RAPPORT DE PROJET DE FIN D'ETUDES

GRENOBLE INP Esisar 2020/2021

Titre du projet

Sécurisation des communications dans un système multi-agents embarqués

Nom et adresse de l'entreprise

LCIS (Laboratoire de Conception et d'Intégration des Systèmes)
50, rue Barthélémy de Laffemas
BP54 26902 VALENCE Cedex 09 France

Nom et prénom de l'étudiant

Adrian BONNET

Dates du stage	01/02/2021 - 09/07/2021
Spécialité	IR&C
Tuteur Entreprise	Oum-El-Kheir AKTOUF
Tuteur ESISAR	Emmanuel BRUN

RAPPORT DE PROJET DE FIN D'ETUDES

GRENOBLE INP Esisar 2020/2021

Mots clés :

Système multi-agents embarqués, sécurité, gestion de clés, environnement décentralisé, Blockchain

Résumé :

Un système multi-agents embarqués est un système distribué dans lequel les agents sont des entités physiques avec des capacités limités. Pour réaliser la tâche à laquelle le système est assigné, les agents doivent coopérer entre eux, ce qui implique parfois d'établir des communications entre les agents. Ces communications forment alors une vulnérabilité vis-à-vis de la sécurité du système dans sa globalité. La sécurisation que l'on cherche à apporter se focalise sur la confidentialité et l'intégrité des échanges. Nous cherchons entre autres à prémunir le système de l'écoute intempestive et de la modification de ces communications.

Keywords:

Embedded multi-agent system, security, key management, decentralized environment, Blockchain

Abstract:

An embedded multi-agent system is a distributed system in which the agents are physical entities with limited capabilities. To perform the task assigned to the system, the agents must cooperate with each other, which sometimes involves establishing communications between the agents. These communications thus constitute a vulnerability regarding the security of the system as a whole. The security that we aim to provide focuses on the confidentiality and integrity of exchanges. Among other things, we seek to protect the system against eavesdropping and modification of these communications.

Introduction

Grâce à l'émergence de nouvelles technologies en termes d'intelligence artificielle, les systèmes multi-agents se répandent dans des domaines d'application de plus en plus nombreux. Cette diversité pourra notamment entraîner dans les années à venir l'apparition de systèmes multi-agents embarqués communicants dans des contextes ouverts, où la coopération peut se faire entre des agents qui suivent des implémentations différentes.

Dans un tel système, les communications entre les agents peuvent constituer un point de défaillance, au niveau des agents comme au niveau de la globalité du système. Ainsi, d'éventuels attaquants peuvent tenter d'interagir avec le système, de l'intérieur ou de l'extérieur, dans le but de perturber son fonctionnement. La sécurité des échanges est donc primordiale afin que la tâche assignée aux agents du système puisse être accomplie malgré la contrainte que représentent les attaques.

Le présent rapport rend compte du stage que j'ai effectué au sein du LCIS sur la problématique de la sécurité des systèmes multi-agents embarqués. Mon travail a consisté en les spécifications et le développement dans une simulation d'une solution de sécurité pour les systèmes multi-agents embarqués, afin de conférer à ces systèmes une robustesse vis-à-vis de la confidentialité et de l'intégrité des communications.

La contrainte forte de décentralisation du système m'a amené à investiguer une solution de sécurité basée sur la Blockchain.

Dans le premier chapitre, nous allons introduire le contexte, les enjeux et les livrables attendus. Le deuxième chapitre détaille le cadre scientifique et les objectifs techniques. Les quatre chapitres suivants relatent ma contribution. Dans le chapitre trois, nous précisons l'état de l'art qui couvre les solutions de sécurité, les modèles d'attaque et les outils de simulation existants. Cet état de l'art m'a permis d'identifier des éléments de base pour mettre en œuvre ma solution dont la modélisation et la conception se trouvent en chapitre quatre. Ce rapport se termine par deux chapitres sur l'implémentation et l'expérimentation de la solution.

Remerciements

Je souhaite remercier en premier Oum-El-Kheir AKTOUF, Arthur BAUDET et Annabelle MERCIER pour m'avoir donné la possibilité de faire ce stage ainsi que pour leur confiance.

Je remercie ensuite les autres stagiaires du laboratoire, ainsi que Caroline PALISSE et Nathalie FRANCK pour avoir contribué à une ambiance un peu plus normale en cette période si particulière.

Enfin, je remercie mes amis les plus proches pour leur aide et leur soutien au quotidien, que ce soit pour des conseils sur mon sujet ou dans un cadre plus privé.

Sommaire

Liste des figures

Liste des tableaux

1 Contexte

1.1 Structure d'accueil

Ce stage a été réalisé au Laboratoire de Conception et d'Intégration des Systèmes (LCIS) à Valence. Le LCIS est un laboratoire de recherche fondé en 1996. Le laboratoire partage ses locaux avec l'école d'ingénieurs Grenoble INP - Esisar, et fait partie de l'établissement Grenoble INP et de l'Université Grenoble Alpes (UGA).

Le LCIS est une structure qui comporte à ce jour 72 personnes, dont 7 Professeurs des Universités, 19 Maîtres de Conférences, 3 Post-doctorants, 19 Doctorants et 15 stagiaires, répartis en trois équipes de recherche :

CTSYS sous la responsabilité de David HÉLY, dans le domaine de la sécurité des systèmes embarqués et distribués critiques ;

CO4SYS sous la responsabilité du Pr Jean-Paul JAMONT, dans la modélisation, la commande et la supervision des systèmes complexes ouverts et décentralisés ;

ORSYS sous la responsabilité de Étienne PERRET, dans les solutions et systèmes RF sans-fil communicants et cognitifs.

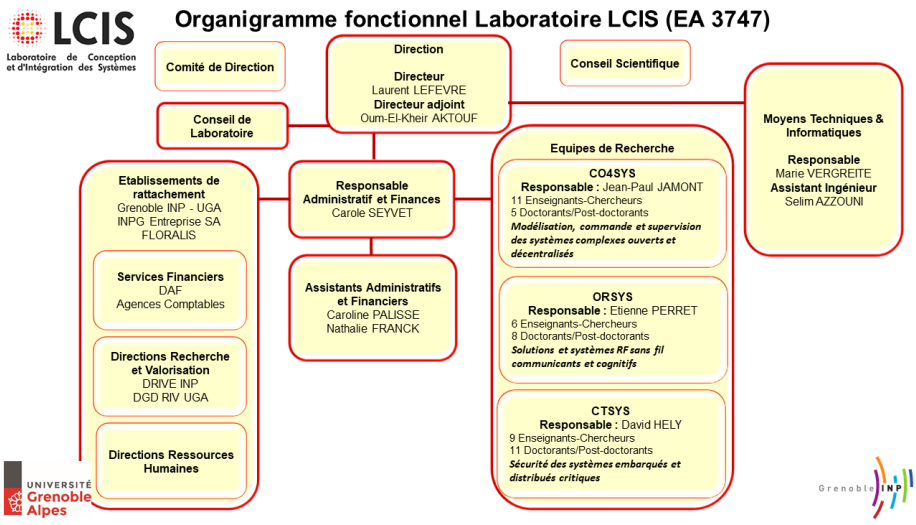

FIGURE 1 – Organigramme du LCIS

Le Pr Laurent LEFÈVRE et Oum-El-Kheir AKTOUF sont respectivement directeur et directrice adjointe du laboratoire.

En lien avec les thématiques de leurs équipes, les chercheurs du laboratoire peuvent travailler divers projets, que l'on peut en général différencier entre la recherche fondamentale dont le but n'est autre que publier les résultats des travaux et ainsi faire gagner en compétences le laboratoire, et le transfert de

technologies pour lequel le laboratoire travaille de concert avec des industriels dans le but de mettre en œuvre une solution innovante à un problème spécifique.

1.2 Enjeux du stage

Le sujet de ce stage a été écrit par Arthur BAUDET dans le cadre de sa thèse sur la sécurité des systèmes multi-agents embarqués. Le stage s'inscrit dans une logique de recherche fondamentale sur le sujet. Les travaux donneront lieu à une publication scientifique dans un workshop, publication qui pourra être citée par Arthur dans sa thèse.

Le sujet se positionne à mi-chemin entre les équipes CTSYS et CO4SYS, puisqu'il est question à la fois de sécurité et de système multi-agent.

1.3 Livrables attendus

Les livrables attendus pour la fin du stage sont un rapport d'état de l'art sur les solutions légères de sécurisation des communications dans un système multi-agent embarqués, les spécifications et le développement de la solution proposée et enfin l'évaluation de cette solution via des simulations.

L'état de l'art devra comprendre une rapide mise en contexte sur les problématiques de confidentialité et d'intégrité des données de manière globale, ainsi qu'une définition précise du cadre de systèmes multi-agents embarqués dans lequel se positionne le sujet du stage, puis mettra en avant différentes solutions à ces problématiques, en les regroupant selon des fonctionnements ou des mécaniques communes et en décrivant pour chaque solution les avantages et les inconvénients que peut représenter leur mise en place. L'état de l'art inclura également une énumération exhaustive des attaques qui peuvent cibler les systèmes multi-agents embarqués dans le monde physique, puis un modèle d'attaque qui posera les hypothèses de sécurité pour l'évaluation de la solution et retiendra plusieurs des attaques précédemment listées en justifiant ces choix. Enfin, l'état de l'art se terminera sur une veille technologique complète sur les outils de simulation de systèmes multi-agents, allant jusqu'à tester les outils qui semblent les plus prometteurs suite à une veille technologique par des critères établis au préalable, dans le but de fournir au laboratoire une vision sur l'état actuel des technologies pour son travail à venir sur les systèmes multi-agents.

Le choix de la solution de sécurité à l'issu de l'état de l'art devra être motivé, et les spécifications concernant son fonctionnement précis seront rédigées. Le développement de la solution sur un système multi-agent embarqués préalablement implémenté fait également partie des livrables du stage.

