Les agents mobiles et la sécurité

Hamed Aouadi et Pr Mohamed Ben Ahmed
Laboratoire RIADI, ENSI
Compus universitaire La manouba, Tunisie
Hamed.Aouadi@ensi.rnu.tn

Résumé. Les agents mobiles sont identifiés comme la future plate-forme de base pour l'architecture de services électroniques. Toutefois une telle technologie ne peut connaître une expansion sans assurer la sécurité des agents eux-mêmes. Dans ce papier, nous présenterons l'état de l'art de la sécurisation des agents logiciels mobiles ainsi qu'une approche par nous proposer en vue de garantir cette sécurité. Dans l'approche proposée, un agent sera transformé en un autre agent crypté mais exécutable par une machine logicielle appelée SVM (Secure Virtual Machine) jouant le rôle d'une plate-forme de confiance.

Mots Clés: agents mobiles, mobilité, sécurité, cryptographie, plate-forme de confiance

1 INTRODUCTION

Les agents logiciels mobiles sont identifiés comme la future plate-forme de base pour l' architecture de services électroniques distribués [Hoh 97] grâce aux avantages que présente cette technologie en terme d'économie en ressources réseaux et à la possibilité de poursuivre leurs fonctionnements même hors connexion [Sau 00]. En raison de l'adaptabilité et de l'autonomie qui les caractérisent, les agents mobiles pourront éviter les contraintes d'hétérogénéité des plates-formes cibles.

Un agent mobile est un logiciel particulier, doté de capacités d'adaptabilité, d'autonomie et de mobilité [Wil 98]. Cet agent se déplace à travers le réseau (en particulier l'Internet) d'un site vers un autre dans le but de fournir un service précis à son propriétaire. Toutefois le déplacement à travers le réseau n'est pas toujours sans risque, en effet l'agent peut être attaqué par un autre agent, par une personne connectée au réseau ou au niveau d'un site visité sur lequel l'agent devra s'exécuter [Hoh 97]. Notre travail se propose d'étudier les problèmes de sécurisation des agents logiciels mobiles. Dans cette étude nous présenterons l'état de l'art ainsi que l'approche par nous suggérée.

Définitions :

Un agent :

Définition 1 : On appelle agent une entité physique (réelle) ou virtuelle

a. qui est capable d'agir dans un environnement,
b. qui peut communiquer directement avec d'autre agents,
c. qui est munie par un ensemble de tendances (sous la forme d'objectifs individuels ou d'une fonction de satisfaction, voire de survie, qu'elle cherche à optimiser),
d. qui possède des ressources propres,
e. qui est capable de percevoir (mais d'une manière limitée) son environnement,
f. qui ne dispose que d'une représentation partielle de cet environnement (et éventuellement aucune),
g. qui possède des compétences et offre des services,
h. qui peut éventuellement se reproduire,
i. dont le comportement tend à satisfaire ses objectifs, en tenant compte des ressources et des compétences dont elle dispose, et en fonction de sa perception, de ses représentations et des communications qu'elle reçoit [Fer 97].

Définition 2 : Une entité réelle ou virtuelle, évoluant dans un environnement, capable de le percevoir et d'agir dessus, qui peut communiquer avec d'autres agents, qui exhibe un comportement autonome, lequel peut être vu comme la conséquence de ses connaissances, de ses interactions avec d'autres agents et des buts qu'il poursuit [Dro01].

Définition 3 : Un agent peut être défini comme une entité (physique ou abstraite) capable d'agir sur elle-même et son environnement, disposant d'une représentation partielle de cet environnement, pouvant communiquer avec d'autres agents et dont le comportement est la conséquence de ses observations, de sa connaissance et de ses interactions avec les autres agents [Sta99].

