.* Table 2.2. (a), (b), (c), (d)), il semble qu'un environnement de confiance d'exécution est la seule mesure susceptible de couvrir toutes les menaces. En effet, 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 discutées. C'est la raison pour laquelle nous nous penchons sur le problème d'estimation de la confiance de l'environnement visité. Par ailleurs, la technique de la clé environnementale nous a parru très intéressante (*cf.* Section 2.5.2). Cette technique peut, en effet, être utilisée lorsque l'hôte d'accueil ne connaît pas les conditions exigées pour l'exécution du service requis. Ce cas correspond à notre problème d'analyse du comportement de l'agent mobile. La

clé environnementale peut donc être employée pour non seulement protéger le service transporté par l'agent mobile mais aussi pour estimer la crédibilité de l'hôte visité puisqu'elle est construite sur la base d'informations environnementales. En outre, il n'est pas possible de décider à priori, au cours du cheminement de l'agent mobile, si un hôte est digne de confiance ou pas. Par conséquent, la sécurité de l'agent mobile exige l'évaluation dynamique de la crédibilité des hôtes visités. L'idée est donc d'utiliser le concept de l'adaptabilité dynamique afin de déterminer le comportement adéquat de l'agent mobile selon le degré de confiance détecté. Ce concept offre à l'agent mobile la capacité de modifier son comportement ; ce qui le rend imprévisible et donc le protège contre les menaces d'analyse.

Classe de menaces	Sous-classe de menaces	Contre-mesures souhaitables
Attaque d'intégrité	Interférence d'intégrité	- Environnement d'exécution de confiance - Fonctions encryptées - Sliding encryption - Etats de référence - Evaluation d'état
	Modification des informations (code ou données)	- Environnement d'exécution de confiance - Matériel de confiance - Enregistrement d'itinéraire - Etats de référence - Fonction cryptographique - Sliding encryption - Clé environnementale - Encapsulation de résultats partiels - Trace d'exécution - Fonction de validation - Nœuds de confiance - Code à la demande - Enregistrement d'itinéraire

Table 2.2. (a). Attaques d'intégrité avec les contre-mesures adéquates

Classe de menaces	Sous-classe de menaces	Contre-mesures souhaitables
Refus de disponibilité	Déni de service	- Environnement d'exécution de confiance - Enregistrement de l'itinéraire - Etats de référence - Trace d'exécution
	Retarder le service	- Environnement d'exécution de confiance - Enregistrement de l'itinéraire - Etats de référence
	Refus de transmission	- Environnement d'exécution de confiance - Enregistrement de l'itinéraire - Etats de référence

Table 2.2. (b). Attaques de disponibilité avec les contre-mesures adéquates

Classe de menaces	Sous-classe de menaces	Contre-mesures souhaitables
Attaques de la confidentialité	Espionnage	- Environnement d'exécution de confiance - Distribution du secret - Sliding encryption - Fonction cryptographique - Clé environnementale
	Vol	- Environnement d'exécution de confiance - Matériel de confiance - Enregistrement d'itinéraire - Sliding encryption - Distribution du secret
	Décompilation	- Environnement d'exécution de confiance - Sliding encryption - Fonction encryptée - Distribution du secret - Obscurcissement pendant un temps limité

Table 2.2. (c). Attaques de confidentialité avec les contre-mesures adéquates

Classe de menaces	Sous-classe de menaces	Contre-mesures souhaitables
Risque d'authentification	Mascarade	- Environnement d'exécution de confiance - Signature digitale - Trace d'exécution - Enregistrement de l'itinéraire - Etats de référence
	Clônage	- Environnement d'exécution de confiance - Sliding encryption - Fonction encryptée - Agent sensible au temps

Table 2.2. (d). Attaques d'authentification avec les contre-mesures adéquates

Les différents efforts menés par la communauté de la sécurité, pour permettre d'obtenir un niveau de sécurité satisfaisant, nous poussent à croire qu'il est possible d'obtenir le niveau de sécurité requis [Gray 1998, Tanenbaum 2004]. De toute façon, plus de recherches sont nécessaires afin d'apporter la confiance suffisante à la technologie basée sur le concept d'agent mobile et justifier son emploi par un large éventail d'utilisateurs.

2.6. Conclusion

La technologie agent mobile est en plein essor bien que la notion d'agent soit relativement ancienne. La maîtrise technologique des agents est un enjeu économique et stratégique très important, ce qui explique la difficulté de trouver une normalisation acceptable. Dans ce chapitre, nous nous sommes intéressés à décrire le contexte général où se situe notre étude sur les agents mobiles.

À travers les différentes caractéristiques des agents mobiles que nous venons d'énumérer, nous pouvons dire qu'ils représentent une possibilité nouvelle pour la construction des applications distribuées. Selon les besoins de performance, de conception, de développement et de sécurité, les agents mobiles pourront apporter une alternative intéressante au classique client/serveur dans certains types de configuration. Leurs avantages les rendent très attrayants particulièrement dans les systèmes où la largeur de bande de réseau est basse et/ou le coût de connexion au réseau est élevé ; Ils sont utiles aussi pour des applications où la quantité des données distantes à traiter est grande ou bien la tâche à exécuter consomme beaucoup de temps.

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 sites 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. Elles incluent, à titre d'exemple, la génération de différents types d'audit pouvant être employés en cas de conflits ou encore les mécanismes et protocoles reliant les actions d'un agent aux hôtes visités. Cependant, une plus large adoption du concept d'agent mobile exige davantage de recherche.

Ce Chapitre a permis de définir la migration des agents mobiles comme proactive afin de préserver leur caractère autonome et les protéger. Il les a aussi dotés d'un caractère adaptatif leur permettant de percevoir leur environnement et de parer aux changements des conditions de sécurité. Il a établi une vue d'ensemble sur les recherches déjà effectuées pour résoudre le problème de sécurité lié aux agents mobiles. 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. Cette dernière sera détaillée au niveau des chapitres suivants.

Chapitre3 – TAMAP : une nouvelle approche de protection des agents mobiles

« Se protéger et faire attention maintient une vie, savoir courir la rallonge! »

[Proverbe africain]

3.1. Introduction

L'étude effectuée au niveau du chapitre précédent 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 l'hôte malicieux. Elle a aussi mis en évidence les avantages des approches de protection logicielles relativement à celles basées sur le matériel de confiance: leur robustesse, leur maintenance facile et leur coût réduit.

L'analyse réalisée à l'issue de cette étude (*cf.* Section 2.4.9) a révélé que la seule protection susceptible de couvrir toutes les menaces est manifestement celle qui consiste à exécuter l'agent mobile dans un environnement de confiance. En effet, l'exécution d'un agent mobile sur un hôte sûr permet de se débarrasser de la plupart des classes de menaces discutées. C'est la raison pour laquelle nous nous penchons sur le problème d'estimation de la confiance de l'environnement visité. En outre, il n'est pas possible de décider à priori, au cours du cheminement de l'agent mobile, si un hôte est digne de confiance ou pas. Par conséquent, la sécurité de l'agent exige l'évaluation dynamique de la crédibilité des hôtes visités pour déterminer le comportement adéquat de l'agent mobile suivant le degré de confiance détecté. Pour cela, l'idée de l'utilisation de la notion de qualité de service (Quality of Service – QoS) qui permet de dégrader le service selon le degré de confiance décelé s'est imposée d'elle-même. De plus, afin de pouvoir estimer la crédibilité de l'hôte visité et décider du comportement de l'agent mobile sans léser[8] ni l'agent ni l'hôte visité par une réaction inadéquate, il est important de s'appuyer sur des informations réelles collectées auprès de l'hôte. En outre, dans le souci de compliquer l'analyse du comportement de l'agent mobile, le concept de l'adaptabilité a été utilisé. Il offre à l'agent mobile la possibilité d'avoir un comportement impévisible. Par ailleurs, les informations transmises (code et données) entre l'agent mobile et son hôte d'origine doivent aussi être protégées. L'utilisation de la notion de clé environnementale a rendu possible cette protection.

Ce chapitre explique les fondements de l'approche de protection proposée. Il introduit un exemple qui sera utilisé pour illustrer l'approche préconisée. Il aborde le problème des attaques d'analyse statique et dynamique en examinant les contre-mesures que notre approche offre pour les contrecarrer. Il décrit le protocole de protection et présente le modèle de comportement de l'agent mobile en spécifiant les propriétés du service fourni.

[8] Une réaction inadéquate vis-à-vis de l'agent pourrait soit lui permettre de s'exécuter malgré que l'hôte soit malicieux ou ne pas s'exécuter alors que l'hôte est fiable. Une réaction inadaptée vis-à-vis de l'hôte serait de ne pas permettre l'exécution de l'agent bien que l'hôte soit de confiance.

3.2. Scénario d'exécution d'un service sécurisé à base d'agent mobile

Le contexte de ce travail est apporté par le scénario dans lequel un pourvoyeur de service utilise un agent mobile pour assurer un service (*cf.* Figure 3.1). Arrivé au niveau de l'hôte client, l'agent mobile essaye de détecter les diverses conditions qui doivent être prises en considération pour l'exécution du service fourni et qui lui permettent de juger du degré de confiance qu'il peut placer en lui. L'agent mobile génère, à partir des données collectées, une clé, dite clé intermédiaire, qu'il utilisera pour l'évaluation de la confiance et donc pour la sélection du comportement le plus approprié. Cette clé sert à la construction de la clé environnementale qui est, à son tour, employée pour encrypter/décrypter la QoS sélectionnée.

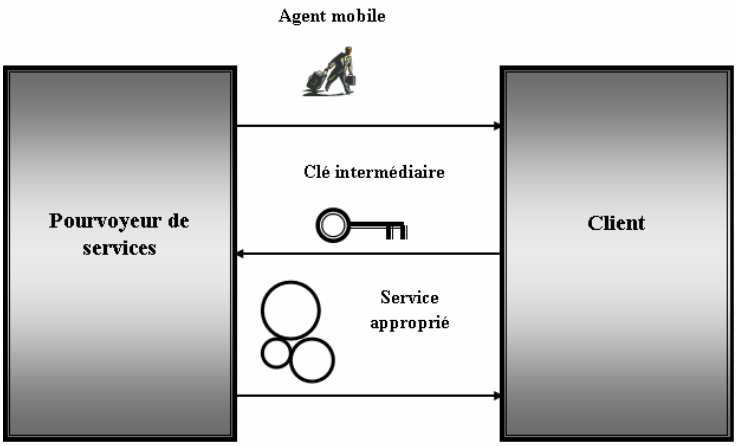

Fig.3.1. Scénario d'exécution d'un service par un agent mobile

3.2.1. Structure du service fourni

Dans notre étude, l'agent mobile est utilisé pour la distribution d'un service donné. Sa tâche générale est décrite comme un ensemble de modules M_i associés à ses différents comportements C_j (*cf.* Figure 3.2).

Le service est constitué de deux composants:

1. Le code (le service)
2. Les données applicables au code.

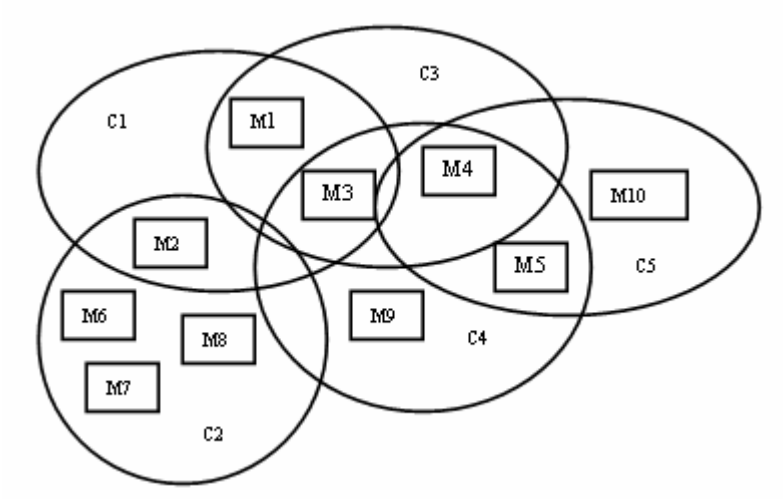

Fig.3.2. Modules participant dans les différents comportements.

Le client dispose des données mais pas du code qui va être fourni par l'agent mobile et exécuté à son niveau. Ainsi, un client adresse une requête au pourvoyeur de service uniquement pour récupérer un service précis afin de l'exécuter localement avec ses propres données. Les deux éléments doivent être réunis sur le site client comme le montre la Figure 3.3. Les résultats obtenus après exécution du service demeurent au niveau de l'hôte client. Ce qui offre l'avantage de les protéger. Le service est transporté par un agent mobile qui l'achemine vers plusieurs clients. Ce schéma représente le cas de figure envisagé dans cette thèse.

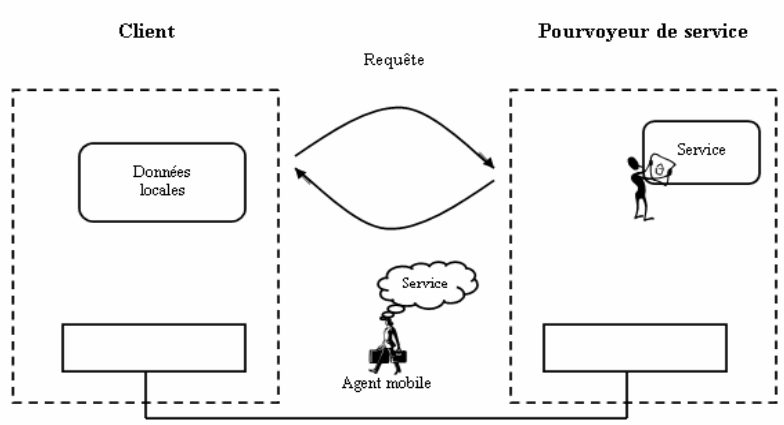

Fig.3.3. Exécution du code (service) transporté par l'agent mobile en utilisant les données fournies par l'hôte client.

3.2.2. Entités sollicitées

Notre approche fait intervenir les entités suivantes:

- Le pourvoyeur de service
- L'agent mobile
- Les clients.

Le pourvoyeur de service est l'entité responsable de la création et de l'émission initiale des agents mobiles vers les hôtes clients. C'est au pourvoyeur de service que revient la tâche de sélectionner la qualité du service à fournir après l'identification du client et l'estimation de

son degré de confiance. Les informations concernant les clients du pourvoyeur de service sont stockées dans une base de données appelée base des évidences. La Table 3.1 montre un exemple d'une instance d'un client stocké dans cette base de données. La liste des services accessibles pour chaque client est rangée dans une autre table de la base de données.

Le client est abonné à un service fourni par le pourvoyeur de service et que l'agent mobile devra exécuter à son niveau. Il dispose de quelques informations confidentielles servant à son authentification[9] et à son identification[10] par le pourvoyeur du service telles que son mot de passe ou pseudonyme. Il possède aussi une portion du code encrypté du service dont il a le droit de bénéficier.

L'agent mobile, quant à lui, est considéré comme le support véhiculant le service. Il peut transporter avec lui éventuellement son itinéraire (la liste des hôtes à visiter) et une portion du code de chaque service susceptible d'être fourni durant son acheminement. Il est capable de générer une clé environnementale en utilisant la clé intermédiaire calculée sur la base des données collectées sur l'hôte et utilisée pour l'estimation de sa crédibilité.

Paramètre	Valeur du paramètre	empreinte (SHA-256)
ID Client	788-19845-GH154	c65ee27c003374618691697b27fa2d8a5ee a2764535aa7f538fbec07eae48c9c
Acronyme du service	Esprit 7	09466cd69275da7a4de53216c1e5adcbe7b 15748c5f96fb5aa73e1eef0f96a8a
Le mot de passe	topsecret	53336a676c64c1396553b2b7c92f3812676 8827c93b64d9142069c10eda7a721
Certificat	< fichier du certificat >	b1b634fd0a09cf8002c52282a3c0f9d5f0cc d8c4566d3e68cfb8e5e46cb4ff0a
La date de validité du certificat	12Décembre2007	72348b5b979a182193b5c44f323abda0928 4ef309329eeec089df9946aae9e47
Identificateur du service	Mobi-Scan	a8186c91696b51664aad9325f74f6dd4a35 d1bae239a4488354f49ab94f95d1d
Répertoire temporaire du client	C:\DOCUME~1\kamel\LOCALS~1\Temp\	3382ec2042a43c44d1dcd3da118348dab72 ae7de2d0f41fdc9c0b54ab83b140c
Répertoire local du client.	C:\Documents and settings\kamel	ddc84da92b129cdec66c8812cb4e32d65bc c6af9400680ab935b8b2c4af5357d
Chemin d'exécution de l'agent mobile	C:\Documents and Settings\ kamel \Bureau	472bedfb45de9683dd5b00daeb4e7973003 b943904969d0473413d71d902e21c
Nom du système d'exploitation	Windows XP	9878693bc56de4c2d77f849aed18c3eedf36 c115447d50516cb08a4ada7f9f91
Version du système d'exploitation	5.1	77065cda8fe1551871ba934cfef4df0f9e8a5 f7fb63909b25bf0ccebbe807e18
Type d'architecture du processeur de l'hôte.	X86	ee914f0378c4a89fd5c2e8e715133ff6d251 2725f76ff05fbd35fd6b470f2753

Table 3.1. Exemple d'une instance client

[9] Processus consistant à vérifier qu'une personne ou un objet correspond bien à son identité déclarée. Dans les réseaux informatiques privés ou publics (dont Internet), l'authentification repose généralement sur un nom de connexion et un mot de passe, ou une signature électronique.

[10] Procédure dont le but est de reconnaître un utilisateur afin de lui accorder les droits correspondant à son profil

3.2.3. Hypothèses formulées

Nous avons établi un certain nombre d'hypothèses qui servent de base à l'approche de protection proposée. Elles concernent la gestion des relations entre les trois entités: pourvoyeur de service, agent mobile et client:

- Un contrat est préalablement établi entre le client et le pourvoyeur de service. Ce qui permet au pourvoyeur de service de créer une instance dans sa base de données pour ce client. Ces informations sont utilisées pour l'authentification du client.
- L'hôte client dispose d'informations sur lui-même ou sur son environnement que l'agent ne connaît pas.
- L'agent mobile doit générer la clé environnementale sans se soucier de la validité des informations collectées au niveau de l'hôte d'accueil.
- Le pourvoyeur de service dispose d'informations concernant la configuration de la machine du client. Toute mise à jour est signalée.
- L'agent mobile ne fait confiance à aucun hôte visité.
- L'hôte du pourvoyeur de service est un environnement considéré comme sûr.
- Le pourvoyeur de service peut avoir une idée préalable sur la crédibilité de l'hôte d'accueil. Elle peut être basée sur sa réputation.

3.3. Fondements de TAMAP

TAMAP (Trust-based Approach for Mobile Agent Protection) est une nouvelle approche de protection des agents mobiles basée sur la confiance. Elle est fondée sur deux concepts : la confiance et l'adaptation dynamique.

L'adaptabilité repose sur une estimation de confiance effectuée par le pourvoyeur de service à partir d'une clé intermédiaire construite sur la base d'informations obtenues suite à une interaction, une inspection et une observation effectuées par l'agent mobile durant son exécution préliminaire au niveau de l'hôte visité. Cette clé permet au pourvoyeur de service d'évaluer le degré de confiance de l'hôte visité et de sélectionner la QoS appropriée. Une clé environnementale est construite à partir de cette clé. Elle est utilisée pour la protection de la QoS émise. Cette dernière sert de support à l'exécution de l'agent mobile qui l'utilise pour déterminer la règle d'adaptation adéquate qui se traduira par un comportement s'accordant avec les conditions de sécurité perçues. La mise en œuvre de TAMAP repose, par voie de conséquence, sur un protocole de protection, un mécanisme d'estimation de la confiance et un processus d'adaptation dynamique:

1 Le protocole de protection a pour objectif de garantir la sécurité de l'agent mobile ainsi que celle de la communication entre l'agent mobile et le pourvoyeur de service. Il utilise, pour cela, des mécanismes fiables tels que l'encryptage, le hachage et la signature digitale.

2 Le mécanisme d'estimation de la confiance permet d'établir la crédibilité de l'hôte et donc la sélection de la QoS la plus appropriée (*cf.* Chapitre 4). Ce mécanisme repose sur des données propres au client et à leur comparaison avec celles détenues par le pourvoyeur de service. Par conséquent, le mécanisme d'estimation de la confiance est basé sur la clé intermédiaire générée, et suit les étapes suivantes :

   – L'identification des paramètres définissant la confiance, et dont les valeurs doivent être vérifiées.

- L’évaluation des paramètres de la confiance. Les valeurs sont collectées par l’agent mobile sur l’hôte client.
- La quantification de la confiance.
- La sélection de la QoS appropriée.

3 Enfin, le processus d’adaptation dynamique utilise la QoS fournie et l’historique de l’exécution de l’agent mobile pour choisir l’exécution la plus adéquate (*cf.* chapitre 5). Cette sélection est basée sur l’aptitude de l’agent mobile à varier son comportement. L’adaptation, dans TAMAP, est conçue de manière statique et exécutée de façon dynamique.

3.4. Exemple

Pour illustrer notre approche, nous considérons l’exemple dans lequel un pourvoyeur de service en ligne offre plusieurs prestations en proposant divers types d’antivirus heuristiques (*cf.* Figure 3.4). L’agent mobile essaye de protéger son comportement pour empêcher son analyse par l’hôte d’accueil. Sa protection est principalement basée sur sa capacité d’estimation de la confiance et son aptitude à s’adapter qui lui permet de générer, pour les mêmes conditions, des comportements variés.

Le service en question fournit différentes QoS. Cette diversité est essentielle à la réaction de l’agent mobile après l’étape de l’estimation de la confiance. Elle permet, en effet, de faire correspondre une QoS à chaque degré de confiance estimé. Par conséquent, le comportement de l’agent mobile est lié à la valeur de la confiance calculée. Des exemples de QoS sont :

QoS0: Ne pas assurer le service.

QoS1: Ne pas assurer le service et en informer l’hôte.

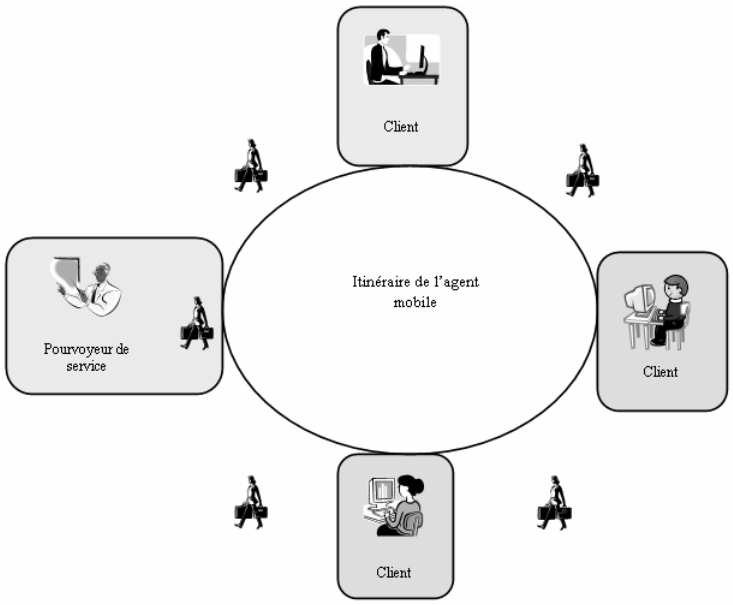

Fig.3.4. Itinéraire de l’agent mobile offrant ses prestations

QoS2: scanner l’ordinateur de l’hôte et afficher le résultat du balayage.

QoS3: scanner l’ordinateur de l’hôte, afficher le résultat du balayage et installer un antivirus de basse performance.

QoS4: scanner l'ordinateur de l'hôte, afficher le résultat du balayage et installer un antivirus de performance moyenne.

QoS5: scanner l'ordinateur de l'hôte, afficher le résultat du balayage et installer un antivirus de haute performance.

QoS6: scanner l'ordinateur de l'hôte, afficher le résultat du balayage, installer un antivirus de basse performance et permettre sa mise à jour.

QoS7: scanner l'ordinateur de l'hôte, afficher le résultat du balayage, installer un antivirus de performance moyenne et permettre sa mise à jour.

QoS8: scanner l'ordinateur de l'hôte, afficher le résultat du balayage, installer un antivirus de haute performance et permettre sa mise à jour.

3.5. Mesures de protection adoptées

Le but primordial de cette approche est de protéger l'agent mobile contre l'analyse dynamique de son code. Cependant, la protection contre l'analyse statique doit être aussi garantie, car les deux catégories de protection sont complémentaires. Par conséquent, il est nécessaire d'introduire des mécanismes de protection classiques tels que l'encryptage, la signature numérique ou le hachage.

3.5.1. Protection vs analyse

Les services informatiques sur le web souffrent de nombreuses attaques (*cf.* Section 2.4.9). Etant donné les dommages importants qu'elles peuvent engendrer [Bierman 2002], ceci porte préjudice à l'image de l'entreprise. L'agent mobile peut subir deux types d'analyse:

(i) L'analyse statique qui concerne l'analyse du code de l'agent avant son exécution
(ii) L'analyse dynamique qui s'occupe de l'analyse du comportement de l'agent durant son exécution.

L'encryptage du code a rendu son analyse statique pratiquement impossible. En effet, face aux attaques par exemple des virus encryptés, l'analyse statique de leurs codes est devenue très difficile et pratiquement inutile dans la plupart des cas. Par contre, l'analyse dynamique de ce genre de virus se révèle intéressante. Si le virus se manifeste toujours de la même façon, il peut être aisément identifié à partir de son comportement. L'analyse dynamique est devenue un outil indispensable pour les anti-virus, anti-spywares et autres systèmes de protection. Elle est classée parmi les attaques les plus efficaces. Elle peut être active (contrôle d'exécution), comme elle peut être passive (apprentissage).

Dans ce qui suit nous allons présenter nos solutions de protection qui permettent de brouiller le comportement de l'agent, et de rendre son code transparent dans le cas aussi bien d'une analyse statique que d'une analyse dynamique.

3.5.1.1. Protection contre l'analyse statique

L'obscurcissement, l'encryptage ainsi que la dispersion du code ont été conjointement utilisés afin d'empêcher l'analyse statique du code de l'agent mobile. En se basant sur des scénarii d'attaques, nous exposons les contre-mesures procurées par notre approche de protection.

Obscurcissement

Le premier scénario d'attaque correspond au cas où un hôte malveillant réussit d'accéder au code de l'agent. Cet hôte peut alors aisément démasquer son comportement. Les algorithmes d'obscurcissement appliqués au code de l'agent mobile, dans TAMAP, auront pour avantage de compliquer son analyse statique durant une période déterminée.

Encryptage

Le second scénario d'attaque concerne la tentative d'analyse du code et des données transportés par un agent mobile durant son déplacement par des hôtes malveillants. Ces attaquants vont pouvoir identifier l'agent, le capturer, le détruire, violer ses données ou y insérer un code malveillant (tel qu'un spyware) et les conséquences peuvent être catastrophiques pour l'organisme qui fournit le service. L'encryptage de tous les modules constituant le code de l'agent est la meilleure protection. Cependant, les opérations d'encryptage/décryptage sont généralement coûteuses en temps d'exécution. Dans TAMAP, seuls les modules sensibles sont encryptés. Nous définissons deux niveaux conceptuels de sensibilité permettant de classer les modules de l'agent mobile selon leur importance (haute sensibilité et basse sensibilité).

L'encryptage des modules est effectué à l'aide d'un algorithme cryptographique symétrique. Notons que les algorithmes de cryptage symétrique sont plus rapides que les algorithmes d'encryptage asymétrique. Le processus d'encryptage des modules utilise l'algorithme d'encryptage AES[11] (Advanced Encryption Standard) ainsi que l'algorithme de hachage SHA (Secure Hash Algorithm) à sens unique[12] qui sert à la protection de la clé symétrique d'encryptage.

Le pourvoyeur de service sélectionne une clé K parmi les clés secrètes disponibles à son niveau, ensuite il encrypte tout module sensible M_i de l'agent avec cette clé:

1. Pour i allant de 1 à n faire $\mathcal{E}_k (M_i) = M_i'$ Finpour
2. Le pourvoyeur de service applique un algorithme de hachage à sens unique sur K tel que SHA-256 et obtient K′: $\mathcal{H}_{SHA256} (K) = K'$.
3. La clé hachée K′ est émise avec l'agent mobile vers l'hôte client.
4. Arrivé au niveau de l'hôte, l'agent mobile utilisera la clé K′ pour restituer la clé K.
5. L'agent mobile décrypte ses modules avec la clé K obtenue, puis il déclenche son exécution.

Chaque hôte client possède une liste des clés secrètes que le pourvoyeur utilise pour encrypter les modules de ses agents mobiles. Cette liste est une table à deux entrées stockée dans un emplacement sûr de l'hôte (*cf.* Table 3.2). Elle est protégée par un mot de passe P de taille 256 bits. Ce dernier est calculé par l'agent mobile au niveau de l'hôte client à partir de l'identificateur ID client, son pseudonyme ainsi que de son mot de passe:

$$\mathbf{P} = H_{SHA\text{-}256}(\textit{ID client} \quad \textit{Pseudo du Client} \quad \textit{Mot de passe Client})$$

[11] AES a été développé par Vincent Rijmen et Joan Daemen. C'est un standard cryptographique depuis 2000. Il a été sélectionné parmi une vingtaine d'algorithmes qui ont participés à un concours lancé par NIST. Il utilise des clés de tailles 128, 192 et 256 bits.

[12] Pour un entier x, il est simple de calculer H(x), mais étant donner H(x), il est pratiquement impossible de déterminer x. La fonction de hachage permet d'extraire une empreinte qui caractérise les données. Celle-ci a toujours une taille fixe indépendamment de la taille des données. Il est pratiquement impossible de trouver deux données ayant la même empreinte.

La clé hachée K′ (SHA-256)	La clé originale K
c83194cf3608b137b85a03b2bc963c73 bdc2a506883acdb348dd18f2775bc617	MC758-HO587469
9a01cd25a07d5c52bc4b43a8d37c7536 3db82d42a205dede68eb3769ec7a6e3c	MC122-HO757445
……	……

Table 3.2. Structure de la table des clés secrètes

L'agent mobile est seul à avoir le droit d'accéder à cette liste. Notons ici que P est utilisé aussi pour accéder au KeyStore[13] (*cf.* Section 3.4.3).

A la suite de l'opération d'encryptage, les modules seront assemblés dans un seul fichier. Pour augmenter la sécurité de l'agent mobile, nous imposons au pourvoyeur de service la contrainte suivante : deux agents instanciés successivement doivent être encryptés avec deux clés différentes. Cette solution améliore la transparence de l'agent dans le réseau et complique son analyse statique.

Dispersion du code

Le troisième scénario a lieu lorsque l'hôte malicieux réussit le décryptage du code de l'agent mobile en utilisant, à titre d'exemple, la méthode de force brute[14]. Il aura en sa possession le code du service transporté par l'agent. Dans cette alternative, l'hôte n'aura pas besoin de re-exécuter l'agent pour bénéficier du service. Il lui suffira de re-exécuter le service lui-même. Afin de pallier à ce problème, TAMAP s'inspire de l'approche basée sur le code à la demande (*cf.* Section 2.4.6) en optant pour la subdivision du code du service (secret) en trois parties; une portion est gardée par le pourvoyeur de service, l'autre transportée par l'agent mobile et la dernière est remise à l'hôte client lors de la signature du contrat (*cf.* Figure 3.5). La première portion du code - la partie cruciale du service - est en clair au niveau du serveur du pourvoyeur de service car celui-ci est considéré comme un environnement sûr. La seconde portion du code est stockée dans la bibliothèque de l'agent mobile, encryptée avec une clé secrète du pourvoyeur de service. Enfin, la troisième portion du code est placée dans le répertoire local du client. Elle lui est remise à la signature du contrat et fait partie des éléments à inspecter par l'agent mobile. Elle est encryptée avec le mot de passe P du KeyStore. Dès que l'agent mobile aura confirmé l'identité de l'hôte auprès du pourvoyeur de service, le code du service sera assemblé puis exécuté. Dans le cas où un hôte malveillant arrive à capturer l'agent, il n'aura en sa possession qu'une portion inutile du code du service.

La dispersion du code est une mesure de protection qui possède un autre avantage puisqu'elle permet d'alléger l'agent mobile. En effet, le service peut avoir un volume important, ce qui

[13] Un KeyStore [knud] est un fichier qui permet le stockage sécurisé de clés publiques, privées et certificats, L'accès au fichier est protégé par un mot de passe.

[14] Si f(x) = y et que y est connu et qu'il est possible de calculer f, on peut trouver x en essayant toutes les valeurs possibles. C'est la recherche par force brute.

l'alourdira, paralysera son déplacement et le rendra par voie de conséquence plus vulnérable aux attaques.

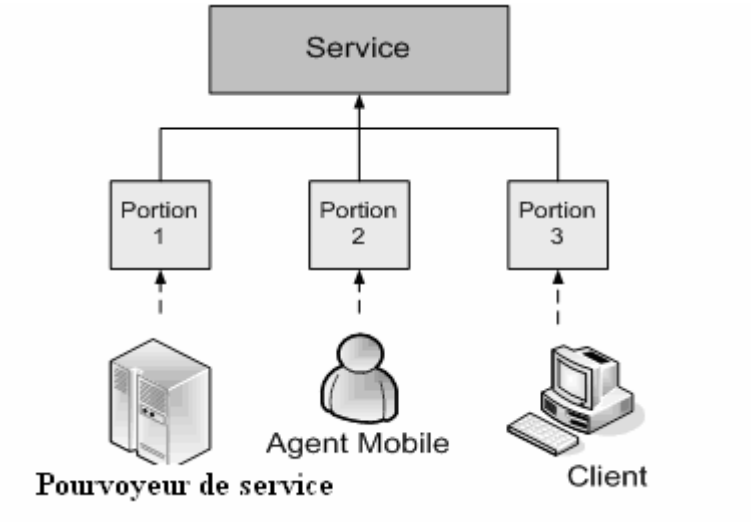

Fig.3.5. Structuration du service.

3.5.1.2. Protection contre l'analyse dynamique

L'analyse dynamique est une attaque qui est susceptible d'être très efficace pour l'identification du service et de son comportement. Ce type d'analyse ne recherche généralement pas à acquérir les données confidentielles du service mais à l'identifier et à contrôler son exécution.