Pour finir, des agents malveillants seront ajoutés dans le système précédemment implémenté, en accord avec le modèle d'attaque défini dans l'état de l'art, et l'évaluation de la solution de sécurité proposée sera faite selon la robustesse du système face à ces attaques, mesurée grâce à des métriques qui devront être définies à l'avance.

1.4 Planning prévisionnel

Le planning du projet, décrit en figure 2, est fait en adéquation avec les livrables attendus. Ainsi, après un mois de prise en main et d'orientation du sujet, on retrouve un état de l'art d'un peu plus d'un mois, comprenant la mise en contexte du projet, l'étude des différentes solutions à envisager et leurs adaptations possibles et le modèle d'attaque utilisé dans la suite. À cet état de l'art se greffe la veille techologique des outils de simulation pour une durée de deux semaines. Suite à cela, on rentre dans le développement de la solution de sécurité, pratiquement jusqu'à la fin du stage, ce qui comprend l'implémentation du système multi-agent, l'implémentation de la solution de sécurité puis les comportements malveillants des attaquants. Le rapport de stage sera rédigé en parallèle du développement, pour une durée d'un mois précédant la date de rendu. En parallèle au développement et en fin de stage, la dernière semaine est réservée à la rédaction de la contribution scientifique et au transfert du travail effectué au laboratoire.

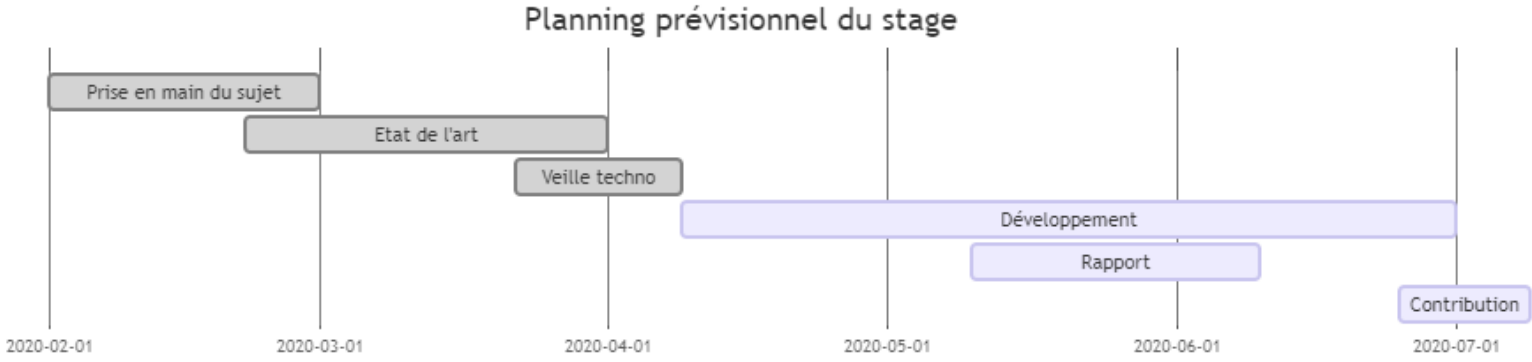

FIGURE 2 – Planning prévisionnel

2 Sujet du stage

2.1 Définitions

Un système multi-agent est un système composé de plusieurs entités, nommées agents, dont le but est d'accomplir une tâche propre au système tout en particiant à un but collectif. Les agents possèdent une vision locale de l'environnement du système et ont la possibilité d'interagir entre eux, de différentes manières selon le système. Les qualités principales des agents sont l'autonomie et la proactivité, c'est-à-dire qu'ils sont capables de prendre seuls des décisions en fonction de leur environnement. Le système est décentralisé, il est capable d'accomplir la tâche à laquelle il est assigné grâce à la coopération entre ses agents.

Par exemple, la simulation d'une foule pour des effets spéciaux, la simulation d'une fourmilière ou d'un banc de poissons ainsi que le problème de recherche du plus court chemin sont des situations pour lesquelles les systèmes multi-agents sont particulièrement efficaces.

On parle d'un système multi-agent embarqués lorsque les agents du système possèdent des capacités restreintes, comme cela peut être le cas sur des cartes embarquées. Ces restrictions sont notamment une capacité de calcul et une mémoire limitées, ainsi qu'un fonctionnement sur batterie, ce qui implique une durée de vie limitée. Nous ajouterons à cette définition la communication sans-fil entre les agents. On suppose donc que les agents du système doivent s'envoyer des messages à intervalles réguliers afin d'accomplir leur tâche. Ces messages sont transmis via une porteuse sans-fil, les agents possèdent donc une portée, c'est-à-dire une distance maximale d'émission, et les messages sont nécessairement retransmis à l'ensemble des voisins à portée de l'agent émetteur.

Les véhicules autonomes, les réseaux de capteurs ainsi que les flottes de drones pour la détection de feux de forêt sont autant d'exemples d'utilisation de tels systèmes multi-agents embarqués.

Dans notre cas, nous aurons recours à la simulation pour faire fonctionner le système multi-agent embarqués. Cette simulation se fera indépendamment d'une implémentation matérielle, un soin particulier est apporté à l'implémentation des problématiques liées aux contraintes de l'embarqué tel que nous l'avons défini.

2.2 Propriétés du système multi-agent

Comme exemple d'application concrète, nous pouvons nous placer dans le cas d'une application de surveillance du trafic urbain aérien de drones dans une ville innovante. Chaque drone possède une tâche qui lui est propre, qui peut être une livraison ou une mission de surveillance par exemple. L'objectif de l'application dans son ensemble est le contrôle de l'ensemble du trafic, afin de répartir la population de drones sur toute la zone, pour éviter des incidents tels des collisions dans les zones denses ou l'absence de service dans les zones peu denses.

Le système multi-agent que nous cherchons à modéliser est donc un système dans lequel les communications se font sans-fil [JOL10] [Jam05]. Nous définissons les propriétés du système suivantes :

décentralisé le système ne contient pas d'entité centralisée ;

embarqué tous les agents du système ont des ressources limitées mais suffisantes en terme de quantité d'énergie, de mémoire et en capacité de calcul si elles sont justifiées ;

mobilité [1] les agents sont mobiles dans le système, leur voisinage peut être amené à changer ;

sans-fil les agents communiquent entre eux sans-fil, ce qui implique que chaque agent possède une distance maximale d'émission, et que les messages sont diffusés à tous les voisins de l'agent ;

dynamique la structure des organisations est dynamique et auto-gérée, c'est-à-dire que l'architecture du système apparaît par émergence ;

ouvert les agents proviennent de fournisseurs qui peuvent ne pas se connaître et ils peuvent apparaître et disparaître du système à tout moment ;

interaction le système permet aux agents d'interagir en suivant des protocoles communs.

En particulier, on suppose que le système multi-agent possède déjà un système de gestion de confiance, ou Trust Management System (TMS) [DJM+19], dont le but est de déterminer quels agents du système sont malveillants. On cherche donc à réaliser une couche de sécurité sur laquelle pourra reposer le TMS. Nous pouvons maintenant faire abstraction des cas d'étude, pour appliquer une solution de sécurisation des communications à un système multi-agent qui respecte les propriétés de ces systèmes.

2.3 Problématique de sécurité

On cherche à assurer la **confidentialité** et l'**intégrité**, c'est-à-dire qu'aucun agent, à part les deux interlocuteurs, ne connaît le contenu des communications et qu'il n'est pas possible que ce contenu soit modifié par un tiers. Pour cela, on cherche à mettre en place un secret partagé entre chaque paire d'agents, généralement un mot de passe ou une clé secrète, qui permet de faire du chiffrement symétrique, afin de rendre incompréhensible le contenu des échanges pour les autres agents. Le chiffrement utilisé est symétrique car il est moins coûteux en nombre d'opérations que le chiffrement asymétrique, en raison de la taille des clés manipulées dans la cryptographie asymétrique qui sont bien plus grandes. On cherche à faire en sorte que les secrets partagés soient négociés après l'initialisation du système.

Afin d'établir ce secret, les agents utilisent chacun des attributs publics et privés car ils n'ont aucune connaissance préalable commune sur laquelle se baser. À partir de ses attributs, il est possible de générer un secret partagé sans avoir

1. il est question ici de mobilité géographique

besoin de négocier celui-ci entre les deux entités, grâce à ce qu'on appelle un échange de clés Diffie-Hellman, ou *DH* [Alv+17]. Cet échange de clés, vastement utilisé sur le web, permet à deux interlocuteurs de se mettre d'accord sur une clé de chiffrement symétrique sur la seule base d'un échange de clés publiques.

Cependant, l'algorithme de Diffie-Hellman ne permet pas d'assurer l'authentification de la clé utilisée lors de l'échange. Toute la problématique consiste donc à identifier correctement l'agent avec lequel on souhaite communiquer, afin de s'assurer de son authenticité. C'est ce point précis que la couche de sécurité pour les systèmes multi-agents embarqués doit résoudre.

Cette problématique est similaire à la sécurité offerte sur le web, lorsqu'un navigateur web vérifie l'authenticité d'un serveur, notamment pour servir du contenu HTTPS par exemple. Pour parvenir à cette authenticité, le web possède une infrastructure à clés publiques (en anglais *PKI* pour *Public Key Infrastructure*) telle que décrite en figure 3, dans laquelle des autorités de certificats, ou *AC*, vont être en mesure d'authentifier les serveurs. L'authenticité des autorités de certificats elles-même est assurée par l'intégrité des navigateurs web qui possèdent dès l'installation une copie de leurs certificats.