Un agent mobile :

Définition 1 : Un agent mobile est un logiciel qui a les trois propriétés suivantes [Wil 98]

1. C'est un objet (ou un ensemble d'objets) constitué de code et d'état (il est actif et il dispose d'un ou plusieurs threads qui le contrôlent).
2. Il peut (de façon autonome) migrer d'une plate-forme d'agents vers une autre.
3. Il exécute une tâche bien définie par son propriétaire (c'est un utilisateur que l'agent représente) et cherche à accomplir cette tâche sans l'intervention de ce dernier.

Objectifs de sécurisation des agents mobiles :

Pour disposer d une solution efficace d'agents il est nécessaire d'assurer la sécurisation de ces agents tout au long des trajets empruntés ainsi que dans les sites visités. A travers le réseau on doit assurer la confidentialité, l'intégrité et la non-répudiation pour chaque agent. Mais sur site, on doit assurer essentiellement la confidentialité d'exécution qui consiste à garder secrètes les actions que l'agent exécute sur un site donné ainsi que l'intégrité d'exécution et qui peut être définie par l'exécution du code de l'agent comme l'a prévu son propriétaire. En d'autres termes, il s'agit de

donner au propriétaire d'un agent le moyen de vérifier l'utilisation non frauduleuse de son agent par des intrus [Lau 01]. Farmer et al [FGS96a] classent les objectifs de sécurité en : objectifs impossibles à atteindre tels qu'assurer l'exécution de l'agent sur la plate-forme cible, objectifs faciles à réaliser et objectifs possibles mais difficiles à obtenir, par exemple donner au propriétaire de l'agent la possibilité de restreindre sa flexibilité

Etat de l'art de la sécurisation des agents mobiles : Les approches de sécurisation des agents mobiles sont multiples et peuvent être classées suivant le moyen de sécurisation en sécurisation matérielle et sécurisation logicielle.

Sécurité basée sur le matériel :
Ce type d'approche est largement utilisé dans le domaine de la sécurisation des logiciels contre le piratage. Dans ce cas 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 [Hoh 97]. La protection de ce logiciel peut être contournée par l'élimination (ou saut) de la partie test du code de l'agent[Man02].

Sécurisation à base d'un environnement d'exécution de confiance :
Une autre méthode de sécurisation consiste à utiliser un coprocesseur dédié à l'exécution de l'agent. L'agent s'exécute exclusivement à l'intérieur de ce périphérique et ne dialogue avec le site visité qu'à travers une interface sécurisée [Wil98]. Une approche de ce type à été proposée par Wilhelm [Wil98] dans laquelle il a appelé le périphérique environnement d'exécution de confiance TPE (Trusted Processing Environment). Le périphérique TPE est fabriqué sous le contrôle d'une autorité de confiance, dispose d'un certificat et d'une politique de sécurité. Dans cette approche les agents sont cryptés par la machine de leurs propriétaire et ne sont plus exécutés par le site visité mais restent cryptés tout au long de leurs chemins de migration et sur chaque site visité. Quand ils arrivent à leur destination ils sont encore cryptés et accèdent au périphérique TPE où ils seront décryptés puis exécutés [Wil98].

Une telle approche constitue une solution très puissante pour sécuriser les agents sur site d'exécution ainsi qu'à travers le réseau. Toutefois le périphérique TPE n'est pas assez 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 et le problème d'exécution de plusieurs agents en parallèle reste posé

Sécurisation à base de cartes à puces :
L'approche proposée par Mana [Man02] consiste à subdiviser le code de l'agent en sections dont certaines seront cryptées par une clé publique d'une carte à puces. Ces dernières seront remplacées par des appels procéduraux vers la carte. Sur site d'exécution, on transmet à la carte comme arguments les sections cryptées qui seront décryptées puis exécutées à l'intérieur de cette dernière. La clé publique et sa clé complémentaire sont générées à l'intérieur de la carte à puce, la clé publique sera publiée tandis que la clé secrète résidera exclusivement à l'intérieure de la carte et ne sera jamais révélée [Man02].