Un hôte malveillant peut avoir réussi à capturer l'agent mobile. Cependant, il n'arrive ni à identifier le service ni à lire clairement le code. Il va tenter de le manipuler afin d'analyser son comportement. Si l'hôte malveillant réussit à déterminer le comportement de l'agent, il sera en mesure d'identifier le service transporté et par conséquent, de contrôler l'exécution de l'agent. Ce dernier doit donc posséder un comportement assez compliqué pour contrecarrer l'analyse de l'hôte malveillant. TAMAP utilise l'adaptabilité dynamique pour faire varier les comportements de l'agent et les rendre imprévisibles (*cf.* Figure 3.6). Pour atteindre cet objectif l'agent va générer, en plus du service lui-même, une séquence aléatoire d'exécution à partir d'un ensemble de sous-tâches fictives. Deux séquences générées successivement ne doivent pas avoir le même ordre ni les mêmes sous-tâches (à l'exception du service lui-même).

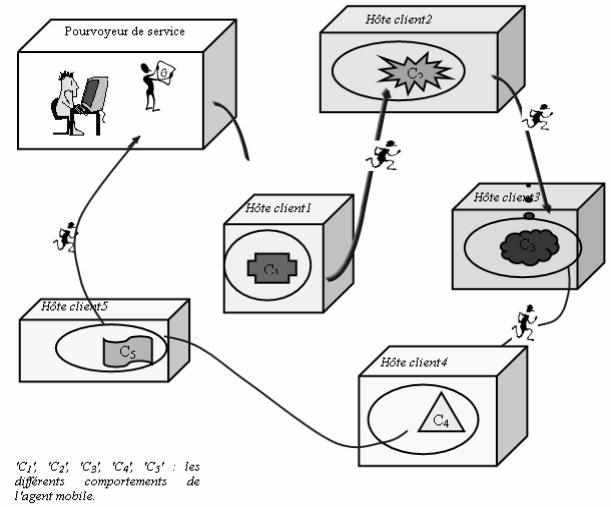

Fig.3.6. Différents comportements de l'agent mobile

3.5.2. Gestion des clés

Etant donné que TAMAP emploie conjointement la cryptographie symétrique et asymétrique, une gestion des clés entre le pourvoyeur de service et ses clients s'impose. Les clés symétriques utilisées pour encrypter/décrypter les parties du code de l'agent mobile sont partagées entre le client et le pourvoyeur de service. La liste de ces clés secrètes est protégée par un mot de passe (*cf.* Section 3.5.1.1). La clé environnementale est aussi utilisée en tant que clé symétrique permettant d'encrypter/décrypter la QoS sélectionnée. Elle est basée sur les informations collectées par l'agent mobile auprès du client visité (*cf.* Section 3.6). Cette clé est calculée aussi bien au niveau du pourvoyeur de service qu'au niveau du client (par l'agent mobile) afin de la protéger en évitant son transfert. L'agent mobile doit générer la clé environnementale sans se soucier de la validité des informations collectées au niveau de l'hôte d'accueil; ce qui permet de le protéger lui ainsi que les informations des clients. La signature digitale, quant à elle, impose l'utilisation de la cryptographie asymétrique. Le pourvoyeur de service dispose de toutes les paires de clés publiques/privées car c'est lui (ou une autorité de certification) qui les génère[15]. Chaque client possède une paire de clés publique/privée utilisées pour l'encryptage et la signature des informations transmises. La liste des clés privées/publiques est périodiquement modifiée. Elle est rangée dans un endroit sûr appelé KeyStore. Celui-ci comprend deux types d'entrées (*cf.* Figure 3.7):

[15] La génération des clés n'est pas le but notre étude.

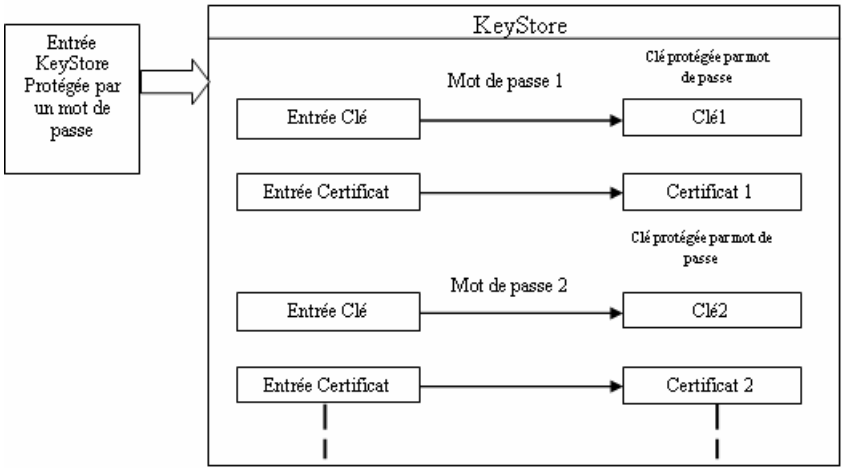

Fig.3.7. Structure du « KeyStore ».

- **Entrée de clés:** elle contient les clés sensibles. Celles-ci sont stockées dans un format protégé pour empêcher l'accès non autorisé.
- **Entrée de certificat:** elle contient un certificat appartenant à une entité.

3.6. Protocole de protection

Ce protocole vise à protéger aussi bien le code d'un agent mobile contre des hôtes malicieux que la communication entre l'agent mobile et le pourvoyeur de service. Il emploie la clé environnementale pour encrypter/décrypter la QoS consécutivement émise/reçue. Cette clé est calculée en utilisant les informations collectées sur l'hôte cible ainsi que l'identificateur de l'agent mobile qui est unique. Son calcul est fondé sur une fonction de hachage couplée à un encryptage à clé publique.

Lorsque le client a besoin d'un service, il émet une demande à tous les pourvoyeurs de service concernés. Ceux-ci lui transmettent leurs propositions. Ces dernières sont analysées par le client qui sélectionne la proposition la plus appropriée. Le client informe le pourvoyeur de service sélectionné qui génère un couple de clés publiques et privées, et fait correspondre une QoS spécifique à chaque comportement de l'agent mobile. Enfin, ce dernier migre vers l'hôte du client.

Nous utilisons en même temps la cryptographie symétrique et la cryptographie asymétrique et supposons l'existence des couples de clés publiques et privées suivantes:

- Les clés du pourvoyeur de service (hôte d'origine): (P_o, S_o)
- Les clés de l'hôte: (P_h, S_h)

Les étapes principales de ce protocole sont (*cf.* Figure 3.8) :

1. Le pourvoyeur de service émet un agent mobile dont le code est constitué de modules encryptés à l'aide d'une de ses clés secrètes (*cf.* Section 3.5) et éventuellement d'un itinéraire comportant la liste des clients à visiter.
2. L'agent mobile interagit avec l'hôte afin d'obtenir les informations nécessaires à la génération de la clé environnementale K_e (*cf.* Boîte 1).
3. L'agent mobile encrypte la clé intermédiaire M à l'aide de la clé publique P_o, la signe avec la clé privée de l'hôte S_h et l'envoie accompagnée de la signature au pourvoyeur de service (*cf.* Boîte 2). Ensuite, il calcule la clé environnementale K_e.

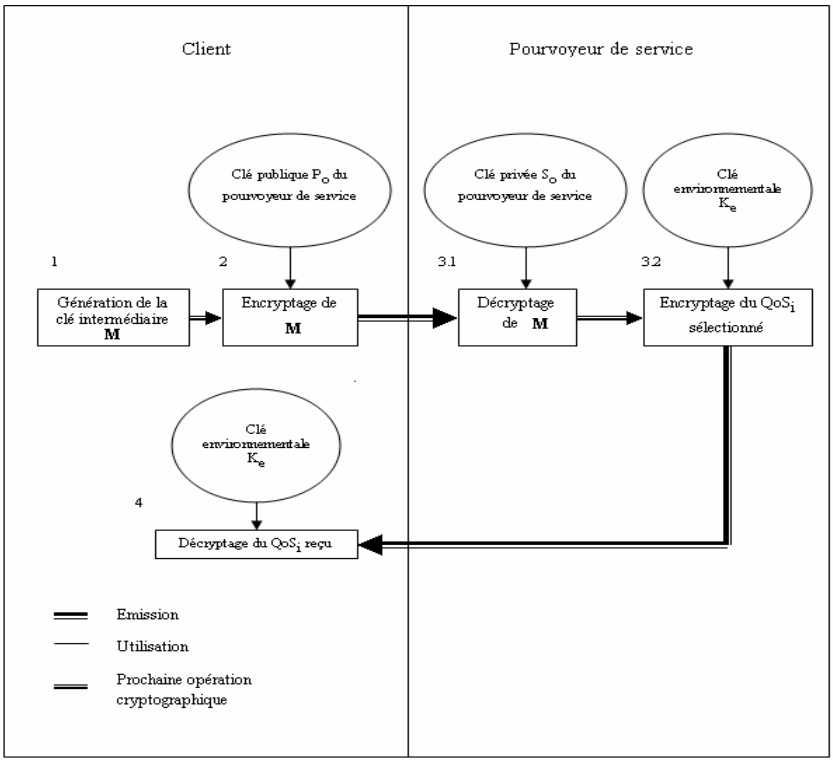

Fig. 3.8. Le protocole de protection de l'agent mobile

4. Le pourvoyeur de service vérifie la signature et décrypte la clé M reçue (*cf.* Boîte 3.1). Il analyse la clé, calcule la valeur de la confiance (*cf.* Section 4.2.6), sélectionne la QoS correspondante ainsi que la portion concernée du code du service, et encrypte l'ensemble à l'aide de la clé environnementale K_e (calculée par le pourvoyeur du service), le signe et l'émet avec sa signature à l'hôte du client (*cf.* Boîte 3.2).
5. Du côté client, l'agent mobile essaye de décrypter la QoS reçue à l'aide de la clé environnementale K_e calculée à son niveau (*cf.* Boîte 4). En cas de succès, il exécute le service demandé.

De plus amples détails sont fournis au niveau du Chapitre 4.

3.7. Modèle de comportement de l'agent mobile

Le comportement de l'agent mobile dépend du service fourni. Le service proposé par l'agent mobile doit pouvoir être déployé sur des environnements variés en prenant en considération leur niveau de sécurité. D'un autre côté, pour assurer un niveau de QoS donné, l'application doit pouvoir prendre en compte l'aspect variable de la sécurité. Par conséquent, l'objectif est de pouvoir adapter la QoS offerte par l'application suivant les variations des niveaux de sécurité de l'hôte. Ces variations dépendent des informations confidentielles détenues par l'hôte visité ainsi que de la disponibilité des ressources affectant directement la réalisation du service choisi.

La sélection de la QoS est basée sur une phase préliminaire représentée par l'estimation de la crédibilité de l'hôte visité. Ainsi, le comportement de l'agent mobile dépend de la catégorie de la QoS sélectionnée. L'agent mobile peut:

1 Assurer le service demandé puis quitter l'hôte (si sa crédibilité est haute).
2 Assurer un service dégradé puis quitter l'hôte (remplacer le service demandé par un autre de moindre utilité car la valeur de la confiance n'est pas significative).
3 Quitter l'hôte sans assurer le service (si la crédibilité de l'hôte est trop basse).

Notre objectif est de compliquer l'analyse dynamique du code de l'agent mobile. Afin de contrecarrer ce type d'analyse, le concepteur de l'agent mobile doit arranger le code de façon à ce qu'il prenne différents chemins d'exécution. En conséquence, déterminer quels sont les blocs de base qui ont été fréquemment invoqués, et dans quelles combinaisons, exigerait un grand effort de la part de l'attaquant.

En se basant sur la décomposition du service en plusieurs modules incluant des modules fictifs et l'emploi de techniques d'obscurcissement [Hacini 2005], [Hacini 2006a], chaque catégorie de QoS peut être exécutée de différentes manières. Cette propriété procure à l'agent mobile la capacité d'avoir des réactions différentes pour la même QoS. Par conséquent, cette aptitude complique son analyse puisque son comportement devient imprévisible.

Soit $E = \{E_1, E_2... E_n\}$ un ensemble de n expressions abstraites[16] utilisées pour la génération des différents comportements de l'agent mobile.

Soit $A = \{A_1, A_2... A_p\}$ l'ensemble des p tâches fonctionnelles (incluant des tâches fictives) participant aux différentes exécutions. Chaque Ej ($j \leq n$) est une séquence d'appels à un sous ensemble de A et peut être vue comme une séquence de p bits (chaque bit indiquant si une tâche spécifique participe ou non à l'exécution d'un comportement de l'agent mobile).

Si nous considérons p tâches, nous pouvons obtenir 2^p-2 combinaisons possibles ; nous ne considérons pas le cas où aucune tâche n'est exécutée. Une expression abstraite est associée à chaque combinaison et désigne un comportement spécifique. Cette propriété augmente le nombre des exécutions possibles de l'agent mobile et complique ainsi l'analyse de son comportement. Les exécutions de l'agent mobile varient d'un service à un autre et dépendent du degré de dégradation du service et du nombre d'alternatives fournies.

Les figures suivantes représentent l'aspect dynamique de chacune des entités du système (l'agent mobile, le client et le pourvoyeur de service). Les diagrammes d'activité de l'agent mobile ainsi que celui du pourvoyeur de service sont représentés séparément.

[16] Une expression abstraite est le chemin d'exécution des différentes tâches (réelles et fictives) participant à un comportement donné.

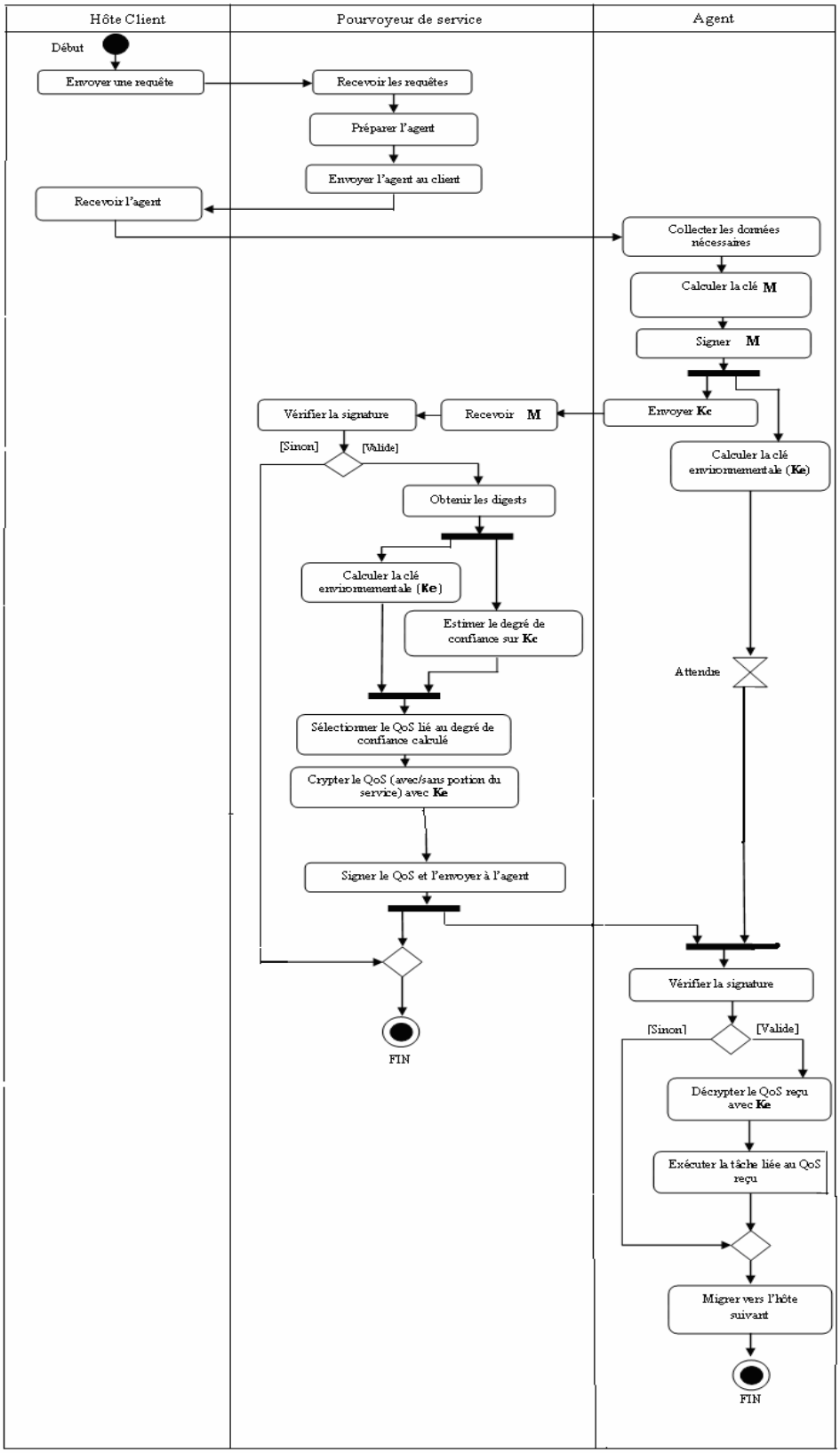

Fig.3.9. Diagramme d'activité UML représentant l'activité globale de chaque entité

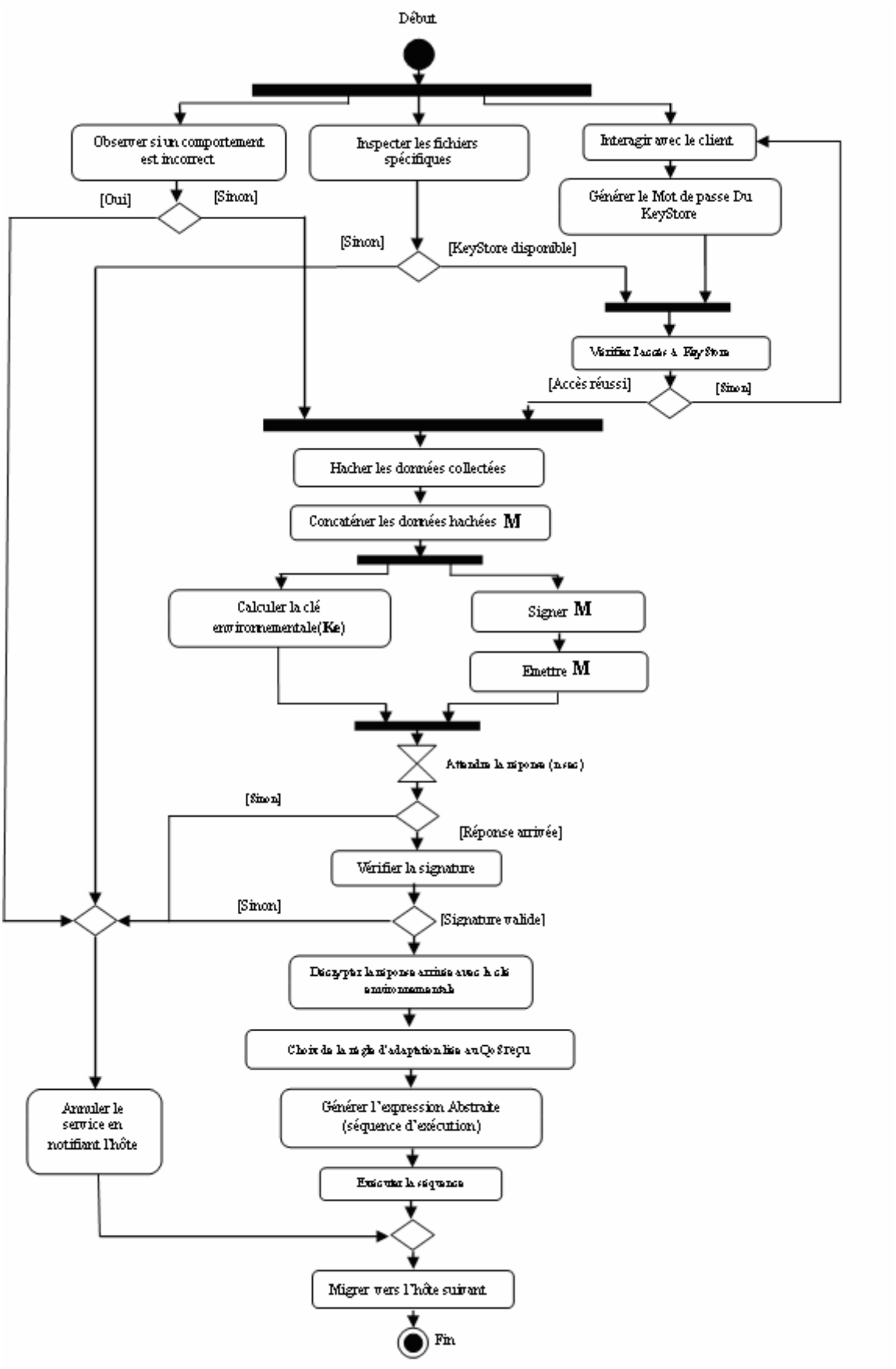

Fig.3.10. Diagramme d'activité UML représentant l'activité détaillée de l'agent mobile

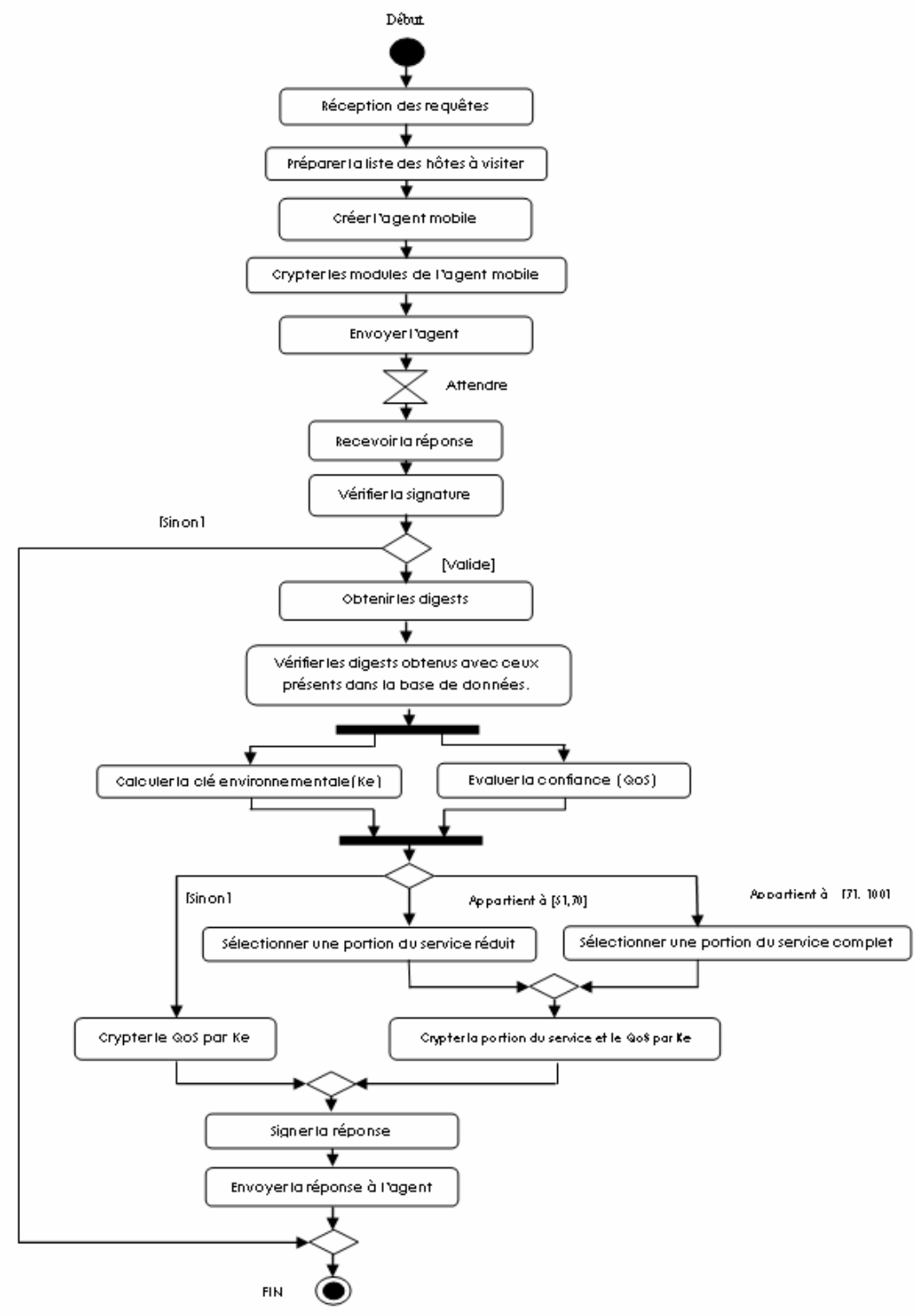

Fig.3.11. Diagramme d'activité UML représentant l'activité détaillée du pourvoyeur de service

3.8. Conclusion

Ce chapitre a permis de donner un aperçu sur les moyens mis en oeuvre pour prévenir l'analyse statique et dynamique du code de l'agent mobile. Les mesures utilisées sont subdivisées en principalement trois catégories :

1. L'ensemble des mesures qui préviennent les attaques contre la communication qui a lieu entre le pourvoyeur de service et l'agent mobile ainsi que celle qui existe entre l'hôte d'accueil et l'agent mobile.
2. l'ensemble des mesures qui permettent de décider de la réaction à adopter vis-à-vis d'un hôte potentiellement malveillant. Elle est basée sur l'estimation du degré de confiance.
3. l'ensemble de mesures qui contribuent à prévenir toute analyse statique/dynamique du code de l'agent mobile.

La première catégorie est supportée par les fonctions de sécurité usuelles qui sont la cryptographie symétrique/asymétrique ainsi que les fonctions de hachage à sens unique. La seconde catégorie est basée sur l'analyse de la crédibilité de l'hôte client et engendre la sélection d'une QoS particulière du service demandé. Enfin, la dernière catégorie utilise principalement la capacité d'adaptation de l'agent mobile qui offre des comportements variés pour des caractéristiques identiques de l'environnement de l'hôte client. Elle emploie aussi la dispersion du code, l'encryptage des modules sensibles ainsi que les techniques d'obscurcissement pour empêcher l'analyse du comportement de l'agent mobile. Les deux dernières catégories vont être amplement détaillées respectivement au sein des chapitres 4 et 5.

Chapitre 4 – Mécanisme d'estimation de la confiance

« Faites confiance à Dieu, mais attachez quand même votre chameau »

[Proverbe persan]

4.1. Introduction

La notion de confiance a été le sujet d'un grand intérêt dans différents domaines de recherches tels que l'économie, la théorie des jeux et les systèmes multi-agents [Braynov 2002], [Dimitrakos 2003a], [Dimitrakos 2003b]. Grandison et Sloman [Grandison 2000] ont souligné l'importance de l'incorporation de la confiance dans les systèmes distribués. Son inclusion permettra des transactions électroniques sécurisées. La confiance joue donc un rôle primordial dans notamment la sécurité du e-commerce et celle du e-business. C'est la clé pour l'acceptation et le déploiement généralisé de ce type d'application. Obtenir et maintenir l'estimation de la confiance est un problème sérieux car elle n'est pas quelque chose qui peut être décrite facilement ; elle dépend plutôt du contexte [Morris 1994].

Ce chapitre souligne le lien intrinsèque existant entre la confiance et la sécurité et met l'accent sur l'importance de la confiance comme facteur améliorant la sécurité des agents mobiles. Il présente notre propre définition de la confiance et décrit les différentes étapes suivies par le mécanisme de la confiance dans notre approche.

4.2. Les agents mobiles et le concept de confiance

Les problèmes entourant le concept de la confiance dans les systèmes d'agents ont fait l'objet de beaucoup de recherche dans la communauté scientifique [Grandison 2000]. L'ostentation dans la littérature est la plupart du temps sur des techniques empêchant les agents malveillants de nuire à leur environnement d'exécution [Castel 2001].

Les décisions fondées sur la confiance reposent fondamentalement sur des paramètres déduits d'observations dérivées de la surveillance d'interactions passées et/ou sur des recommandations de tiers de confiance. Ces deux principales sources d'information de confiance permettent la constitution dynamique d'une opinion sur une entité. Les mécanismes de confiance basés sur la réputation et/ou le risque sont aussi suggérés par un certain nombre d'auteurs [Schillo 2000], [Birk 2001].

Un agent mobile a besoin de décider s'il peut migrer et s'exécuter sur un hôte particulier. Il doit pouvoir prendre ces décisions en se basant sur l'évaluation de la crédibilité de l'information ou de la source de l'information. Toutefois, créer et établir une confiance entre des agents sans aucune (ou très limitée) histoire commune ou connaissance l'un de l'autre, ce qui est le cas dans un système MAS ouvert, n'est pas une tâche évidente.

Cette section explique l'intérêt du concept de confiance comme facteur améliorant la sécurité des systèmes basés agents mobiles.

4.2.1. Confiance et sécurité

L'objectif des mécanismes de sécurité est de fournir une protection contre des parties malveillantes. Les mécanismes de protection traditionnels protègent les ressources contre les utilisateurs malicieux [Yahalom 1993], [Blaze 1996], [Karjo 1997], [Jansen 2000a]. Toutefois, dans beaucoup de situations dans les systèmes distribués, les applications doivent être protégées contre ceux qui offrent des ressources de sorte que le problème est en fait inversé. A titre d'exemple, une ressource fournissant une information peut agir de façon malhonnête en fournissant une information fausse ou abusive. De plus, étant donnée que les ressources sont distantes, les mécanismes de sécurité traditionnels ne sont pas adaptés à ce type de menaces.

Il existe une vaste source d'information sur la théorie et l'application de la confiance [Castel 2000b], [Waidner 2002], [Bosworth 2002], [Nixon 2003], [Jensen 2004], [Hermann 2005]. Dans le monde du Web, la confiance a été reconnue en tant qu'aspect important de prise de décision pour le e-commerce [Grandison 2000], [Josang 2005]. Les clients doivent avoir l'assurance que les vendeurs fourniront les services qu'ils annoncent et ne révéleront pas leurs informations privées (nom, adresse, les détails de la carte de crédit, etc.). Ce volet est important car la confiance dans la compétence et l'honnêteté du fournisseur influence la décision du client quant au choix de celui-ci. De même, les vendeurs doivent aussi être sûrs que l'acheteur est en mesure de payer les produits ou services demandés, qu'il est autorisé à effectuer des achats au nom d'un organisme ou n'est pas en dessous de l'âge spécifiant son autorisation à accéder à certains services ou acquérir certains produits.

Par ailleurs, si on examinait les diverses définitions de la sécurité offertes par différents dictionnaires de la langue française ou anglaise, on remarquerait qu'elle est considérée comme la propriété d'être hors de portée d'un risque, d'un péril ou d'un échec. Un système absolument sécurisé est donc un système en lequel on a une totale confiance ou une certitude absolue. La sécurité est en fait une propriété d'un système tandisque la confiance est conçue comme une relation représentant la perception d'une entité au sujet d'une autre entité ou d'une chose. Les solutions basées sur la confiance produisent par définition directement de la sécurité. D'ailleurs, Camp [Camp 2002] écrit: "*security is not a separable element of trust*". Cette phrase adhère avec le fait que la confiance et la sécurité sont deux mécanismes qui sont classifiés tous deux en tant que mesures protectrices [Ratnasingham 2000]. Irrémédiablement, une stratégie de confiance est reconnue comme moyen d'empêcher un comportement malveillant (ou du moins incongru) et de construire et de sauvegarder le concept de confiance. En fait, La confiance et la crédibilité sont fondamentales pour n'importe quelle solution de sécurité. Le besoin à ces deux aspects de la confiance et les moyens utilisés pour les implémenter affectent le mécanisme de sécurité de n'importe quel système de commerce. Toutefois, il faut distinguer entre ces deux aspects. La confiance est un acte de celui qui recherche s'il peut accorder sa confiance. En revanche, la crédibilité est la caractéristique de quelqu'un ou de quelque chose qui est l'objet de la confiance. L'estimation de la confiance est supportée par l'identification, l'authentification, la responsabilité, l'autorisation, et la disponibilité [Andert 2002].

4.2.2. Profusion de définitions

La confiance a été étudiée sur la base de différentes considérations et a été abordée selon diverses vues. En fait, il n'existe pas de consensus, dans la littérature, quant à la définition de

la confiance [Mcknight 1996] car nous sommes face à un phénomène qui est traité par différentes disciplines des sciences sociales avec chacune leur spécificité. Elle a été reconnue en tant que sujet important et complexe [Usunier 1998] faisant intervenir l'honnêteté, la compétence, la fiabilité, etc. d'une personne, d'un organisme ou d'un service.

La sociologie met à notre disposition une définition de la confiance qui pourrait être utile dans la sécurité informatique. La définition de Coleman [Coleman 1990] explique l'action rationnelle des individus dans des situations sociales. Elle est comportementale plutôt qu'affective ce qui la rend intéressante dans le contexte des agents mobiles. D'un autre côté, Bhattacharya et al. [Bhattacharya 1998] ont fourni une discussion très stimulante sur la notion de confiance en se basant sur la définition suivante : « *Trust is an expectancy of positive (or nonnegative) outcomes that one can receive based on the expected action of another party in an interaction characterized by uncertainty* ». Selon les auteurs, cette définition recèle au moins six caractéristiques qu'ils considèrent comme essentielles :

1 La question de la confiance ne se pose que dans un environnement d'incertitude.
2 La confiance renvoie à une attente, donc à la possibilité d'une erreur ou d'une réaction décevante (vulnérabilité).
3 Dans la relation de confiance il est important de tenir compte de l'enjeu, de l'envergure de l'objet de confiance (importance).
4 Il convient également de tenir compte du degré de confiance accordée. La confiance accordée peut être quasi-totale ou très faible (degré).
5 La relation de confiance requiert la mutualité, le répondant du partenaire (réciprocité).
6 La confiance est établie en relation avec une fin qui est « bonne » (non négative).

L'un des travaux les plus influents ayant apporté une définition pratique de la confiance est celui de Gambetta [Gambetta 1990]: "*When we say we trust someone or that someone is trustworthy, we implicitly mean that the probability that he will perform an action that is beneficial or at least not detrimental to us is high enough for us to consider engaging in some form of cooperation with him. Correspondingly, when we say that someone is untrustworthy, we imply that that probability is low enough for us to refrain from doing so*." Cette définition repose sur le fait que la confiance est fondamentalement une croyance ou une estimation qui a inspiré l'utilisation d'une logique subjective comme un moyen de mesurer la confiance [Josang 1999a], [Josang 1999b]. Castelfranchi et Falcone [Castel 1998] ont étendu la définition de Gambetta en incluant la notion de compétence avec la crédibilité.

Kini et Choobineh [Kini 1998] ont examiné la confiance du point de vue de la théorie de la personnalité, de la sociologie, des sciences économiques et de la psychologie sociale. Ils soulignent les implications de ces définitions et combinent leurs résultats pour créer leur propre définition de la confiance dans un système. Ils définissent la confiance comme: "*a belief that is influenced by the individual's opinion about certain critical system features*". Leur analyse couvre des aspects variés de la confiance humaine dans les systèmes d'ordinateurs mais ils n'abordent pas la question de la confiance entre des parties (humains ou processus) concernées par des transactions du e-commerce.