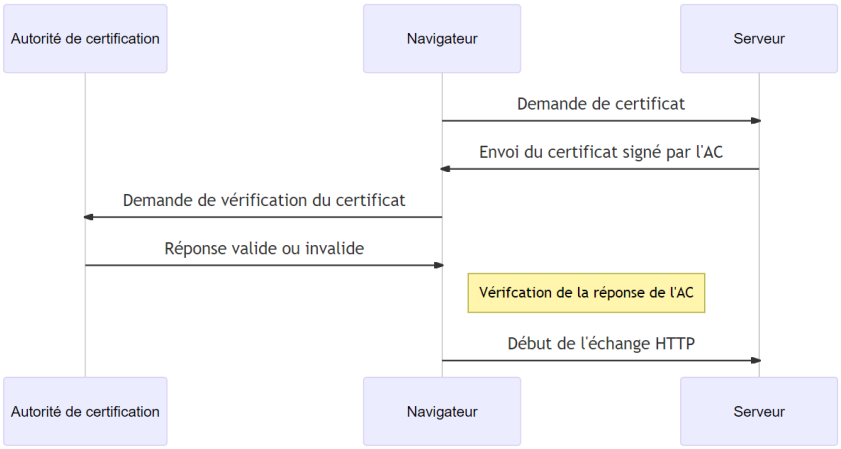

FIGURE 3 – Fonctionnement de la PKI sur le web

Dans notre cas de sécurité sur les systèmes multi-agents embarqués, il n'est pas possible d'avoir recours à une PKI car c'est une solution hiérarchique et centralisée. Ce que l'on recherche est donc une solution de sécurité qui doit garantir la confidentialité et l'intégrité des données tout en étant décentralisée [Won+18] [Mor+06].

De plus, l'infrastructure de sécurité que l'on cherche à mettre en place sur les systèmes multi-agents embarqués a pour but d'authentifier les deux agents de la communication. Contrairement à une architecture semblable au web, les environnements décentralisés n'offrent pas d'entité qui joue le rôle d'un serveur.

Chacun des participants à la conversation devrait donc authentifier son interlocuteur.

2.4 Objectifs

Le stage se situe dans le cadre d'un projet de recherche en collaboration entre le LCIS et l'Institut Fourier (IF) sur la sécurité des systèmes embarqués. Dans ce cadre, l'objectif de mon stage est d'abord de définir la sécurité des communications dans un système multi-agent embarqués avec des agents malveillants, en termes à la fois de contraintes liées à l'embarqué, de domaines de la sécurité concernés et de modèle d'attaque, ainsi que les spécifications et le développement d'une solution de sécurité correspondant à ces critères.

Afin de ne pas avoir un champ de recherche trop large, le stage étant limité dans le temps, nous avons choisi d'appliquer les simplifications suivantes aux contraintes définies précédemment pour livrer une première version complète du travail :

- les agents ne sont pas mobiles ;
- certains agents possèdent une portée plus grande ;
- les agents de longue portée ont une durée de vie plus grande ou infinie ;
- les agents ne peuvent apparaître dans le réseau qu'à l'initialisation ;
- les agents ont une mémoire et une capacité de calcul suffisantes pour le traitement demandé ;
- la durée de vie des agents n'est limitée que par un nombre d'opérations ;
- les interactions ne se font que selon un unique protocole.

Dans la même logique de simplification, nous ne prenons pas en compte les attaques qui portent atteinte à la disponibilité des systèmes multi-agents embarqués.

3 État de l'art

Le rapport d'état de l'art constitue l'un des livrables du stage. Il est d'autant plus important qu'il pourra être utilisé par la suite dans le laboratoire comme point d'entrée sur la sécurité des systèmes multi-agents embarqués.

Cet état de l'art passe d'abord en revue des solutions techniques au problème de confidentialité et d'intégrité en les regroupant selon des fonctionnements communs puis en donnant pour chaque solution les avantages et les inconvénients de leur mise en place. L'état de l'art liste ensuite de manière détaillée les attaques possibles sur un système multi-agent embarqués afin de motiver le choix du modèle d'attaque. Enfin, l'état de l'art est accompagné par une veille technologique qui compare les outils existants pour simuler un système multi-agent.

3.1 Solutions de sécurité

Pour garantir la confidentialité et l'intégrité des communications, les agents qui veulent communiquer doivent établir un *secret partagé*, grâce à l'échange de clés de Diffie-Hellman, vu au paragraphe 2.3. Le point à résoudre est l'authenticité de cet échange.

Comme vu au paragraphe 2.3, la solution communément utilisée sur le web est la PKI, qui a recours à une liste hiérarchique d'autorités de certification. Une autre solution est d'utiliser l'architecture PGP (pour *Pretty Good Privacy*), notamment utilisée pour le chiffrement de bout-en-bout de mail, qui consiste à diffuser les clés publiques sur Internet, par exemple en signature de mails ou sur des serveurs de clés PGP qui répertorient les clés de n'importe quel utilisateur qui en fait la demande. PGP utilise donc le principe de confiance lors de la première utilisation (en anglais *TOFU* pour *Trust On First Use*). Ces deux architectures ont donc pour but de diminuer la confiance que les utilisateurs doivent avoir en le système. Plusieurs solutions de sécurité listées ci-dessous s'inspirent de ces architectures.

— Les **mécanismes de vérification partielle** sont diverses méthodes qui permettent de vérifier l'identité de l'interlocuteur, notamment grâce à la redondance spatiale et temporelle [AK09], la latence [Rah17] et la fragmentation [Pim+04]. Ces mécanismes peuvent être mis en place tout au long de l'échange ou uniquement lors de l'échange de clés. Ils possèdent l'avantage d'être relativement simples à mettre en œuvre. Cependant, s'il existe dans le système un ensemble d'agents malveillants qui collaborent entre eux de telle sorte qu'il n'existe aucun chemin entre le ou les agents attaqués et le reste du système, alors ces méthodes ne permettent pas de se prémunir d'une attaque sur une communication entre des agents situés de part et d'autres des attaquants. Pour résoudre cette attaque, il serait alors nécessaire d'avoir recours à un système de détection d'intrusion, (en anglais *IDS* pour *Intrusion Detection System*), tout en préservant la décentralisation [Meh+18].

— Les **solutions purement cryptographiques** permettent également de vérifier la provenance d'une clé publique, grâce aux cryptosystèmes à

seuil [Fou11] et au calcul multipartite sécurisé (en anglais *multi-party computing*). Par exemple, le fonctionnement pourrait être de générer un couple de clés maîtresses, *master secret key* et *master public key*, dont la clé privée serait un secret réparti entre tous les agents du système de telle sorte qu'il ne soit pas possible de la reconstituer sans l'accord de tous les agents. Ensuite, à partir du couple de clés maîtresses, il serait possible de créer les clés publiques et privées des agents, de telle sorte que n'importe quel agent puisse calculer la clé publique de n'importe quel autre agent grâce à sa connaissance de la *master public key*. Ces solutions pourraient être les plus adaptées car elles offrent la possibilité de décentraliser la vérification de certificats, mais les outils qui doivent être maniés sont à développer pour être utilisés sur des systèmes multi-agents.

— Les **autorités de certification distribués** servent à distribuer une autorité de certification parmi les agents du système. On cherche donc à construire une structure de données répartie qui permette de vérifier l'intégrité des données qu'elle contient, offre la possibilité d'ajouter de nouvelles données tout en étant vigilant à la quantité de mémoire utilisée par les agents.

Bien que les mécanismes de vérification partielle soient des solutions très intéressantes pour le domaine de l'embarqué, le fait qu'elles possèdent des failles dès la conception nécessite d'avoir recours à un système qui gère la confiance et détecte les intrus. Or, nous souhaitons utiliser une solution qui soit indépendante d'un tel système. Les solutions purement cryptographiques quant à elles sont trop complexes pour être envisagées sur la durée d'un PFE si l'on souhaite tester la solution via des simulations, car les outils à manipuler doivent être conçus, au niveau cryptographique, pour répondre aux attendes du projet.

Nous avons donc choisi la solution d'une autorité de certification décentralisée. La proposition la plus simple était d'avoir recours à une table de hachage distribuée (en anglais *DHT* pour *Distributed Hash Table*) pour stocker les certificats [Tak+08], d'une manière similaire au stockage des informations dans un réseau pair-à-pair, comme utilisé par le protocole *BitTorrent*. Cependant, cela implique de calculer les clés publiques avant l'initialisation du système, afin d'injecter sur chaque agent l'équivalent d'un fichier *torrent*. Il n'est pas possible d'ajouter de nouveaux agents dans le système une fois que celui-ci est lancé.

Afin de garder la possibilité d'ajouter des agents à tout moment dans le système, nous nous sommes tournés vers l'autorité de certification décentralisée basée sur la Blockchain. La Blockchain (en français *chaîne de bloc*) est une structure de données fiable, à laquelle il est possible d'ajouter des informations, et qui offre de la traçabilité et de la non-répudiation. Les données de la Blockchain sont contenues dans des blocs, le lien entre chaque bloc est fait par un hachage cryptographique. L'idée est d'utiliser la technologie Blockchain, (BCT) pour stocker les certificats des agents [YSS18] [SB18] [Qin+20] [Cal+18]. De cette manière, chaque agent peut accéder à la clé publique de tout autre agent sans lui faire de requête directement. Pour ajouter des blocs dans la Blockchain, il est nécessaire de parvenir à un consensus sur la validité des informations que contiendra le bloc à ajouter.

Pour parvenir au consensus dans la Blockchain, il existe diverses méthodes détaillées dans une section dédiée dans le rapport d'état de l'art. Pour autant, ce sujet est très technique, le présent rapport n'abordera que la méthode implémentée dans la section 4.6 [BMZ18] [HM19].

3.2 Modèle d'attaque

Un système multi-agent tel que nous l'avons défini au paragraphe 2.2, c'est-à-dire **décentralisé**, **embarqué**, **mobile**, **sans-fil**, **dynamique** et **ouvert**, peut être la cible de nombreuses attaques. Ces attaques ont pour but soit de nuire au système multi-agent, afin d'entraver son fonctionnement, soit de collecter des informations qui seront traitées ultérieurement par le ou les attaquants.