De cette façon on assure la confidentialité du code et on évite l'exécution de la totalité de l'agent sur le périphérique de sécurisation. En cas d'exécution de plusieurs agents sur site la carte peut jouer le rôle d'une ressource critique par laquelle on gère l'accès par les moyens connus. Malgré la robustesse apparente de l'approche, le code est caché en partie et ce qui reste est lisible ce qui permet la cryptanalyse et la déduction des fonctionnalités du code clair ainsi que du code caché. Après déduction des fonctionnalités des sections cachées, ce qui permet une attaque à boîte noire [BL96]. Les sections cachées peuvent être appelées par un code écrit par celui qui attaque en vue de réaliser des tâches au nom de l'agent et sur le compte de son propriétaire (par exemple la fonction signature peut être exploitée pour signer un document autre que l'agent a choisi).

Sécurisation basée sur le logiciel :
Dans ce type d'approches, on cherche à sécuriser les agents de façon purement logicielle ce qui réduit le coût de la sécurité et facilite sa maintenance. Des approches les plus référencées, nous citons : l'obscurcissement, le calcul par fonction cryptographique, les traces cryptographiques, l'appréciation d'état et les fonctions de validation.

Obscurcissement :
Comme approche nous présentons celle proposée par Fritz Hohl [Hoh97] dans laquelle il suggère une solution de sécurisation de l'agent par son propre code. L'approche consiste à produire d'un agent A un agent B analogue au premier sur le plan fonctionnalités mais ayant un code difficile à analyser [Wil98]. Ceci peut être assurer 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 fiable. Ces règles seront respectées au niveau de l'agent source A et pour passer à l'agent sécurisé B, tout d'abord on génère à partir des variables originelles, de nouvelles variables ayant des noms non significatifs et leur nombre est différent de celui des premiers. 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, on remplace l'appel procédural par le corps des procédures et on utilise la structure de contrôle "GOTO" pour remplacer les autres structures [Hoh97]. Enfin, on peut insérer des fragments de code mort pour améliorer la protection. Seulement il faut se méfier des mécanismes de détection de codes morts. Ces mécanismes 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[Hoh97].

Dans cette approche, une solution logicielle efficace pour sécuriser les agents a été avancée. Toutefois, on peut lui reprocher le manque de fondement théorique. De plus la sécurité de l'agent est temporaire et n'est valide que pour les agents qui transportent des données à courte durée de vie. Autrement, l'agent sera enregistré et analysé lentement afin de déduire les données qu'il transporte. Enfin, on peut reprocher à cette approche l'élimination de l'appel procédural ce qui entraîne la perte de l'utilisation des bibliothèques sur site d'exécution.

Calcul par fonction cryptographique:
L'idée proposée par Sander et Tschudin [ST 98] est la suivante : soit une fonction f matérialisée par un agent A, cette fonction sera cryptée en E(f) qui cache les fonctionnalités et les détails de f. Un programme P(E(f)) sera écrit pour implémenter E(f), ce qui produit un nouvel agent B. L'agent B migre sur un site distant où il sera exécuté sur une donnée x et retourne à son site d'origine qui exécute l'algorithme de décryptage $E^{-1}(P(E(f)))(x)=f(x)$. Au niveau du site distant, on exécute P(E(f))(x) qui cache les détails, ce qui assure la sécurité de l'agent. Cette approche définit un homomorphisme entre l'espace des données en clair et celui des données cryptées tout en se basant sur la fonction PLUS et la fonction MIXED-MULT. Ces deux fonctions sont valides uniquement sur les polynômes ce qui représente la défaillance de l'approche de Sander sur les données ordinaires. De plus, le travail de l'agent se limite au calcul d'un résultat sur un site distant puis l'agent retourne vers son site d'origine, ce qui ne permet pas la négociation sur site ni la possibilité de signature ou de prise de décision.