Au niveau du projet Trust-EC de la ECJRC[17], Jones [Jones 1999] définit la confiance en tant que: "*the property of a business relationship, such that reliance can be placed on the business partners and the business transactions developed with them*". Jones établit l'importance des questions de l'identification et de la fiabilité des partenaires commerciaux ; de la confidence des informations sensibles; de l'intégrité de l'information précieuse; de la prévention de la copie et de l'utilisation non autorisées de l'information ; de la garantie de la qualité des

[17] ECJRC : European Commission Joint Research Centre, voir http://dsa-isis.jcr.it/TrustEC

produits digitaux; de la disponibilité des informations critiques; de la gestion des risques liés aux informations critiques ainsi que de la fiabilité des services informatiques et des systèmes.

Grandison et Sloman [Grandison 2000] ont aussi examiné diverses définitions de la confiance. En suivant une brève analyse de ces définitions, ils ont énoncé: *"the firm belief in the competence of an entity to act dependably, securely and reliably within a specified context"*. Ils déduisent que la confiance est une combinaison de différents attributs – la fiabilité, l'honnêteté, la crédibilité, la sécurité, la compétence et la conjoncture – qui peuvent être considérés et définis selon l'environnement dans lequel la confiance est spécifiée.

Dimitrakos [Dimitrakos 2001] a défini la confiance comme suit: "*Trust of a party A in a party B for a service X is the measurable belief of A in B behaving dependably for a specified period within a specified context in relation to X*". Dans sa définition, la crédibilité est utilisée dans un sens large afin d'inclure la sécurité, la sûreté, la fiabilité, la conjoncture ou l'opportunité.

Josang et al. [Josang 2005] ont défini la confiance comme: "*the extent to which one party is willing to depend on something or somebody in a given situation with a feeling of a relative security, even though negative consequences are possible*". Leur définition inclut des aspects tels que la dépendance ou la crédibilité en une partie ou entité de confiance, l'utilité qui peut être positive ou négative selon qu'elle découle d'un résultat positif ou négatif ; et une certaine attitude du risque dans le sens que la partie de confiance est disposée à accepter le risque résultant des éléments précédents.

Quelques aspects de ces définitions sont communs, d'autres sont complémentaires. A titre d'exemple, [Gambetta 1990] déduit que la confiance est en partie subjective. Dans d'autres définitions comme celles de [Grandison 2000], [Dimitrakos 2001] et [Josang 2005] elle est vue comme un présent caractéristique. Grandison [Grandison 2000] souligne que la confiance est une croyance dans la compétence d'une entité dans un contexte donné, tandisque [Kini 1998] indique que l'entité qui manifeste la confiance (trustor) est un humain – pas un système. La définition de [Jones 1999] focalise sur l'aspect que la confiance dans le domaine du e-commerce est relative aux relations des affaires. Une entité peut faire confiance à une autre entité pour une affaire spécifique et non en général. Enfin, la définition donnée dans [Dimitrakos 2001] met en évidence un point important qui est le fait que la confiance évolue dans le temps et qu'elle est mesurable.

En tant que phénomène unidimensionnel, la confiance a été systématisée comme un état psychologique ou un comportement choisi. Le point de vue de l'état psychologique définit la confiance en termes de processus cognitifs corrélés et d'orientations vers des croyances ou espoirs positifs par rapport aux autres. Quelques définitions s'appliquent aux contextes où elles sont identifiables, comme dans [Rousseau 1998] "*trust is a psychological state comprising the intention to accept vulnerability based upon positive expectations of the intentions or behavior of another*". Dans d'autres définitions, la confiance est présentée comme une attitude plus générale ou une perspective au sujet d'autres personnes ou au sujet du système social général. Le point commun à toutes ces définitions est la référence aux états de vulnérabilité, de confiance (confidence) et d'espérances positives. La confiance en tant que comportement bien choisi peut être vue comme la bonne volonté à prendre des risque en agissant sur la base de mots, d'actions, de décisions ou autres [Zand 1972]. Dans quelques définitions, la confiance est perçue comme une décision plus ou moins rationnelle motivée principalement par la perception des risques concernant soit la probabilité d'une coopération réussie ou la possibilité de la réduction du coût de la transaction.

La confiance peut être de deux types : confiance molle (soft trust) ou confiance dure (hard trust) en se basant sur les définitions données par Lin et al. [Lin 2004]:

Définition 1: « *Hard trust is the trust that is established via cryptographic Mechanisms* ».

Définition 2: « *Soft trust is the trust that is derived from non-cryptographic mechanisms such as social control via recommendation protocols, observation and direct experiences* ».

La confiance peut être composée de différents éléments, chacun ayant un caractère différent. Les diverses prédispositions, caractéristiques et intentions de celui dont on évalue la crédibilité et des conditions situationnelles telles que le degré de risque peuvent déterminer le niveau de la confiance. L'importance relative de ces facteurs est déterminée par le type des relations de confiances préétablies.

4.2.3. Travaux traitant de l'estimation de la confiance

De nombreux modèles généraux traitant la confiance ont été proposés pour introduire la notion de confiance dans le contexte des applications à systèmes distribués [Abdul-Rahman 1997], [Gambetta 1990], [Lin 2003]; cependant, seulement un nombre restreint de ces modèles ont abordé la question d'intégration de la confiance avec la sécurité dans les systèmes basés agents mobiles [Grandison 2000]. Beth [Yahalom 1993] a proposé l'un des modèles de confiance les plus précoces pour l'authentification dans les systèmes distribués tandis que Abdul-Rahman et al. [Abdul-Rahman 1997] a produit un modèle général basé sur des recommandations. Blaze et al. [Blaze 1996], [Blaze 1999] ont proposé un système de gestion de la confiance bien connu pour l'amélioration de l'application de l'autorisation dans les systèmes distribués en uniformisant le traitement des qualifications de sécurité, des politiques et des rapports de confiance. Toutefois, ces modèles ne se sont pas intéressés à la dynamique de la confiance basée sur le comportement. Ils se sont concentrés sur l'évaluation de la sécurité et l'expression de la confiance dans les systèmes d'agents mobiles. C'est aussi le cas des modèles proposés par Karjo [Karjo 1997] et Jansen [Jansen 2000a].

Wilhelm et al. [Wilhelm 1998], [Wilhelm 2000] ont publié l'une des discussions les plus complètes sur la question de la confiance dans les systèmes mobiles. Ils ont identifié quatre fondements de la confiance à savoir: la confiance sans visibilité (blind trust), la confiance basée sur la « bonne » réputation, la confiance basée sur le contrôle et la punition et la confiance basée sur l'application d'une politique. Le protocole CryPO constitue leur solution au problème de la confiance dans les systèmes à agents mobiles. Il est basé sur le matériel sécurisé (tamper-proof material) pour fournir des environnements sécurisés qui constituent la base pour l'exécuteur de l'agent. Ce protocole utilise des certificats ainsi que des techniques de cryptographie pour garantir la sécurité. Il est une extension du framework de certification. Cette solution offre, à un fournisseur de service, une manière plus aisée pour s'établir dans un marché; elle lui permet également de protéger des données spécifiques d'un agent mobile. Cependant, la notion de la confiance utilisée est statique et elle correspond à un constructeur de matériel particulier; par conséquent, cette approche ne permet pas la mise à jour de la confiance et limite donc la flexibilité dans l'application.

Manchala [Manchala 1998], [Manchala 2000] a proposé un modèle basé sur des variables relatives à la confiance telles que le coût de la transaction et son historique et définit des matrices de décision basées sur la relation entre le risque et la confiance. Celles-ci sont employées avec les règles d'inférence de la logique floue pour déterminer s'il est possible ou non de traiter avec une partie particulière.

Tan et Moreau [Tan 2001] ont développé un modèle de confiance et de croyance, pour les systèmes d'agents, plus complet. Il est basé sur des mécanismes d'authentification distribués établis par Yaholm et al. [Yahalom 1993] et utilisant des tiers de confiance (Trusted Third Parties-TTP) en tant que serveurs de vérification. Leur modèle de dérivation est fondé sur les similitudes entre l'authentification distribuée dans les infrastructures à clé publique et la

confiance de l'agent mobile. Cependant, leur framework est simple et limité puisqu'il incorpore uniquement la notion de la confiance associée au mécanisme étendu de sécurité basé sur la trace de l'exécution [Tan 2002a] et ne prend pas en compte des mécanismes ou conditions de sécurité génériques. De plus, la propriété de transitivité utilisée dans les règles d'inférence pour la dérivation de la confiance est basée sur la disponibilité des TTPs. Cette hypothèse semble être très contraignante pour des applications d'agents mobiles dans un réseau ouvert où une telle structure de regroupement de serveurs de vérification n'est pas toujours disponible. Leur unique contribution est l'un des seuls travaux qui incorporent la confiance dans le développement du système d'agent mobile. Ce travail définit des rapports de confiance exprimés dans la plate-forme d'agents mobiles Marism-a[18] [Robles 2002a, Robles 2002b]. Bien qu'initialement ils aient établi une expression statique de la confiance, Marism-a emploie, à l'heure actuelle, des rapports de confiance définis parmi des entités et des actions dans une transaction d'agent mobile.

Braynov et al. [Braynov 2002] ont apporté une solution qui ne se fonde pas sur la collecte et l'analyse des informations sur les agents non fiables. Ils proposent, plutôt, un mécanisme d'incitation dans lequel les agents révèlent leur degré de confiance au début de chaque interaction. Dans ce mécanisme, les agents rapportent leur véritable niveau de confiance même s'ils ne sont pas fiables.

Kagal et al. [Kagal 2001] ont suggéré l'ajout de la confiance pour améliorer la sécurité dans les scénarios de calcul mobiles et ont défini des activités de gestion de la confiance telles que celles définies par [Blaze 1996] : le développement des politiques de sécurité, l'assignation des qualifications et leur vérification par rapport aux conditions de politique et la délégation de la confiance à d'autres parties. Kagal a étendu son modèle de délégation de la confiance aux scénarios d'agents mobiles [Kagal 2002] et prend en charge les limites de la FIPA pour sécuriser le système de gestion d'agents. Bien qu'il s'intéresse à la manière dont les agents établissent la confiance quand ils sont inconnus, il se concentre sur principalement l'authentification et ne considère ni les exigences de sécurité spécifiques à la mobilité ni l'expression de la confiance.

Cahill et al. [Cahil 2003] ont proposé le projet SECURE (SEcure Collaboration among Ubiquitous Roaming Entities) – un projet qui s'intéresse principalement à la construction d'infrastructures de confiance pour les réseaux sans fil ad hoc – qui étudie la conception des mécanismes de sécurité pour le calcul dominant (pervasive computing) fondé sur la notion humaine de la confiance. La principale contribution de SECURE est de fournir à des entités une base de raisonnement sur la confiance et le risque incorporés dans un framework informatique pouvant être adapté à une variété de scénarios d'application. Ils précisent également que l'information de confiance est basée sur deux sources : observations personnelles (interactions précédentes) et recommandations d'autres parties de confiance (confiance transitive ou déléguée). Cependant, il n'y a aucun lien fourni entre les conditions de sécurité, la confiance, et les buts d'application pour les agents mobiles. De plus, la manière dont ils développent l'adaptation dynamique de la confiance n'est guère claire. Carbone et al. ont développé le modèle de confiance pour SECURE dans [Carbone 2003] et fournissent une manière formelle de raisonner au sujet du rapport de confiance entre des parties. Leur utilisation d'un intervalle de confiance de 0 à 1 peut être considéré comme une mesure d'incertitude.

De manière similaire, Capra [Capra 2004] a défini un modèle formel pour le hTrust — la gestion de la confiance dans un réseau mobile qui définit la formation, la diffusion, et l'évolution de la confiance. Capra a passé en revue plusieurs cadres de gestion de la confiance

[18] Marism-a: Architecture for Mobile Agents with Recursive Itinerary and Secure Migration.

qui ont les limitations suivantes : elles sont conçues pour les serveurs centralisés, elles ont un coût de calcul prohibitif (pour les dispositifs mobiles), elles ne prennent pas en compte l'évolution dynamique de confiance et elles ne détaillent pas l'utilisation locale de la politique. Le hTrust remédie à ces imperfections et incorpore directement, au cadre de gestion de la confiance, une notion humaine de la confiance. Il modélise une gamme des valeurs de confiance de sorte qu'un manque d'évidence ou de décision de la connaissance puisse être distingué d'une décision basée sur la confiance et permet de se méfier d'une autre partie. Le modèle de données de confiance incorpore également la notion du temps de sorte que les rapports de confiance peuvent se dégrader. Des recommandations sont employées quand une partie n'a pas d'historique. Comme dans des interactions humaines, le hTrust utilise la notion de recommandations des personnes qui ont fourni de bonnes recommandations dans le passé et rejette celles émanant de personnes qui ont déçu dans le passé. Enfin, un aspect clé de ce modèle est son incorporation dans un contexte social et transactionnel (le réseau des services qui sont assurés par des parties dans le système). Malgré que le hTrust fournisse un cadre générique pour l'expression de la confiance des applications mobiles, il ne traite pas directement les conditions spécifiques de sécurité des agents mobiles.

Dimitrakos [Dimitrakos 2003a] a introduit les métriques, les fonctions de coût et de service comme paramètres d'un algorithme produisant la politique de confiance en vue d'une décision basée sur la confiance. Néanmoins, ce travail a besoin d'une définition quantitative des diverses mesures impliquées.

Josang [Josang 2001], [Josang 2004] a proposé un schéma pour la propagation de la confiance à travers un réseau informatique basée sur des certificats à clé publique ainsi que des rapports de confiance. Il démontre la manière dont les mesures résultant de la confiance peuvent être utilisées pour prendre des décisions au sujet des transactions électroniques. Il définit également un modèle de confiance composé de la fiabilité de la confiance comme une probabilité du succès d'une transaction et une décision de confiance dérivée de la surface de décision. L'adaptabilité de la confiance dans le temps n'a pas été considérée.

Chin Lin et al. [Lin 2003], [Lin 2004], [Mu 2004], [Lin 2005] ont suggéré un modèle de confiance hybride employant des mécanismes logiciels de la confiance avec des constructions telles que la recommandation, les expériences directes par l'intermédiaire des interactions ainsi que les observations. Ces mécanismes sont utilisés pour estimer la confiance "dure" (basée sur les mécanismes cryptographiques) afin d'améliorer la sécurité de l'agent mobile dans des situations d'indisponibilité d'une totale authentification de la confiance due à l'absence ou à la non disponibilité d'une tierce partie de confiance.

Antonopoulos et ses collègues [Antonopoulos 2000] développent un mécanisme général de contrôle d'accès qui est de nature distribuée et est concentré sur les agents mobiles. Cette approche exprime des rapports de confiance parmi des parties, mais elle est fondamentalement basée sur des listes de contrôle d'accès pour prévenir l'activité malveillante du code.

Castelfranchi et al. [Castel 2000a] plaident en faveur d'une vue cognitive de la confiance comme structure complexe de croyance et de buts, impliquant que l'administrateur de la confiance (trustor) doit avoir une "théorie de l'esprit" de l'administré (trustee). Une telle structure de croyance détermine un degré de confiance et une évaluation de risque et engendre une décision pour compter ou pas sur l'autre. La décision est également basée sur un seuil du risque personnel à accepter ou à éviter. Ils expliquent les composants et les utilisations raisonnables et irrationnels de la confiance. Leur travail représente un appui de notre approche où la valeur de la confiance est basée sur les ingrédients de confiance (évaluation de la situation et du comportement).

Tous les modèles de confiance décrits dans cette section utilisent exclusivement la confiance molle [Abdul-Rahman 1997], [Manchala 2000], [Castel 2000a], [Antonopoulos 2000], [Braynov 2002], [Cahil 2003], [Capra 2004] – basée sur des techniques non cryptographiques – ou bien la confiance dure [Wilhelm 2000], [Kagal 2001], [Tan 2001], [Mu 2004], [Josang 2004] – basée sur les mécanismes cryptographiques conventionnels. En général, les modèles utilisant la confiance molle ne considèrent pas les problèmes de la confiance dure ou de l'authentification qui sont pourtant importantes pour les propriétés de sécurité des applications distribuées telles que les systèmes d'agents mobiles.

4.2.4. Synthèse

Les problèmes de sécurité sont principalement liés à l'exécution de l'agent mobile sous le contrôle d'un hôte potentiellement malicieux. Ceci est lié au fait que les agents ont besoin d'agir en fonction de l'information reçue à partir de diverses entités. Par conséquent, la crédibilité de cette information doit être garantie par le système et être considérée par l'agent. En outre, l'estimation de la confiance permet d'avoir une idée sur les risques encourus, de se protéger par voie de conséquences contre eux et de prévenir certains abus.

La majorité des modèles de confiance, étudiés dans cette section, génèrent une valeur unique qui, en cas d'échec, ne reflète pas la cause exacte du défaut de confiance. Par conséquent, la décision résultant de cette estimation pourrait être inappropriée. Par ailleurs, certains modèles utilisent également la réputation de l'hôte comme facteur intervenant dans l'estimation de la confiance. Nous considérons qu'il n'est pas nécessaire que l'agent mobile (ou le pourvoyeur de service) connaisse la réputation des hôtes visités qui peuvent être nouveaux dans le réseau. Dans notre approche, l'évaluation de la confiance est dynamique et elle est employée pour permettre à l'agent mobile d'adapter son exécution dans les environnements non fiables.

Différentes définitions de la confiance ont été proposées. Nous avons ressenti le besoin de présenter notre propre définition qui est ajustée au problème de protection d'un agent mobile. Elle a été inspirée par un certain nombre de définitions. En premier lieu, la définition donnée par Gambetta [Gambetta 1990] nous a intéressée car elle fait apparaître l'idée de la quantification de la confiance et l'allie à la compétence. Cependant, elle la désigne comme étant une probabilité subjective donc ne pouvant pas engendrer une décision totalement rationnelle. La définition de Josang nous a aussi interpellés car elle convient aux environnements dynamiques [Josang 2005], et associe la confiance à une relative sécurité. En second lieu, nous approuvons l'idée de Castelfranchi et al. [Castel 2000a] qui clament: "*the richness of the mental ingredients of trust cannot and should not be compressed simply in the subjective probability estimated by the actor for its decision*". Nous attestons que la crédibilité de l'hôte repose sur non seulement plusieurs paramètres mais aussi sur des comportements malicieux. La question est donc: pourquoi a-t-on besoin d'un compte explicite des ingrédients mentaux de la confiance?

La confiance peut être composée de différents éléments, chacun ayant un caractère différent. Les diverses prédispositions, caractéristiques et intentions de celui dont on évalue la crédibilité et des conditions situationnelles telles que sa compétence à mener à terme une tâche donnée. L'importance relative de ces paramètres est déterminée par le type des relations de confiance préétablies. A notre avis, il est primordial d'avoir à priori un consensus, entre les deux parties faisant intervenir la confiance, qui établit les caractéristiques constituant la base d'une estimation de la confiance. Dans notre cas : « *La confiance d'une partie A en une partie B pour un service X est la croyance mesurable de A en B à un moment donné en se basant sur les paramètres aidant à son authentification et prouvant sa fiabilité et sa compétence dans un contexte donné pouvant évoluer dans le temps* ». (« *The trust of a part A in a part B for a*

service X is the measurable belief of A in B in a given moment based on the parameters helping to its authentication and proving its reliability and its competence in a given context which can evolve in time »). Notre mécanisme d'estimation de la confiance est hybride puisqu'il utilise la confiance dure (établie via des mécanismes cryptographiques) et la confiance molle (établie via des mécanismes non cryptographiques).

Dans le contexte courant, la définition de la confiance exige de répondre aux questions suivantes :

- Comment un agent peut percevoir son environnement de sorte qu'il puisse émettre une juste opinion sur la crédibilité et le profile de l'hôte visité?
- Comment diverses perceptions peuvent être agrégées pour générer une valeur de la confiance?
- Comment cette valeur peut déterminer exactement la catégorie de ce client?
- Comment cette valeur peut, dans le cas d'un mauvais comportement, expliquer l'origine de l'échec (défaut de confiance) pour éviter une réaction inadéquate?

La section suivante de ce chapitre répond à ces interrogations en expliquant le mécanisme d'estimation de la confiance que nous proposons pour améliorer la sécurité des agents mobiles.

4.3. Estimation de la confiance

Lorsqu'un exécuteur reçoit un agent mobile, le propriétaire perd tout contrôle sur le code et les données de l'agent. L'exécuteur peut décompiler le code, analyser ses données, ou modifier l'un ou l'autre. S'il n'y a pas de possibilité d'attaques directes, l'exécuteur peut toujours expérimenter l'agent en le réinitialisant et en lui fournissant des données arbitraires pour observer ses réactions. L'exécuteur peut également réaliser cette opération avec une copie de l'agent placée sur une plate-forme isolée. Par conséquent, le propriétaire de l'agent doit être sûr de la confiance qu'il peut placer en l'exécuteur afin d'empêcher l'utilisation illégale de ces méthodes. Pour cette raison, l'évaluation de la crédibilité de l'exécuteur représente une étape primordiale de notre approche.

L'agent mobile fait confiance à l'hôte lorsqu'il considère qu'il est sécurisé. D'où, il est important que les agents mobiles puissent authentifier et identifier leurs hôtes de manière à pouvoir leur faire confiance (ou du moins pouvoir se construire une opinion sur leur crédibilité).

Un contrôle est exigé pour l'établissement de la confiance et l'amélioration de la sécurité. Ainsi, si un agent mobile ne fait pas confiance à un hôte, il utilise, par exemple, l'observation pour empêcher ou tout au moins détecter son mauvais comportement. En outre, si l'hôte sait qu'il est observé, il devient plus fiable.

La confiance accordée par l'agent mobile est fondée sur les paramètres qui renseignent sur la crédibilité de l'hôte visité. La quantification de la confiance se base sur les valeurs attribuées à ces paramètres. Cette section révèle le mécanisme d'estimation de confiance proposé par notre approche de protection. Elle dévoile, en premier lieu, les spécificités des paramètres qui contribuent à l'estimation de la confiance. Ensuite, elle présente les algorithmes relatifs à l'évaluation de la confiance et à la génération de la clé environnementale. Elle explique enfin la phase de prise de décision pour la sélection de la QoS la plus appropriée.

4.3.1. Bases de l'estimation de la confiance

TAMAP doit permettre à l'agent mobile de transporter le service offert par le pourvoyeur de service en toute sécurité. Le service est caractérisé par des QoSs qui correspondent directement au degré de confiance estimé par l'agent mobile; plus les conditions de sécurité sont satisfaites, plus le niveau de confiance est bon, meilleure est la QoS offerte. L'estimation de la confiance repose donc sur une modulation du service selon la confiance accordée. Elle exige, par ailleurs, l'identification de tous les paramètres permettant de se forger une opinion sur la crédibilité de l'hôte d'accueil (appelés paramètres de confiance), leur évaluation et enfin la quantification de la confiance. En effet, l'information détenue par le client a une incidence sur la décision d'exécuter ou non le service demandé. La valeur de la confiance est donc le résultat obtenu à la suite du contrôle de l'hôte cible par une observation, une inspection et une interaction et engendre une réaction de la part de l'agent mobile qui peut soit assurer le service, le réduire ou l'annuler. Afin d'évaluer la confiance, chaque agent mobile emploie donc (*cf.* Figure 4.1) :

- Un mécanisme d'observation qui capture tous les paramètres susceptibles de contribuer à la perception de son environnement et d'informer sur le degré de confiance de l'hôte client. C'est un mécanisme de prévention et de détection de toute tentative de manipulation malveillante contre l'agent mobile. Cela est effectué en surveillant tous les paramètres aptes à renseigner sur l'environnement. A ce propos, nous utilisons le concept de «comportement typique» qui correspond au comportement normal d'un client et l'opposons à celui d'un «comportement atypique» qui pourrait permettre la détection d'un agissement malicieux mais qui n'est pas toujours dangereux [Zanero 2004].
- Un mécanisme d'interaction qui permet à l'agent mobile de poser à l'hôte des questions dont les réponses sont préalablement connues par le pourvoyeur de service (ou le propriétaire de l'agent). Dans ce cas, la confiance peut être basée sur les informations privées du client.
- Un mécanisme d'inspection qui concède à l'agent mobile la possibilité d'examiner l'environnement à la recherche d'une information particulière (telle qu'un fichier ou un dossier spécifique).

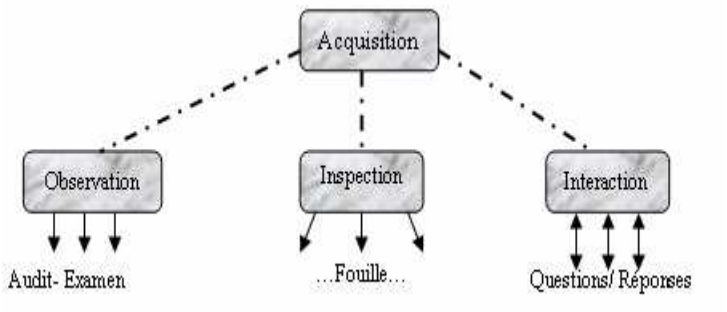

Fig. 4.1. Les différents mécanismes d'acquisition de la confiance

Il existe des techniques variées d'observation et d'inspection. Nous nous référons, par exemple, à la méthode Red Pill de Rutkowska [Rutkowska 2006] et plus généralement aux

techniques de rootkit[19] basées sur la visualisation ainsi que les questions relatives à la détection [Hoglund 2006], [ht1], [ht2], [ht3].

4.3.2. Paramètres de la confiance

Pour protéger l'agent mobile et éviter ainsi que le service ne soit intercepté, celui-ci doit être encrypté. Pour cela, une clé symétrique doit être utilisée. Elle servira à encrypter et à décrypter le service transporté du côté consécutivement du pourvoyeur de service et de celui du client. Cette clé doit être elle-même protégée pour prévenir l'interception du service par un hôte malveillant. Il faut donc éviter qu'elle ne soit transportée par l'agent mobile et faire en sorte qu'elle soit calculée aussi bien au niveau du client que celui du pourvoyeur de service sur la base d'informations environnementales collectées auprès de l'hôte visité. Cette clé désignée par "clé environnementale"[20] possède un double rôle car elle devra non seulement protéger le service mais aussi renseigner sur la crédibilité de l'hôte. Son analyse doit donc mettre en évidence les raisons ou causes du résultat obtenu (la valeur de la confiance); particulièrement, lorsque le degré de la confiance estimée est bas. Par conséquent, cette clé ne doit pas être une simple valeur mais l'agrégation d'un ensemble de valeurs significatives. Ce qui rejoint l'idée de Castelfranchi qui stipule que les dimensions quantitatives de la confiance sont basées sur celles de ses constituants cognitifs [Castel 2000a]. Dans notre approche, l'estimation de la confiance est basée sur l'analyse d'un ensemble de valeurs de paramètres qui représentent un prérequis à l'exécution du service demandé. Ces paramètres ont été identifiés en tant qu'éléments importants dont les valeurs doivent dénoter les conditions de sécurité caractérisant l'environnement d'accueil et signaler un comportement malicieux. Ainsi, pour déterminer la crédibilité de l'hôte, différentes sortes de paramètres doivent être identifiés. Nous les avons classifiés comme suit:

1 Les paramètres qui rendent une interaction digne de confiance.
2 Les paramètres qui déterminent la catégorie des clients à laquelle l'hôte appartient.
3 Les paramètres de logiciels et de matériels qui peuvent affecter la perception de la confiance et l'exécution du service.
4 La réputation (peut ne pas exister) de l'hôte client visité fourni par le propriétaire de l'agent ou un tiers de confiance et qui est relative à l'historique de ses transactions.

Nous notons que les types de paramètres dans, par exemple, le second et le quatrième point sont différents. Le second point exprime tous les paramètres qui sont relatifs à l'intimité de l'hôte (tel que le mot de passe) tandis que le troisième point fait référence aux paramètres qui sont associés à l'infrastructure de l'hôte et qui ne sont pas privés. Ceci nous amène à distinguer deux catégories de paramètres. La première catégorie correspond aux caractéristiques privées de celui qui subit la confiance (trustee) et renseigne sur la confiance dite interne. Tandisque la seconde catégorie, correspondant à la confiance externe, désigne les difficultés techniques et environnementales qui entravent la bonne exécution d'un service. En effet, des obstacles peuvent affecter la manière dont doit être gérée l'exécution du service.

[19] Un rootkit est un ensemble de programmes et de code permettant d'établir une présence permanente et indétectable sur un ordinateur [hog]. Les rootkits sont capables de surveiller ce qui se passe dans un système. Leur spécificité est qu'ils sont pour l'instant impossible à détecter et pratiquement impossible à supprimer. C'est le cas des malwares qui peuvent être très difficiles à démasquer et parfois, à éradiquer. Ils possèdent deux caractéristiques originales: (i) ils modifient en profondeur le fonctionnement du système d'exploitation; (ii) ils se rendent invisibles à ce système d'exploitation. Ils peuvent rendre aussi invisibles divers autres parasites qu'ils auront installés: spyware, porte dérobée, cheval de Troie... et c'est ce qui constitue en général la raison majeure de leur existence. Utilisés correctement ce type de programmes permet la collecte d'information, le contrôle des systèmes infectés ou bien sûr la détection d'une attaque directe.

[20] Cette clé est construite en utilisant des informations environnementales propres à l'hôte visité.

Elles peuvent concerner les ressources réseau comme la largeur de la bande passante, la latence, le temps CPU, la mémoire ou la puissance de la batterie d'un ordinateur portable. Nous précisons que cette catégorie peut aussi englober la réputation de l'hôte qui est considérée comme un paramètre intervenant au niveau de la confiance indirecte[21].

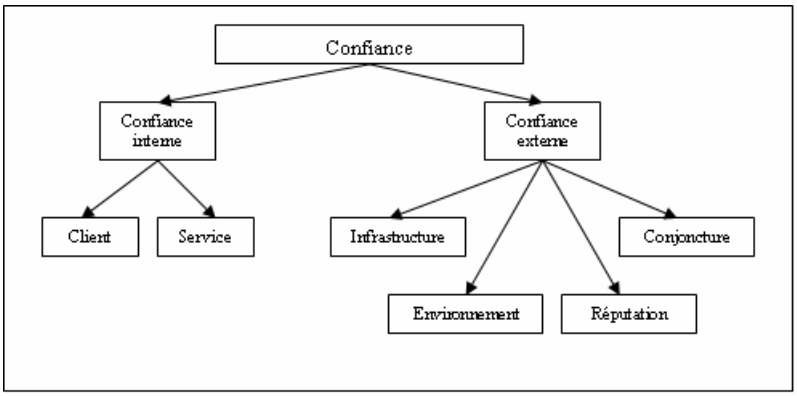

Fig.4.2. Modélisation des paramètres de la confiance

Nous subdivisons donc les paramètres de confiance en deux groupes (*cf.* Figure 4.2, Figure 4.3):

- Les paramètres internes définissant la confiance interne. Ils impliquent toutes les informations susceptibles d'authentifier la relation existant entre le client et le pourvoyeur de service telles que l'identité (nom, organisme…), contrat (type, référence, acronyme du service, période de validité...), certificat, informations privées (mot de passe...), etc.
- Les paramètres externes désignant la confiance externe (environnement, infrastructure, conjoncture, réputation …).

Nous notons que les valeurs des paramètres concernant la réputation ou la conjoncture ne sont pas collectées par l'agent mobile mais détenues par le pourvoyeur de service.

[21] La confiance indirecte est généralement fournie par une tierce partie. Elle n'est donc pas directement estimée par l'agent mobile de façon dynamique. Par conséquent, elle peut être périmée et n'est donc pas forcément crédible. C'est la raison pour laquelle nous avons choisi de l'insérer parmi les paramètres attribués à la confiance externe.

<table>
<tr><th></th><th></th><th>Objet d'authentification</th><th>Exemples de paramètres de confiance</th></tr>
<tr><td rowspan="8">Confiance</td><td rowspan="2">Confiance interne</td><td>Client</td><td>• L'identificateur unique du client.
• Le pseudonyme.
• Le mot de passe.
• Le certificat.
• La date de validité du certificat.
• Répertoire temporaire du client.
• Répertoire local du client.
• …</td></tr>
<tr><td>Service</td><td>• L'acronyme du service autorisé.
• Numéro de référence du contrat.
• Date de validité du service.
• …</td></tr>
<tr><td rowspan="6">Confiance externe</td><td>Infrastructure</td><td>• Version du système d'exploitation.
• Type de système d'exploitation.
• …</td></tr>
<tr><td>Environnement</td><td>• Type d'architecture du processeur.
• Taille de la RAM
• Vitesse du processeur
• Débit de la connexion
• …</td></tr>
<tr><td>Réputation</td><td>Valeur quantitative ou qualitative</td></tr>
<tr><td>Conjoncture</td><td>• Guerre.
• Etat de siège.
• Embargo.
• Tremblement de terre.
• Inondations
• …</td></tr>
</table>

Fig. 4.3. Exemples de paramètres de confiance

La composition entre les paramètres internes et externes produit différentes estimations de la confiance. En effet, pour opter pour la meilleure réaction à adopter, le pourvoyeur de service doit distinguer entre les deux catégories de paramètres. La raison repose sur le fait que si l'hôte est hésitant ou en panne ceci ne signifie pas qu'il est automatiquement malveillant. L'analyse de la clé environnementale générée permet de réaliser un diagnostic méticuleux du manque de confiance.

4.3.3. Modélisation de la confiance

L'agrégation de données constitue un élément important dans les mécanismes de raisonnements humains dans l'élaboration d'une décision finale telle que l'estimation de la confiance qu'il est possible d'accorder à une entité. Le problème posé ici est comment combiner les valeurs des paramètres collectées par l'agent mobile afin d'estimer la confiance à accorder à l'hôte visité ? Quelle pondération donner à chacun des paramètres inventoriés ?

Quels opérateurs utiliser ? Nous avons pour cela choisi l'approche intuitive qui consiste à préciser les propriétés souhaitables des paramètres de confiance intuitivement et à répertorier les opérateurs qui les satisfont.

Soit $P = \{p_1, p_2, \ldots, p_j, \ldots p_n\}$ l'ensemble des paramètres participant à l'évaluation de la confiance.

En premier lieu, l'introduction d'une pondération des différents paramètres à agréger est proposée. Nous notons Wj le poids et Ij l'importance affectés au paramètre p_j. Le poids exprime la puissance du paramètre tandisque l'importance révèle son impact dans la décision finale ; spécialement lorsque la valeur de la confiance est basse ou non significative (cas de suspicion de malveillance). Par ailleurs, il ne faut pas perdre de vue notre objectif qui est celui de vérifier la crédibilité de l'hôte visité. Une corroboration des valeurs des paramètres collectées auprès de l'hôte visité avec celles détenues par le pourvoyeur de service (enregistrées au cours de l'élaboration du contrat du service avec le client) s'impose. Le résultat de cette comparaison est consigné par un facteur de validation S_J. Ce dernier possède une valeur binaire (0 pour une valeur non valide ou 1 pour une valeur valide) correspondant au résultat du test de validation de la valeur collectée relative au paramètre p_j.