Les attaques que nous avons répertoriées sont classées selon leur niveau, parmi *physique*, *hardware*, *communication* et *software* :

physique La **Jamming** attack est une attaque physique sur les réseaux de capteurs qui porte atteinte à la disponibilité [Kik+17].

hardware Les **Side channel** attacks sont des attaques hardware qui se font forcément sur les composants matériels, elles portent atteinte à la confidentialité ou à l'intégrité.

communication L'**Acknowledgement spoofing**, la `HELLO` **flood** attack, la **Routing information** attack, le **Selective forwarding** et la **Sinkhole** attack portent sur les communications. Elles portent toutes atteinte à la disponibilité [KW03].

software La **51%** attack, le **DoS**, la **Long-range** attack, le **Man-in-the-Middle**, le **Nothing-at-Stake**, la **Sleep deprivation**, la **Sybil** attack, le **White washing** et la **Wormhole** attack sont des attaques software. Les attaques software peuvent se faire sur tout type d'architecture, certaines de ces attaques sont exclusives à la Blockchain, aux systèmes de confiance ou aux systèmes multi-agents. Elles peuvent porter atteinte à la disponibilité ou à l'intégrité des données.

Dans le cadre de ce stage, nous nous intéressons uniquement aux attaques portant atteinte à la confidentialité et l'intégrité des données du système, nous mettons de côté celles sur la disponibilité.

Nous choisissons également de ne pas prendre en compte les attaques physiques et matérielles, telles que les attaques par canaux cachés (en anglais *side channel attack*) ou les attaques par brouillage sur la porteuse du signal (en anglais *jamming*) car celles-ci ne sont pas spécifiques aux systèmes multi-agents, et les mesures pour s'en prémunir dépassent le sujet de ce stage. Enfin, nous ne prendrons pas en compte les nombreuses attaques portant sur les informations de routage, nous faisons ainsi l'hypothèse que les messages sont toujours acheminées correctement dans le système pourvu qu'il existe un chemin entre les deux agents. Enfin, les attaques logicielles, c'est-à-dire celles qui reposent sur des agents malveillants au sein du système, sont celles qui nous intéressent particulièrement. Nous faisons également l'hypothèse que les attaquants du système

n'auront pas accès à une quantité de ressources supérieure à celle des agents, ce qu'on appelle des attaques *mote-class*.

Nous cherchons entre autres à nous prémunir de l'écoute intempestive et de la modification des communications du système. Nous considérons donc que l'attaquant cible certaines communications en particulier dans le but de récolter des informations ou d'usurper une identité. Pour parvenir à la sécurité, il est nécessaire d'authentifier les entités qui cherchent à communiquer. Étant donné le délai assez court du projet, nous nous intéresserons uniquement aux attaques de fork sur la Blockchain, à l'eavesdropping, au Man-in-the-Middle et à l'attaque Sybil.

- Le **fork** est une attaque sur une Blockchain dans laquelle le ou les attaquants possèdent suffisamment d' agents ou des ressources pour prendre le contrôle momentané ou définitif du système. Si les attaquants sont majoritaires, ils forment alors un consensus, c'est l'attaque par majorité, ou *51% attack*.
- L'**eavesdropping** ou écoute intempestive est une attaque sur n'importe quel type de réseau dans laquelle l'attaquant parvient à récupérer les messages d'un échange sans interférer dans la conversation, afin d'obtenir des informations qui ne lui sont pas destinées.
- Le **Man-in-the-Middle** (*MitM*) ou attaque de l'homme du milieu est une attaque sur n'importe quel type de réseau dans laquelle l'attaquant se positionne entre les deux entités qui tentent de communiquer, en se faisant passer auprès de chacun pour l'autre participant, il peut ainsi déchiffrer les communications.
- L'attaque **Sybil** est une attaque sur un système de confiance, l'attaquant crée et attribue plusieurs identités à un seul agent, et procède ainsi sur plusieurs agents, pour obtenir un contrôle disproportionné du système. Si l'attaquant parvient à isoler un ou plusieurs agents, il peut notamment procéder à une attaque MitM.

3.3 Étude des outils de simulation

Enfin, une veille technologique sur les différents outils de simulations vient clôturer l'état de l'art. Le but de cette comparaison est double : d'une part déterminer le simulateur à utiliser dans la suite du projet, et d'autre part fournir au laboratoire une note de prise en main des outils les plus intéressants, afin de faciliter la suite des travaux dans le domaine des systèmes multi-agents. Ce travail s'est découpé en deux phases.

D'abord, la comparaison de treize outils de simulation qui s'est faite sur les huit critères suivants :

- la possibilité de simulation des agents ;
- l'existence de notions d'agents embarqués comme la portée, l'énergie et la sérialisation ;
- l'ouverture du code source ;
- les communications selon des spécifications existantes dans le domaine de l'embarqué ;

— le langage de programmation utilisé ;
— l'utilisation de l'outil dans la communauté scientifique ;
— la documentation ;
— la popularité ;
— la date de sortie.

Cette veille technologique a permis de mettre en avant cinq outils de simulation, que sont JADE [BPR01], MadKit [GF00], PADE [Mel+19], SPADE [GCB06] et MESA [MK15], dont les deux premiers sont en Java et les trois autres en Python.

L'étape suivante a été de procéder à l'installation et l'utilisation sur la programmation d'un cas d'étude simple avec ces outils. L'intérêt était de vérifier par la pratique qu'ils étaient utilisables et adaptés dans notre contexte, mais également pour rédiger la note de prise en main de ces solutions ainsi que donner mon avis, en avantages et inconvénients, sur chacun d'entre eux.

Le problème utilisé pour tester les outils est celui des enchères : un agent Vendeur annonce un article à son prix de départ auprès de tous les agents Acheteurs. Lorsqu'il reçoit une offre supérieure à la dernière valeur proposée, le Vendeur met à jour la valeur de l'article puis diffuse cette offre auprès de tous les Acheteurs. Lorsque le Vendeur attend une durée supérieure à 10 secondes, il envoie un message à tous les Acheteurs en annonçant le gagnant ainsi que le montant de l'article. Le Vendeur passe ensuite au deuxième article. Lorsqu'ils reçoivent une offre, les agents Acheteurs attendent une durée aléatoire, puis si la dernière offre est issue d'un autre acheteur et qu'ils peuvent surenchérir, alors ils font une nouvelle offre dont le montant est choisi aléatoirement entre le montant actuel et la somme qu'ils ont en possession. Lorsqu'un agent Acheteur reçoit l'annonce d'un gagnant et s'il est l'agent qui a remporté l'article, alors il met à jour la somme qu'il possède.

Suite à ces tests, nous avons d'abord mis de côté MESA en Python au profit de MadKit en Java car c'est une plateforme abordée pendant un TP de l'école et parce que MESA ne supporte pas de manière native la fonctionnalité d'envoi de message d'un agent à un autre. Cette couche logicielle pour la communication étant très importante dans le système multi-agent, son absence nous avait conduit à écarter cet outil trop rapidement. Ainsi, après un mois de développement sur MadKit et sur suggestion d'Arthur BAUDET, nous avons préféré changer au profit de MESA en implémentant nous même la fonctionnalité d'envoi de message. Les raisons qui ont motivé ce choix sont, entre autres, la préférence d'Arthur BAUDET, qui reprendra mon travail, et de moi-même pour le langage Python, une plus grande lisibilité et compréhension du code de l'outil et une meilleure indépendance de l'environnement de développement. Les fonctionnalités manquantes et celles déjà mises en place avec MadKit ont été implémentées par Arthur afin de ne pas prendre de retard dans le développement.

4 Modélisation et conception

La solution de sécurité a pour but de permettre aux agents du système de réaliser l'application pour laquelle ils sont conçus en chiffrant les communications. Nous allons représenter le système avec deux types d'agents : d'un côté les agents prévus initialement pour l'application du système, que l'on appellera les *Application Agents*, et de l'autre côté des agents ajoutés au système dont le seul but est de faire fonctionner la Blockchain présentée en section 4.5 qui permettra de stocker les clés publiques nécessaires au chiffrement, que l'on appellera les *Blockchain Agents*.

On souhaite que le travail supplémentaire à fournir par les *Application Agents* pour utiliser la couche de sécurité soit minimal. Dans le cas nominal, ces agents doivent choisir une paire de clés cryptographiques au démarrage, déclarer la clé publique auprès des *Blockchain Agents*, puis récupérer les informations dont ils ont besoin dans la Blockchain qui est diffusée autour d'eux.

Les *Blockchain Agents* forment un système à l'intérieur du système, donc le but est de trouver un consensus sur l'état de la Blockchain. Dans le cas nominal, les *Blockchain Agents* commencent par ajouter leurs propres clés publiques dans la Blockchain, puis ils ajoutent celles des *Application Agents* qui en font la demande, tout en garantissant le consensus. Ils doivent également diffuser chaque nouveau bloc auprès des *Application Agents*.

4.1 Couche application

La partie du système qui nous intéresse ici est la solution de sécurité mise en place par les *Blockchain Agents*, l'utilisation qui en est faite par les *Application Agents* ainsi que la robustesse face à des éventuels attaquants.

Pour les besoins de la simulation, les *Application Agents* auront pour but de délivrer des messages de manière aléatoire à travers le système. À chaque tour, une fois que l'initialisation de la Blockchain est terminée, chaque *Application Agent* a 5% de chance d'entamer une conversation avec un autre agent tiré au hasard parmi le système. Lorsqu'un agent reçoit un message qui lui est destiné, il a 75% de chance de répondre à son tour. Ainsi, la longueur moyenne d'un échange est de 3 messages.

Ces échanges ont pour but de simuler le fonctionnement des *Application Agents* dans l'accomplissement de leur tâche quelle qu'elle soit.

4.2 Agents de longue portée

Afin de diffuser au mieux la Blockchain, nous avons rapidement fait la supposition que les *Blockchain Agents* avaient à leur disposition une portée d'émission largement supérieure à celle des *Application Agents*, comme sur la figure 4. De cette manière, dans le cas nominal, les *Applications Agents* se retrouvent à portée d'au moins un *Blockchain Agent*, et ils peuvent recevoir les blocs de la Blockchain sans avoir besoin de faire confiance aux agents qui transmettraient les blocs.