Traces cryptographiques :
Dans cette approche, chaque site visité par l'agent, génère une trace d'exécution de ce dernier. Cette trace contient chaque ligne de code exécutée ainsi que toutes les valeurs externes lues par l'agent [VIG98]. Avant la migration de l'agent vers sa nouvelle destination, le site calcule par fonction de hachage (exemple MD5 ou SHA [FIPS180]) une représentation condensée de la trace, la signe et l'accompagne avec l'agent mobile vers le site suivant. Afin d'alléger son approche, l'auteur divise le code de l'agent en segments blancs et en segments noirs. Les segments noirs sont ceux qui résultent des interactions avec la plate-forme visitée. Et au lieu de la trace d'exécution du code entier, on ne génère que la trace relative aux segments noirs [Lau 01]. Après retour de l'agent, son propriétaire aura une trace de l'exécution de l'agent sur le dernier site et l'adresse du premier site visité (vers lequel il a envoyé l'agent). De cette façon s'il a une doute sur un site donné, il pourra suivre le chemin de l'agent et disposer de la trace d'exécution sur ce site. Le propriétaire de l'agent pourra simuler son exécution sur le site douteux et comparer la simulation avec la trace reçue[Vig98].

Cette approche permet d'assurer la non répudiation et la détection de toute manipulation de l'exécution de l'agent après son retour. Toutefois l'approche reste détective et n'évite en aucun cas l'utilisation frauduleuse de l'agent. De plus, il faut avoir de bonnes raisons de douter d'un site particulier pour lancer le mécanisme de trace sinon l'utilisation systématique de ce mécanisme devient assez coûteuse.

Appréciation d'état :
Dans cette approche Farmer et al. [FGS96a] 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 suppose l'existence d'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 arrive sur un site donné et permet de vérifier la stabilité de l'état de l'agent. L'existence de telles fonctions 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.

Une amélioration a été proposée par Jansen [Jan01][Jan2 01] [Jan3 00] . Elle 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 X.509[ISO9594-8]. Dans le certificat on définit les droits et les responsabilités d'un agent sur un site donné. On distingue les certificats d'attributs dans lesquels on définit le comportement de l'agent et les certificats de politique servant à définir 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 propriétaire mais n'offre aucune protection des données que l'agent transporte.

Fonction de validation :
Les approches à base de fonctions de validation ont été proposée par Blum [Blu88] afin de vérifier la fiabilité d'un code. L'approche qui consiste à vérifier la validité du résultat retourné par le code peut être exploitée dans le cas d'agents mobiles dans le but de vérifier sur la base du résultat reçu l'intégrité d'exécution de l'agent sur un site distant. L'approche se base sur la propriété de réductibilité de la fonction qu'implémente l'agent. La propriété de réductibilité permet selon Yao [Yao90] de vérifier certaines relations entre les résultats d'un agent qui implémente une fonction f appliquée à un point x_0, si ces relations ne sont pas vérifiées alors l'exécution de l'agent a été manipulée, en cours de route.

Une approche plus récente proposée par Laurero [Lau 01] qui définit une fonction de vérification V comme suit : Soient un programme P qui implémente une fonction f , Y l'ensemble des résultats possibles de $f(x)$, x, avec x appartenant 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.

Comparaison entre les différentes approches :
La comparaison entre les approches précédemment présentées sera effectuée en fonction des critères relatifs à la solidité de la sécurité apportée aux agents et de ceux relatifs au coût de cette sécurité. Comme critères relatifs à la sécurité, nous examinerons tout d'abord les apport de chaque approche en terme de ***la confidentialité d'exécution*** du code mobile sur chaque site visité en suite nous examinerons est ce que l'approche en question assure ***l'intégrité d'exécution***. Une telle intégrité sera-t-elle assurée tout au long du voyage de l'agent ou juste vérifiée après son retour. Concernant les coûts supportés par l'approche de sécurité, nous nous intéresserons à l'augmentation de la ***taille du code***, à l'influence de l'approche de sécurité sur le ***temps d'exécution*** de l'agent et essentiellement aux ***communications additionnelles*** à travers le réseau en raison des coûts de ces dernières et particulièrement les vulnérabilités qu'elles peuvent engendrer.