Les opérateurs adoptés sont les opérateurs de multiplication et d'addition. La multiplication est employée pour évaluer l'impact d'un paramètre donné alors que la somme est utilisée pour évaluation globale de l'ensemble des valeurs des paramètres. Nous posons la contrainte suivante qui trouve son utilité dans la phase de prise de décision:

$$\sum_{j=1}^{n} W_j I_j = 100 \qquad (1)$$

La formule de la confiance T (Trust) s'écrit alors :

$$T = \sum_{j=1}^{k} w_j \; I_j \; s_j \; . \qquad (2)$$

Nous remarquons que lorsque le résultat de la validation de la valeur d'un paramètre p_j produit S_J=0, celui-ci n'est pas comptabilisé (puisque Wj Ij S_J =0) et la valeur de la confiance T se trouve diminuée. De plus, la formule d'évaluation de la confiance proposée est linéaire est de complexité O(n) car pour calculer T, on effectue 2n multiplications et n-1 additions donc 3n-1 calculs élémentaires. La complexité de 3n est O(n) donc la complexité de T est O(n). Par conséquent, elle n'engendre pas de surcoût en temps d'exécution.

Soient VC, VD, W, I et S des vecteurs associés aux n paramètres de confiance et qui comprennent respectivement les valeurs collectées par l'agent mobile, les valeurs détenues par le pourvoyeur de service, leurs poids, leurs importances et enfin leurs facteurs de validation. La Figure 4.4 montre l'algorithme de la fonction d'évaluation de la confiance.

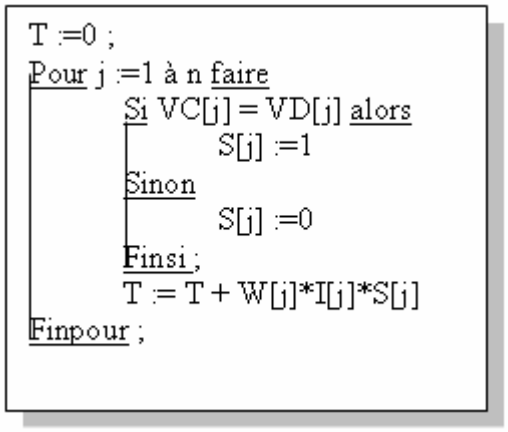

Fig.4.4. Evaluation de la confiance

L'identification des paramètres de la confiance ainsi que leur pondération sont réalisés par le concepteur de l'agent mobile. Ces deux opérations sont intimement liées aux critères de sécurité du service. La décision qui découle de l'estimation de la confiance est présentée au niveau de la Section 4.3.5.

4.3.4. Génération de la clé environnementale

Cette section présente les différentes étapes suivies aussi bien par le pourvoyeur de service que par l'agent mobile pour assurer le service requis. Elle détaille principalement l'opération de génération de la clé environnementale qui représente le support du mécanisme d'estimation de la confiance puisqu'il allie la confiance dure et la confiance molle conjointement utilisées par notre approche. Elle offre un double avantage : l'évaluation de la confiance et la protection du service transporté par l'agent mobile. La tâche de la génération de la clé environnementale est basée sur les données collectées durant l'interaction entre l'agent mobile et l'hôte visité.

Dés que l'agent mobile arrive au niveau de l'hôte client, il exécute des actions qui lui permettent l'acquisition des valeurs des paramètres de la confiance (*cf.* Table 4.2). Ces actions consistent principalement en :

1. La collecte des valeurs des paramètres de confiance.
2. Le calcul et l'émission de la clé intermédiaire qui est constituée de valeurs hâchées pour protéger la confidentialité des informations des clients.
3. Le calcul de la clé environnementale.

Afin d'évaluer le degré de confiance relatif à un hôte, le pourvoyeur de service doit aussi exécuter des actions (*cf.* Table 4.3) permettant l'estimation de la crédibilité du client courant et donc, la sélection du service approprié (QoS). A ce titre, le pourvoyeur de service établit une reconnaissance du client sur la base de sa signature, évalue la confiance en utilisant les informations de la base des évidences (*cf.* Section 4.3.5) et décide de la réaction à entreprendre en sélectionnant une QOS. La Figure 4.5 montre clairement le processus de reconnaissance du client qui permet, grâce au diagnostic de la clé intermédiare reçue, l'estimation de la confiance et découle sur la sélection d'une QoS adaptée aux conditions de sécurité décelées.

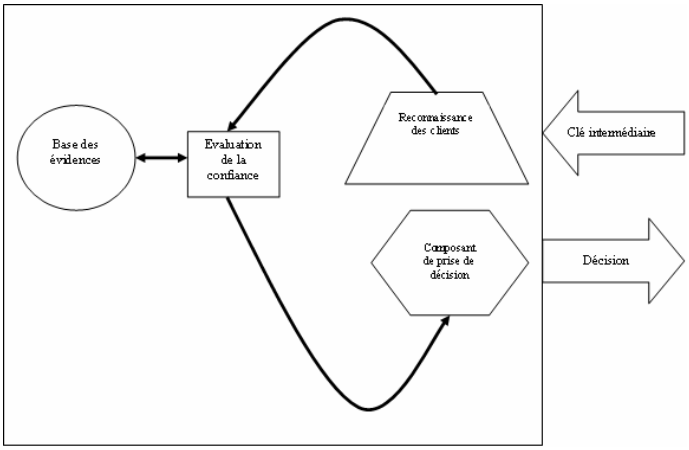

Fig.4.5. Processus de sélection d'une QoS

Nous employons, dans les algorithmes suivants, les lettres majuscules pour indiquer les opérations exécutées ($\mathcal{H}$, $\mathcal{E}$, $\mathcal{D}$, $\mathcal{S}$ et $\mathcal{V}$ pour respectivement le hachage, l'encryptage, le décryptage, la signature numérique et sa vérification) et les indices pour désigner l'entité qui effectue l'opération ("a" pour l'agent mobile et "h" pour l'hôte du pourvoyeur de service).

La clé environnementale K_e est calculée en utilisant les données collectées ainsi que l'identificateur de l'agent mobile. Il est important d'identifier, sans ambiguïté, chaque agent mobile. Nous adoptons la définition suivante de l'identification des agents mobiles:

« Un agent est défini par un identifiant unique non réutilisable engendré lors de sa création et qu'il conservera jusqu'au terme de son cycle de vie. On utilise par exemple la date et le site de création de l'agent ».

Entrée : la clé publique P_o du propriétaire de l'agent mobile, les clés de l'hôte courant (P_h, S_h), les données collectées, EQ et SQ.
Sortie : EM et ES.

1 : Collecter les données (correspondant aux valeurs des paramètres).
Soit $\{d_1, d_2 \ldots d_k\}$ l'ensemble des données collectées.

2 : Appliquer la fonction à sens unique SHS (Secure Hash Standard) à chaque donnée

For i := 1 to k do $M_i = \mathcal{H}_a(d_i)$ End For

3 : Concaténer toutes les empreintes et obtenir la clé intermédiaire :

$M = (M_1.M_2 \ldots M_k)$

4 : Encrypter M en utilisant la clé publique du propriétaire de l'agent mobile P_o:

$EM = \mathcal{E}_a(M)$

5 : Utiliser la clé privée S_h de l'hôte courant pour signer EM : $SM = \mathcal{S}_a(EM)$

6 : Emettre EM et SM vers le pourvoyeur de service

7 : Appliquer le hachage au résultat de la troisième étape, soit $D = \mathcal{H}_a(M)$ le l'empreinte finale

8 : Appliquer $D \oplus id$ (où id est l'identificateur unique de l'agent mobile) afin de générer la clé environnementale K_e. K_e sera utilisée pour décrypter la QoS_i sélectionnée.

9 : Recevoir EQ avec sa signature SQ

10 : Vérifier SQ en utilisant l'opération booléenne $\mathcal{V}(EQ, SQ)$:

10.1. Décrypter SQ en employant la clé publique P_o du pourvoyeur de service et obtenir l'empreinte reçue : $RD = \mathcal{D}_a(SQ)$

10.2. Appliquer le hachage au EQ reçu et obtenir l'empreinte calculée:

$CD = \mathcal{H}_a(EQ)$

10.3. Comparer CD à RD : $\mathcal{V} \leftarrow (RD=CD)$

11 : Essayer de décrypter la QoS reçue en utilisant K_e: $\mathcal{D}_a(EQ) = QoS_i$

12 : Si le décryptage réussit alors Exécuter le service sélectionné

Table 4.2. Algorithme Du comportement de l'agent mobile

Entrée : les clés (P_o, S_o) du propriétaire de l'agent mobile, la clé publique P_h de l'hôte courant, EM et ES
Sortie : EQ et SQ

1 : Recevoir EM et SM
2 : Vérifier SM en utilisant l'opération booléenne $\mathcal{V}$(EM, SM) :
    2.1. Décrypter SM en employant la clé publique P_h de l'hôte courant et obtenir l'empreinte reçue : RD = $\mathcal{D}_h$ (SM)
    2.2. Appliquer le hachage au EM reçu et obtenir l'empreinte calculée : CD = $\mathcal{H}_h$ (EM)
    2.3. Comparer CD à RD : $\mathcal{V} \leftarrow$(RD=CD)
3 : Décrypter EM en utilisant la clé privée du pourvoyeur S_o de service et obtenir : M= $\mathcal{D}_h$ (EM)
4 : Obtenir, à partir de M, k empreintes $\mathcal{H}$ (d_1), $\mathcal{H}$ (d_2)… $\mathcal{H}$ (d_k) (en se basant sur le nombre de paramètres, leur position et la taille de chaque empreinte)
5 : Comparer les empreintes obtenues avec celles présentes dans la base de données (voir Table 4.4)
6 : Estimer la confiance en calculant la valeur de T (voir la Sous section 4.3.3)
7 : Comparer la valeur de T avec les limites des intervalles d'estimation de la confiance et sélectionner l'action à entreprendre (voir Table 4.5)
8 : Sélectionner une QoS_i
9 : Appliquer le hachage au résultat de la quatrième étape, soit D = $\mathcal{H}_h$ ($\mathcal{H}$ (d_1), $\mathcal{H}$ (d_2)… $\mathcal{H}$ (d_k)) l'empreinte finale
10 : Appliquer D $\oplus$ id (où id est l'identificateur unique de l'agent mobile) afin de générer la clé environnementale K_e
11 : Encrypter la QoS_i sélectionnée à l'aide de la clé K_e : EQ = $\mathcal{E}_h$ (QoS_i)
12 : Signer EQ en utilisant la clé privée S_o du pourvoyeur de service : SQ = $\mathcal{S}_h$ (EQ)
13 : Envoyer EQ et sa signature SQ au client

Table 4.3. Algorithme de l'exécution du pourvoyeur de service

De plus amples détails de la description des principales étapes de ces algorithmes sont apportés au niveau de l'annexe A.

4.3.5. Prise en compte de la confiance

La décision de la réaction à entreprendre en vertu des conditions de sécurité détectée se traduit par un certain nombre d'étapes (*cf.* Figure 4.6). Elle débute par une étape préalable qui est transcrite par l'acquisition des valeurs des paramètres et leur corroboration avec celles détenues par le pourvoyeur de service (prétraitement). En effet, ce dernier possède une base des évidences (*cf.* Table 4.4) qui fait correspondre à chaque client ses propres valeurs de paramètres accompagnés de leurs empreintes (résultats de la fonction de hachage). Une comparaison entre les empreintes existantes et reçues est effectuée. Cette opération permet d'attribuer la valeur 1 ou 0 au facteur Sj relatif à chaque paramètre p_J dans le cas respectivement du succès ou de l'échec de la comparaison. Ensuite, la confiance T est évaluée en utilisant l'algorithme mentionné au niveau de la Section 4.3.3.

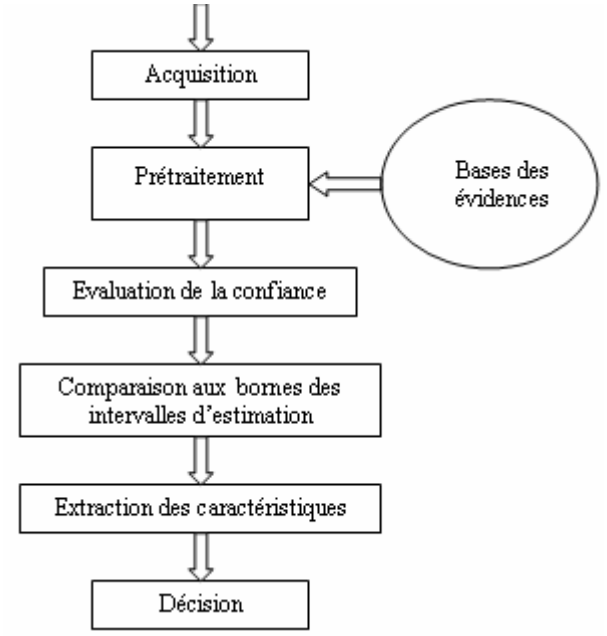

Fig. 4.6. Mécanisme d'évaluation et de prise décision

Paramètres	Valeurs des paramètres	Empreintes (SHA-256)
Identificateur du service	Mobi-Scan	a8186c91696b51664aad9325f74f6dd4a35d1bae 239a4488354f49ab94f95d1d
Acronyme	Esprit 7	09466cd69275da7a4de53216c1e5adcbe7b1 5748c5f96fb5aa73e1eef0f96a8a
Date de validité	December 31th,2007	d68c89bbc02ccb054642d08a9c18092b478df1f2 7a80b2821875fc77f3d9cb77

Table 4.4. Exemple des enregistrements d'un client

Enfin, la sélection de la QoS appropriée repose sur la valeur de T. Sa valeur est comprise entre 0 et 100 en fonction de la contrainte (1) préalablement établie. Les variations de cette valeur sont décrites par une granulation de l'intervalle [0, 100]. La Table 4.5 donne un exemple d'une telle granulation. Ainsi, notre estimation de la confiance est fondée sur des intervalles de valeurs et non sur des seuils. Ceci fournit au pourvoyeur de service une plus grande liberté de réaction. En effet, plutôt que d'avoir des réactions rigides (assurer ou non le service demandé) comme c'est le cas dans les modèles utilisant des seuils, la réaction ici est adaptée aux valeurs appartenant à un intervalle particulier. A titre d'exemple, si la valeur de la confiance appartient au bon intervalle (tel que [71, 100]), le client est crédible. Trois cas peuvent alors apparaître :

1 La valeur de la confiance T tend vers 71 et QoS_6 est sélectionnée.

2 La valeur de la confiance T tend vers 100 et QoS_8 est sélectionnée.

3 La valeur de la confiance T tend vers le milieu de l'intervalle et QoS_7 est sélectionnée.

Intervalles d'estimation de la confiance	Rétroactions
0-30	QoS0 QoS1 QoS2
31-70	QoS3 QoS4 QoS5
71-100	QoS6 QoS7 QoS8

Table 4.5. Exemple des intervalles d'estimation de la confiance avec leurs relatives rétroactions

L'agent mobile peut alors, après réception de la QoS appropriée, exécuter le service sélectionné. D'un autre côté, si la valeur de la confiance obtenue est considérée comme basse (appartenant à l'intervalle [0-30]), il est possible de rechercher, avant de choisir la QoS appropriée, la cause exacte de cet échec. Ceci est réalisé très simplement en parcourant les paramètres qui ont un facteur S égal à zéro.

Les paramètres sont subdivisés en trois catégories (*cf.* Table 4.6) :

- Les paramètres de la plus haute importance (3)
- Les paramètres d'importance moyenne (2)
- Les paramètres de moindre importance (1)

Etant donné l'importance de certains paramètres, ils peuvent influencer fortement le choix de l'action à entreprendre et ainsi la sélection de la QoS appropriée. Les valeurs assignées au poids et à l'importance de chaque paramètre de confiance définissent son impact sur la décision finale (*cf.* Table 4.6). Une discussion sur ce point est présentée au niveau de l'annexe B.

Paramètres	Poids	Valeur de l'importance (3, 2, 1)	Reaction en cas d'échec (relié à l'importance du paramètre)
identificateur	20	3	QoS0
acronyme	10	2	QoS3
e-mail	5	1	QoS6

Table 4.6. Exemple de valeurs des attributs des paramètres

Il est à noter qu'il n'existe qu'un seul cas qui génèrera l'exécution du service complet. C'est celui dans lequel l'attaquant connaît toutes les données du client (celles collectées par le biais du mécanisme d'interaction). En outre, les données collectées par le biais du mécanisme d'inspection sont aussi valides. De plus, nous devons supposer aussi que cet attaquant a intercepté la clé privée du client. Ce cas semble très peu probable.

4.4. Conclusion

Ce chapitre a permis d'expliquer le rôle de l'estimation de la confiance dans un système distribué et de montrer sa répercussion dans l'amélioration de la sécurité des agents mobiles. Il a aussi détaillé le mécanisme d'évaluation de la confiance proposé par notre approche de protection. Les principales contributions de notre travail sont donc:

1) la proposition d’un modèle d’estimation hybride combinant la confiance molle (soft trust) qui est basée sur les mécanismes non-cryptographiques (avec des mécanismes comme l'observation, l'interaction directe et l'inspection), et la confiance dure (hard trust) fondée sur les mécanismes cryptographiques conventionnels;

2) la proposition d’une solution d’estimation de confiance directe (sans tierce partie de confiance) pour prévenir des décisions inappropriées et améliorer le niveau global de la sécurité des agents mobiles.

En plus de sa faculté à établir une réaction justifiée dûe principalement à la collecte d’informations concrètes, cette solution a l’avantage de la simplicité et est donc facilement implantable dans n’importe quelle application distribuée.

Alors que le présent chapitre a détaillé l’emploi du concept de confiance pour l’amélioration de la protection du service transporté par l’agent mobile tout simplement en prévenant son exécution sur un hôte hostile, le prochain chapitre s’intéresse à l’utilisation de l’adaptabilité en tant qu’outil permettant de compliquer l’analyse dynamique du code.

Chapitre 5 – Adaptabilité : un outil pour la protection des agents mobiles

"Quand le rythme du tamtam change, le pas de danse doit s'adapter."
[Proverbe africain]

5.1. Introduction

Dans le domaine des systèmes distribués, le réseau et les environnements d'exécution deviennent de plus en plus hétérogènes, changeants et hostiles. De plus, la complexité croissante des applications modernes, la multiplication des environnements d'exécution et la diversification des attentes des utilisateurs lancent de nouveaux défis à la recherche informatique. L'adaptabilité des applications à des modifications (prévisibles ou non) de leur environnement est l'un de ces défis. En effet, l'adaptabilité se révèle être une réponse prometteuse aux besoins de gestion de la qualité de service (QoS), tel que la performance, la sûreté, la tolérance aux pannes et surtout au problème de sécurité qui nous préoccupe.

L'adaptabilité présente un intérêt évident pour la protection des agents mobiles. Lorsque l'agent mobile migre vers un nouvel hôte, il peut rencontrer de nouvelles conditions de sécurité. Il doit être en mesure de détecter ces nouvelles conditions, de décider de l'action à entreprendre et, par conséquent, de se protéger. En d'autres mots, l'agent est capable de réguler ses aptitudes communicationnelles ou comportementales en fonction de l'environnement dans lequel il évolue [Guessoum 2003], [Guessoum 2004]. L'adaptabilité peut être vue aussi comme un mécanisme permettant à un système (en l'occurrence un agent) de fournir ses services sous des conditions particulières ou nouvelles et lui permet d'assurer les modifications nécessaires d'une manière transparente [Occello 2002].

L'objectif de ce chapitre est d'étudier les besoins et les bénéfices de l'adaptabilité dans la protection de l'agent mobile. Il introduit l'adaptabilité dynamique en s'appuyant sur la réflexivité en tant qu'approche permettant son implantation. Il présente une sélection non exhaustive mais néanmoins, à notre sens, représentative de quelques architectures adaptatives. Enfin, il expose l'architecture adaptative de notre agent mobile et montre son exécution.

5.2. Notion d'adaptation

L'adaptation désigne l'action de s'ajuster et de réagir face aux variations des contraintes de l'environnement et des besoins des utilisateurs [Belaramani 2002]. Il est essentiel de différencier entre les termes adaptabilité et adaptation. L'adaptabilité représente la capacité de changer la caractéristique d'une exécution en réponse à un déclencheur. C'est le but que nous voulons atteindre. En effet, si l'architecture de l'agent mobile repose sur l'adaptabilité basée sécurité, ceci signifie qu'il peut adapter son comportement selon des environnements différents (malicieux ou non). L'adaptation est, quant à elle, l'action entreprise pour atteindre l'adaptabilité par le changement d'une certaine propriété, paramètre ou métrique. Elle

représente le moyen. Dans le cas où un agent mobile modifie la QoS du service qu'il exécute selon les caractéristiques de sécurité, il emploie une adaptabilité comportementale.

Dans la littérature, il n'y a pas de classification standard pour l'adaptation. Les chercheurs utilisent différentes classifications. Nous avons choisi celle qui la catégorise en adaptation structurelle et adaptation comportementale [Guessoum 2003], [Guessoum 2004]:

- L'adaptation structurelle permet d'adapter la structure de l'agent à l'évolution de son environnement. L'architecture d'agent mobile dynamiquement adaptable, à base de micro-composants remplaçables et spécialisables présentée par Leriche et al. [Leriche 2004] est un exemple de ce type d'adaptation.
- L'adaptation comportementale permet d'adapter le processus de décision de l'agent à l'évolution de son environnement. L'adaptation du choix des actions des agents à la situation du marché comme pour le cas des firmes dans [Rejeb 2005] en est un exemple.

Nous nous intéresserons, dans notre cas, à l'adaptation comportementale.

5.2.1. Adaptation comportementale

Pham et al. [Phan 2004] considèrent que l'adaptation comportementale peut être statique ou dynamique. L'adaptation est statique quand on dote les agents de règles de comportements obtenues, à titre d'exemple, par la simulation de l'évolution de l'environnement et des réactions de l'agent à cet environnement. Ces règles sont figées et sont établies en fonction des connaissances détenues de l'environnement de déploiement. Elle correspond aussi, selon Sornn-Friese [Sornn 2001], à une réaction automatique aux pressions et aux changements de l'environnement. Cette réaction peut aussi être basée sur un ensemble statique de règles définies a priori. A titre d'exemple, Jonathan [Dumant 1998] est un système qui s'intéresse à l'implémentation des interfaces dédiées à une "personnalité" donnée et s'appuie sur le choix statique du type de la communication en fonction des besoins de l'application.

L'adaptation dynamique correspond, selon Malenfant [Malenfant 2004] à des formes d'adaptabilité qui se manifestent au cours de l'exécution des programmes. Elle peut correspondre aussi, pour le cas des Systèmes Multi-Agents, à un changement organisationnel incluant d'un côté un comportement organisationnel proactif [22] et de l'autre côté un comportement réactif et protecteur [Rejeb 2005]. Elle représente la capacité de l'agent à s'adapter aux changements imprévus de son environnement par son évolution continuelle sans l'interruption de son exécution.

Il existe deux types de solutions :

1 La première solution consiste à prendre en compte, pendant la conception de l'application, les différentes variations de l'environnement et à définir les règles d'adaptation en conséquence. Toutefois, les adaptations sont effectuées dynamiquement. C'est l'approche utilisée dans le cas des applications réparties "classiques" où les constructeurs décident d'utiliser un protocole donné pour la gestion d'un problème relié aux fluctuations de l'environnement. Le choix est statique alors que le fonctionnement de ce protocole prend en compte des informations dynamiques. Ce type d'adaptation peut se faire par connexion avec des bibliothèques dynamiquement chargeables et en tenant compte des ressources disponibles, ou par modification du code exécutable - ajout, optimisations particulières tenant compte des ressources disponibles ou de l'utilisation particulière qui en est faite [Malenfant 2004]. Elle est réalisée lors du

[22] L'agent a sa propre activité et son propre but et ne fait pas que réagir à l'environnement.

lancement/chargement, de la compilation juste à temps, de la création des objets ou encore lors de l'appel des procédures/fonctions/méthodes. Dans cette catégorie, se trouvent les projets utilisant le principe de réflexivité permettant un choix dynamique entre les implémentations existantes [Ledoux 2000]. Un exemple d'une telle stratégie est donné par MultiSpace [Gribble 1999] dans lequel des adaptations dynamiques sont réalisées par l'administrateur qui charge du code à distance et installe des composants/services sur les noeuds du système.

2 La seconde solution, appelée aussi auto-adaptabilité [Malenfant 2004], correspond à des formes d'adaptabilité ou le programme réalise lui-même sa propre adaptation. Elle suppose une connaissance de soi, de son contexte d'exécution. Elle exige le moyen de se modifier soi-même et la capacité à décider quand et comment s'adapter. Cette solution utilise des règles de comportement qui sont construites dynamiquement au fur et à mesure que l'environnement change [Phan 2004]. Elle aspire donc, non seulement, à effectuer de l'adaptation pendant l'exécution mais aussi à définir et mettre en place la stratégie d'adaptation de façon dynamique. Ce type d'adaptation peut nécessiter des capacités d'apprentissage.

5.2.2. Prise en compte de l'adaptabilité

La conception d'une application exige la prise en compte de ses aspects fonctionnels et non fonctionnels. De ce fait, on distingue, dans le code constituant un programme, le code fonctionnel implémentant la logique de l'application et sa sémantique, du code non fonctionnel qui implémente les divers services utilisés pour supporter le bon fonctionnement de l'application. Il est possible de rendre cette séparation explicite, et jouir de nombreux avantages qu'elle offre telles que la réutilisation. C'est d'ailleurs le cas des approches de séparation des préoccupations (Separation of Concerns) telles que la programmation par aspects, la programmation par sujets ou la réflexivité [Lenglet 2001]. Cette séparation au niveau conceptuel est particulièrement intéressante. Elle permet, en effet, de distinguer deux alternatives différentes sur lesquelles se concentre le processus d'adaptation. La première alternative choisit de porter les efforts d'adaptation sur le code fonctionnel lui-même. Le programmeur d'application fournit alors plusieurs implémentations différentes de certains éléments du code fonctionnel et spécifie sous quelles conditions elles doivent être utilisées. Cette approche permet, au programmeur, de conserver un contrôle total sur le code exécuté, mais au prix d'un travail supplémentaire. A l'inverse, la seconde alternative choisit de se focaliser sur le code non fonctionnel dont le rôle est de modifier, dans une certaine mesure, la façon dont le code fonctionnel sera interprété, mais sans altérer sa sémantique. Le code non fonctionnel est supposé être organisé sous forme de modules. La seconde alternative consiste à modifier les services non fonctionnels et donc l'interprétation du code fonctionnel.

La séparation entre code fonctionnel et non fonctionnel peut aussi être vue comme la séparation de ce qui est spécifique à une application particulière et de ce qui en est indépendant. De ce point de vue, la première alternative ne peut pas être aussi bien généralisée que la seconde. Elle exige de fournir de nouvelles implémentations pour chaque application. Par contre, la seconde alternative permet de développer des services non fonctionnels indépendamment d'une application particulière. En ayant des services non fonctionnels réutilisables, le travail à fournir pour adapter une nouvelle application est beaucoup plus réduit.

5.2.2.1. Fonctionnalités requises pour l'adaptabilité dynamique

Cette section présente brièvement les différentes fonctionnalités nécessaires pour un système adaptatif, à savoir: l'observation, la décision, et l'action [David 2002]. Un système adaptatif doit être capable d'observer l'environnement d'exécution. Cette fonctionnalité est obligatoire si l'on veut que les modifications apportées à l'application soient liées à cet environnement. Plus précisément, le système doit être capable de détecter les variations de l'environnement. Ce sont ces variations qui déclenchent les adaptations. Le système doit, en outre, avoir une bonne connaissance de lui-même pour pouvoir effectuer des transformations ayant un sens tout en conservant sa cohérence.

Ayant une bonne connaissance à la fois de l'environnement extérieur et de lui-même, le système doit aussi être capable de décider quelles modifications de cet environnement ont un impact sur l'application en cours d'exécution et quelles modifications mettre en oeuvre pour y répondre. Ces informations étant en règle générale, dépendantes des applications, elles ne peuvent pas être toujours prévues à l'avance. Certains systèmes se contentent, à ce niveau, de laisser la main au programmeur d'application. Cette fonctionnalité est donc réduite au minimum dans ces systèmes; ils sont, tout au plus, capables de détecter des changements dans l'environnement d'exécution et d'en notifier l'application.

Pour que la prise en charge de cette fonctionnalité soit automatique, le programmeur doit fournir au système, certaines règles définissant une politique d'adaptation. Ces dernières peuvent avoir différentes formes suivant les solutions adoptées, mais leur rôle est d'indiquer quelles sont les variations de l'environnement qui ont un impact sur les applications en cours d'exécution, et quelles mesures il faut prendre pour adapter ces applications aux nouvelles conditions. Le but recherché est de concevoir des politiques d'adaptation simples avec un niveau d'abstraction élevé.

Une fois que le système a détecté une modification significative de l'environnement d'exécution et qu'il a décidé que celle-ci devrait entraîner une réaction de sa part (en se basant sur une certaine politique d'adaptation), reste à déterminer quels types de modifications du système sont possibles. Là encore, suivant le sujet d'adaptation choisi, les types d'actions possibles varient. Dans le cas où le sujet d'adaptation est le code fonctionnel, les modifications possibles vont consister à remplacer certains éléments de ce code fonctionnel par une version alternative, plus adaptée aux nouvelles conditions détectées. Les éléments de code fonctionnel que l'on peut ainsi modifier dépendent du système; cela peut être la classe d'un objet, remplacée par une classe alternative. Dans le cas où, par contre, le sujet de l'adaptation est le code non fonctionnel, les différentes actions possibles correspondent à la reconfiguration des associations entre composants fonctionnels et les services non fonctionnels. Par exemple, il est possible de modifier la sémantique de l'invocation de messages pour certains composants en la rendant asynchrone. Il doit être possible également de spécifier de façon précise quels services doivent (ou ne doivent pas) être associés à tel ou tel composant. Ces services peuvent être configurés, en spécifiant par exemple des paramètres. Ces actions ne doivent pas être globales au système, mais locales aux composants pour lesquels elles ont un sens.

5.2.2.2. Approches de réalisation de l'adaptabilité

Deux approches principales qui traitent le problème de la stratégie dans les travaux autour de l'adaptabilité sont identifiées. Ces dernières sont nommées approche de la stratégie implicite et approche de la stratégie explicite [Marangozova 1999].

Approche implicite

La stratégie implicite est l'approche la plus fréquemment utilisée. Il s'agit du cas où l'application fait usage d'un certain protocole encapsulant une solution à l'adaptation vis à vis d'un problème donné sans avoir le droit de contrôle, ni de remplacement. Il s'agit donc de bénéficier de l'adaptabilité offerte par ce protocole particulier pour gérer son propre problème. Les exemples à citer sont nombreux: utilisation d'un service de communication donné (facilités proposés par CORBA, JAVA-RMI ou autre), d'un type de sécurité donné ou d'un type de gestion de la persistance.

L'adaptabilité proposée dans ces cas a un domaine de validité limité. Elle ne concerne que les variations de l'environnement considéré. Si l'application doit faire face à des changements non prévus dans cet environnement d'exécution ou être déployée dans un environnement hostile où ces protocoles ne sont pas efficaces et ne gèrent pas les problèmes de manière adaptée, elle peut perdre beaucoup au niveau de la performance et même se retrouver en état de blocage.

Approche explicite

Les applications réparties capables de contrôler la définition ou/et le choix des activités d'adaptation exhibent une stratégie explicite. Cette stratégie peut être fournie de manière procédurale ou de manière déclarative ou encore d'une manière réflexive [Marangozova 1999] :

- La manière procédurale consiste à programmer, dans l'application, la gestion des aspects liés à l'adaptation. Dans ce cas, le programmeur de l'application manipule, directement dans le code, les mécanismes fournis par le système. La nature des mécanismes peut être très diverse et en conséquence les moyens fournis au programmeur pour leur manipulation peuvent différer considérablement**,** ce qui complique notoirement la tâche du programmeur.
- La manière déclarative, quant à elle, consiste à fournir une spécification de la qualité de service et de sa gestion sans nécessairement considérer les mécanismes effectifs à mettre en place lors de la réalisation. L'approche vise donc la possibilité de pouvoir exprimer et documenter explicitement les aspects liés à la qualité de service pendant la phase de conception et d'éviter la pratique d'implémentation directe dans le code.

La manière réflexive est une solution intermédiaire qui gagne de plus en plus de popularité. La visibilité qu'elle offre du système peut être placée entre la transparence de l'approche déclarative et la visibilité totale dans l'approche procédurale où tout le code est écrit manuellement. La réflexivité (appelée aussi réflexion) répond au principe d'implémentation ouverte où certains points décisifs du système sont accessibles et adaptables. Elle permet de maintenir automatiquement la liaison entre un niveau abstrait (méta-modèle) et le code. Avec la réflexivité il est possible non seulement de rajouter des services dans le code mais aussi de modifier l'implémentation des services existants.

5.2.3. Réflexivité

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 [Marangozova 1999]. 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.

Malenfant [Malenfant 2004] la définit en tant que : **«** *la capacité pour un programme de manipuler comme données quelque chose représentant l'état de ce programme durant son exécution* ». Autrement dit, c'est la capacité pour un programme de raisonner et d'agir sur lui-même. Maes [Maes 1987] la décrit comme : « *the process of reasoning about and/or acting upon oneself* ».

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 réification.

La démarche de la réflexivité structure un système réflexif en deux niveaux d'abstractions : Un niveau de base et un niveau méta (ou méta-niveau) [Malenfant 2004]**.**

- Le niveau de base désigne l'application standard. C'est un niveau applicatif qui manipule les entités du domaine d'application et décrit la fonctionnalité de l'application. Il englobe le code fonctionnel de l'application (ce que l'application fait).
- Le méta-niveau réfléchit sur le niveau de base et manipule des abstractions de ses entités. Il comprend le code non fonctionnel de l'application (comment elle le fait). Il peut être vu comme un processeur.

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.

Réflexivité structurelle / comportementale

La réflexivité peut être classifiée en réflexivité structurelle ou comportementale. La réflexivité structurelle s'intéresse aux aspects les plus statiques. Tandis que la réflexivité comportementale concerne tout ce qui a trait aux aspects les plus dynamiques, liés à l'état d'exécution et à la façon d'exécuter les programmes. La réflexivité peut aussi être mixte et allier les deux aspects structurel et comportemental [Malenfant 2004].