Si l'on retire l'hypothèse d'émission à plus longue portée, et si la densité dans le système des *Blockchains Agents* est inférieure à celle des *Applications Agents* alors les *Applications Agents* ne reçoivent plus les blocs de la Blockchain directement, mais les blocs doivent être transmis de proche en proche à tous les agents. Il faudrait alors que les blocs transmis soient signés de la part d'un des *Blockchain Agents* et vérifiés lors de leur réception par les *Application Agents*. Le premier problème de sécurité que cela pose est l'injection des clés publiques des *Blockchain Agents*, c'est-à-dire comment être certain que les clés reçues n'ont pas été compromises par un agent malveillant, qui aurait alors un contrôle total sur les agents attaqués. Aussi, le problème de disponibilité des blocs se pose, car s'ils doivent être transmis, des agents malveillants peuvent choisir de ne pas faire leur part du travail et isoler artificiellement des agents du système.

C'est pour ne pas avoir besoin de gérer ces problématiques, donc dans un soucis de simplification, que nous avons fait l'hypothèse que les *Blockchains Agents* aient un portée plus grande que celle des *Application Agents*.

4.3 Disposition du système

On s'intéresse ici à la disposition des agents, c'est-à-dire à leur répartition géographique dans le système. Nous souhaitons que le système, et plus particulièrement la solution de sécurité, soit résilient peu importe cette répartition. Au minimum, si certaines dispositions particulières posent un problème, nous aimerions pouvoir caractériser ces difficultés.

Pour cela nous avons besoin d'une répartition géographique aléatoire des agents, mais sous certaines contraintes : les agents doivent posséder un certain rayon autour d'eux libre de tous voisins, et la répartition doit être uniforme dans l'espace. De cette manière, les agents ne sont jamais trop proches les uns des autres, afin d'éviter les collisions par exemple, mais aussi aucun agent n'est isolé du réseau, de telle sorte qu'il soit toujours possible de relier à travers le réseau toute paire d'agents.

Pour répondre à ces critères, nous avons recours à la répartition en disques de Poisson, ce qui donne des dispositions semblables à la figure 4. Sur cette représentation, les points les plus gros représentent les *Blockchain Agents*, tandis que les plus petits montrent les *Application Agents*. Deux agents sont reliés s'ils sont tous les deux voisins l'un de l'autre.

4.4 Durée de vie des agents

Pour modéliser la durée de vie des agents induite par la quantité limitée d'énergie électrique, nous avons choisi de représenter la batterie des agents sous forme d'unité abstraite. Pour cela, les agents possèdent lors de l'initialisation une quantité initiale d'énergie, puis au fur et à mesure cette quantité est décrémentée du coût des opérations qu'ils effectuent.

Nous avons pour hypothèses que le coût d'une opération est identique d'une fois sur l'autre, que les agents en veille ont une consommation nulle et que l'unité de mesure est uniforme au court de la vie de l'agent.

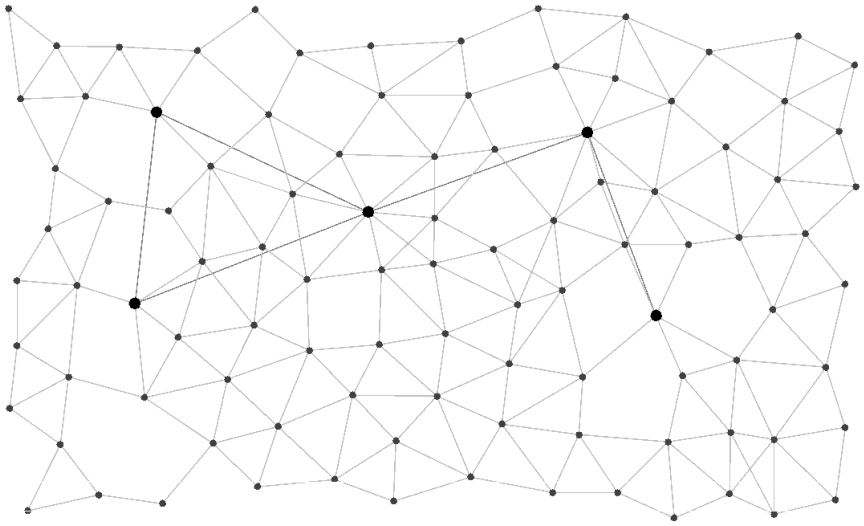

FIGURE 4 – Exemple de répartition des agents dans le système

Ces coûts ont été déterminés avec des chercheurs du laboratoire compétents sur les domaines de la communication sans-fil et de la cryptographie [KAK17] [Bui+17] [Wu+15]. Pour autant, elles sont issues de nombreuses approximations, et n'ont pas pour vocation à être exactes.

Les coûts des opérations dans notre unité arbitraire sont les suivants :

— envoi et réception d'un message sans-fil : 30
— opération cryptographique asymétrique : 20
— opération cryptographique symétrique : 6
— opération simple sur le processeur : 1

Ce que nous appelons "opération simple sur le processeur" est typiquement le coût associé au stockage en mémoire. De manière plus générale, ces quatre types d'opérations ont pour but d'être assez générales et ne prennent pas en compte certaines subtilités de consommation d'énergie.

4.5 Présentation de la Blockchain

La Blockchain est une technologie apparue pour la première fois en 2008 dans un article scientifique publié sous le nom de Satoshi Nakamoto [Nak+08], dans lequel est expliqué le fonctionnement du Bitcoin. Dans les années qui ont suivi, la Blockchain a continué de faire beaucoup parler d'elle, notamment grâce à l'émergence d'innombrables cryptomonnaies similaires au Bitcoin, et parfois innovantes. Puisque la Blockchain est associée depuis l'origine aux cryptomonnaies, les utilisations qui en sont faites sont liées aux cryptomonnaies dans l'écrasante majorité des cas. C'est en partie pour cette raison que la Blockchain fait

l'objet de très nombreux fantasmes et idées reçues, même dans le domaine de l'informatique.

La Blockchain est une structure de données distribuée capable d'évoluer dans le temps. La problématique principale de la Blockchain est de parvenir à maintenir un consensus sur les données qu'elle contient. La Blockchain fonctionne comme un registre dans lequel sont inscrites les données : contrairement à d'autres structures de données distribuées, il est possible d'y ajouter des informations, tant que le consensus est maintenu. En revanche, il est impossible de modifier ou supprimer une donnée qui a été ajoutée et validée dans la Blockchain.

Lorsqu'une entité souhaite ajouter des informations dans la Blockchain, ces informations sont signées grâce à une signature cryptographique puis diffusées auprès des autres entités qui possèdent la Blockchain. Une fois que ces informations ont été entendues, un *mineur* va toutes les ajouter dans un bloc qu'il ajoutera ensuite à la Blockchain. Un bloc contient donc une liste d'informations - des transactions dans le cas des cryptomonnaies - signées et les unes à la suite des autres. Un bloc contient également un identifiant, qui se trouve être le hachage cryptographique du bloc précédent, de telle sorte qu'une légère modification dans l'un des blocs résulterait en une Blockchain totalement différente, l'intégrité des données est assurée de cette manière. Enfin, les blocs sont signés par le mineur qui les a produits et ajoutés à la Blockchain, afin de garder une trace de qui a ajouté quelle information.

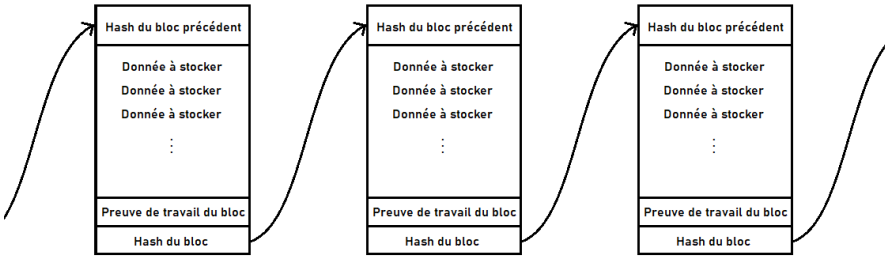

FIGURE 5 – Enchaînement des blocs dans la Blockchain

Lorsqu'un bloc est "ajouté à la Blockchain", cela signifie que le mineur qui a créé le bloc va le diffuser auprès de toutes les entités qu'il connait et qui contribuent à la Blockchain. À leur tour, ces entités diffusent le dernier bloc reçu, de sorte que de proche en proche toutes les entités aient connaissance du bloc. La Blockchain n'est donc pas stockée sur un serveur centralisé qui servirait d'hébergeur, mais elle est répliquée à l'identique sur toutes les entités qui y contribuent, et diffusée lorsque de nouveaux blocs sont minés.

Jusqu'ici, le fonctionnement de la Blockchain est entièrement décorrélé des informations qu'elle contient. Mais pour l'instant, le consensus n'est pas assuré, des entités peuvent décider d'inscrire des informations erronées dans la Blockchain et de les diffuser. Pour éviter que cela ne se produise, on cherche à faire

en sorte qu'à chaque bloc, le mineur qui va l'ajouter soit déterminé selon son implication dans la Blockchain. Dans son papier original sur le Bitcoin, Satoshi Nakamoto propose la preuve de travail (*Proof-of-Work*), dans laquelle les entités qui souhaitent ajouter des blocs dans la Blockchain doivent être les premiers à résoudre un problème cryptographique complexe. Le premier qui parvient à résoudre ce challenge gagne le droit d'ajouter un bloc dans la Blockchain, donc de le diffuser auprès des autres, mais surtout empoche une récompense qui sert de motivation pour inciter les gens à miner. De manière générale sur les cryptomonnaies, la récompense du minage est une certaine somme dans la devise supportée par la Blockchain, c'est d'ailleurs souvent la seule manière de "créer" la devise. Dans la Blockchain du Bitcoin, le challenge cryptographique à résoudre consiste à trouver un nombre tel que le hachage cryptographique commence par environ une trentaine de bits à zéro, ce qui représente 2^{32} possibilités, c'est de là que vient l'image d'un mineur.