Critères / Approches	la confidentialité d'exécution	l'intégrité d'exécution	taille du code	temps d'exécution	Communications additionnelles	Remarques
Environnement d'exécution de confiance	forte	forte	Non modifiée	Dépend du TPE	Aucune	Les performances du site dépendent du TPE qui est difficile à maintenir
Cartes à puces	forte	forte	Ajout d'instructions d'appel de la carte	Faiblement ralentit à cause de l'accès à la carte	Aucune	Le site visité doit disposer de toutes les cartes relatives aux agents qu'il accueilli
Obscurcissement	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	Le manque de base théorique ne met pas en évidence la robustesse de l'approche
Calcul par fonction cryptographique	Forte	forte	Relativement acceptable	Relativement acceptable	L'agent doit retourner à son site après chaque offre	L'approche se limite aux agents implémentant une fonction polynomiale
Traces cryptographiques	Non assurée	Vérifiable	Relativement acceptable	Relativement acceptable	Aucune	L'approche se limite à vérifier la bonne exécution de l'agent
Appréciation d'état	Non assurée	Partiellement assurée	Relativement acceptable	Relativement acceptable	Aucune	Malgré sa faiblesse l'approche permet de vérifier les intentions réelles du propriétaire d'un agent
Fonctions de validation	Non assurée	Vérifiable	Relativement acceptable	Relativement acceptable	Aucune	L'approche se limite à vérifier la bonne exécution de l'agent

Notre approche :

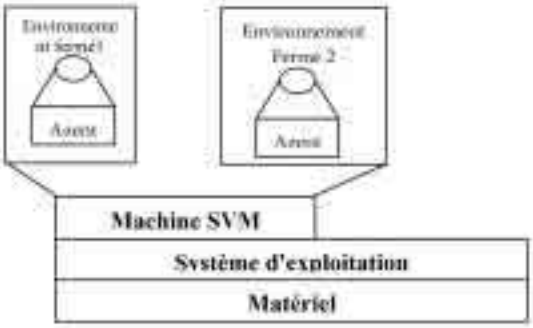
Dans l'approche proposée nous cherchons à remplacer l'environnement d'exécution de confiance TPE proposé par Wilhelm par une solution logicielle. Ainsi, nous bénéficions d'une solution de sécurisation solide (analogue à celle que présente un périphérique), à des coûts abordables et avec des performances meilleures (possibilité de parallélismes). Pour cela nous proposons un environnement logiciel qui se charge d'exécuter l'agent. Cet environnement sera matérialisé par un interpréteur assimilé à une machine virtuelle. Un agent sera tout d'abord écrit d'une façon ordinaire, avec un langage ordinaire puis un transformateur (sorte d'obscurcisseur) génère dans le langage de la machine virtuelle, un nouvel agent analogue au premier de point de vue fonctionnalités, crypté et interprétable en fonction d'une clé publique par la machine virtuelle appelée SVM (Secure Virtual Machine).

Avant son exécution, des clones seront créés à partir de l'agent et envoyés vers les autres sites à visiter. De cette façon, nous augmentons la fiabilité par répartition de la tâche sur plusieurs agents, nous augmentons l'efficacité du système par l'exécution en parallèle de plusieurs agents et nous protégeons la réponse qui ne sera plus rapportée par un seul agent visitant tous les sites, mais chaque clone transporte un fragment de la liste et par suite un site ne peut pas déterminer la meilleure offre reçue par le système globalement. Dans la suite nous présenterons les détails de notre approche.