La réflexivité structurelle permet la réification des aspects structuraux du programme, comme les graphes d'héritage et de composition, ou encore les types de données. Cela oblige à maintenir une description complète du niveau de base d'un programme ; ce qui peut se traduire par un surcoût important. De plus, les processus de réification peuvent être parfois assez compliqués à manipuler. C'est pourquoi cette technique est relativement peu employée dans les systèmes d'exploitation, mais est assez largement répandue dans les langages extensibles (langages dont la sémantique et la syntaxe peuvent être étendues) [Andrade 2003]. A titre d'exemple, dans les langages de programmation à objet les structures de données à manipuler sont des objets. Les objets sont décrits par des classes. Par conséquent, pour manipuler sans contrainte les objets à l'exécution, il faut aussi manipuler, et donc réifier, les classes. La solution consiste à les représenter comme des objets donc accessibles à

l'exécution. Les classes deviennent alors des entités de plein droit[23]. Exemple en Java, représenter les classes en tant qu'objets, à l'exécution, assure de pouvoir utiliser les classes comme n'importe quel objet, dont leur envoyer des messages.

La réflexivité comportementale permet, quant à elle, la réification des aspects de calcul et de comportement d'une application. Un système de ce type propose, par exemple, deux implémentations alternatives d'un seul et même module [Andrade 2003]. Ce type de réflexivité est utilisé principalement dans les systèmes d'exploitation et les systèmes de communication, pour, à titre d'exemple, modifier les politiques de gestion de la mémoire ou de synchronisation, ajouter des instructions de contrôle, remplacer le protocole de communication ou changer de comportement. Un exemple d'élément à réifier est l'évaluateur qui comprend les procédures pour l'évaluation des programmes. Le plus simple serait que les données de l'évaluateur utilisent directement des structures de données du langage et que ces mêmes structures puissent être manipulées par le programme.

Dans un certain nombre de systèmes récents, le type de réflexivité offert est mixte (à la fois structurel et comportemental). Blair [Andrade 2003], propose une architecture réflexive pour les futurs middleware exhibant les deux types de réflexivité.

5.2.4. Quelques travaux utilisant l'adaptabilité dynamique

Il existe de nombreux travaux utilisant l'adaptabilité dynamique. Son application touche plusieurs domaines dont les plus connus sont les domaines de l'hypermédia ou celui du e-learning. La plupart de ces travaux utilisent une architecture basée composants. La liste que nous vous présentons n'est guère exhaustive cependant, elle inclut des travaux basés sur les agents mobiles.

Ledoux et Bouraqadi-Saâdani [Ledoux 2000] présentent leur contribution bâtie avec la mobilité du code. Leur étude porte sur l'adaptation des systèmes d'agents mobiles au cours d'exécution, dans le but de faciliter la construction des systèmes ouverts et garantir une meilleure QoS. Ils utilisent la *réification* des différents aspects du système permettant ainsi la "*séparation des problèmes*". L'idée principale est que *l'introspection* du réseau peut être utilisée pour choisir dynamiquement la meilleure politique de l'exécution. L'approche proposée permet cette faculté d'adaptation de l'infrastructure des systèmes d'agents mobiles en se basant sur le modèle de la réflexivité.

Amano et Watanabe [Amano 2001] Considèrent que l'adaptabilité dynamique constitue un mécanisme de base d'évolution du logiciel. À cet effet, ils ont proposé un modèle adaptable nommée DAS et son langage de description nommée LEAD++. Ils ont insisté sur les besoins de sûreté de l'adaptabilité dynamique, via un modèle amélioré basé-composants nommé *Safe DAS*. La sûreté vérifie que le comportement adaptable du système ne viole pas sa consistance. Le modèle *"Safe DAS*" supporte des exceptions et les mécanismes des assertions.

Par ailleurs, afin d'encourager le développement des applications devant s'adapter aux changements de l'environnement, Chefrour et André [Chefrour 2003a, Chefrour 2003b] proposent un modèle de composants auto-adaptatif qui se repose sur un mécanisme de notification par événement des variations de l'environnement. Ils permettent l'accès à l'adaptation des différents niveaux depuis l'intergiciel. Les services offerts par ce dernier ne

[23] Dans un langage de programmation, se dit d'une entité qui est représentée par une valeur d'un type dont la manipulation admet toutes les opérations normales sur les valeurs des types standards : création à l'exécution, affectation à une variable, passage en paramètre à des fonctions (procédures ou méthodes), retour comme résultat de fonctions, etc.

doivent pas être implantés sous forme de boite noire qui rend l'adaptation transparente. Cela permet aux applications d'intervenir et de faire des choix d'adaptation. Le modèle ACEEL est conçu selon un modèle réflexif. Il comporte deux niveaux séparés : Le niveau de base incluant l'objet « *context*», réalise l'interface, alors que le niveau méta, contenant l'objet « *adapter* » prend en charge l'auto-adaptation en utilisant une politique d'adaptation sous forme de script s'appuyant sur le système de détection-notification.

Zhi et Zhogwen [Zhi 2004] Présentent un mécanisme d'adaptation dynamique de sécurité en utilisant le modèle de conception orienté objets. Utilisant ce mécanisme, différentes implémentations de sécurités peuvent être remplacées à tour de rôle de façon dynamique. Le but étant l'adaptation des procédures de sécurité telles que les différentes fonctions cryptographiques symétriques (DES, AES…) ou asymétriques (RSA, DSA, …). L'approche proposée met en évidence un cadre de travail dans lequel les agents mobiles peuvent choisir dynamiquement une implémentation de sécurité parmi plusieurs. La solution de l'adaptation dynamique de sécurité, mise en évidence, fait que l'agent mobile change l'implémentation de sécurité selon le changement de l'état de l'environnement dans lequel il s'exécute. L'adaptation dynamique de la sécurité, selon Zhi et Zhogwen, augmente l'efficacité de l'agent mobile.

Amara-Hachmi et El Fallah-Seghrouchni [Amara 2005] considèrent que le paradigme orienté composants est mieux adapté pour la conception des systèmes ouverts, complexes et évolutionnaires. Il permet d'augmenter la productivité et diminuer le coût de développement des logiciels. Elles ont proposé à cet effet un modèle d'architecture basé composants pour un agent mobile de manière à accroître la modularité, la réutilisation, l'extensibilité, et l'auto-adaptabilité. Lorsque l'agent mobile se déplace d'un hôte à un autre, le contexte change. Le résultat de l'adaptation dynamique est la sélection des composants adéquats et la mise en relation des composants sélectionnés pour le nouveau contexte.

Le travail de Rejeb et Guessoum [Rejeb 2005] a permis de concevoir et d'implémenter un système multi-agents complet intégrant les firmes, les formes organisationnelles et leurs interactions. Ce système est principalement basé sur la modélisation de l'adaptation des firmes en les dotant d'une capacité d'apprentissage pour la résolution des problèmes inhérents tels que le dilemme d'exploration/exploitation.

Dans le cadre de leurs travaux sur l'aide au développement d'applications réparties ouvertes, Plus récemment, Leriche et Arcangeli [Leriche 2006] ont proposé un modèle d'agent mobile auto-adaptable ainsi que son implémentation au sein d'un middleware Java. Ils ont étudié une évolution de ce modèle afin de lui donner davantage de flexibilité. Ils ont discuté le besoin de flexibilité et la possibilité de définir différents modèles d'agents par le biais d'un méta-niveau configurable. Puis, ils ont présenté les éléments d'une architecture à base de micro-composants permettant l'adaptation dynamique.

Tous ces travaux ont confirmé le rendement et l'efficacité que présente la réflexivité pour la conception et l'implantation de l'adaptabilité.

5.2.5. Synthèse

La première partie de ce chapitre a permis d'abord d'expliquer le concept de l'adaptabilité puis de motiver l'utilisation de la réflexivité en tant que support pour la réalisation de l'adaptabilité, du fait qu'elle fournit une bonne représentation de soi tout en rendant possible la manipulation et par conséquent la modification de cette représentation.

Nous désirons que le comportement de l'agent mobile puisse changer durant son exécution pour adapter le service aux conditions de sécurité latentes au sein de l'environnement visité. Il

devrait, pour se protéger, pouvoir détecter les changements de sécurité de l'environnement et y répondre convenablement. C'est l'objectif vers lequel nous aspirons.

Dans le cadre de cette thèse, nous proposons donc de protéger l'agent mobile contre son analyse statique/dynamique, par une politique de protection basée sur l'adaptabilité dynamique. L'agent mobile sera doté d'une capacité d'adaptation de son comportement, conçue statiquement mais exécutée dynamiquement. Il s'agit d'estimer les différentes variations possibles de l'environnement d'exécution, que peut rencontrer l'agent mobile durant son cycle de vie, et de définir un ensemble de règles lui permettant de décider et d'entreprendre les actions appropriées. Le concept de réflexivité est un moyen qui permet de contrôler son comportement et d'adapter son exécution selon les conditions de sécurité courantes. Ainsi, cette adaptation est réalisée à l'exécution et elle est supportée par une démarche réflexive.

La Section suivante s'occupe de la présentation de l'architecture adaptative de l'agent mobile sur laquelle s'appuie l'approche de protection évoquée.

5.3. Agents mobiles adaptatifs

Dans notre contexte, on entend par adaptabilité la capacité de réagir face aux variations des contraintes (de l'environnement d'accueil) ou à des besoins de l'application en sécurité [Hacini 2007a], [Hacini 2007b]. Notre agent mobile doit être sensible à l'environnement d'accueil puisqu'il doit détecter ses caractéristiques. Ces dernières sont relatives au niveau de sécurité de cet environnement.

La stratégie d'adaptation définit les paramètres importants influençant la décision de l'exécution du service demandé et engendre un comportement spécifique de l'agent mobile particulièrement lorsque ces paramètres ne vérifient pas certaines propriétés relatives aux conditions de sécurité requises et que le niveau de sécurité fourni est jugé non satisfaisant.

L'adaptabilité repose sur une estimation de confiance effectuée par l'agent mobile durant son exécution préalable, en interagissant avec l'hôte visité. A partir de cette estimation, le pourvoyeur de service va fournir à l'agent mobile une QoS qu'il utilise pour choisir la règle d'adaptation adéquate qui se traduira par un comportement approprié. Dans l'approche proposée, l'adaptation est statiquement conçue et dynamiquement exécutée. Cette solution consiste à estimer, durant la conception de l'application, les différentes variations de l'environnement et à définir les règles d'adaptation correspondantes. Parmi les diverses techniques permettant la mise en œuvre de l'adaptation, la réflexivité est apparue comme une solution prometteuse. Elle constitue un support pour le développement d'applications et fournit des mécanismes permettant d'exprimer les traitements en termes extrêmement génériques. Les propriétés intrinsèques de la réflexivité (l'introspection et l'intercession) offrent à l'agent mobile la possibilité de raisonner et d'agir sur lui-même. Notre agent mobile observe son comportement interne pour modifier la tâche relative à la QoS reçue et ajouter ou supprimer des fonctions fictives. Par conséquent, son adaptation consiste à modifier son comportement interne en remplaçant le code des fonctions fictives ou en y ajoutant/supprimant d'autres. En effet, plusieurs implémentations différentes des tâches du service constituant le code fonctionnel sont fournies grâce à l'emploi de différents algorithmes d'obscurcissement. Le code non fonctionnel spécifie sous quelles conditions elles doivent être utilisées. L'architecture adaptative proposée met en évidence deux niveaux conceptuels (*cf.* Figure 5.1):

Niveau de base. Il contient le code fonctionnel et formule l'implémentation logique de l'application ainsi que sa sémantique. Il comprend cinq composants : *une interface*, *une*

mémoire, une bibliothèque, un générateur de clé et *un adaptateur*. L'interface permet la communication de l'agent mobile avec l'hôte visité ainsi que l'acquisition de données. Ces dernières sont utilisées pour la sélection du comportement de l'agent mobile (relatif à la QoS choisie). Les données collectées et la trace de l'exécution sont rangées dans la mémoire et les différents comportements sont stockés en bibliothèque.

Un méta-niveau. Il contient le code non fonctionnel et décrit le processus d'adaptation. Ce dernier est assuré par le composant *manager* qui se charge de la sélection de la règle d'adaptabilité qui va spécifier le comportement de l'agent mobile.

5.3.1. Architecture adaptative de l'agent mobile

L'architecture adaptative de l'agent mobile proposée (*cf.* Figure 5.1.) comporte les composants suivants :

5.3.1.1. Interface

Elle contrôle la communication entre l'agent mobile et son environnement courant ainsi qu'avec le pourvoyeur de service. Elle correspond au comportement préliminaire (avant l'exécution du service) de l'agent mobile et s'occupe essentiellement de:

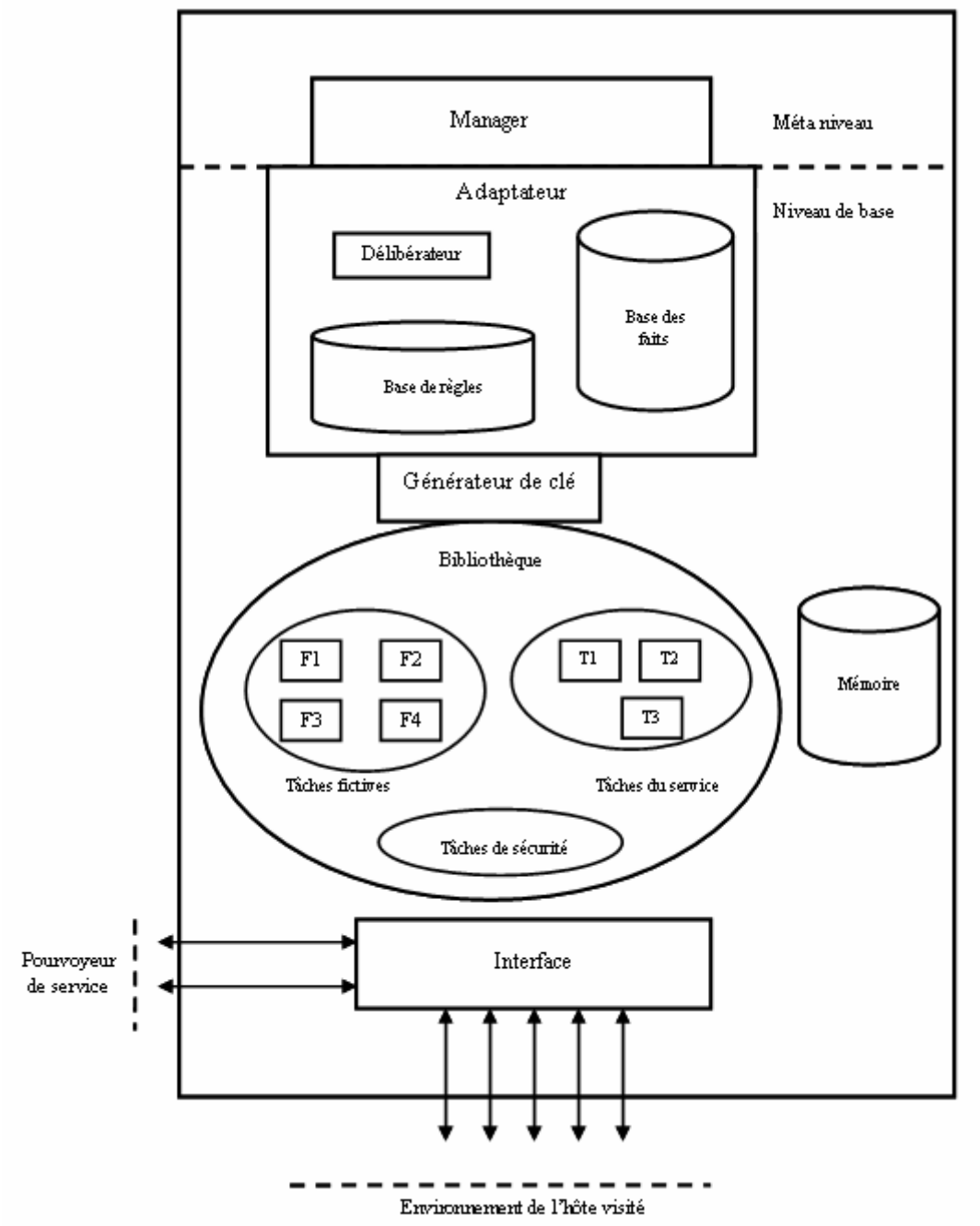

Fig.5.1. Architecture adaptative de l'agent mobile

- La perception de l'environnement et de l'acquisition des données relatives à l'hôte visité. Ces données sont utiles à l'estimation de la crédibilité de l'hôte. Trois types d'acquisition peuvent être considérés: l'observation, l'inspection et l'interaction. Le type d'acquisition dépend de la donnée exigée. A 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 l'hôte 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. Afin d'améliorer le niveau de sécurité, les informations collectées durant l'interaction ne sont pas toutes utilisées pour la construction de la clé environnementale.

- L’exécution de la séquence d’actions relative au comportement sélectionné. Ces actions peuvent être une simple alarme, une tâche (service ou fictive), un message de transmission/réception, un arrêt d’exécution ou/et une migration.

Ce composant est donc responsable de la transmission et de la réception des messages échangés par l’agent mobile et le pourvoyeur de service tels que l’émission de la clé intermédiaire ou la réception de la QoS et de la portion du service sélectionné. Par ailleurs, la tâche d’émigration de l’agent vers les autres hôtes de l’itinéraire lui incombe.

5.3.1.2. Bibliothèque

Elle comprend les tâches du code de l’agent mobile. La composition de ces différentes tâches génère les différents comportements de l’agent mobile. L’objectif est d’offrir plus de confidentialité et plus de flexibilité du comportement. Afin de varier les comportements de l’agent mobile, des alternatives de ces tâches sont proposées. Ces alternatives sont principalement fournies par les algorithmes d’obscurcissement. En outre, quelques tâches fictives sont utilisées pour compliquer l’analyse du comportement de l’agent mobile et améliorer le niveau de sécurité. Des combinaisons variées de ces tâches sont fournies par un ensemble d’expressions abstraites employées pour la génération de séquences d’exécution correspondant aux comportements de l’agent mobile. La bibliothèque contient également toutes les tâches de sécurité utilisées par le protocole de protection.

Soit A= {A1, A2…Aq} un ensemble de tâches. L’ensemble A peut être subdivisé en deux sous-ensembles : F et S. F= {A1, A2…Ap} est l’ensemble des tâches fonctionnelles qui correspondent à l’application et S= {Ap+1…Aq} représente l’ensemble des tâches de sécurité (ex: encryptage/décryptage, hachage, signature digitale, etc.).

La bibliothèque est en fait un composant statique dans l’agent mobile. Elle est représentée par un fichier compacté qui accompagne l’agent mobile. Elle est constituée d’une table de triplets (Table 5.1). Chaque triplet correspond à une tâche et comporte son nom, le code machine (exécutable) de la tâche obscurcie et cryptée à l’aide d’un algorithme cryptographique symétrique) ainsi que la signature numérique de la tâche pour garantir son authenticité et son intégrité. Les noms des tâches doivent être codifiés selon un système de codification qui permet d’attribuer à chaque tâche un nom non significatif. La bibliothèque comporte principalement des tâches fictives (de petite taille), une portion du code égale au 1/3 de chaque service (tâche réelle) susceptible d’être fourni au cours de l’exécution de l’agent mobile.

Nom codifié de la tâche	**Code exécutable de la tâche**	**La signature de la tâche**
Fonc_01	‰÷ˮ,خمهusفا)ج#R÷'³** ڑ‘طL…	OQs96oD2jGdfgtGP29QD8gV2tsM
Fonc_02	سL¶ظڑدg°€^û"†}¾DqQ.±mo…	hjLA9WHuiZ4p+lLjnAo5WS2iL+g
…	…	…

Table.5.1. Structure de la bibliothèque

5.3.1.3. Mémoire

Elle comporte les données hachées (collectées), la trace de l'exécution qui se limite à la séquence d'exécution sélectionnée ainsi qu'à éventuellement l'itinéraire de l'agent mobile. L'ensemble est signé à l'aide de la clé privée de l'hôte et est utilisé par le pourvoyeur de service pour vérifier la validité des données collectées et pour confirmer l'exécution du comportement choisi. Elle est constituée en fait d'un vecteur hétérogène et dynamique. Ce vecteur peut contenir des objets de types différents (Table 5.2).

L'identificateur de l'objet	Le contenu d'objet
Données_hachées	gGRCBNemqViZeMhng9/6yzCz8dw=
Trace d'exécution	…..
Itinéraire	…..

Table.5.2. La structure de la mémoire.

La mémoire reçoit le nom de l'objet demandé à partir du manager, et lui renvoie l'objet en question (*cf.* Figure 5.2).

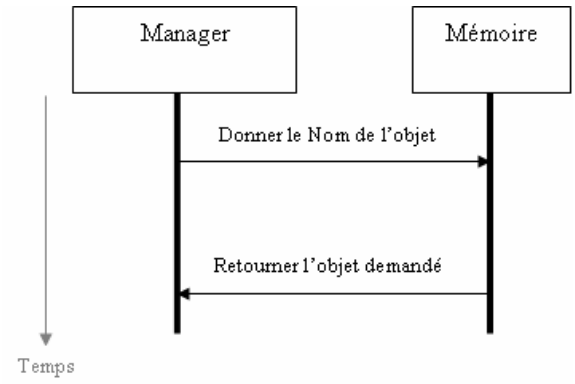

Figure 5.2. Fonctionnement de la mémoire.

5.3.1.4. Générateur de clé

C'est la partie de l'agent mobile qui garantit la sécurité des données échangées entre le client et le pourvoyeur de service, cette sécurité est assurée à l'aide des fonctions cryptographiques : encryptage, hachage, signature numérique et aussi par l'utilisation de certificats numériques. Ce composant a pour rôle, la génération de la clé environnementale. Il permet la sauvegarde des données hachées en mémoire et leur émission au pourvoyeur de service par le biais de l'interface. Il est également utilisé pour décrypter la QoS reçue. L'algorithme de chiffrement symétrique *Blowfish*[24] [ht8] a été utilisé pour l'encryptage/décryptage des tâches sensibles

[24] Il utilise une longueur de clé variant entre **32 bits** et **448 bits**. Depuis sa conception, il a été grandement analysé et il est aujourd'hui considéré comme étant un algorithme de chiffrement robuste. Il n'est pas breveté et ainsi son utilisation est libre et gratuite.

ainsi que pour l'encryptage/décryptage de l'ensemble {QoS, portion de service}. L'algorithme de hachage utilisé est *SHA-256*. Afin d'échapper à l'attaque par force brute et éviter les clés de tailles 384 ou 512 plus coûteuses en temps CPU et en débit réseau (lors du transfert), la taille de la clé choisie est de 256. Les spécialistes en cryptographie recommandent d'utiliser des fonctions de hachage plus récentes telles que le *SHA-256*. La signature numérique est utilisée pour authentifier la source d'une information et garantir son intégrité. Elle possède également la propriété de non-répudiation. Autrement dit, elle permet d'assurer que l'expéditeur a bien envoyé le message. Nous optons pour l'algorithme *SHA1withRSA*. Ces deux algorithmes sont considérés par la communauté scientifique comme sécuritaires [Rhee 2003]. La Figure 5.3 illustre le processus de la signature numérique. Le certificat numérique, quant à lui, est une identité électronique émise par une tierce partie de confiance pour une une entité réseau. Il permet d'associer une clé publique à une entité (une personne, une machine, ...) afin d'en assurer la validité. Le certificat est en quelque sorte la carte d'identité de la clé publique. Il est délivré par un organisme appelé *autorité de certification* (souvent notée **CA** pour *Certification Authority*). Il est divisé en deux parties : Une partie contenant les informations (détenteur du certificat, validité, algorithme de signature, etc.), l'autre contenant la signature de l'autorité de certification. Chaque certificat est signé avec la clé privée de signature d'une autorité de certification. Il garantit l'identité d'un individu, d'une entreprise ou d'une organisation. En particulier, il contient la clé publique de l'entité et des informations associées à cette entité. Il existe plusieurs formats de certificat. Le format X509 choisi dans TAMAP est le format le plus largement utilisé par la plupart des protocoles [ht6].

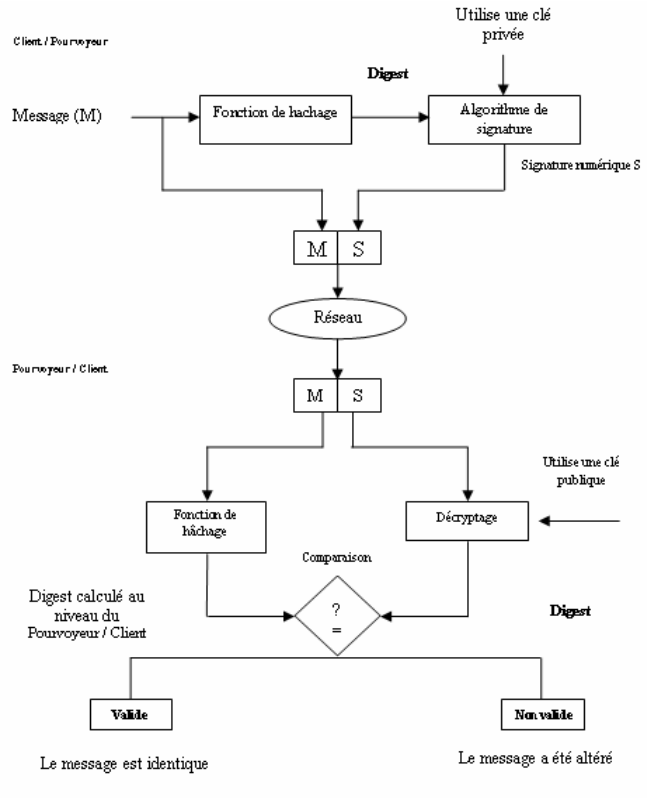

Fig.5.3. Signature numérique

5.3.1.5. Adaptateur

Il s'occupe de l'application de la politique d'adaptation. Il comprend *une base de règles, une base de faits* qui se limite au seul fait représenté par la QoS reçue et un moteur de l'adaptation appelé *délibérateur*.

- *La base de règles:* elle comprend les règles d'exécution. Ces règles sont décrites comme suit: Si <condition> Alors <Action>. La partie gauche de la règle comprend la QoS sélectionnée et sa partie droite spécifie la tâche correspondante telle qu'une tâche (i,j) désigne le service i ayant le comportement j. La base de règles inclut les règles comportementales (Table 5.3) affectées d'un poids ainsi que les règles relatives aux exceptions telles que:

  Si <le temps d'exécution a expiré> Alors < notifier et quitter>

  Le poids de la règle désigne son nombre d'activations à un instant donné du cheminement de l'agent mobile.

Poids	Règle
0	R0: si QoS0 alors exec (task0.1)
0	R1: si QoS0 alors exec (task0.2)
0	R2: si QoS1 alors exec (task1.1)
0	R3: si QoS1 alors exec (task1.2)
0	R4: si QoS1 alors exec (task1.3)
0	R5: si QoS1 alors exec (task1.4)
0	R6: si QoS2 alors exec (task2.1)
0	R7: si QoS2 alors exec (task2.2)
0	R8: si QoS2 alors exec (task2.3)
0	R9: si QoS3 alors exec (task3.1)
…	…
0	…

Table 5.3. Exemple de règles comportementales en phase initiale

- *Le délibérateur*: c'est le moteur de l'adaptation. Il utilise la QoS reçue ainsi que la base de règles pour choisir l'ensemble des règles qui peuvent être déclenchées. Il sélectionne les règles les plus appropriées parmi celles qui ont été choisies. Cette sélection détermine le comportement de l'agent mobile. Il utilise le poids des règles. Le poids de chaque règle est initialisé à zéro puis incrémenté à chaque fois que la règle correspondante est déclenchée tout le long du chemin emprunté par l'agent mobile. Un générateur de nombres aléatoires sélectionne les règles appropriées parmi celles qui ont le poids le plus bas. L'historique du comportement de l'agent mobile est lié aux poids des règles. L'activité du Délibérateur s'effectue en trois phases successives:
  - **Sélection** : il sélectionne toutes les règles déclenchables par la QoS reçue.
  - **Filtrage :** il détermine les règles ayant le poids minimal P_m parmi les règles sélectionnées, car le filtrage est basé sur P_m. Toutes les règles ayant un poids équivalent à P_m seront regroupées dans une liste $\mathbf{L_{filtrée}}$ pour la prochaine phase**.**
  - **Résolution de conflit :** dans cette phase, le Délibérateur choisit une règle à partir de $L_{filtrée}$ de manière aléatoire, il incrémente son poids puis il retourne la tâche correspondante au manager.

5.3.1.6. Manager

Le manager lance l'adaptateur qui sélectionne la tâche réelle du service et lui fournit la tâche effective. Le manager lui associe alors des tâches fictives et génère une séquence d'exécution. Le rôle du manager est de veiller à ce qu'il n'y ait pas deux séquences d'exécution successives analogues. Il détermine les actions à exécuter en sélectionnant les tâches stockées en bibliothèque. En effet, pour compliquer l'analyse du comportement de l'agent mobile, chaque tâche sélectionnée est associée à diverses expressions abstraites définies comme des combinaisons de tâches utiles et de tâches fictives. Une expression abstraite sélectionnée évoluera en une séquence d'exécution. A titre d'exemple, si la règle sélectionnée est R8 et (F0, F1, F2) sont les tâches fictives, la séquence d'exécution pourrait être (F0, task2.3, F1), (F1, task2.3, F3), (F2, task2.3, F0, F1), (F3, task2.3, F2) ou (F1, task2.3, F0, F2). En outre, le manager permet l'utilisation des tâches de sécurité et il est le seul à pouvoir utiliser les composants sensibles. Pour cela, le manager comporte deux sous composants: un chargeur et un séquenceur.

- **Le chargeur** : il est responsable du chargement des composants de haute sensibilité de l'agent mobile. Le diagramme de la Figure 5.3 explique ce processus.
- **Le séquenceur** : il s'occupe de la génération de la séquence d'exécution; C'est une combinaison de tâches, dont une seule est la tâche effectivement sélectionnée par l'adaptateur, et les autres sont fictives. L'utilisation des tâches fictives a pour but de compliquer le comportement de l'agent afin d'améliorer sa protection, et de contrecarrer l'analyse dynamique. Cependant, pour que ce mécanisme soit fiable, certaines contraintes doivent être respectées :
  - La séquence d'exécution doit comporter, en plus de la tâche effective (le service), au moins trois fonctions fictives.
  - Deux séquences générées successivement ne doivent pas avoir le même ordre de tâches ni la même tâche effective (grâce à l'utilisation de l'obscurcissement).
  - Soit E_i, E_{i+1} deux séquences d'exécution successivement générées, et soit T et T' deux ensembles de tâches fictives telles que :

$$T \in E_i, T' \in E_{i+1} \Rightarrow T \cap T' \neq T \cup T'$$

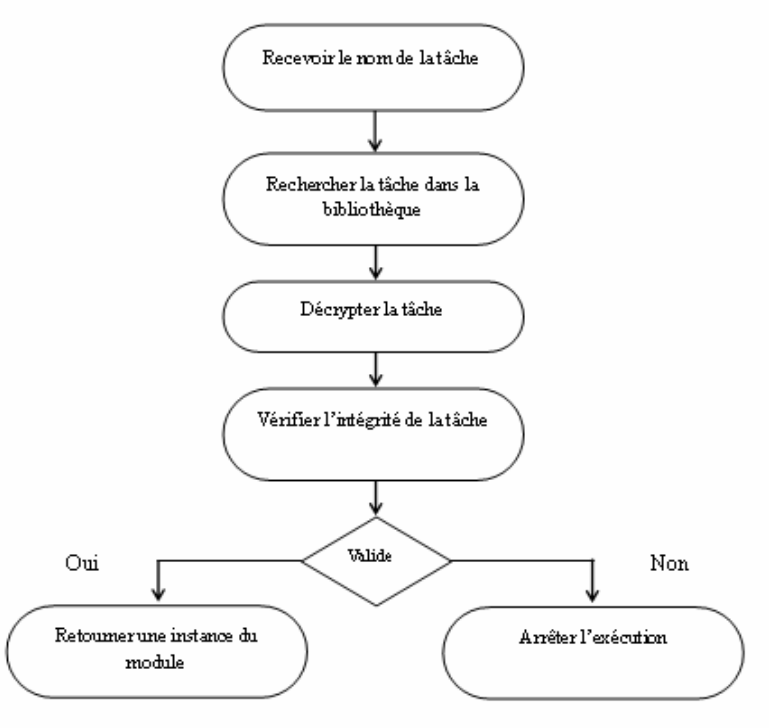

Fig.5.3. Chargement d'une tâche

5.3.2. Aperçu de l'exécution de l'agent mobile

Dés que l'agent mobile arrive au niveau de l'hôte, il entame l'opération de génération de la clé environnementale (*cf.* Figure 5.4). Durant cette étape, l'agent mobile collecte les données en utilisant le composant capteur. Le générateur de clé applique les opérations de hachage, d'encryptage et de signature digitale aux données collectées en employant les tâches correspondantes présentes au niveau de la bibliothèque. Il émet le résultat au pourvoyeur de service et sauvegarde une copie au niveau de la mémoire de l'agent mobile. Puis, il calcule la clé environnementale. Ensuite, l'agent mobile attend l'arrivée de la QoS sélectionnée et de la portion de code manquante à partir du pourvoyeur de service. Dés que le message transportant la QoS choisie arrive, l'agent mobile l'identifie (vérifie la signature digitale) et utilise la clé environnementale pour le décrypter. Puis, le délibérateur emploie la QoS reçue et la base de règles pour sélectionner l'ensemble des règles à déclencher. En se basant sur un générateur de nombres aléatoires, le délibérateur choisit les règles appropriées parmi celles ayant le poids le plus bas. Enfin, le manager génère une séquence d'exécution et charge les tâches choisies à partir de la bibliothèque. L'interface s'occupe alors de l'exécution de la séquence d'actions relatives au comportement sélectionné. Le rapport de l'exécution renfermant la suite des tâches réellement exécutées est sauvegardé au niveau de la mémoire de l'agent mobile. Il servira comme trace d'exécution.

Dans le but d'augmenter la flexibilité du système, un ensemble d'exceptions est défini. Si un problème donné surgit durant l'une des étapes d'exécution, le manager remarque la non progression pendant une durée déterminée. Il demande alors au délibérateur de vérifier l'ensemble des exceptions dans le but d'une possible restauration. Si aucune des exceptions ne correspond au problème mentionné, il arrête immédiatement l'exécution et quitte l'hôte courant vers le prochain hôte ou bien l'hôte d'origine.