L'idée de la *Proof-of-Work* est d'obliger les mineurs à fournir une puissance de calcul telle qu'il ne peut pas être rentable de tenter d'ajouter de faux blocs dans la Blockchain. Lorsque des mineurs malveillants tentent d'ajouter des blocs erronés dans la Blockchain, on appelle cela un fork de la Blockchain : au delà d'un certain point - le dernier bloc précédant l'attaque - il existe deux versions de la Blockchain qui coexistent. Lorsqu'une telle situation se produit, les mineurs "bienveillants", c'est-à-dire ceux qui ne font pas partie de l'attaque, ont pour règle de toujours poursuivre sur la branche de la Blockchain la plus longue. De cette manière, afin de faire perdurer un fork malicieux, les attaquants doivent déployer une puissance de calcul supérieure à la somme des puissances des autres mineurs. Si jamais cela se produisait, les attaquants auraient un contrôle total sur la Blockchain et pourraient y inscrire des informations fausses, on appelle cela l'attaque par majorité, voir paragraphe 3.2. Puisqu'une telle attaque est très difficile à mettre en place, les attaques sur la Blockchain ont généralement pour but de réaliser un fork de la Blockchain pendant suffisamment de temps pour arriver à y inscrire plusieurs opérations frauduleuses, comme des fausses transactions, et ainsi récupérer des devises qui ne leur appartiennent pas dans la cryptomonnaie.

Dans le cas de la sécurité des systèmes multi-agents, le consensus est réalisé par une autre méthode car la *Proof-of-Work* est très énergivore. La Blockchain possèdera une valeur équivalente à la cryptomonnaie, basée sur la réputation au sein de la Blockchain, décrite au paragraphe suivant.

4.6 Fonctionnement de la Blockchain

La Blockchain telle que décrite au paragraphe 4.5 possède une méthode de consensus basé sur la *Proof-of-Work*, dont le but est de parvenir à "casser" des hachages cryptographiques le plus rapidement possible. C'est pour cette raison que le Bitcoin et d'autres cryptomonnaies ont vu l'émergence de "fermes" de minage, dans lesquelles sont entreposées des centaines de machines qui n'ont pour but que le minage cryptographique sur carte graphique.

Ce type de comportement est très indésirable dans le cadre des systèmes

multi-agents embarqués, car les ressources en termes de capacité de calcul sont limitées et que les opérations sont coûteuses en énergie. Pour cela, nous allons utiliser une autre méthode de consensus inspirée de la preuve d'enjeu (*Proof-of-Stake*). On peut notamment citer la cryptomonnaie Ethereum qui a entamé une transition de la *Proof-of-Work* vers la *Proof-of-Stake*.

Dans la *Proof-of-Stake*, les nœuds, c'est-à-dire les participants à la Blockchain, peuvent mettre en jeu une partie de leurs cryptomonnaies pour tenter de devenir un validateur et gagner le droit de valider un bloc. Les nœuds sont dépossédés de cette somme qui va dans leur mise, et plus la mise d'un nœud est importante, plus ce nœud a de chance de valider un bloc. Lorsqu'un nœud valide un bloc, c'est à dire qu'il ajoute dans un bloc toutes les nouvelles informations à mettre dans la Blockchain puis qu'il ajoute ce bloc à la Blockchain, alors sa mise augmente. Les nœuds peuvent à tout moment retirer leur mise. Cependant, si un validateur ajoute un bloc erroné, c'est-à-dire avec des informations qui sont fausses, sa mise est "brûlée" et donc perdue. Ce mécanisme se met en place lorsque des validateurs tentent de créer un fork de la Blockchain. Les informations sont vérifiées en continu par les autres nœuds qui possèdent une mise, et qui créent un fork légitime de la Blockchain, puisqu'ils ne souhaitent pas voir eux-même leur mise disparaître. Lorsque le fork légitime est suffisamment plus grand que celui erroné, tous les validateurs qui ont misé sur le fork erroné perdent leur mise.

La *Proof-of-Stake* possède cependant elle aussi des défauts, dont un phénomène "d'enrichissement des riches" qui se produit. L'idée est que les nœuds qui possèdent la plus grande mise dans la Blockchain ont plus de chance que les autres de valider un bloc et donc d'augmenter leur mise, augmentant ainsi leur chance de valider un bloc. Pour contrer ce problème, de nombreuses alternatives à la *Proof-of-Stake* ont émergé au stade académique, mais peu d'entre-elles ont eu une application concrète dans une cryptomonnaie répandue. Parmi ces initiatives, on peut notamment citer la preuve d'enjeu et de temps, *Proof-of-Stake-Time*, dans laquelle l'âge de la mise est aussi pris en compte et où le choix du validateur se fait alternativement parmi ceux qui ont la plus grosse mise et ceux qui en ont le moins [Pik+15], ou la preuve d'enjeu âgé minimal, *Proof-of-Minimum-Aged-Stake*, dans laquelle tous les nœuds qui possèdent une mise suffisamment âgée et supérieure au seuil minimal ont la même chance de valider un bloc [PP21].

Le second défaut de la *Proof-of-Stake* qui nous intéresse est le fait qu'elle est intrinsèquement liée à la cryptomonnaie. Une Blockchain basée sur la preuve d'enjeu ou une méthode dérivée ne peut pas être utilisée autrement que pour une cryptomonnaie si elle n'est pas modifiée, sinon le concept de mise et d'enjeu n'a pas de sens. Nous souhaitons adapter la *Proof-of-Stake* pour le cadre des systèmes multi-agents. Pour cela, nous définissons la **réputation** comme la grandeur intrinsèque à la Blockchain. Cette réputation est une valeur non-cessible, c'est-à-dire qu'on ne peut pas la transférer directement d'un agent à un autre. Tous les agents en possèdent une certaine quantité lors de l'initialisation de l'agent. Comme il n'est pas possible d'échanger de la réputation, le seul moyen d'en gagner est de valider des blocs. Cette réputation est propre

à la Blockchain et est donc indépendante d'un éventuel système de gestion de confiance.

Dans ce cas, le phénomène d'enrichissement des riches n'est plus réellement un problème, car les agents du système qui possèdent la meilleure réputation seront davantage sollicités, et donc useront plus vite leur batterie. Le système étant ouvert, de nouveaux agents arriveront pour remplacer les anciens. Comme la réputation n'est pas cessible d'un agent à un autre, la réputation des agents qui quittent le système disparaît en même temps qu'eux.

Lors de son initialisation, chaque agent possède dans la Blockchain une réputation minimale. Pour les *Blockchain Agents*, qui vont ajouter les informations des agents dans des blocs, la réputation va augmenter au fil du temps grâce aux récompenses des blocs ajoutés à la Blockchain. Si un *Blockchain Agent* tente d'ajouter un bloc erroné, sa réputation diminue, de telle sorte à ce qu'il ait moins de chance de pouvoir ajouter un bloc de nouveau. Un *Application Agent* ne peut pas ajouter de blocs à la Blockchain, sa réputation ne peut donc pas augmenter. En revanche, en fonction de la véracité des informations qu'il transmet, comme les *Blockchain Agents*, il est susceptible de perdre de la réputation.

4.7 Transmission de la Blockchain

Pour les Blockchain sur Internet, comme c'est le cas pour les cryptomonnaies par exemple, les données sont envoyées directement via Internet. De cette manière, n'importe quel nœud du réseau peut échanger des données avec n'importe quel autre nœud, que ce soit pour l'envoi d'un bloc ou la réception des informations à ajouter dans un bloc. Les problématiques de nœuds malveillants ne se posent pas sur la transmission des blocs, mais uniquement sur la calcul et l'ajout des blocs eux-mêmes.

Dans notre cas de systèmes multi-agents embarqués, il n'est pas possible de passer par Internet pour échanger des données. Les informations à ajouter dans la Blockchain doivent donc être propagées de proche en proche à l'intérieur du système. Par exemple, lorsqu'un *Application Agent* souhaite faire ajouter son certificat de clé publique dans la Blockchain, il transmet à ses voisins son certificat signé, en précisant les destinataires finaux qui sont tous les *Blockchain Agents* dont l'*Application Agent* a connaissance. Les voisins de l'*Application Agent* vont eux-mêmes transmettre le certificat à leurs voisins à condition qu'ils connaissent eux-mêmes chacun des destinataires. De cette manière, et en gardant en mémoire le dernier message transmis, on évite les problèmes d'inondation de type tempête de *broadcast*. Une fois que l'information est arrivée aux *Blockchain Agents*, ils se transmettent alors eux-mêmes les informations de proche en proche, de manière similaire, afin que le *Blockchain Agent* ajoutant le prochain bloc ait accès à l'information. Une fois que ce *Blockchain Agent* a ajouté un bloc à la Blockchain, il le diffuse auprès de ses voisins, qui vérifient à la fois que les informations sont correctes et qu'il possédait bien le droit d'écrire dans la Blockchain. Les autres *Blockchain Agents* peuvent ensuite diffuser à leur tour, dans tout le système, le nouveau bloc.

Dans cette chaîne de transmission, on remarque que comme les informations sont transmises de proche en proche, tout agent malveillant sur la chaîne, qu'il soit *Application Agent* ou *Blockchain Agent*, est susceptible de modifier les informations. Ce serait pour lui un comportement risqué, car si par un autre chemin l'information arrive à destination et est ajouté au prochain bloc, l'agent malveillant risque de perdre de sa réputation. En revanche, un comportement malveillant moins risqué consiste à ne pas transmettre les informations dans la chaîne. Ces deux comportements sont à la base des attaques qui peuvent avoir lieu dans la sécurisation des communications d'un système multi-agent embarqués par autorité de certificat basée sur la Blockchain.