La machine virtuelle SVM:

La machine virtuelle SVM est un logiciel installé (il peut être transporté avec l'agent pour être installé sur le site visité si ce dernier ne contient pas cette machine virtuelle) sur les sites qui seront visités par l'agent comme couche

entre le système d'exploitation et les agents visiteurs. Ce logiciel aura pour rôle de recevoir l'agent totalement crypté et créer une instance SVM propre à cet agent. L'instance se charge d'exécuter l'agent qui reste crypté dans un espace mémoire alloué proprement à
cette instance SVM qu'on appellera environnement fermé. Dans son environnement fermé l'agent sera exécuté suivant un chemin d'exécution défini par la clé publique de l'agent. Au cours de son exécution l'agent contient une (plusieurs) instruction(s) qui permet sa migration et le vidage de l'environnement fermé. Le vidage de l'environnement fermé se fait suite à la demande de migration de l'agent ou à la détection d'une tentative de manipulation ou à l'expiration de la durée de vie de l'agent. La notion d'environnement fermé permettra ainsi la confidentialité de l'agent et son indépendance des autres agents de façon à simuler une exécution de chaque agent par machine (physique) autonome. En cas d'agent de recherche d'information sans prise de décision, chaque offre sera cryptée par la clé publique du propriétaire de l'agent avant d'être insérée dans la liste.



La machine virtuelle SVM

Le transformateur de programmes :
Le transformateur a pour objectif de générer un agent crypté C-A depuis un agent ordinaire A. Le principe consiste à segmenter l'agent A et à crypter différemment les segments. On utilisera un cryptage symétrique (vue l'économie de ressources et de temps que présente un cryptage symétrique) au niveau des différents segments dont la clé change d'un segment au segment suivant. Un segment S_i assure en plus de ses fonctions le décryptage et le saut vers le segment S_{i+1}, sachant que S_{i+1} est le successeur de S_i sur le plan d'exécution et non par rapport à l'emplacement en mémoire. Avant de procéder au cryptage symétrique, on appliquera l'approche proposée par Hohl [Hoh97] vue les performances qu'elle offre sans augmenter considérablement la taille des agents ou leur temps de réponse. Ainsi, on procédera tout d'abord à la création de nouvelles variables à partir des variables originales comme présenter dans le paragraphe « Obscurcissement », on remplacera en suite chaque appel procédural par le corps de la procédure (ou fonction, ou méthode…) et enfin on insérera des fragments de codes morts. En plus de cette technique, on insérera à l'intérieur du code de l'agent des instructions d'auto-destruction. Ces dernières seront exécutées suite à l'expiration de la durée de vie de l'agent, de la manipulation de sa date de création ou d'une éventuelle détection de la manipulation de son code par des intrus. Après ces opérations on décomposera le corps de l'agent en segments S_i de tailles aléatoires, on procèdera au cryptage symétrique des différents segment comme présenté précédemment et nous réécrivons ces segments dans un nouvel ordre. Seul le premier segment S_1 sera crypté par la clé secrète de l'agent. Enfin, nous "compilons" l'agent dans le langage interprétable par notre machine SVM.

Conclusion :

Dans cette étude de l'état de l'art, nous avons défini les agents logiciels et nous avons expliqué la notion de mobilité tout en mettant l'accent sur les problèmes de sécurité de tels agents mobiles tout en présentant les deux grandes approches proposées à savoir :

- les approches basées sur le matériel et
- les approches basées sur le logiciel.

Notre étude a montré que les approches basées sur un périphérique constitue une solution solide mais coûteuse et difficile à maintenir. Tandis que les approches logicielles sont faciles à maintenir et leurs coûts sont réduits, toutefois

elles restent moins robustes. Dans notre travail, nous cherchons une solution logicielle qui offre un degré de sécurité aussi solide que les approches basées sur le matériel. Pour cela, nous avons conçu une machine virtuelle SVM qui se charge d'exécuter l'agent sans le décrypter et de créer des clones de cet agent qui seront envoyés vers de nouvelles destinations. Notre approche consiste à une amélioration du travail de Hohl qui propose une approche de sécurisation facile à mettre en œuvre, sans augmenter considérablement la taille des agents ni augmenter l'occupation des ressources matérielles. De plus, l'approche de transformation est irréversible, ce qui protège l'agent contre une analyse suite à une opération de reverse engineering. Dans la suite de notre travail, nous chercherons à renforcer encore notre approche de cryptage solide en nous basant, par exemple, sur le calcul par fonction cryptographique. De plus, nous cherchons à utiliser des agents qui sécurisent l'agent principal en exploitant le principe d'agents gardes de corps ou de leurres.