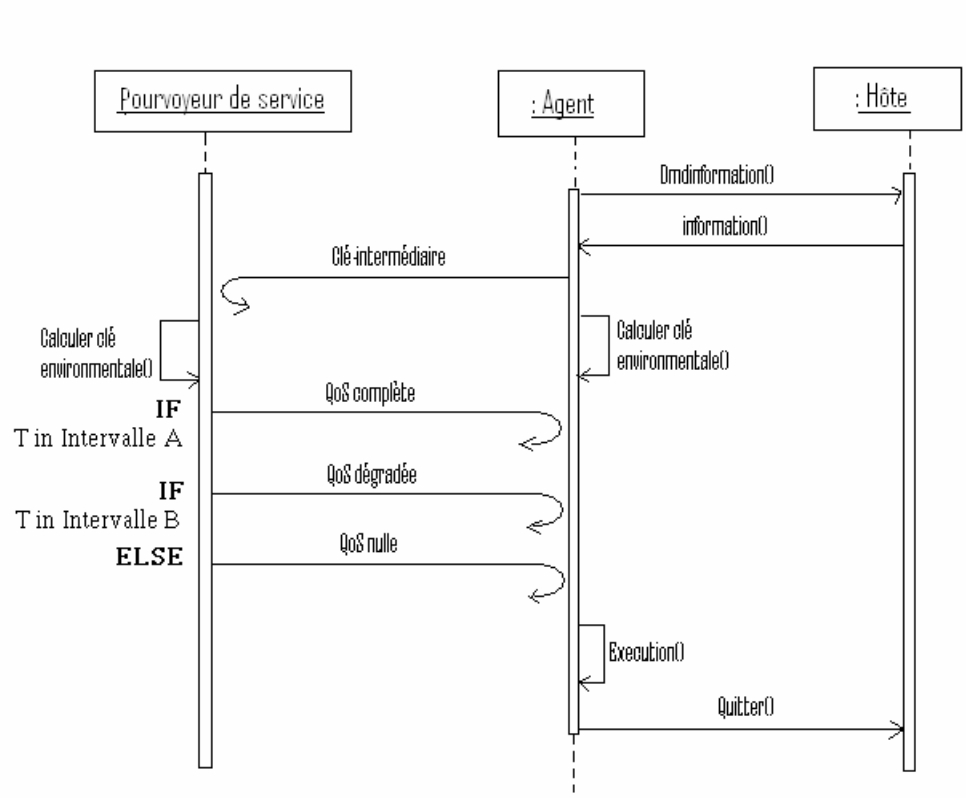

Fig.5.4. Diagramme de séquence.

5.4. Conclusion

L'adaptation proposée, dans TAMAP, est supportée par une architecture réflexive offrant à l'agent mobile la capacité de s'auto-protéger contre l'analyse dynamique de son code en détectant les variations de son environnement d'exécution et d'y répondre en sélectionnant la QoS du service et en adoptant le chemin d'exécution le moins récemment emprunté.

Le choix de la QoS est basé sur la vérification d'un certain nombre de critères, définissant la sécurité d'une application donnée, appelés paramètres de confiance. Ces derniers sont utilisés dans le calcul du degré de confiance. Ils dépendent du service offert et peuvent donc changer d'une application à une autre. Tandis que la sélection du chemin d'exécution (comportement de l'agent mobile) est régie par un ensemble de mesures:

- A chaque QoS correspondent plusieurs exécutions différentes. Cette diversité est offerte par l'existence, pour chaque tâche, au niveau de la bibliothèque de l'agent mobile de différentes versions obscurcies.
- A chaque exécution correspond une règle d'adaptation
- Chaque règle est pourvue d'un poids dénotant le nombre de déclenchements
- Le choix de la tâche à exécuter est fourni par un générateur de nombres aléatoires ayant comme entrée l'ensemble de règles possédant le poids minimal et produisant comme sortie le numéro de la règle la plus appropriée. Celle-ci correspond au comportement de l'agent mobile (tâche effective).
- La tâche effective sélectionnée est utilisée en combinaison avec des tâches fictives pour constituer une séquence d'exécution. Pour éviter deux exécutions successives

identiques, cette dernière doit être choisie en fonction de l’historique de l’agent mobile matérialisé par sa trace d’exécution.

Cette solution offre à l'agent mobile la faculté de s'adapter et de diversifier ses comportements au cours de son cycle de vie en tenant compte des variations sécuritaires de l'environnement d’exécution. Il se caractérise ainsi par un comportement changeant et imprévisible et donc inintelligible. Ce qui rend toute tentative d'analyse difficile.

Par ailleurs, l'utilisation des techniques de sécurisation classiques telles que la cryptographie, la signature numérique ou le hachage, permet de préserver la confidentialité et l'intégrité de l'agent mobile et de renforcer le niveau de sécurité proposé. Ainsi, notre approche répond indirectement à d'autres types d'attaques telles que l’altération des données.

L’adaptation, telle qu’elle a été réalisée dans TAMAP, permet au concepteur de l’agent mobile, de conserver un contrôle total sur le code exécuté, mais au prix d'un travail supplémentaire. Nous notons aussi qu’elle est simpliste et que la réflexivité a été appliquée de façon très limitée. Cependant, elle ouvre la voie à des utilisations plus conséquentes du concept de l’adaptation dynamique pour la protection des agents mobiles.

Chapitre6 – Expérimentation et étude des performances

"Votre force ne saurait être supérieure au maillon le plus faible de votre défense."
[Proverbe chinois]

6.1. Introduction

Après avoir détaillé les concepts de notre approche protégeant le code de l'agent mobile contre l'analyse statique/dynamique des hôtes malveillants et avoir discuté de l'architecture adaptative de l'agent mobile, nous décrivons, au sein de ce chapitre, son implémentation qui a été réalisée à l'aide de la plate-forme agent Jade sous l'environnement Eclipse. En outre, ce chapitre étudie les performances de l'approche de protection TAMAP. Cette étude est subdivisée en deux points : les performances du protocole de protection vis-à-vis des différentes attaques et celui relatif à l'efficacité du processus d'adaptation à masquer le comportement de l'agent mobile.

6.2. Outils d'implémentation

6.2.1. Plate-forme d'Agent

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

- La migration de l'agent doit être souple.
- La plateforme doit offrir une large gamme d'utilitaires.
- Le langage de programmation sous jacent doit fournir des fonctionnalités de sécurité assez suffisantes telles que les fonctions cryptographiques 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*. 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**[25], 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 seule application Java, et donc, 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'agent et permet à plusieurs agents de s'exécuter en parallèle sur le même serveur.

La plate-forme JADE inclut tous les composants qui permettent de contrôler un **SMA**[26]. Ces composants sont :

[25] Foundation for Intelligent Physical Agents
[26] Système Multi-Agents

- Système de Gestion d'Agent (**AMS – Agent Management System**) : Agent central qui supervise les autres agents ainsi que les accès à la plate-forme (Pages Blanches).
- Canal de Communication entre Agents (**ACC – Agent Communication Channel**) : Agent qui ouvre la voie aux interactions entre les agents en dedans et en dehors de la plate-forme.
- Facilitateur d'Annuaire (**DF – Directory Facilitator**) : Agent qui fournit un service de Pages Jaunes (recensement des services disponibles).

Les agents sont implémentés sous forme d'un *Thread* (l'**Agent Scheduler**) qui exécute à la suite et en boucle les comportements liés à l'agent (jusqu'à la fin).

La création d'un agent sous JADE passe par les 4 étapes suivantes:

- faire hériter la classe de notre agent de la classe *jade.core.Agent*.
- définir les méthodes *setup ()* et *takeDown ()*.
- effectuer l'initialisation de l'agent dans la méthode *setup ()* :
  - enregistrer l'agent dans le DF à partir de l'initialisation.
  - initialiser et lier les behaviours ("comportements") principaux de l'agent dans la méthode *setup ()*.
- créer les behaviours spécifiques de l'agent.

Le package *Jade.core* implante le noyau du système. Il possède la classe *agent* qui doit être étendue par les applications des programmeurs. Une classe *behaviour (comportement)* est contenue dans *jade.core.behaviours*, une sous classe de *jade.core*. Les comportements implémentent les tâches ou intentions des agents.

En utilisant JADE, les créateurs des applications peuvent construire des agents mobiles qui peuvent migrer ou se clôner à travers le réseau. Les agents mobiles doivent vérifier la disponibilité des destinations afin de décider quand et où se déplacer ; pour cela, JADE fournit une ontologie, appelée *jade-mobilité-ontology*, définissant les concepts et les actions nécessaires. Cette ontologie est contenue dans la classe *jade.domain.MobilityOntology*. Les deux méthodes *doMove ()* et *doClone ()* de la classe *Agent* permettent à un agent de JADE de migrer ailleurs ou de générer une copie à distance de lui-même sous un nom différent :

- La méthode *doMove ()* prend *jade.core.Location* en tant que paramètre, qui représente la destination prévue pour l'agent.
- La méthode *doClone()* prend également *jade.core.Location* comme paramètre, mais ajoute un *String* contenant le nom du nouvel agent qui sera créé comme copie de la courante.

6.2.2. Langage de programmation

Nous avons choisi JAVA pour implémenter l'agent mobile. C'est un langage multi plate-forme 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 un langage orienté objet qui apporte un support efficace et élégant en définissant de manière très stricte la façon dont les objets communiquent entre eux. Le principal avantage est que chaque objet puisse être mis au point séparément.

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 sites 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. Ceci est réalisé grâce au mécanisme d'interprétation [Knudsen 2004]. JAVA permet la gestion dynamique de tableaux, c'est-à-dire le changement de leur taille au cours de l'exécution du programme. Il comporte aussi le Garbage Collector (ramasseur de miettes) ; un programme qui se charge de restituer la mémoire lorsque la vie d'un objet est terminée. 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 [Gong 1998a], [Wang 2000]. 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 dynamique des tâches contenues dans la bibliothèque de l'agent mobile. Par ailleurs, Java possède une bibliothèque riche qui comporte de nombreuses bibliothèques de sécurité parmi lesquelles nous mentionnons javax.security, javax.CryptoPackage ou javax.crypto.

6.2.3. Environnement de programmation

Un environnement de développement intégré (*EDI* ou *IDE - Integrated Development Environment*) est un programme regroupant un éditeur de texte, un compilateur, des outils automatiques de création, et souvent un débogueur. Pour le langage Java, il existe plusieurs EDI tels que NetBeans (de Sun), JBuilder (de Borland), JCreator ou Eclipse (d'IBM).

L'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. Eclipse est écrit en Java (à l'aide de la bibliothèque graphique SWT, d'IBM), et ce langage, grâce à des bibliothèques spécifiques, est également utilisé pour rajouter des extensions [ht7]. 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, en effet, développées en tant que plug-in.

6.3. Implémentation de l'agent mobile

Nous présentons, au sein de cette section, l'adaptateur et le manager qui constituent les deux composants les plus importants du code de l'agent mobile puisqu'ils s'occupent tous les deux de la mise œuvre de l'adaptation. L'adaptateur est composé principalement du moteur d'adaptation - délibérateur - ainsi que de la base de règles. Le délibérateur est représenté par la classe AgRunEngine et la base des règles correspond à la classe AgRuleBase (*cf.* Figure 6.). Les parties les plus importantes du code du moteur d'exécution sont:

La selection des règles qui correspondent à la QoS sélectionnée (*cf.* Section 5.3.1.6):

```
for(int i=0; i<ruleBaseEng.getRuleBaseSize();i++){
if(ruleBaseEng.getRule(i).getQosValue()==qosValue){
matchedQosValueList.add(new
AgMatchedQos(ruleBaseEng.getRule(i).getWeight(),i)); } }
```

La recherche du poids minimum parmi les règles dont les parties gauches correspondent à la QoS sélectionnée:

```
minWeight=((AgMatchedQos)matchedQosValueList.getFirst()
).getRuleWeight(); for(int i=0;i<matchedQosValueList.size();i++)
{if(((AgMatchedQos)matchedQosValueList.get(i) ).getRuleWeight()
<minWeight)
minWeight=((AgMatchedQos)matchedQosValueList.get(i)
).getRuleWeight();
}
```

La collecte des règles de poids minimum parmi la liste des règles correspondant à la QoS sélectionnée:

```
for(int i=0;i<matchedQosValueList.size();i++){
if(((AgMatchedQos)matchedQosValueList.get(i)).getRuleWeight()
==minWeight)matchedWeightList.add((AgMatchedQos)
matchedQosValueList.get(i));
}
```

La sélection de la règle à déclencher

```
selectedRuleIndex = ((AgMatchedQos)matchedWeightList.get
(safeRandom(matchedWeightList.size()))).getRuleIndex();
selectedRule= ruleBaseEng.getRule(selectedRuleIndex);
```

La classe AgRuleBase possède plusieurs méthodes. Un exemple est la méthode updateRuleWeight() qui met à jour le poids de la règle sélectionnée en l'incrémentant:

```
public void updateRuleWeight(int index){ updatedRule=(AgRule)
ruleBase.get(index);
updatedRule.setWeight(updatedRule.getWeight()+1); }
```

Dans ce qui suit, nous présentons les diagrammes de classes des principaux composants de notre Agent Mobile (*cf.* Chapitre 5).

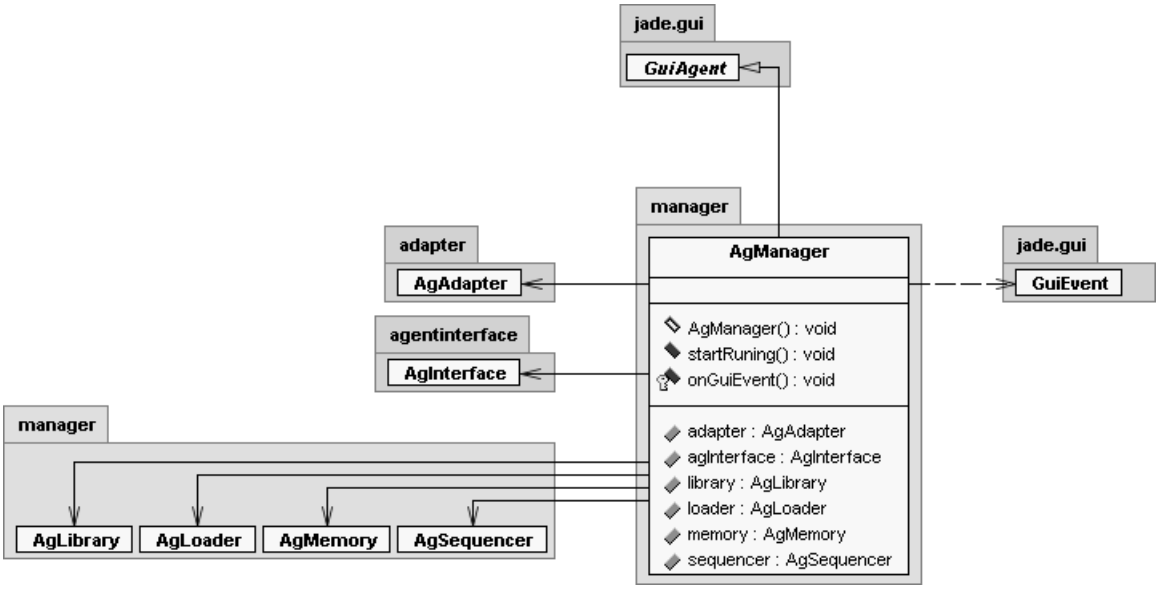

Fig.6.1. Diagramme de classe du composant manager.

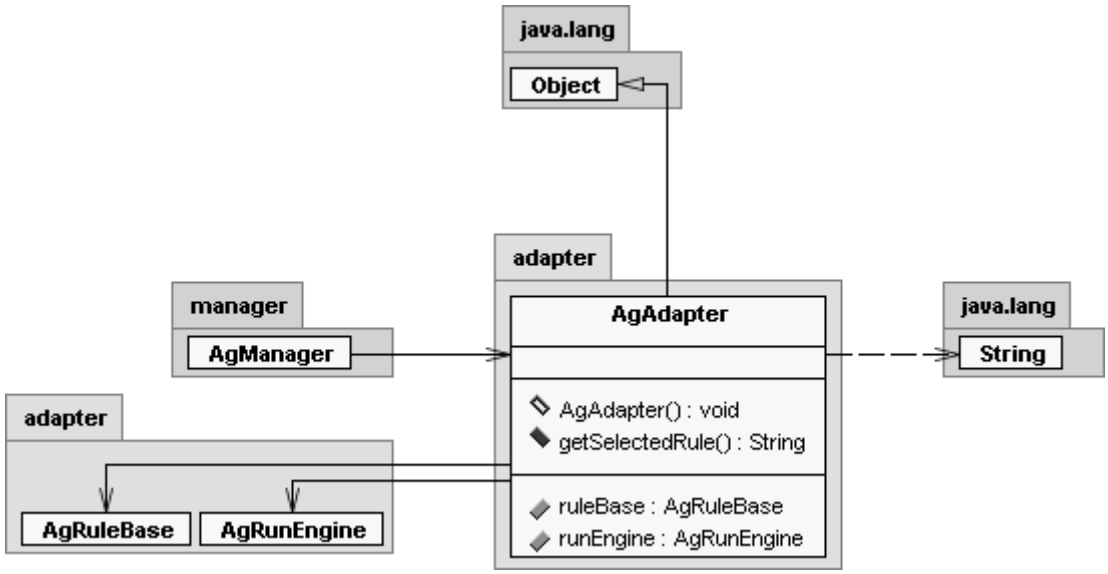

Fig.6.2. Diagramme de classe du composant adaptateur.

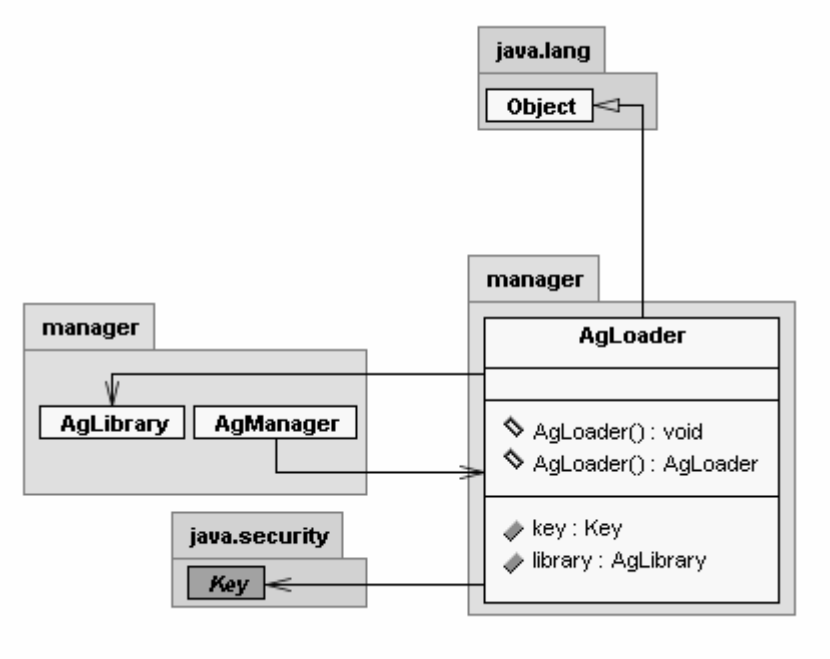

Fig.6.3. Diagramme de classe du composant chargeur.

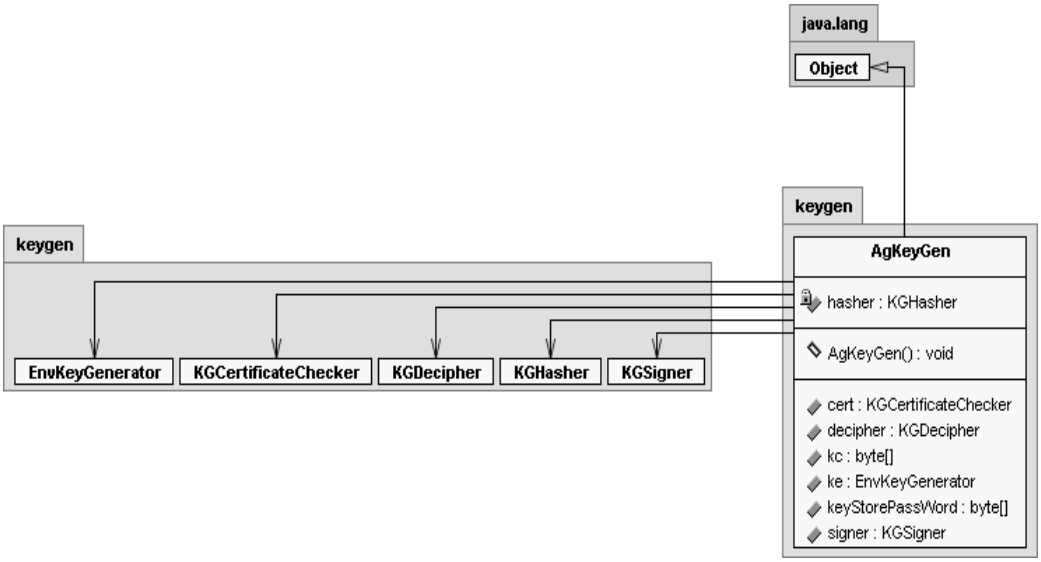

Fig.6.4. Diagramme de classe du générateur de clé.

6.4. Etude de performance

Dans cette section, nous allons discuter la fiabilité, la performance et la robustesse de notre protection vis-à-vis des attaques présentées antérieurement (*cf.* Chapitre 3).

6.4.1. Résistance contre les attaques

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

Lorsque l'agent mobile est envoyé vers les hôtes cibles, il passe par plusieurs hôtes avant d'atteindre sa cible. Certains hôtes sur le chemin, sont équipés d'analyseurs de paquets qui leurs permettent d'intercepter des objets spécifiques. Notre agent mobile peut en être victime. L'hôte malicieux va analyser le code de l'agent pour en tirer quelques informations sur l'identité et le comportement.

Discussion 1

Les modules reflétant l'aspect comportemental complexe de l'agent mobile sont encryptés avec une clé de taille 256 bits qui est modifiée périodiquement. Lorsque l'agent mobile désire exécuter un module sensible, ce dernier sera chargé, déchiffré en mémoire puis passé directement à l'exécution sans laisser aucune trace sur l'hôte. Dès que le module est hors portée de la fonction appelante ; il sera détruit automatiquement par le *Garbage Collector.*

Cet encryptage permet de contrecarrer l'analyse, pour que l'agent mobile puisse passer inaperçu à travers ces hôtes et éviter une attaque sérieuse. L'analyse statique est donc non efficace contre cette protection.

Scénario 2

Un hôte malicieux a réussi à intercepter l'agent mobile, il va tenter de l'exécuter pour l'analyser et déduire son comportement.

Discussion 2

Une fois l'agent mobile est exécuté, il commence par inspecter l'environnement et interagir avec l'utilisateur pour obtenir des informations nécessaires avant de fournir le service qu'il possède. Une interface de l'agent mobile est affichée à l'utilisateur en demandant quelques informations privées. Si une de ces informations est fausse (identificateur de client par exemple), l'agent n'arrivera pas à ouvrir le fichier KeyStore et par conséquent, l'agent va terminer son exécution.

Scénario 3

L'hôte malicieux a acquis les informations nécessaires par interaction de l'agent mobile (ID du client, pseudo du client, mot de passe). Il re-exécute l'agent mobile.

Discussion 3

Dès que l'agent mobile termine son exécution préliminaire (interaction, inspection et observation), il envoie la clé intermédiaire au pourvoyeur de service, et poursuit son exécution en attendant la réception de la QoS (avec/sans portion du code). Le pourvoyeur de service va détecter un défaut de paramètres car les informations perceptibles à l'utilisateur ne sont pas les seules à collecter ; il existe aussi celles collectées par inspection et par

observation de l'hôte visité. Par conséquent, le pourvoyeur de service envoit à son agent mobile un ordre (correspondant à la QoS0) d'arrêt de l'exécution et à une migration vers l'hôte suivant (*cf.* Section 3.4).

Scénario 4

L'hôte malicieux est le client lui-même ; il a acheté le service pour pirater son code. Il lance une requête au pourvoyeur de service ; il reçoit l'agent mobile qui débute son exécution. Il analyse l'exécution de l'agent en observant ses opérations d'entrée/sortie, nombre de processus,…, etc.

Discussion 4

Chaque fois que l'agent est exécuté, il se comporte différemment (adaptabilité). Au cours de l'observation de l'agent, les tâches (fictives et effectives) sont exécutées en parallèle (threads) afin d'embrouiller l'analyseur. Celui-ci va avoir l'impression que toutes ces tâches sont nécessaires pour accéder au service, alors qu'en fait, seulement quelques unes sont effectives. Cette aptitude permet de compliquer le comportement de l'agent mobile et donc d'améliorer sa protection contre l'analyse dynamique.

Scénario 5

L'analyse dynamique n'a pas donné de bons résultats. Cependant, elle a fourni au pirate quelques pistes (le premier module qui s'exécute, le dernier,…, etc.). Le pirate a aussi réussi à casser le cryptage des modules. Il est maintenant prêt à les décompiler pour restituer leurs codes sources et découvrir le comportement effectif de l'agent mobile.

Discussion 5

Tous les modules importants de l'agent mobile sont obscurcis, cela permet de rendre leurs restitutions très coûteuses en temps et en argent.

Scénario 6

L'hôte malicieux a pu restituer quelques modules de l'agent mobile, il va essayer de restituer le code du service.

Discussion 6

Le code du service qui accompagne l'agent ne fait que 1/3 du code global du service, et ce dernier est réuni seulement au cours de son exécution, donc sa restitution est inutile, et le service reste toujours protégé.

6.4.2. Les performances de l'agent mobile

Nous allons discuter, dans cette section, les performances des mécanismes de protection employés en termes du temps d'exécution et d'espace mémoire.

Obscurcissement

Un module obscurci n'a rien de différent qu'un module normal sauf que son code compilé ne permet plus de retrouver facilement le code source original même avec un décompilateur évolué. L'outil que nous avons utilisé pour l'obscurcissement est *RetroGuard*. Ce dernier permet de réduire la taille de l'agent mobile sans modifier son comportement. Ce qui accélère le transfert de l'agent mobile ainsi que son chargement sur l'hôte client.

Encryptage

Les algorithmes symétriques *AES* ou Blowfish peuvent, à titre d'exemple, être utilisés pour l'encryptage des modules. AES est un standard depuis 2000. *Blowfish* [ht8] est un logiciel libre qui utilise une taille de bloc de **64 bits** et la clé de longueur variable peut aller de **32** à **448 bits**. Elle est basée sur l'idée qu'une bonne sécurité contre les attaques de cryptanalyse peut être obtenue en utilisant de très grandes clés pseudo-aléatoires. *Blowfish* présente une bonne rapidité d'exécution, il est environ **5 fois** plus rapide que *Triple DES* et **deux fois** plus rapide qu'*IDEA*. Malgré son âge, il demeure encore solide du point de vue cryptographique avec relativement peu d'attaques efficaces sur les versions avec moins de tours. La version complète avec 16 tours (que nous utilisons) est à ce jour entièrement fiable [ht8]. Une clé de 128 bits est suffisante pour protéger les modules, car une clé de taille supérieure (256 ou 448 bits) va consommer plus de temps CPU pour décrypter les modules. Etant donné que ce type d'algorithme est en constante évolution, son choix revient au concepteur de l'agent.

Hachage

Nous avons utilisé le *SHA-256* pour équilibrer entre la sécurité et la vitesse d'exécution, le *SHA-1* (160 bits) est devenu moins résistant, cependant, le *SHA-512* est très robuste ; mais très lourd en exécution, cela nous a amené à choisir le *SHA-256*.

Signature numérique

Nous avons utilisé pour la signature numérique le *SHA-1withRSA*, le RSA est un algorithme asymétrique. Pour la signature, il est très fiable par rapport au DES (algorithme symétrique). Le *SHA-1withRSA* est l'algorithme le plus utilisé pour la signature numérique.

Dispersion du code

Cette technique permet de protéger le code de l'agent mobile d'une manière très efficace, car le code est distribué entre plusieurs entités. Nous aurions pu partager le code du service entre uniquement le pourvoyeur de service et l'agent mobile ; cependant, cette solution aurait présenté les inconvénients suivants :

- La portion du service émise par le pourvoyeur de service serait plus grande, et nécessiterait donc plus de temps pour la transférer vers l'agent, ainsi qu'une bande passante plus large pour assurer une fluidité du trafic convenable sur le réseau
- Le risque de piratage du code augmenterait, dans le cas où l'agent mobile serait capturé par un hôte malicieux.
- La taille de l'agent mobile serait plus grande.

Cependant, si le code est réparti en trois portions, nous notons un gain en temps de transfert du service, en espace mémoire (taille de l'agent mobile) et en matière de sécurité.

6.4.3. Les performances du processus d'adaptation

Nous étudions, dans cette section, l'influence de l'approche d'adaptation proposée sur la protection de l'agent mobile. Etant donné que l'adaptation repose sur la modification du comportement qui est essentiellement basée sur l'emploi des tâches fictives et des tâches effectives obscurcies, nous nous intéressons à leur nombre pour faire cette étude. Pour cela, nous avons réalisé un simulateur de l'adaptateur dynamique basé sur TAMAP (*cf.* Figure 6.5) ainsi qu'un programme malicieux réalisant l'analyse dynamique du code de l'agent mobile.

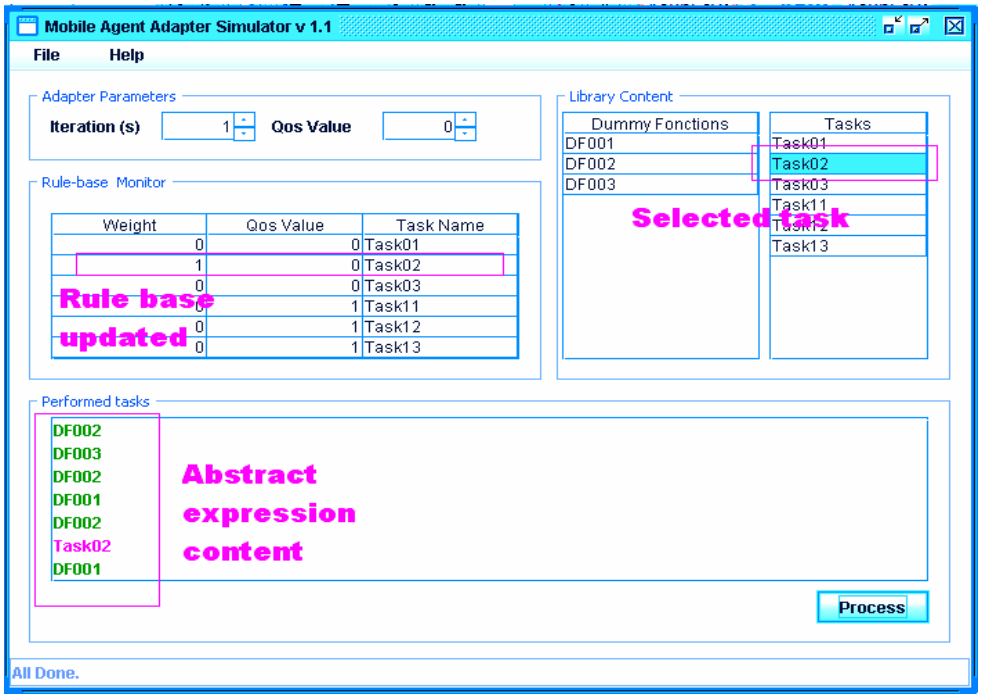

Fig. 6.5. Simulateur de l'adaptateur dynamique basé sur TAMAP

Nous avons réalisé cette simulation sur une machine de configuration suivante :

- Processeur Intel Pentium 4 520, 2856 Mhz (14 x 204)
- RAM 512 Mo
- Système d'exploitation : Microsoft Windows XP Professionnel.

Instruction	Durée moyenne en ns	Durée moyenne en s
Calcul	489	$489\ 10^{-9}$
Allocation mémoire	1485447	$1485447\ 10^{-9}$
Entrée/Sortie	8523	$8523\ 10^{-9}$

Table 6.1. Types et durée d'instructions

Nous supposons qu'une tâche est une suite d'instructions qui réalise un calcul, une allocation mémoire ou une Entrée/Sortie (*cf.* Table 6.1). Nous associons à chaque opération un temps d'exécution qui varie selon son type. Chaque tâche est une combinaison de ces instructions.

L'hôte essaye de découvrir la tâche effective de l'agent en se basant sur l'analyse de la séquence d'exécution ainsi que sur le temps d'exécution de chaque tâche. Pour que l'analyse converge, nous établissons les hypothèses suivantes qui sont prises en considération par l'analyseur de l'hôte malicieux:

1 Les tâches sont perçues par l'hôte qui a lancé l'analyse.

2 L'analyseur découvrira avec certitude la tâche réelle. Notre contribution tente uniquement de le retarder aussi longtemps que possible.

3 La consommation mémoire des tâches fictives ne dépasse pas 50% de la mémoire allouée à l'agent.

4 Le temps total consommé par les tâches fictives ne dépasse pas 50% du temps total réservé à l'exécution de l'agent mobile.

5 La séquence d'exécution est à chaque fois différente.

Les hypothèses (3) et (4) garantissent que les tâches fictives ne gêneront pas le fonctionnement de la tâche effective.

Dans la première simulation, nous avons utilisé une seule tâche effective et nous avons augmenté au fur et mesure le nombre de tâches fictives.

En s'appuyant sur la quatrième hypothèse, l'analyseur oriente son analyse en éliminant, à chaque itération, la tâche qui a consommé le moins de temps d'exécution. La tâche qui reste est considérée comme la tâche effective. Les tâches constituant la séquence d'exécutions sont des threads. Nous remarquons que dans cette simulation, la seconde hypothèse est respectée puisque la tâche effective est toujours découverte. La Figure 6.6 présente le résultat des deux tests. Nous notons que la durée de l'analyse est proportionnelle au nombre de tâches fictives et qu'elle peut varier d'un test à l'autre de manière aléatoire. De plus, la tâche effective est toujours découverte car elle possède une durée d'exécution maximale relativement aux durées d'exécution des tâches fictives.