4.8 Récupération des certificats

Le but de la Blockchain est de créer une autorité de certification décentralisée afin que les *Application Agents* puissent s'envoyer des messages chiffrés. Pour autant, on aimerait que les *Application Agents* n'aient pas la nécessité de stocker la totalité de la Blockchain. Pour cela, on met en place des mécanismes de requêtes de bloc. Pour cela, nous aurons besoin que les *Application Agents* stockent, pour chaque bloc reçu, le hachage cryptographique du bloc. Les agents n'ont pas besoin de le calculer puisque ce hachage se situe en fin de bloc.

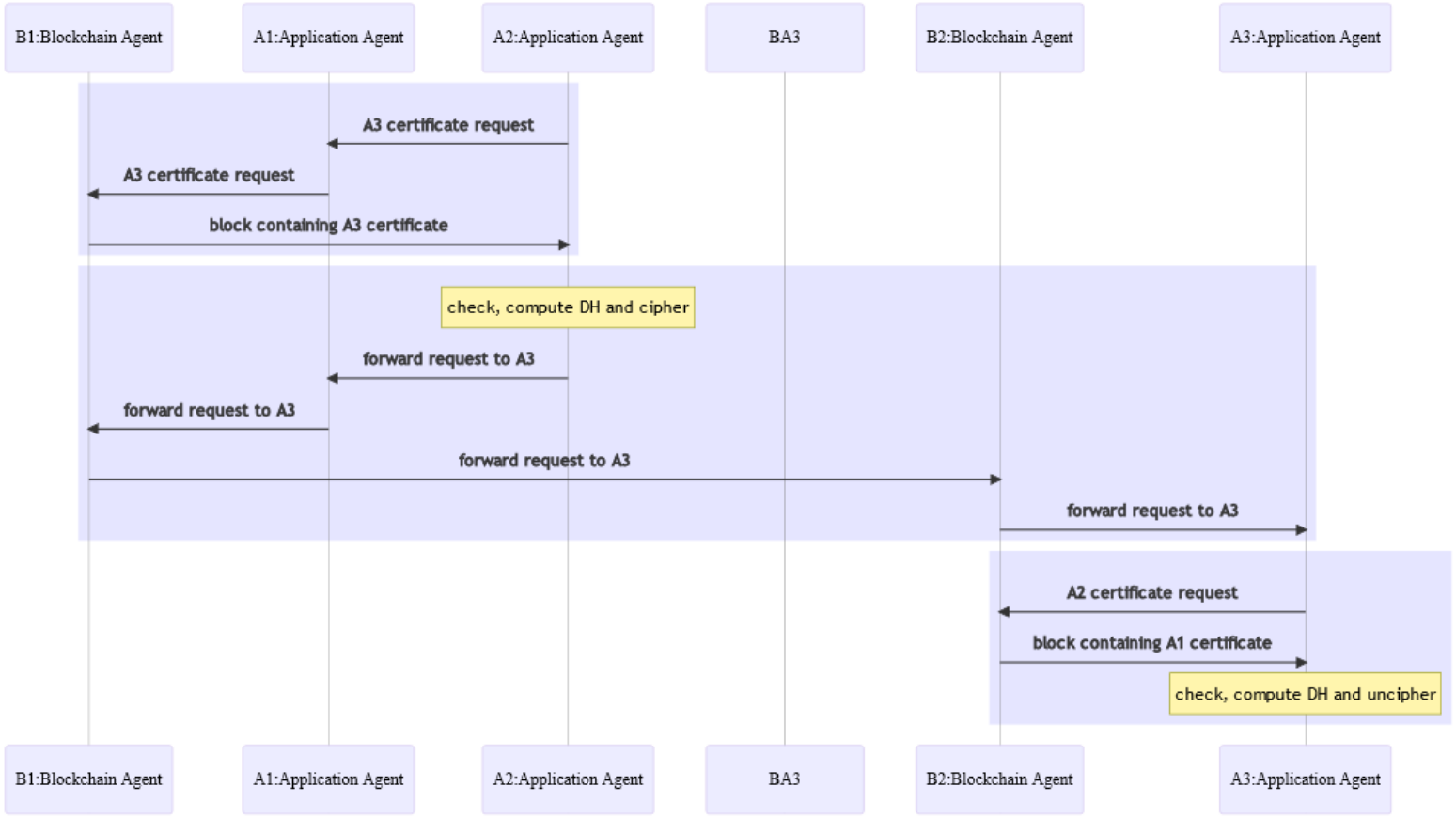

FIGURE 6 – Diagramme de séquence d'une requête de certificat

La figure 6 décrit un diagramme de séquence de cette requête. Un *Application Agent* qui souhaite entrer en communication avec un autre va d'abord émettre un message de type requête de certificat vers les *Blockchain Agents* qu'il connaît. Cette requête est transmise de proche en proche à travers les *Application Agents* qui connaissent également ces *Blockchain Agents*, pour enfin arriver jusqu'à eux. Les *Blockchain Agents*, qui possèdent chacun une copie intégrale de la Blockchain doivent ensuite la parcourir jusqu'à trouver le bloc qui contient le certificat demandé et le retransmettent. L'*Application Agent* qui a fait la requête récupère le bloc et vérifie dans la liste des hachages cryptographiques de blocs que celui qu'il vient de recevoir fait effectivement partie de la Blockchain.

Cet agent peut ensuite réaliser un échange de clé de Diffie-Hellman avec le certification reçu, puis réaliser le chiffrement symétrique et envoyer le message qui contient ses données chiffrées. Le message est transmis de proche en proche jusqu'aux *Blockchain Agents*, qui se le transmettent de proche en proche jusqu'à l'agent qui connaît le destinataire, qui le transmet une dernière fois. L'*Application Agent* destinataire reçoit le message chiffré, réalise une requête de certificat et attend la réponse. Une fois qu'il reçoit le bloc contenant le certificat de l'émissaire du message et qu'il a vérifié qu'il reconnaît ce bloc comment étant valide, il réalise un échange de clés de Diffie-Hellman et déchiffre le message.

Le détail des messages existants est décrit dans le tableau 2.

Dans le fonctionnement de la couche applicative que l'on a défini, les *Application Agents* choisissent aléatoirement à chaque tour s'ils envoient un message ou non, et s'ils le font ils piochent le destinataire du message. Dans une application réelle, on peut supposer que les *Application Agents* peuvent avoir une liste de certificats qui seront vraisemblablement utiles. Ils pourraient alors stocker ces certificats lors du calcul de la Blockchain, afin d'éviter au plus possible le mécanisme de requête de certificat qui est coûteux pour lui-même et pour ses voisins.

5 Implémentation

Avant de commencer l'implémentation, nous avons défini un plan de travail en quatre grandes phases, en ayant pour but de travailler de manière incrémentale sur la simulation, pour partir d'un système multi-agent embarqués non sécurisé à un système sécurisé, robuste face aux attaques et possédant toutes les propriétés définies plus tôt. Voici ce plan de travail :

1. **système multi-agent** : développer un système multi-agent embarqués dont les agents communiquent de manière non sécurisée, en répartissant les agents dans le système, puis en ajoutant les propriétés de batterie et de portée des agents, en ajoutant les *Blockchain Agents*, et enfin faire communiquer les agents entre eux de manière non sécurisé via l'organisation du système.
2. **Blockchain** : intégrer la gestion d'une Blockchain dans le système multi-agent en tant qu'autorité de certificats décentralisée, d'abord en implémentant une Blockchain, basée sur la *Proof-of-Work* puis la *Proof-of-Stake* indépendamment du système, en intégrant la Blockchain au système multi-agents et enfin en ajoutant les certificats à la Blockchain et en chiffrant les messages.
3. **Attaques** : évaluer la robustesse du système face aux attaques en ajoutant les comportements malveillants des *Application Agents* et des *Blockchain Agents*, puis en évaluant les performances du système face aux attaques et en étudiant les dispositions qui facilitent les attaques.
4. **Améliorations incrémentales** : ajouter des fonctionnalités au système multi-agent embarqués, comme l'apparition de nouveaux agents dans le système, l'affinage du modèle de consommation d'énergie ou le déplacement géographique d'agents dans le système.

Le dernier point a pour but d'ajuster la simulation déjà implémentée.

5.1 Disposition du système

Le premier élément sur lequel nous avons travaillé en début d'implémentation est la disposition des agents dans le système. La problématique est d'avoir une disposition qui soit aléatoire afin de tester la solution de sécurité sur différents cas, tout en respectant une répartition sensiblement uniforme, pour qu'il n'y ait ni zone vide ni zone trop dense. Pour cela nous utilisons un échantillonnage en disques de Poisson, c'est-à-dire de telle sorte que chaque point soit à une distance minimale de n'importe quel autre point, comme en figure 6.

La méthode naïve de l'échantillonnage de Poisson consiste à placer des agents de manière aléatoire dans l'espace puis à vérifier en bouclant sur chaque point précédemment placé afin de vérifier que la distance est respectée. Cette implémentation possède une complexité algorithmique de l'ordre de $O(n!)$. Pour améliorer ce résultat, nous utilisons l'algorithme d'échantillonnage rapide en disques de Poisson [Bri07], qui consiste en la manipulation d'une grille conçue

de telle sorte qu'il ne soit possible d'avoir qu'un unique point dans chaque case de la grille, comme illustré en figure 7. De cette manière, lorsqu'on ajoute un nouveau point tiré aléatoirement, au lieu de vérifier parmi tous les points précédemment placés, il suffit de chercher dans les cases de la grille autour du point tiré.