Bibliographie

[Fer 97] Jacques Ferber, "Les systèmes multi-agents", InterEditions 1997.

[FGS96a] : Farmer, William; Guttmann, Joshua; Swarup, Vipin: "Security for Mobile Agents: Issues and Requirements", in: Proceedings of the National Information Systems Security Conference (NISSC 96), 1996.

[Hoh 97] : Fritz Hohl, "An approach to solve the problem of malicious hosts", www.informatik.uni-stuttgart.de

[Wil 98] : http://lsewww.epfl.ch/~wilhelm/thesis/

[Dro01] : Alexis Drogoul,"Systèmes multi-agents", Laboratoire LIP6 Paris France, cours 2001-2202.

[Sta99] : Stan Franklin, "realizing consciousness in software agents", Thèse de doctorat, Université de Memphis, Décembre 1999.

[Sah 98] : N. Sahli, "A synthesis of the state of the art in Internet Security and guidelines for developing countries", African 1998.

[Sta&S96] : David J. Stang, Silvia Moon, "Sécurité réseaux", Dunod Paris 1996.

[Bal01] : Jai Sundar Balasubramaniyan, Jose Omar Garcia-Fernandez, David Isacoff, Eugene Spafford, Diego Zamboni, COAST Laboratory Purdue University, West Lafayette, IN 47907-1398

[BL96] : Dan Boneh et Richard J.Lipton, "Algorithms for black-box fields and their application to cryptography (extended abstract) ", Advances in cryptology, CRYPTO'96, volume 1109 of Lecture notes in computer sciences, pages
283-297, Août 1996.

[Mar02] : Antonio Maña, Ernesto Pimentel, "An efficient software protection scheme", University of Málaga, Spain, 2002.

[VIG98] : G. Vigna. Cryptographic traces for mobile agents. In 'Mobile Agents and Security' [Vig98b], pages 137{153.

[FIPS180] : Federal Information Processing Standard, 'Secure Hash Standard' FIPS PUB 180, Mai 1993.

[ST98] Tomas Sander and Christian Tschudin. Towards mobile cryptography. Proceeding1998 IEEE symposium on security and privacy, pp. 215-224, Oakland, California, May 1998.

[Lau 01] Sergio Laureiro, Mobile code protection, thèse de doctorat, Institut Eurocom, Janvier 2001.

[Blu 88] : Manuel Blum et Sampath Kanan, "Designing programs to check their work", Technical report TR-88-009, International Computer science Institute, Décembre 1988.

[Jan01] : Determining Privileges of Mobile Agents, Proceedings of the Computer Security Applications Conference, December 2001. Wayne Jansen.

[Jan2 01] : A Privilege Management Scheme for Mobile Agent Systems, First International Workshop on Security of Mobile Multiagent Systems, Autonomous Agents Conference, May 2001. Wayne Jansen.

[Jan3 01] : Countermeasures for Mobile Agent Security, Wayne A. Jansen, National Institute of Standards and Technology, Gaithersburg, MD 20899, USA, October 2001.

[ISO9594-8] : ITU-T Recommendation X.509 | ISO/IEC 9594-8: Information Technology – Open Systems Interconnection - The Directory: Public Key and Attribute Certificate Frameworks, March 2000.

[Sau 00] Sau-Koon-Ng, "Protecting Mobile Agents Against Malicious Hosts", A thesis for the degree of Master of Philosophy, Division of Information Engineering The Chinese University of Hong Kong, June 2000.