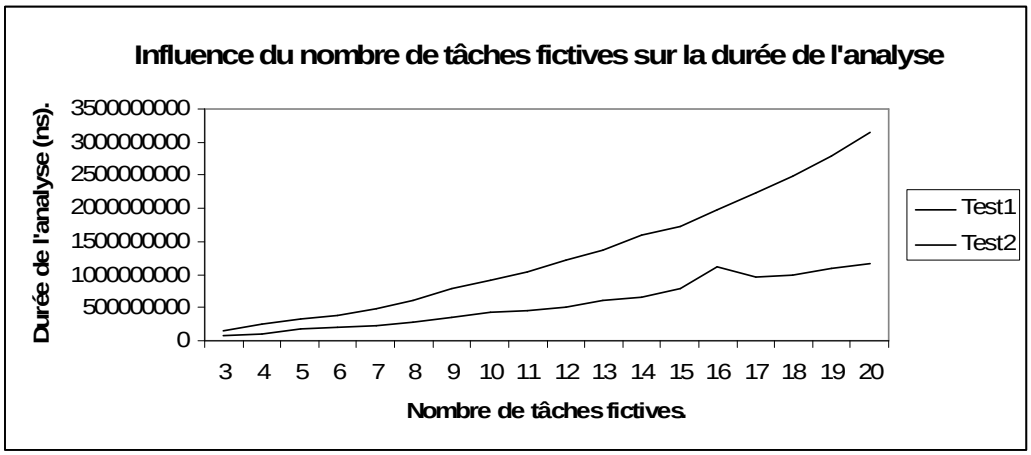

Fig.6.6. Résultat de la première simulation

Au niveau de la seconde simulation, nous nous intéressons au temps d'occupation du processeur. Pour cela, nous faisons en sorte que la quatrième contrainte soit toujours respectée mais en temps d'occupation du processeur. Nous rajoutons donc au niveau du code de chaque tâche fictive une instruction Sleep() durant un temps donné de façon à ce qu'on ait l'impression que toutes les tâches (fictives et effective) commencent et terminent leur exécution en même temps et minimiser ainsi l'écart existant entre elles. Ceci a pour but de compliquer la découverte de la tâche effective.

Dans un premier temps, nous avons remarqué qu'il arrive que l'analyseur élimine la tâche effective lorsque son temps d'exécution est le plus petit. Dans ce cas, nous considérons qu'il a échoué dans la découverte du service. Cependant, le succès ou l'échec de l'analyse est très aléatoire.

Nous avons donc fixé, le nombre de tâches fictives à 3 et modifié le critère de l'analyse. Etant donné, que les temps d'exécution de toutes les tâches sont devenus approximatifs, la découverte du service ne pourra plus reposer sur la durée d'exécution de la tâche. Elle dépendra plutôt du nombre de changements des tâches : la tâche qui apparaîtra toujours dans la séquence d'exécution sera la tâche effective. Nous affectons pour cela un compteur à chaque tâche qui est initialisé à zéro et incrémenté à chaque fois que la tâche n'apparaît pas dans la séquence d'exécution. L'analyse s'arrête lorsqu'il ne reste plus qu'une tâche dont la valeur du compteur est nulle. Elle représente la tâche effective. Nous nous fixons comme temps d'arrêt de la simulation 100 analyses (*cf.* Table 6.2). Le temps moyen de la durée de l'analyse ainsi que le nombre de tentatives avant de découvrir la tâche effective ont été calculés. Cette simulation a été répétée 3 fois de suite. Nous avons remarqué une variation de tentatives à chaque analyse tout à fait aléatoire. Ceci est dû au fait que la séquence d'exécution respecte la 5ème hypothèse et que la séquence est générée de manière aléatoire.

Durée moyenne de l'analyse (ns) pour 100 analyses	**Nombre moyen de tentatives pour la découverte de la tâche effective**	**Variation du nombre de tentatives**
380815619	8	2-29
362088652	8	2-27
311396010	7	2-23

Table 6.2. Résultat de la seconde Simulation

Au niveau de la troisième simulation, nous avons utilisé deux tâches effectives. Chacune d'elles représente, grâce à l'obscurcissement, deux implémentations différentes du même service. Nous nous fixons comme temps d'arrêt de la simulation 200 itérations pour pouvoir observer l'apparition des tâches au niveau de la séquence d'exécution. Nous avons réalisé trois simulations dans lesquelles nous avons compté le nombre de fois qu'une tâche apparaisse ou disparaisse de la séquence d'exécution.

F1	F2	F3	T1	T2	Durée de l'analyse pour 200 itérations (ns)
58	54	41	103	103	9520583638
48	48	62	106	106	10920173258
42	60	57	97	97	9554986209

Table 6.3. Résultat de la troisième Simulation

Nous remarquons que T1 et T2 apparaissent beaucoup plus souvent au niveau de la séquence d'exécution que F1, F2 et F3 (*cf.* Table 6.3); ceci est dû au fait qu'elles sont effectives donc une parmi elles doit apparaître à chaque fois. L'écart au bout de 200 itérations devient pratiquement le double. Par conséquent, facilement observable.

Au niveau de la quatrième simulation, nous avons, dans un premier temps, utilisé un nombre de tâches effectives égal au nombre de tâches fictives. Nous avons observé qu'au bout du même nombre d'itérations (200), l'écart diminue sans pour cela devenir négligeable (*cf.* Table 6.4).

F1	F2	F3	T1	T2	T3	Durée de l'analyse pour 200 itérations (ns)
50	58	48	83	79	72	17546567657
49	62	48	84	83	83	18688972904
72	50	50	96	86	94	18666805180

Table 6.4. Résultat de la quatrième Simulation (a)

Nous avons augmenté le nombre de tâches effectives (4 au lieu de 3). Nous avons alors observé que l'écart devient de plus en plus négligeable (T3 et F3) et même rendre la valeur du compteur de la tâche effective inférieur à celui de la tâche fictive (T4 et F2). Par conséquent, on ne peut plus découvrir facilement le service. Cette solution est intéressante, cependant elle risque d'empiéter sur la taille de l'agent et la 3ème hypothèse n'est plus respectée (*cf.* Table 6.5).

F1	F2	F3	T1	T2	T3	T4
47	44	**68**	74	74	**68**	78
35	60	**64**	**71**	72	87	70
50	**72**	56	94	81	74	**65**

Table 6.5. Résultat de la quatrième Simulation (b)

Nous avons conclu que le nombre de tâches fictives influe certainement sur l'analyse. Cependant, tout repose sur l'algorithme de choix de la séquence d'exécution. Il faut utiliser un algorithme qui sache équilibrer les apparitions de toutes les tâches (fictives et effectives) au niveau de la séquence d'exécution et ne pas permettre à l'analyseur dynamique de retrouver le service en s'appuyant sur l'observation de la séquence.

6.5. Conclusion

Le problème de la sécurité des agents mobiles qui est un problème sous-jacent à celui de la sécurité informatique, n'a jamais été totalement résolu. Notre approche de protection ne prétend pas l'avoir totalement résolu car elle présente certainement des faiblesses qui pourraient être exploitées par des pirates. Cependant, TAMAP est une approche hybride qui a apporté des solutions à plusieurs étapes du cycle de vie de l'agent mobile. Elle a recherché d'abord à vérifier la crédibilité de l'hôte et la considérer comme prérequis à son exécution complète. Ensuite, elle s'est penchée sur la protection de la communication qui s'instaure entre l'agent et le pourvoyeur de service ou entre l'agent et l'hôte visité. Elle a aussi compliqué son analyse statique en instaurant des obstacles basés sur la dispersion du code, l'obscurcissement, la cryptographie et le hachage. Enfin, elle a offert à l'agent mobile la possibilité de se comporter de manière imprévisible et a compliqué ainsi son analyse dynamique. L'étude des performances réalisée au sein de ce chapitre a permis d'avoir une idée sur sa robustesse. Toutefois, elle a été conçue en regard des attaques actuelles; ce qui pourrait la rendre vulnérable aux nouvelles attaques. Nous notons que l'efficacité de TAMAP dépend fortement de son implémentation, car une mauvaise implémentation risque d'engendrer des failles qui peuvent être exploitées par des hôtes hostiles.

Chapitre 7 – Conclusions et perspectives

7.1. Conclusions

L'utilisation massive d'applications mobiles, employant notamment des agents mobiles, a été engendrée par des phénomènes récents tels que le nomadisme des utilisateurs, la mobilité des terminaux ou l'activité en mode déconnectée. Ces applications évoluent dans des environnements de plus en plus dynamiques, imprévisibles et hostiles et engendrent donc un important problème de sécurité dont la résolution constitue un véritable challenge. Ce défi est expliqué par l'importance de l'utilisation des agents mobiles dans les applications distribuées et l'intérêt de pouvoir pleinement profiter de leurs atouts.

Le travail effectué dans cette thèse s'inscrit dans le cadre de cette problématique. Il vise particulièrement à protéger les agents mobiles contre l'analyse dynamique de leur code. Pour résoudre ce problème, nous avons commencé par une étude des approches de protection existantes. Cette analyse a montré l'inexistence de solution de protection universelle au problème de l'hôte malicieux et a révélé que la seule protection susceptible de couvrir toutes les menaces est indubitablement celle qui consiste à exécuter l'agent mobile dans un environnement de confiance. C'est la raison pour laquelle nous nous sommes penchés sur le problème d'estimation de la confiance de l'environnement d'accueil. La sécurité de l'agent mobile exige de surcroît une évaluation dynamique de la crédibilité des hôtes visités pour déterminer le comportement adéquat de l'agent mobile suivant le degré de confiance détecté. Pour cela, la notion de QoS a été employée puisqu'elle permet de dégrader le service selon le degré de confiance décelé. En outre, dans le souci de compliquer l'analyse du comportement de l'agent mobile, le concept de l'adaptabilité dynamique a été utilisé. Il offre à l'agent mobile la possibilité d'avoir un comportement imprévisible. Par ailleurs, les informations transmises (code et données) entre l'agent mobile et son hôte d'origine doivent aussi être protégées. L'utilisation de la notion de clé environnementale a rendu possible cette protection.

7.1.1. Contributions

Un hôte malveillant peut déduire la stratégie d'exécution d'un agent mobile en analysant son code. Pour empêcher ce genre d'attaques, nous nous sommes appliqués au cours de cette thèse à concevoir une technique de protection hybride faisant intervenir les notions de confiance et d'adaptabilité et ajustant des approches de protection existantes. Nous citons principalement l'approche de la clé environnementale [Riordan 1998] et celle de la distribution de secret [Beimel 2000], [Schneider 1997]. L'approche de protection proposée intervient en trois points : un protocole de protection, un mécanisme d'évaluation de la confiance et un processus d'adaptation dynamique.

Protocole de sécurité

Le protocole de protection a pour objectif d'utiliser des mécanismes fiables pour la protection du code de l'agent mobile tels que l'encryptage, le hachage et la signature digitale. Il prend en charge aussi bien la protection de l'agent mobile que celle de la communication entre l'agent

mobile et le pourvoyeur de service. Il préserve ainsi la confidentialité des données de l'agent mobile et renforce son niveau de sécurité. Ce protocole est principalement basé sur l'utilisation de la clé environnementale.

Un mécanisme d'estimation et de prise en compte de la confiance

Le mécanisme d'estimation de la confiance permet d'établir la crédibilité de l'hôte et donc la sélection de la qualité de service la plus appropriée. Ce mécanisme repose sur des données propres au client et à leur comparaison avec celles détenues par le pourvoyeur de service. En outre, les données collectées au niveau de l'hôte client sont utilisées pour la génération de la clé environnementale. Pour éviter son transfert dans un réseau non sûr, celle-ci est calculée au niveau du pourvoyeur de service ainsi que par l'agent mobile au niveau du client. Elle est utilisée en tant que clé symétrique pour l'encryptage et le décryptage de la QoS émise. Par conséquent, le mécanisme d'estimation de la confiance est basé sur la clé environnementale générée et il est identifié par les étapes suivantes :

- L'inventaire des paramètres définissant la confiance et dont les valeurs doivent être vérifiées
- L'évaluation des paramètres de la confiance qui se traduit par l'attribution d'une valeur à chacun des paramètres inventoriés.
- La quantification de la confiance
- La sélection de la QoS appropriée.

Un processus d'adaptation

Le processus d'adaptation dynamique permet l'adaptation du comportement de l'agent mobile en se basant sur la crédibilité de l'hôte d'accueil, l'historique du comportement de l'agent mobile et de la QoS fournie. La sélection du comportement adéquat est basée sur l'aptitude de l'agent mobile à varier son comportement. En outre, le processus d'adaptation dynamique est basé sur le degré de dégradation du service et sur la capacité de proposer diverses solutions de rechange à la même QoS.

7.1.2. Apports

Nous avons présenté une technique pour permettre à l'agent mobile de protéger le comportement de l'agent mobile en compliquant son analyse dynamique. Aucune approche, à notre connaissance, n'a utilisé l'adaptabilité pour résoudre ce problème. Cette technique repose principalement sur l'estimation de la crédibilité de l'hôte visité. Elle permet une réaction basée sur des valeurs de paramètres tangibles. Ce mécanisme d'évaluation de la confiance est lié au contenu de la clé intermédiaire (obtenue à la phase préliminaire de la construction de la clé environnementale).

Comparé aux modèles usuels de la confiance, notre mécanisme fournit plusieurs avantages :

- Il simplifie beaucoup d'infrastructures complexes et coûteuses pour l'évaluation des risques comme des bases de données pour la maintenance de la réputation.
- L'évaluation de la confiance est basée sur des valeurs concrètes. Ainsi, en cas de déduction d'hôte non fiable, il est possible d'identifier la source exacte du résultat obtenu. Cette capacité est très importante puisqu'elle permet d'éviter une réaction inadéquate.

- Le modèle de confiance proposé est flexible : le pourvoyeur de service peut modifier les paramètres de confiance, les intervalles d'estimation de la confiance en se basant sur l'importance du service (le montant mis en jeu, la sensibilité du service, les risques encourus et les dommages qui en résultent) ou sa péremption (s'il est ou non périmé). Toutes ces modifications préservent la sécurité du code de l'agent mobile en rendant ces informations obsolètes.
- Le processus d'adaptation dynamique fournit un intérêt certain puisqu'il offre à l'agent mobile la capacité de réagir avec un comportement imprévisible. Cette aptitude empêche l'hôte visité de déduire la stratégie empruntée. En conséquence, n'importe quelle tentative d'analyse de comportement devient difficile.
- Le pourvoyeur de service a la possibilité, à tout instant, de modifier la politique d'adaptation en augmentant ou diminuant le nombre de tâches fictives ou tâches effectives et ainsi compliquer ou simplifier son analyse selon le risque évalué.
- La formule de calcul de la confiance est linéaire et n'influe donc pas sur le coût de l'approche de protection.
- Les algorithmes de calcul de la clé environnementale et d'estimation de la confiance sont très simples et donc réalisables par n'importe quel organisme.

7.2. Perspectives

Notre approche de protection offre une première tentative de protection employant l'adaptabilité afin de prévenir l'analyse dynamique du code de l'agent mobile. Ce qui explique que les perspectives de ce travail soient nombreuses. Nous citons dans cette section quelques unes d'entre elles.

7.2.1. Augmenter l'autonomie de l'agent mobile

L'agent mobile lui-même pourrait prendre l'initiative de réagir après l'évaluation de crédibilité de l'hôte. Ceci peut être réalisé par une adaptation dynamique comportementale. L'agent mobile estimerait donc la crédibilité de l'hôte et modifierait les intervalles d'estimation de la confiance selon le niveau de sécurité de l'environnement. Ainsi, il pourrait décider quelle QoS adopter. Ceci aurait l'avantage d'augmenter l'autonomie de l'agent mobile dans notre modèle. Par ailleurs, l'autonomie peut aussi être améliorée par la faculté d'apprentissage de l'agent mobile.

7.2.2. Adapter les intervalles d'estimation à la sensibilité du service

La confiance constitue une méthode pour construire un agent mobile dont le comportement est sensible aux conditions de sécurité rencontrées au niveau de l'environnement de l'hôte visité. Le mécanisme d'estimation de la confiance proposée est simple, il permet de juger la crédibilité de l'hôte d'accueil. Cependant, il gagnerait à être amélioré en modélisant la phase de prise en compte du degré de confiance.

7.2.3. Modélisation des métriques de la confiance

Nous nous sommes basés, dans l'estimation de la confiance, sur deux types de paramètres : les paramètres internes et les paramètres externes. Toutefois, le modèle de confiance proposé exige une grande aptitude aussi bien dans les phases d'inventaire des paramètres de confiance que celle de leur évaluation. Ces phases constituent donc une voie de recherche à explorer pour augmenter la généricité de notre technique de protection.

Références bibliographiques

[Abadi 1990] M. Abadi and J. Feigenbau, “Secure circuit evaluation: a protocol based on hiding information from an oracle,” Journal of Cryptology, vol. 2, 1990.

[Abdul-Rahman 1997] A. Abdul-Rahman and S. Hailes, “Using Recommendations for Managing Trust in Distributed Systems”, In Proceedings of IEEE Malaysia International Conference on Communication'97 (MICC'97). Kuala Lumpur, Malaysia, 1997.

[Alfalaleh 2004] M. Alfalaleh and L.Brankovic, “An Overview of Security Issues and Techniques in Mobile Agents”, 2004.

[Algesheimer 2001] J. Algesheimer, C. Cachin, J. Camenisch and G. Karjoth, “Cryptographic Security for Mobile Code”. IEEE Symposium on Security and Privacy, 2001.

[Allée 2001] G. Allée, “Sécurité des Agents Mobiles: Protocole d’Enregistrement d’Itinéraire par Agents Coopérants”. Mémoire de maîtrise, École Polytechnique de Montréal, Canada, Février 2001.

[Amano 2001] N. Amano and T. Watanabe "Towards Constructing Component-based Software Systems with Safe Dynamic Adaptability". Proceedings of the 4th International Workshop on Principles of Software Evolution. ISBN: 1-58113-508-4. Vienna, Austria, page 178 – 181, 2001.

[Amara 2005] N. AMARA-HACHMI1 AND A. EL FALLAH-SEGHROUCHNI, “Towards a generic architecture for self-adaptive,” Proceedings of 5th European Workshop on Adaptive Agents and MultiAgent Systems (AAMAS’05), Paris, 2005.

[Andert 2002] D. Andert, R. Wakefield and J. Weise, “Professional services security practice”, Sun BluePrints, OnLined December 2002.

[Andrade 2003] L-F. Andrade, J-L. Fiadeiro. "Architecture Based Evolution of Software Systems". Third International School on Formal Methods for the design of Computer, communication and Software Systems: Software Architectures, SFM 2003, Italy, 2003.

[Antonpoulos 2000] N. Antonopoulos, K. Koukoumpetsos and K. Ahmad, “A Distributed Access Control Architecture for Mobile Agents”, In Proceedings of the International Network Conference 2000, Plymouth, UK, July 2000.

[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.

[Barak 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'2001, Lecture Notes in Computer Science, Vol. 2139, pages 1—18, 2001.

[Bazzi 1998] R. A. Bazzi, "Code Hiding for Mobile Agents Security", Technical Report TR11 12998, Department of Computer Science and Engineering, Arizona State University, Novembre 1998.

[Beaucamps 2007] P. Beaucamps, E. Filiol, “On the Possibility of Practically Obfuscating Program - Towards a Unified Perspective of Code Protection”. WTCV'06

Special Issue, G. Boufante I \& J. Y. Marion editions, Journal in Computer Vrology, 3(1), 2007.

[Beimel 2000] A. Beimel, M. Burmester, “Computing Functions of a Shared Secret”. SIAM J. Discrete Math., Vol. ~13, No.3, 324—345, 2000.

[Belaramani 2002] N. M. Belaramani, “A component-based software system with functionality adaptation for mobile computing”. Master’s thesis, 2002.

[Bella 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.

[Bennet 1999] Y.S. Bennet, “A Sanctuary for Mobile Agents”. Secure Internet Programming, LNCS, Vol. 1603, pages 261-273, 1999.

[Biehl 1998] I. Biehl, B. Meyer and S. Wetzel, “Ensuring the Integrity of Agent-Based Computations by Short Proofs”. Proceedings of the second International Workshop on Mobile Agents. LNCS, Vol. 1477, pages 183-194, 1998.

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

[Birk 2001] A. Birk, “Learning to trust”. In R. Falcone, M. Singh, and Y. H. Tan, editors, Trust in Cyber-societies, number 2246 in LNAI, pages 27-54. Springer-Verlag, Berlin, 2001.

[Birrell 1984] A.D. Birrell and B.J. Nelson, “Implementing remote procedure calls”. ACM Transactions on Computer Systems, 2(1), pages 39–59, February 1984.

[Bhattacharya 1998] Bhattacharya Rajeev, Timothy M. Devinney, Madan M. Pillutla, “A Formal Model of trust based on outcomes”, in Academy of Management Review, Vol 23, No3, pages 459-472, 1998.

[Blaze 1996] M. Blaze, J. Feigenbaum and J. Lacy, “Decentralized trust management”. In Proceedings of the 1996 IEEE Conference on Security and Privacy, Oakland, CA, pages 164 –173, May 1996.

[Blaze 1999] M. Blaze, J. Feigenbaum, J. Ioannidis and A.D. Keromytis, “The Role of Trust Management in Distributed Systems Security”. In Secure Internet Programming, J.Vitek, C. Jensen, Ed., Springer-Verlag, pages 185-210, 1999.

[Boneh 1996] D. Boneh, R.J. Lipton, “Algorithms for black-box fields and their application to cryptography”. Advances in cryptology, CRYPTO’96

[Blum 1988] M. Blum and S. Kanan, "Designing programs to check their work", Technical report TR-88-009, International Computer Science Institute, December 1988.

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

[Bosworth 2002] K. P. Bosworth and N. Tedeschi, “Public key infrastructures: the next generation”. In Robert Temple and John Regnault, editors, Internet and wireless security, BTexact Communications Technology series 4, IEEE, London, pages 95-120, 2002.

[Braynov 2002] S. Braynov and T. Sandhol, “Trust Revelation in Multiagent Interaction”. Proceedings of CHI'02, Workshop on the Philosophy and Design of Socially Adept Technologies, Minneapolis, 2002.

[Cahil 2003] V. Cahill et al., “Using Trust for Secure Collaboration in Uncertain Environment”. IEEE Pervasive Computing, 2(3), pages 52-61, 2003.

[Camp 2002] L.J. Camp, “Designing for Trust”. In: R. Falcone, S. Barber, L. Korba and M. Singh (eds.): Trust, Reputation, and Security: Theories and Practice, Springer-Verlag; Berlin Heidelberg, New York, pages 15-29, 2002.

[Capra 2004] L. Capra, “Engineering Human Trust in Mobile System Collaborations”, In Proceeding of the 12th International Symposium on the Foundations of Software Engineering (SIGSOFT 2004/FSE-12). Newport Beach, CA, November 2004.

[Carbone 2003] M. Carbone, M. Nielsen and V. Sassone, “A Formal Model for Trust in Dynamic Networks”, In Proceeding Of 1st International Conference on Software Engineering and Formal Methods (SEFM’03), Brisbane, Australia, pages 54–63, September 2003.

[Carzaniga 1997] A. Carzaniga, G. P. Picco and G. Vigna, “Disigning Distributed Applications with Mobile Code Paradigms”, 19th International Conference on Software Engeneering. ACM Press, May 1997.

[Castel 2000a] C. Castelfranchi, R. Falcone, “Trust is much more than Subjective Probability: Mental Components and Sources of Trust”. The 32nd Hawaii International Conference on System Sciences - Mini-Track on Software Agents, Maui, Hawaii, 2000.

[Castel 2001] C. Castelfranchi and Y. Tan, editors, “Trust and Deception in Virtual Societies”. Kluwer Academic Publishers. The Netherlands, 2001.

[Castel 2000b] C. Castelfranchi, R. Falcone, B. Sadighi and Y-H Tain, Guest Editorial, Applied Artificial Intelligence, 14(9), Taylor & Frances, 2000.

[Castel 1998] C. Castelfranchi and R. Falcone, “Principles of Trust for MAS: Cognitive Anatomy, Social Importance, and Quantification”, In Y. Demazeau (editor), Proceedings of the Third International Conference on Multi-Agent Systems. IEEE C.S., Los Alamitos, 1998.

[Chefrour 2003a] D. Chefrour et F. André, “Auto-adaptation de composants ACEEL coopérants”. In 3ème Conférence Française sur les Systèmes d'Exploitation (CFSE'3), La Colle sur Loup, France, octobre 2003.

[Chefrour 2003b] D. Chefrour et F. André, “Développement d'applications en environnements mobiles à l'aide du modèle de composant adaptatif ACEEL”. In Langages et Modèles à Objets LMO'03. Publié dans STI, série L'objet, vol. 9, Hermes. Vannes, France, 2003.

[Chess 1995] D. Chess, B. Grosof, C. Harrison, D. Levine, C. Parris and G. Tsudik, “Itinerant Agents for Mobile Computing”. Technical Report, IBM T.J. Watson Research Center, NY, 1995.

[Chess 1994] D. Chess, C. Harrison and A. Kershenbaum, “Mobile Agents : Are They a Good Idea ?” Technical Report RC 19887, IBM Research Division, T.J. Watson Research Center, 1994.

[Coleman 1990] J. Coleman, “Foundations of Social Theory”, Belknap Press, Cambridge, MA, 1990.

[Collberg 1997] C. Collberg, C. Thomborson and D. Low, “A Taxonomy of Obfuscating Transformations”. In Technical Report 148, Department of Computer Science, University of Auckland, July 1997.

[Cubat 2005] C. CUBAT, “Agents Mobiles Coopérants pour les Environnements Dynamiques”. Thèse de doctorat de l’Institut National Polytechnique de Toulouse (ENSEEIHT), Décembre 2005.

[Dadon-Elichai 2004] A. Dadon-Elichai, “RDS: Remote Distributed Scheme for Protecting Mobile Agents”. In: The Third International Joint Conference on Autonomous Agents and Mutli-Agent Systems, 2004.

[D’Anna 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, T. Ledoux, "An Infrastructure for Adaptable Middleware ". DOA’02, Springer Verlag, LNCS 2519, Irvine, California, 2002.

[Dimitrakos 2003a] T. Dimitrakos, “A Service-Oriented Trust Management Framework”. Falcone R., Barber S., Korba L., and M. Singh, editors, Trust, Reputation, and Security: Theories and Practice, LNAI 2631. Springer, pages 53-72, 2003.

[Dimitra 2003b] T. Dimitrakos, “Trust, Security and Accounting”, Business & Information Technology. CCLRC Rutherford Appleton Laboratory, 2003.

[Dimitrakos 2001] T. Dimitrakos, “System Models, e-Risk and e-Trust”. Towards Bridging the Gap in Towards the ESociety: E-Business, E-Commerce, and E-Government, eds. B. Schmid, K. Stanoevska-Slabeva, V. Tschammer. Kluwer Academic Publishers, 2001.

[Dumant 1998] B. Dumant, F. Dang Tran, F. Horn, J.-B. Stefani. “Jonathan: an open distributed processing environment in java. “Middleware'98: IFIP International Conference on Distributed Systems Platforms and Open Distributed Processing, The Lake District, U.K., 1998.

[Farmer 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.

[Farmer 1996b] W. Farmer, J. Guttmann and V. Swarup, "Security for Mobile Agents: Issues and Requirements", in: Proceedings of the National Information Systems Security Conference (NISSC 96), 1996.

[Ferber 1995] J. Ferber, “Les Systèmes Multi-Agents”, Informatique Intelligence Articielle, InterEdition, 1995.

[Filiol 2005] E. Filiol, “Strong Cryptography Armoured Computer Viruses Forbidding Code Analysis: the Bradley virus”. Proceedings of the 14th EICAR Conference, 2005.

[Filiol 2007] E. Filiol, “Techniques virales avancees (chapter 8)”, collections IRIS, Springer Verlag, january 2007.

[Filiol 2006] E. Filiol, “Malware Pattern Scanning Schemes Secure Against Black-box Analysis”. EICAR 2006 Special Issue, V. Broucek I \& Paul Turnee eds, Journal in Computer Virology, 2 (1), 2006.

[FIPS 2000] FIPS PUB 186-2, Digital Signature Standard (DSS). Gaithersburg, MD, January 2000.

[Fuggetta 1998] A. Fuggetta, G. P. Picco and G. Vigna, “Understanding Code Mobility”. IEEE Transactions on Software Engineering, 24(5), pages 342–361, May 1998.

[Funfrocken 1999] S. Funfrocken, “Protecting Mobile Web-Commerce Agents with Smartcards”. In Proceedings of the First International Symposium on Agent Systems and Applications / Third International Symposium on Mobile Agents (ASA/MA’99), IEEE Computer Society, pages 90-102, 1999.

[Gambetta 1990] D. Gambetta, “Can we Trust Trust? Trust: Making and Breaking Cooperative Relations”, Gambetta, D (ed.), Basil Blackwell, Oxford, 1990.

[Gong 1998a] L. Gong, “Secure java class loading. IEEE Internet Computing”, pages 56—61, 1998.

[Gong 1998b] L. Gong, “Java Security Architecture (JDK1.2)”, Technical Report, Sun Microsystems, Inc., 901 San Antonio Road, Palo Alto, California 94303, U.S.A, 1998.

[Gong 1997] L. Gong, M. Mueller, H. Prafullchandra and R. Schemers, “Going Beyond the Sandbox: An Overview of the New Security Architecture in the Java Development Kit 1.2”, In Proceedings of the USENIX Symposium on Internet Technologies and Systems, Monterey, California, December 1997.

[Grandison 2000] T. Grandison and M. Sloman, “A Survey of Trust in Internet Applications”, IEEE Communications Surveys and Tutorials, Fourth Quarter, 2000.

[Gray 2002] R.S. Gray, G. Cybenko, D. Kotz, R. A. Peterson and D. Rus, “D’agents : Applications and performance of a mobile-agent system”. Softw., Pract. Exper., 32(6), pages 543–573, 2002.

[Gray 2000] R. S. Gray, G. Cybenko, D. Kotz and D. Rus, “Mobile agents: Motivations and state of the art”. Technical Report TR2000-365, Department of Computer Science, Dartmouth College, Hanover, New Hampshire, USA, April 2000.

[Gray 1998] R. S. Gray, G. Cybenko, D. Kotz and D. Rus, “D’Agents : Security in a multiple-language, mobile-agent system”. In Giovanni Vigna, editor, Mobile Agent Security, volume 1419 of Lecture Notes in Computer ScienceSpringer-Verlag: Heidelberg, Germany, , pages 154–187. 1998.

[Gribble 1999] S.Gribble, M.Welsh, E.Brewer, D.Culler “The MultiSpace: an Evolutionary Platform for Infrastructural Services”. Proceedings of the 1999 Usenix Annual Technical Conference, Monterey, CA, 1999.

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

[Guessoum 2003] Z.Guessoum, “Modèles et architectures d'agents et de systèmes multi-agents adaptatifs”. Thèse d'habilitation de l'université de paris 6, Laboratoire d'Informatique de Paris 6, 2003.

[Guessoum 2004] Z. Guessoum, M. Ziane and N. Faci, “Monitoring and Organizational-Level Adaptation of Multi-Agent Systems”. AAMAS'04, ACM, New York City pages 514—522, 2004.

[Guy 1999a] B. Guy, “Applicabilité et performances des systèmes d’agents mobiles dans les systèmes répartis”. In Première Conférence Française en Systèmes d’Exploitation (CFSE’1), Rennes, France, pages 8-11, Juin 1999.

[Guy 1999b] B. Guy, “Technologie du code mobile : état de l’art et perspective”. In Colloque Francophone sur l’Ingénierie des Protocoles (CFIP’99), Nancy, France,pages 29-29, Avril 1999.

[Hacini 2004] S. Hacini, "Prise en charge de la sécurité dans une association d’entreprises partenaires véhiculant des agents mobiles" Actes des 2èmes journées d’informatique pour l’entreprise, Blida, pages 287-296, 2004.

[Hacini 2005] S. Hacini, “Using Adaptability to Protect Mobile Agents Code”. IEEE International Conference on Information Technology ITCC 2005, Las Vegas, pages 49-53, USA, 2005.

[Hacini 2006a] S. Hacini, Z. Guessoum and Z Boufaida, “Using a Trust-Based Key to Protect Mobile Agent Code”. Transactions on Engineering, Computing and Technology, vol. ~16, ISSN 1305-5313 © 2006 World Enformatika Society, CCIS'2006, Venice, Italy, pages 326-332, 2006.

[Hacini 2006b] S. Hacini, C. Cheribi and Z. Boufaïda, "Dynamic Adaptability using Reflexivity for Mobile Agent Protection", in Transactions On Engineering, Computing And Technology Enformatika V17 2006 ISSN 1305-5313, Cairo, Egypte, pages 222-227, December 2006.

[Hacini 2007a] S. Hacini, Z. Boufaïda and C. Cheribi,”Mobile Agent Protection in e-business Applications: A Dynamic Adaptability Based Approach”. The Second Symposium On Information Security (IS’07). Springer Lncs. Vilamoura, Algarve, Portugal, November 2007.

[Hacini 2007b] 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, n°4/Nov. 2007, pages 267-283. DOI 10.1007/s11416-007-0056-y.

[Hauswirth 2000] M. Hauswirth, C. Kerer and R. Kurmanowytsch, “A secure execution framework for Java,” In Proceedings of the 7th ACM conference on computer and communications security (CCS 2000), Athens, Greece, pages 43-52, November 2000.

[Hermann 2005] P. Hermann, V. Issarny and S. Shue (editors), Third International Conference on Trust Management. Lecture Notes in Computer Science, vol. 3477, Springer, 2005.

[Herzberg 1985] A. Herzberg, S.S. Pinter, “Public Protection of Software”. Advances in Cryptology: Crypto 85, Springer-Verlag, Berlin, pages 158-179, 1985.

[Hoglund 2006] G. Hoglund and J. Butler, “Rootkits Infiltrations du noyau windows”. CampusPress-2006.

[Hohl 1997] F. Hohl, “An approach to solve the problem of malicious hosts”. Universität Stuttgart, Fakultät Informatik, Fakultätsbericht Nr. 1997/03, 1997.

[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.

[Hohl 2000] Hohl, F.: A framework to Protect Mobile Agents by Using References States. In Proceedings of ICDCS 2000, 2000

[Hohl 1999] F. Hohl, “A Protocol to Detect Malicious Hosts Attacks by Using Reference States”, Technical Report No.09/1999, Faculty of Informatics, University of Stuttgart, Germany, August 1999.

[Ismael 1999] L. Ismael and D. Hagimont, “A performance evaluation of mobile agent paradigm”, In Proceedings of the International Conference on Object-Oriented Programming, Systems, Languages, and Applications (OOPSLA’ 99), pages 306–313, November 1999.

[Jansen 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 Multiagent 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

[Jennings 1995] N. R. Jennings and M. Wooldridge, “Intelligent agents: Theory and practice”. The Knowledge Engineering Review, 10(2), pages 115-152, 1995.

[Jennings 1998a] N. R. Jennings, M. Wooldridge and K. Sycara, “A Roadmap of Agent Research and Development”, International Journal of Autonomous Agents and Multi-Agent Systems, vol. 1, n°1, pages 7-38, 1998.

[Jennings 1998b] N. R. Jennings and M. J. Wooldridge, “Applications of Intelligent Agents”. In Nicholas R. Jennings and Michael J.Wooldridge, editors, Agent Technology: Foundations, Applications, and Markets. Springer-Verlag Berlin Heidelberg New York, pages 3–28, 1998.

[Jensen 2004] C.D. Jensen, S. Poslad and T. Dimitrakos (editors), “Second International Conference on Trust Management”, Lecture Notes in Computer Science, vol. 2995, Springer, 2004.

[Johansen 1998] D. Johansen, “Mobile agent applicability”. Lecture Notes in Computer Science, 1477, pages 80–98, 1998.

[Jones 1999] S. Jones, “TRUST-EC: Requirements for Trust and Confidence in E-Commerce”, European Commission Joint Research Centre, 1999.

[Josang 1999a] A. Josang, “Trust-Based Decision Making for Electronic Transactions”, The 4th Nordic Workshop on Secure ITSystems (NORDSEC'99), Stockholm University Report 99-005, Stockholm, 1999.

[Josang 2001] A. Josang, “A Logic for Uncertain Probabilities” International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems, 9(3), pages 279-311, 2001.

[Josang 2OO4] A. Josang, S. Lo Presti, “Analyzing the Relationship between Risk and Trust”, T. Dimitrakos (editor) the Proceedings of the Second International Conference on Trust Management, Oxford, 2004.

[Josang 2005] A. Josang, R. Ismail and C. Boyd, “A Survey of Trust and Reputation Systems for Online Service Provision”, In Decision Support Systems, 2005.

[Josang 1999b] A. Josang, “An Algebra for Assessing Trust in Certification Chains”, In J. Kochmar (editor), Proceedingsof the Network and Distributed Systems Security Symposium (NDSS’99), The Internet Society, 1999.

[Jun 2002] L. Jun, L. Xianlang, H. Hong and Z. Xu, “Application of mobile agent in wide area network”, In IEEE, 2002.

[Kagal 2001] L. Kagal, T. Finin and A. Joshi, “Moving from security to distributed trust in ubiquitous computing environments”. IEEE Computer, December 2001.

[Kagal 2002] L. Kagal, T. Finin and A. Joshi, “Developing secure agent systems using delegation based trust management”. In Fischer, K. and Hutter, D., Eds. Proc. of 2nd Intl Workshop on Security of Mobile MultiAgent Systems (SEMAS'2002), Bologna, Italy, July 2002.

[Karjo 1997] D. Karjoth, D. B. Lange and M. Oshima, “A security model for Aglets”. IEEE Internet Computing, 1(4), pages 68–77, July-August 1997.

[Karjo 1998] G. Karjoth, N. Asokan and C. Gülcü, “Protecting the computation results of free-roaming agents”, In Kurt Rothermael and Fitz Hohl, editors, proc. Of the second International Workshop, Mobile Agents 98, Springer-Verlag Lecture Notes in Computer Science No.1477, pages 195-207, 1998.

[Karjo 2000] G. Karjoth and J. Posegga, “Mobile Agents and Telcos' Nightmares,” Annales des Télécommunications Vol. 55, No. 7/8, pages 29-41, 2000.

[Karnik 1998] N. Karnik, “Security in Mobile Agents Systems”, PhD thesis, Department of Computer Sciences and Engineering, University of Minnesota, Minneapolis, USA, 1998.

[Kini 1998] A. Kini and J. Choobineh, “Trust in Electronic Commerce: Definition and Theoretical Consideration”, Proceedings of 31st International Conference on System Sciences, IEEE, 1998.

[Knudsen 2004] J. B. Knudsen, “Java Cryptography”. O’Reilly 2004

[Kotz 1999] D. Kotz and R. S. Gray, “Mobile agents and the future of the internet”. Operating Systems Review, 33(3), pages 7–13, July 1999.

[Lange 1999] D. B. Lange and M. Oshima, “Seven good reasons for mobile agents”, Commun, ACM 42, 3, pages 88-89, March 1999.

[Ledoux 2000] T. Ledoux and N. M.N.Bouraqadi-Saâdani, “Adaptability in mobile agent systems using reflection,” in ECOOP 2000, Workshop on Reflection and Metalevel Architectures, Cannes, France, 2000.

[Lee 1997] P. Lee and G. Necula, “Research on Proof-Carrying Code on Mobile-Code Security,” In Proceedings of the Workshop on Foundations of Mobile Code Security, 1997.

[Lee 2003] P. Lee. (n.d.), “Proof-carrying code,” Retrieved December 28, 2003, from Web site: http://www-2.cs.cmu.edu/~petel/papers/pcc/pcc.html

[Lenglet 2001] R. Lenglet, "Conteneurs adaptables pour le déploiement de composants" Effectué dans le département DTL/ASR de France, 2001.

[Leriche 2004] S. Leriche, J. Arcangeli, “Une architecture pour les agents mobiles adaptables”. Dana Journées Composants JC’2004, Lille, pages 1-9, 2004.

[Leriche 2006] S. Leriche, J. Arcangeli, “Vers un modèle d'agent flexible,” In : Journées Multi-Agent et Composant”, JMAC'06, Nîmes, mars 2006.

[Libsie 2002] M. Libsie and H. Kosch, “Content adaptation of multimedia delivery and indexing using MPEG-7”, In Proceedings of the tenth ACM international conference on Multimedia (MM-02). ACM Press, New York, pages 644–646, December 2002.

[Lin 2003] C. Lin and V. Varadharajan, “Modelling and Evaluating Trust Relationships in Mobile Agent Based Systems”. In Proceedings of First International Conference on Applied Cryptography and Network Security (ACNS03), Lecture Notes in Computer Science, Vol. ~2846, Springer-Verlag, Kunming, China, pages 176-190, 2003.

[Lin 2004] C. Lin, V. Varadharajan, Y. Wang Y. Mu, “On the Design of a New Trust Model for Mobile Agent Security”, The 1st International Conference on Trust and Privacy in Digital Business (TrustBus04), Lecture Notes in Computer Science, Vol. ~3184, Springer Verlag, Zaragoza, Spain, pages 60-69, 2004.

[Lin 2005] C. Lin, V. Varadharajan, Y. Wang and V. Pruthi, “Trust Enhanced Security for Mobile Agents”. Manuscript, 2005.

[Loureiro 2001a] S. Loureiro, R. Molva and Y. Roudier, "Mobile Code Security," Institut Eurecom, 2001.

[Loureiro 2001b] S. Loureiro, "Mobile code protection", thèse de doctorat, Institut Eurocom, January 2001.

[Lucco 1995] S. Lucco, O. Sharp and R. Wahbe, "Omniware: A Universal Substrate for Mobile Code", in Fourth International World Wide Web Conference, MIT, December,1995.http://www.w3.org/pub/Conferences/WWW4/Papers/165/.

[Luo 2002] C. Luo, “Multi-Layred protection of mobile code”, Fraunhofer Centre for Research in Computer Graphics, Inc, 2002.

[Lyon 1985] B. Lyon, G. Sager, J. M. Chang, D. Goldberg, S. Kleiman, T. Lyon, R. Sandberg, D. Walsh and P. Weiss, “Overview of the Sun network file system”, Technical Report CMU-CS-84-137, Sun Microsystems, Inc., January 1985.

[Maes 1987] M. Maes, “Computational Reflection”. PhD Thesis, Vrije Universiteit, Brussel, 1987.

[Maes 1994] M. Maes, “Modeling adaptative autonomous agent”. Artificial life journal, 1(1 et 2), pages 135-162, 1994.

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

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

[Manchala 1998] D.W. Manchala,“Trust Metrics, Models and Protocols for Electronic Commerce Transactions”. The 18th International Conference on Distributed Computing Systems, 1998.

[Manchala 2000] D.W. Manchala, “E-Commerce Trust Metrics and Models”, IEEE Internet Computer, pages 36-44, 2000.

[Marangozova 1999] Marangozova, “Adaptabilité”, Rapport de magister, 1999.

[McGrath 2001] S. McGrath, D. Chacón and K. Whitebread, “Intelligent Mobile Agents in Military Command and Control,” MILCOM 2001, Vol.1, 2001.

[McGraw 1996] G. McGraw and E. Felten, “Securing JAVA [Electronic version]”. John Wiley and Sons,(1996-9). http://www.securingjava.com

[Mcknight 1996] D.H. McKnight and N.L. Chervany, “The Meaning of Trust”, Technical Report MISRC Working Paper Series 96-04, University of Minnesota, Management Information Systems Research Center, 1996.

[Milojicic 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”, In K. Rothermel and F. Hohl, editors, Proceedings of the 2nd International Workshop on Mobile Agents, volume 1477 of Lecture Notes in Computer Science. Springer-Verlag: Heidelberg, Germany, pages 50–67, 1998.

[Milojicic 2000] D. Milojicic, F. Douglis, Y. Paindaveine, R. Weeler and S. Zhou, “Process migration”. ACM Compting Surveys, 32(3), pages 241–299, September 2000.

[Milojicic 1999] D. Milojicic, F. Douglis and R. Wheeler, “Mobility on the Internet”. In Dejan S. Milojicic, F. Douglis, R. Wheeler, editors, Mobility: Processes, Computers, and Agents. ACM Press, pages 451-456, February 1999.

[Mir 2002] J. Mir and J. Borrell, “Protecting General Flexible Itineraries of Mobile Agents”, in K. Kim (Ed.): ICICS 2001, LNCS 2288. © Springer-Verlag Berlin Heidelberg 2002, pages 382-396, 2002.

[Morris 1994] J.H. Morris, D.J. Moberg, “Work organizations as contexts for trust and betrayal”, In: R. Theodore Sarbin, Ralph M. Carney, Carson Eoyang (Eds.): Citizen Espionage: Studies in trust and betrayal, US. Praeger Publishers/Greenwood Publishing Group, pages 163-187, 1994.

[Mu 2004] Y. Mu, C. Lin, V. Varadharajan, Y. Wang, “On the Design of a New Trust Model for Mobile Agent Security”, Trust and Privacy in Digital Business. Lecture Notes in Computer Science, Vol. 3184. Springer-Verlag, Berlin Heidelberg Germany, pages 60-69, 2004.

[Necula 1998] G. C. Necula, P. Lee, “Untrusted Agents using Proof-Carring Code”.Lecture Notes in Computer Science, Vol. 1419, Springer-Verlag, 1998.

[Nelson 1981] B. J. Nelson, “Remote procedure call”, PhD thesis, Carnegie-Mellon University, 1981.

[Nixon 2003] P. Nixon, S. Terzis (editors), “First International Conference on Trust Management”. Lecture Notes in Computer Science, vol. 2692, Springer, 2003.

[Occello 2002] Audrey Occello, "Composants: Vers une adaptation dynamique cohérente" DEA d'Informatique. Université de Nice Sophia Antipolis. Rapport de stage présenté en juillet 2002.

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

[Phan 2004] D. Phan, “Cognitive Economics, chapter From Agent-Based Computational Economics towards Cognitive Economics”, pages 371-398. Springer Verlag, 2004.

[Picco 1998] G. P. Picco, “Understanding, Evaluating, Formalizing, and Exploiting Code Mobility”. PhD thesis, Politecnico di Torino, Italy, February 1998.

[Perret 1997] S. PERRET, “Agents mobiles pour l’accès nomade à l’information répartie dans les réseaux de grande envergure”, Thèse de Doctorat. Université Joseph Fourier - Grenoble I, 1997.

[Poslad 2000] S. Poslad and M. Calisti, “Towards improved trust and security in FIPA agent platforms”, In Autonomous Agents 2000, June 2000.

[Ratnasingham 2000] P. Ratnasingham, K. Kumar, “Trading Partner Trust in Electronic Commerce Participation”, 2000. http://portal.acm.org/citation.cfm?id=3598

[Reiser 2000] H. Reiser, “Security Requirements for Management Systems using Mobile Agents”. Proceeding of the Fifth IEEE Symposium on Computers and Communications: ISCC 2000, Antibes, France, pages 160-165, 2000.

[Rejeb 2005] L. Rejeb, “Simulation multi-agents de modèles économiques :Vers des systèmes multi-agents adaptatifs”. Thèse de doctorat, Université de Reims Champagne-Ardennes, France, 2005.

[Rhee 2003] Man Young Rhee, “Internet Security”, John Wiley & Sons 2003.

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

[Robles 2002a] S. Robles, J. Mir, and J. Borrell, “MARISMA-A: An architecture for mobile agents with recursive itinerary and secure migration”, In 2nd Information Workshop on Security of Mobile Multiagent Systems, Bologna, July 2002.

[Robles 2002b] S. Robles, “mobile Agent Systems and Trust, a Combined View toward Secure Sea-of-Data Applications”, Submitted to Universitat AutonÒma De Barcelona in partial fulfillment of the requirements for the degree of doctor of philosophy in computer science. July 2002.

[Roth 1998] V. Roth, “Secure Recording of Itineraries Through Cooperating Agents”, Proceedings of the ECOOP Workshop on Distributed Object Security and 4th Workshop on Mobile Object Systems: Secure Internet Mobile Computations, INRIA, France pages 147-154, 1998.

[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.

[Rothemel 1998] K. Rothermel and M. Schwehm, “Mobile Agents”. In: A. Kent and J. G. Williams (Eds.): Encyclopedia for Computer Science and Technology, New York: M. Dekker Inc., New York, 1998.

[Rousseau 1998] M.T. Rousseau, S.B. Stikin, S.B Burt, C. Camerer, “Not so different after all: a crossdiscipline view of trust”. Academy of Management Review, Vol. 23. pages 393-404, 1998.

[Rouverais 2002] S. Rouvrais, “Utilisation d'Agents Mobiles pour la Construction de Services Distribues”. These de doctorat de l'universite de Rennel, France, 2002.

[Rubin 1998] D. Rubin and D. E. Geer, "Mobile code security," IEEE Internet Computing, 1998., Li Gong,”Secure java class loading,” IEEE Internet Computing, pages 56-61, 1998.

[Rutkowska 2006] J. Rutkowska, “Red pill... or how to detect VMM using (almost) oneCPU instruction”, 2006. http://invisiblethings.org/papers/redpill.html

[Sahai 1998] A. Sahai C. and Morin, “Mobile agents for enabling mobile user aware applications”, In Katia P. Sycara and Michael Wooldridge, editors, Proceedings of the 2nd International Conference on Autonomous Agents (Agents’98), ACM Press. pages 205–211, New York, 1998.

[Sander 1998a] Sander, T., Tschudin, C.: Toward Mobile Cryptography. IEEE Symposium Security and Privacy, IEEE Computer Soc. Press, Los Alamitos, California, pages 215-224, 1998.

[Sander 1998b] T. Sander, C. Tschudin, “Protecting Mobile Agent against Malicious Hosts”, G.Vigna (Ed.), Mobile Agents and Security, Lecture Notes in Computer Science, Vol. 1419, ©Springer-Verlag Berlin Heidelberg, Berlin, pages 44—60, 1998.

[Satoh 2002] I. Satoh, “Physical mobility and logical mobility in ubiquitous computing environments”. Lecture Notes in Computer Science, 2535, pages 186–202, 2002.

[Sau-Koon 2000] Ng. Sau-Koon, “Protecting Mobile Agents Against Malicious Hosts”, Thesis submitted for the degree of Master of Philosophy Division of Information Engineering, The Chinese University of Hong Kong, June 2000.

[Sayeb 2002] M. Sayeb, M. Hamza, 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.

[Schillo 2000] M. Schillo, P. Funk, and M. Rovatsos, “Using trust for detecting deceitful agents in artificial societies”. Applied Artificial Intelligence, 14(8), pages 825-848, September 2000.

[Schneider 1997] F. B. Schneider, “Towards fault-tolerant and secure agentry”, In M. Mavronicolas and P. Tsigas, editors, Proceedings of the Eleventh International Workshop on Distributed Algorithms, number 1320 in LNCS. Springer-Verlag, Berlin, pages 1-14, 1997.

[Schneider 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 2001, Pages 86-101, 2001.

[Security 2003] “Security Model for the Next-Generation secure computing Base”. White paper, Microsoft Corporation, 2003.

[Sekar 2001] R. Sekar, C.R. Ramakrishnan, I.V. Ramakrishnan and S.A. Smolka, “Model Carrying Code (MCC)”: A new Paradigm for mobile code security.In Proceedings of 2001 IEEE Workshop on New Security Paradigms, NSPW01, pages 23-30, 2001, ACM Press.

[Shamir 1994] A. Shamir, “Efficient signature schemes based on birational permutations”. In D. R. Stinson, editor, Proc. CRYPTO 93, Springer, Lecture Notes in Computer Science No. 773, pages 1–12, 1994.

[Smith 1998] S.W. Smith, V. Austel, “Trusting Trusted Hardware: Towards a Formal Model for Programmable Secure Processors”. The 3rd USENIX Workshop on Electronic Commerce, 1998.

[Sornn 2001] H. Sornn-Friese. “Learning in firms and markets: Organizational Adaptation and Industry dynamics in the road Haulahe industry in Denmark in the 1990s”. Phd thesis, Copenhagen business school, 2001.

[Swarup 1997] V. Swarup, “Trust Appraisal and Secure Routing of Mobile Agents”, DARPA Workshop on Foundations for Secure Mobile Code, Position Paper. Monterey, CA, USA, March 1997.

[Tan 2001] Tan, H. K., Moreau, L.: 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 2002a] H. K. Tan and L. Moreau, “Extending execution tracing for mobile code security”, In Fischer, K. and Hutter, D., Eds. Proc. of 2nd Intl Workshop on Security of Mobile MultiAgent Systems (SEMAS'2002), Bologna, Italy, pages 51-59, July 2002.

[Tan 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

[Tanenbaum 2004] A. S. Tanenbaum, F. M. T. Brazier, and G. J. Van’t Noordende, “Security in a mobile agent system”, citeseer in line, July 2004. http ://citeseer. ist.psu.edu/705581.html.

[Tclet 1996] Sun Microsystems, "Tcl/Tk: Create Web Apps In Internet Time”, Sun Microsystems Online, August, 1996. http://www.sun.com/960710/cover/

[TCG 2001] Trusted Computing Group | Main Specification, Version 1.1a, 2001.

[Tschudin] C. Tschudin, “Apoptosis - the programmed death of distributed services” In Secure Internet Programming.

[Usunier 1998] Usunier Jean-Claude. “Un examen du concept de confiance à travers la littérature”. In Usunier J.C. (coord.) Confiance et performance, Vuibert, Paris, 1998.

[Vigna 1997] G. Vigna, “Protecting mobile agents through tracing”. In Proceedings of the Third ECOOP Workshop on Operating System support for Mobile Object Systems, Finland, pages 137-153, June 1997.

[Vigna 1998] G. Vigna, edition, “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), pages 298-299. IEEE Computer Society, pages 19-22, January 2004.

[Vitek 1999] J. Vitek and G. Castagna, “Mobile Computations and Hostile Hosts”, In Journées Francophones des Langages Applicatifs (JFLA99), pages 113-132, February 1999.

[Wahbe 1993] R. Wahbe, S. Lucco, T. E. Anderson and S. L. Graham, “Efficient software-based fault isolation,” In Proceedings of the 14th ACM Symposium on Operating Systems Principles, pages 203-216, December 1993.

[Waidner 2002] M. Waidner (editor), “Ercim News, Special Theme: Information Security”, No 49, 2002.

[Wang 2000] T. Wang, S. Guan, T. Khoon Chan, “Integrity Protection for Code-On-Demand Mobile Agents in E-Commerce”, The Journal of Systems and Software 60, pages 211-221, 2000.

[White 1994] J. E. White, “Telescript technology : The foundation for the electronic marketplace”, White paper, General Magic, Inc., 2465 Latham Street, Mountain View, CA 94040, 1994.

[Wilhelm 2000] U. G. Wilhelm, S.M. Staamann, L. Buttyan, “A Pessimistic Approach to Trust in Mobile Agent Platforms”. IEEE Internet Computing, Vol.4, No. 5, ISSN: 1089-7801, pages 40-48, 2000.

[Wilhelm 1998] U. G. Wilhelm, S. Staamann and L. Buttyan, “On the Problem of Trust in Mobile Agent Systems”. IEEE Symposium on Network and Distributed System Security, San Diego, California, 1998.

[Wilhelm 1997] U. G. Wilhelm, “Cryptographically Protected Objects”.A french version appeared in the Proceedings of RenPar’9.
http://lsewww.ep.ch/wilhelm/Papers/CryPO.ps.gz.

[Wilhelm 1999] U. G. Wilhelm, “A technical Approach to Privacy based on Mobile Agents Protected by Tamper-Resistant Hardware”, PhD. dissertation, No. 1961, Swiss Federal Institute of Technology, Lausanne, Switzerland, 1999.

[Yahalom 1993] R. Yahalom, B. Klein, T. Beth, “Trust Relationships in Secure Systems - A Distributed Authentication Perspective”. The Proceedings of IEEE Conference on Research in Security and Privacy, 1993.

[Yee 1995] B. Yee, D. Tygar, “Secure Coprocessors in Electronic Commerce Applications”, The Proceeding of First Usenix Workshop on Electronic Commerce, Usenix Assoc., Berkeley, California, pages 155—170, 1995.

[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.

[Young 1997] A. Young and M. Yung, “Encryption Tools for Mobile Agents: Sliding Encryption,” In: E. BIHAM (ed), Fast Software Encryption. Lecture Notes in Computer Science, no. 1267. Springer-Verlag, Germany, 1997.

[Yunguick 2002] H. Yunguick Lee and J. Alves, “The Use of encrypted functions for mobile agent security”, Proceedings of SAICSIT 2002, pages 141 – 148, 2002.

[Zachary 2003] J. Zachary, “Protecting mobile code in the wild”. IEEE Internet Computing, 7(2), pages 78–82, 2003.

[Zand 1972] Zand, D.E.: Trust and managerial problem solving. Administrative Science Quarterly, Vol. 17, pages 229-239, 1972.

[Zanero 2004] S. Zanero, “Behavioral Intrusion Detection”. In Cevdet Aykanat, Tugrul Dayar, and Ibrahim Korpeoglu, editors, Proceedings of ISCIS2004, volume 3280

of Lecture Notes in Computer Science, Springer, Kemer-Antalya, Turkey, pages 657-666, October 2004.

[Zhi 2004] W. Zhi, G. Zhogwen, “A Dynamic Security Adaptation Mechanism For Mobile Agents”, Proceedings of the International Computer Congress, China, page 334-339, 2004.

Sites Web

[ht1] http://en.wikipedia.org/wiki/Rootkit

[ht2] http://3psilon.info/Les-rootkits.html

[ht3] http://www.rootkit.com

[ht4] “Introduction to Code Signing,” (n.d.), Retrieved December 15, 2003, from Microsoft Corporation, Microsoft Developer Network (MSDN) Web site: http://msdn.microsoft.com/library/default.asp?url=/workshop/security/authcode/intro_authenticode.asp

[ht5] “Signed Code,” (n.d.), Retrieved December 15, 2003, from James Madison University, IT Technical Services Web site:
http://www.jmu.edu/computing/infosecurity/engineering/issues/signedcode.shtml

[ht6] http://www.securiteinfo.com/cryptographie/pki.shtml

[ht7] http://www.eclipse.org/

[ht8] http://fr.wikipedia.org/wiki/Blowfish

Annexe A

Détails de la génération de la clé environnementale

Cet exemple souligne les principales étapes des algorithmes présentés dans les tables 4.2 et 4.3 de la Section 4.3.4. nous avons utilisé, pour cet exemple précis, la fonction de hachage SHA-1 parcequ'elle génère un résultat de 40 caractères plus facile à représenter que celui généré par SHA-256 (caractères)

1. Côté Client

1 Collecter les données:
Identificateur du service = Mobi-Scan
Acronyme = Esprit 7
Date de validité = December 31th,2007

2 Appliquer SHA-1 à chaque donnée :
Identificateur du service = 6654b57274e1bdfc2953324999cbaef0bb470be9
Acronyme = bca3fbec4d267cacddeaa0dcbc091f72d3f32fd7
Date de validité = de01cecfc4fc72d91128210329614afab1ff09ce

3 Concaténer les trois empreintes:
M=6654b57274e1bdfc2953324999cbaef0bb470be9bca3fbec4d267cacddeaa0dcbc091f72d3f32fd7de01cecf
c4fc72d91128210329614afab1ff09ce

4 Encrypter M en utilisant l'algorithme à clé publique RSA (taille de la clé est de 1024). Le résultat obtenu est:

EM=69710f7d78369be7e6823c00be39174cd1eeb4c8c091ddb616380d02d163ba29a046a68d430fba45bb30
b751a81f697d6e254613559f0ccc555e8eea6bfa3e75bb48f96b1771a5ac07444497290460b22b98b97c64b61
d16b5c7ec95e56c9b5d627f7b253b50448d35ed1fa592cb0d5778fbd8cda2643385ecf5aec7e7f4c764

Notons que la taille de EM est de 256 caractères.

5 Signer EM en utilisant l'algorithme DSA :
SM=302c021451b1f4af3cd47c252ba00d184176fab9c7d5cfba0214010a4360324f91b5fcbc791a80d9ab7
bff73fc8d

6 Emettre EM et SM vers le pourvoyeur de service
7 Appliquer le hachage sur M :
$D = \mathcal{H}(M) = 6ec45dd290571c95b03e09037fa628fd9872a7ea$

8 Appliquer D⊕id (où id est par exemple "Ag543hdmk"):
$Ke = D \oplus id = 2fa368e6a33f78f8db7f6e364b954099f519e68d$

Les étapes restantes correspondent à la vérification de SQ et le décryptage de EQ en utilisant la clé environnementale.

2. Côté pourvoyeur de service

1 Recevoir EM et SM :

EM=69710f7d78369be7e6823c00be39174cd1eeb4c8c091ddb616380d02d163ba29a046a68d430fba45b
b30b751a81f697d6e254613559f0ccc555e8eea6bfa3e75bb48f96b1771a5ac07444497290460b22b98b97
c64b61d16b5c7ec95e56c9b5d627f7b253b50448d35ed1fa592cb0d5778fbd8cda2643385ecf5aec7e7f4c7
64

SM=302c021451b1f4af3cd47c252ba00d184176fab9c7d5cfba0214010a4360324f91b5fcbc791a80d9ab7
bff73fc8d

2 Vérifier SM
3 Décrypter EM et obtenir :

M=6654b57274e1bdfc2953324999cbaef0bb470be9bca3fbec4d267cacddeaa0dcbc091f72d3f32fd7de01
cecfc4fc72d91128210329614afab1ff09ce

4 En se basant sur le nombre des paramètres de confiance, la fonction de hachage utilisée (la taille du résumé ici est de 40 caractères) et la position de chaque paramètre, il est possible de récupérer les différentes empreintes:

Position1:
Identificateur du service = 6654b57274e1bdfc2953324999cbaef0bb470be9

Position2:
Acronyme = bca3fbec4d267cacddeaa0dcbc091f72d3f32fd7

Position3:
Date de validité = de01cecfc4fc72d91128210329614afab1ff09ce

Les étapes restantes permettent l'estimation du degré de confiance, la sélection de la QoS appropriée et le calcul de la clé environnementale. Cette dernière est recalculée de la même façon que du côté du client et sera utilisée pour encrypter le QoS sélectionné.

Annexe B

Impact de la pondération des paramètres sur la valeur de la confiance

Nous avons réalisé quelques tests faisant apparaître l'impact des différents facteurs présents dans la formule de la confiance (2).

Le premier test utilise quatre exemples de paramètres de confiance dont deux sont internes (identité et mot de passe) et deux sont externes (RAM et fichier spécifique). Le poids et l'importance attribués à chaque paramètre sont fixés en respectant la contrainte (1).

I	W	S	**Identité**	I	W	S	**Mot de passe**	I	W	S	**RAM**	I	W	S	**Fichier**	**TRUST**
3	**10**	**1**	**30**	**3**	**12**	**1**	**36**	**2**	**10**	**1**	**20**	**1**	**14**	**1**	**14**	**100**
3	10	1	30	3	12	1	36	2	10	0	0	1	14	1	14	**80**
3	10	1	30	3	12	1	36	2	10	1	20	1	14	0	0	**86**
3	10	1	30	3	12	0	0	2	10	1	20	1	14	1	14	**64**
3	10	0	0	3	12	0	0	2	10	1	20	1	14	1	14	**34**
3	10	0	0	3	12	0	0	2	10	1	20	1	14	0	0	**20**

Table A.1. Impact de la non-conformité des paramètres de confiance

Les intervalles d'estimation ainsi que les actions à entreprendre pour chaque intervalle sont aussi établis. Des exemples d'intervalles sont:

- Ne pas executer de service : 0 – 40
- Exécuter un service dégradé : 41 – 70
- Exécuter un service complet : 71 – 100

Notons que plus le dernier intervalle est réduit plus le niveau de sécurité est élevé.

Dans ce premier test (*cf.* Table A.1), nous avons expérimenté l'échec de chacun des facteurs et observé son influence sur la valeur de la confiance et par conséquent sur la réaction à entreprendre. Nous remarquons que plus le poids et/ou l'importance d'un paramètre sont élevés plus son impact dans la valeur de la confiance est grande. A titre d'exemple, la non-conformité des deux paramètres internes (l'identité et le mot de passe) mène à un refus d'exécution du service demandé alors que celle du paramètre externe RAM n'influe que légèrement sur la valeur de la confiance et mène à une exécution complète du service si bien sûr le problème de la taille mémoire est résolu.

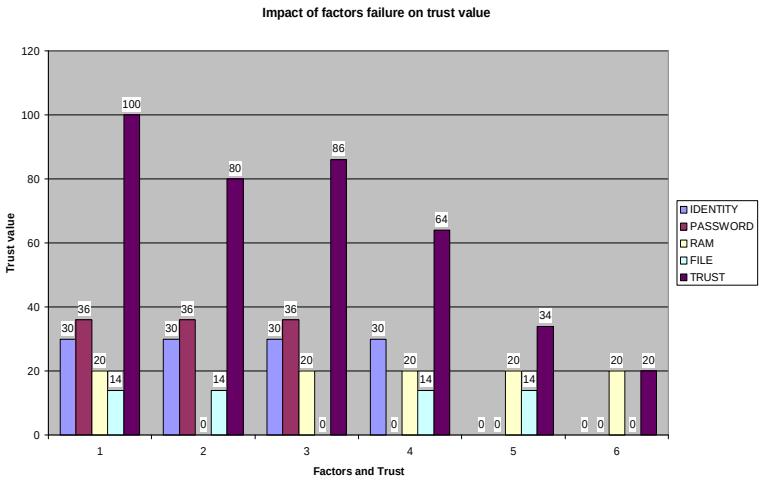

Fig.A.1. Histogramme mettant en évidence l'impact de la non-conformité des paramètres de confiance

Dans le second test, l'importance d'un facteur externe – le fichier spécifique – est modifiée et l'impact qu'elle peut avoir sur la fiabilité (en cas d'échec) est observé.

I	W	S	Identité	I	W	S	Mot de passe	I	W	S	RAM	I	W	S	Fichier	T
3	**10**	**1**	**30**	**3**	**12**	**1**	**36**	**2**	**10**	**1**	**20**	**1**	**14**	**1**	**14**	**100**
3	10	1	30	3	12	1	36	2	10	1	20	1	14	0	0	86
3	10	1	30	3	12	1	36	2	5	1	10	2	14	0	0	76
3	8	1	24	3	8	1	24	2	5	1	10	3	14	0	0	58

Table A.2. Impact de l'importance d'un paramètre externe sur la valeur de la confiance

Nous remarquons que plus nous augmentons l'importance de ce paramètre plus sa valeur augmente (plus nous devons diminuer le poids des autres paramètres pour respecter la contrainte (1)) et plus son impact sur la valeur de la confiance augmente. Cependant, nous notons aussi que la valeur de la confiance appartient toujours aux intervalles d'exécution complète ou réduite du service.

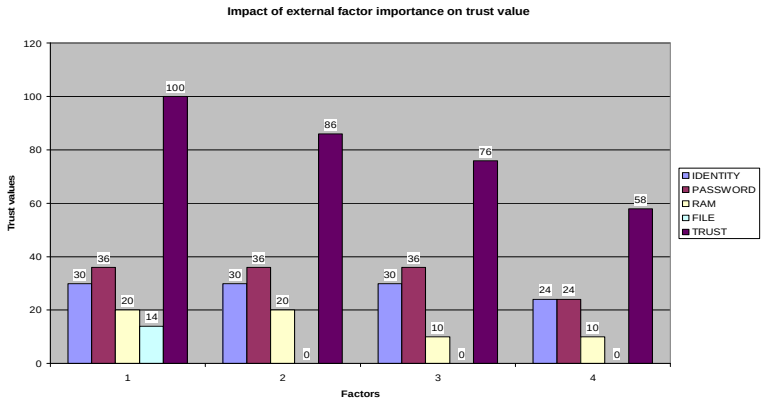

Fig.A.2. Histogramme mettant en évidence l'impact de l'importance d'un paramètre externe sur la valeur de la confiance

Au niveau du troisième test, nous choisissons un paramètre interne de poids fort tel que le mot de passe et nous faisons varier son importance et observons l'impact de son échec. Nous remarquons que plus son importance diminue (nous gardons son poids inchangé et nous modifions le poids des autres paramètres afin de respecter la contrainte (1)) plus la valeur de la confiance augmente et évolue vers l'intervalle d'exécution complète. Nous notons aussi, dans le cas de la dernière ligne de la Table A.2, que si l'importance du paramètre est la plus grande et son poids le plus grand alors son échec a un plus grand impact puisque la valeur de la confiance n'appartient plus aux intervalles d'exécution et engendre un refus d'exécution du service.

I	W	S	Identité	I	W	S	Mot de passe	I	W	S	RAM	I	W	S	fichier	T
3	10	1	30	3	15	1	45	2	8	1	16	1	9	1	9	100
3	10	1	30	3	15	0	0	2	8	1	16	1	9	1	9	55
3	12	1	36	2	15	0	0	2	10	1	20	1	14	1	14	70
3	15	1	45	1	15	0	0	2	12	1	24	1	16	1	16	85
2	10	1	20	3	20	0	0	2	6	1	12	1	8	1	8	40

Table A.3. Impact de l'importance d'un paramètre interne de poids fort sur la valeur de la confiance

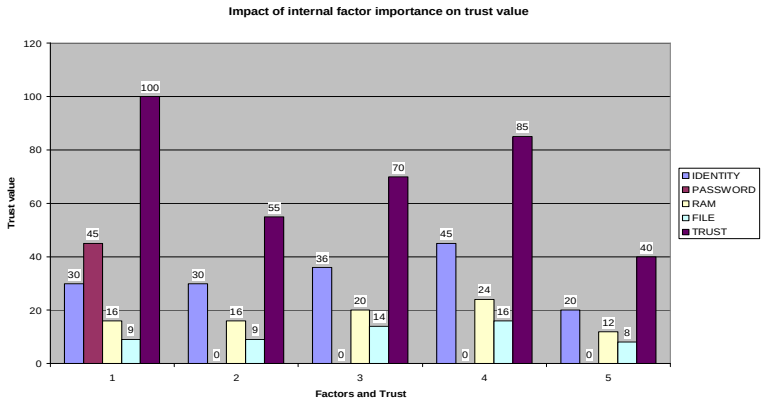

Fig.A.3. Histogramme mettant en évidence l'impact de l'importance d'un paramètre interne de poids fort sur la valeur de la confiance