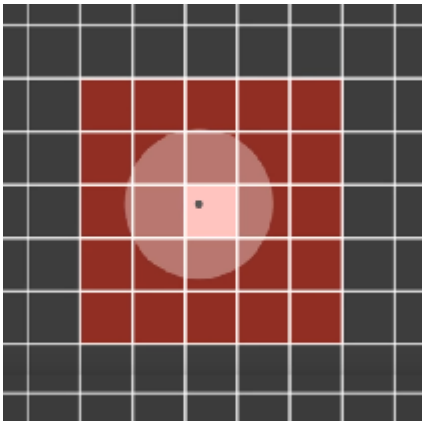

FIGURE 7 – Illustration de l'algorithme rapide d'échantillonnage de Poisson [2]

Nous avons adapté cet algorithme à nos besoins. Premièrement, nous faisons deux échantillonnages, le premier pour les *Blockchain Agents* et le second pour les *Application Agents*, de telle sorte que les *Application Agents* ne puissent pas être placés trop près des *Blockchain Agents*. Ensuite, nous avons ajusté l'algorithme pour que le nombre d'agents soit fait par pallier de cinq points, c'est-à-dire que le nombre de points est toujours un multiple de cinq. Sans cette modification, le nombre de points répartis par l'algorithme est aléatoire, en fonction de la distance minimale entre les points et les dimensions de l'espace. De cette manière, nous aurons plusieurs dispositions de systèmes multi-agents avec des nombres d'agents comparables.

La figure 8 donne un exemple de répartition des agents avec MESA. Sur cette représentation, les *Application Agents* sont en bleu et les *Blockchain Agents* en rouge.

Pour finir, nous avons séparé le script de répartition des agents et de la simulation en elle-même. Ainsi, le script de répartition génère un fichier au format JSON, duquel sont chargées les positions par le simulateur. De cette manière, nous pouvons réaliser plusieurs simulations sur une même disposition des agents, mais surtout nous pouvons créer à la main des dispositions que nous pensons particulièrement intéressantes.

5.2 Architecture logicielle

La figure 9 présente le diagramme de classes de la simulation.

2. crédit image : Sebastian Lague

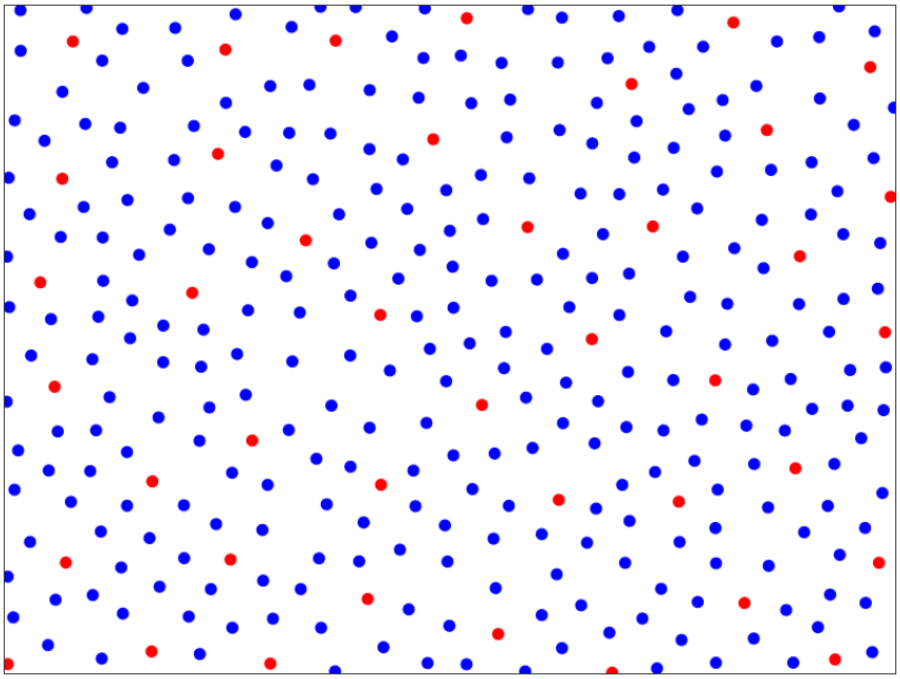

FIGURE 8 – Répartition des agents dans l'interface de MESA

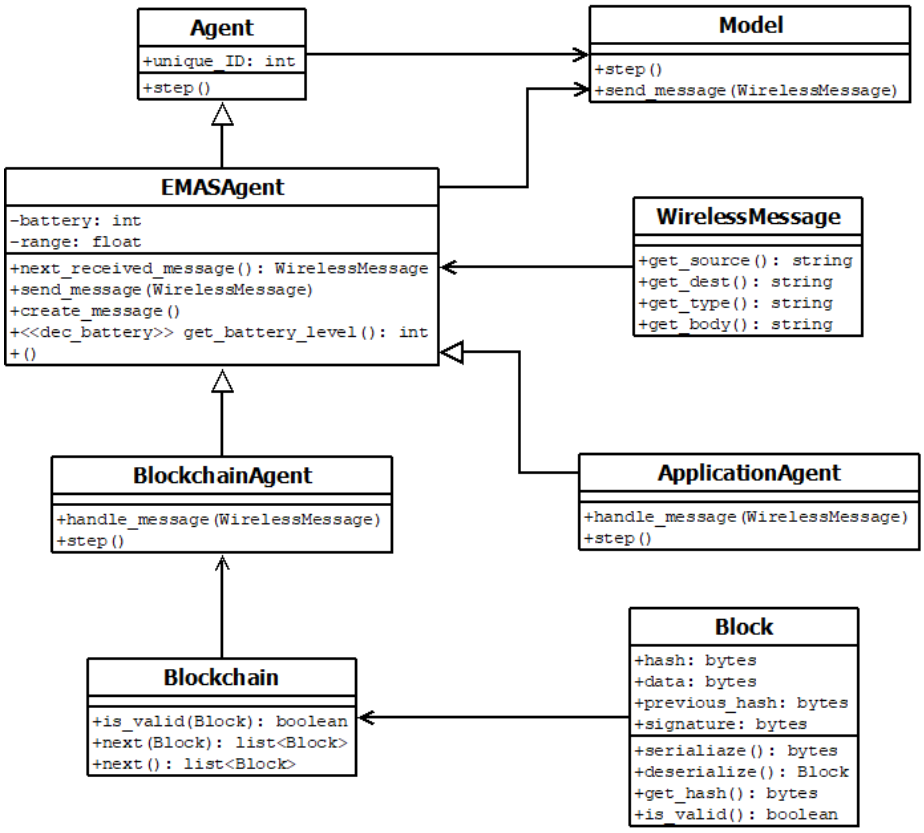

FIGURE 9 – Diagramme de classes de la simulation

5.3 Fonctionnement des messages

Les messages envoyés entre les agents sont des messages sans-fil, c'est-à-dire des objets de la classe `WirelessMessage`. Ces messages contiennent quatre champs : la source, le destination, le type du message et le contenu. Le contenu du message varie en fonction de son type comme d'après le tableau 1. Les champs source et destination contiennent respectivement les identifiants des agents émetteur et destinataire du message. Nous supposons que chaque identifiant est unique. Pour le champs destination, une valeur spéciale sera utilisée pour signifier qu'un message ne possède pas de destinataire, c'est-à-dire qu'il doit être traité pour tous les voisins de l'agent émetteur. Comme les messages ont vocation à être transmis sans-fil, dont l'émission et la réception passent par de l'électronique numérique, ils sont envoyés et reçus sous la forme de chaînes d'octets.

Table 1 : Type de messages sans-fil

Type de message	**Description**	Contenu
public key	Partage de la clé publique d'un agent et découverte des voisins	Clé publique
forward request	Transmission d'un message à destination d'un autre agent	Message à transmettre
block	Envoi d'un bloc de la Blockchain	Bloc
application	Envoi d'un message utile de l'application, transmis dans un forward request	Ping
help	Annonce d'une attaque sur l'agent	Clé publique
certificate request	Demande de transmission du bloc contenant le certificat d'un agent	Identifiant de l'agent

Les agents du système possèdent une boîte aux lettres sous la forme d'un objet liste. Lorsqu'un agent souhaite envoyer un message, il fait appel à la méthode `sendMessage()` de sa classe. Cette méthode fait ensuite appel au modèle de la simulation, qui connaît tous les agents du système, et qui va ajouter le message dans la boîte aux lettres de tous les agents à portée de l'émetteur. Les agents à portée pourront alors lire le message dans leur boîte aux lettres.

Table 2 : Longueur des champs d'un message

Source	Destination	Type	Contenu
48 bits	48 bits	4 bits	taille variable

Le tableau 2 décrit la longueur des champs d'un message en octets. Nous considérons la taille de l'espace d'adressage des identifiants comme celui des

adresses dans le protocole Bluetooth, ce qui correspond à $2 * 10^{14}$ identifiants, mais il est fort possible dans le cas d'une application à grande échelle que l'on observe des découpages de l'espace d'adressage, ce qui réduit le nombre d'identifiants utilisés. De plus, les identifiants pourraient être utilisés autrement, pour encoder des chaînes de caractères par exemple. La taille du contenu pourra varier énormément, entre 48 bits pour un message de type *certificate request* et 1k bits pour un message de type *block*, soit l'équivalent de cinq clés publiques de taille 256 bits. Le nombre de clés publiques par bloc n'est pas limité dans la simulation.

5.4 Comportement des agents

Lorsque les agents entrent dans le système lors de l'initialisation, ils doivent faire en sorte de rapidement mettre en place la Blockchain afin d'éviter les attaques de type *Man-in-the-Middle*. Pour ceci, les agents doivent rester le plus discrets possible tant que leurs certificats ne sont pas inscrits dans la Blockchain. Pour cela, un agent du système peut se considérer dans un des trois états suivants :

incertain lorsque le certificat de l'agent n'est pas encore dans la Blockchain, l'agent va émettre le moins possible et tenter d'inscrire son certificat dans la Blockchain ;

sûr lorsque son certificat est inscrit dans la Blockchain, l'agent peut alors relayer les requêtes des autres agents ;

non sûr lorsque le certificat présent dans la Blockchain est erroné ou qu'il est toujours absent après la phase d'initialisation, l'agent va alors tenter de faire remonter l'erreur jusque dans la Blockchain.

Le diagramme d'état en figure 10 décrit les transitions entre ces états, et vaut à la fois pour les *Blockchain Agents* et pour les *Application Agents*.

Les premiers agents à inscrire leurs certificats dans la Blockchain sont les *Blockchain Agents* qui vont chacun s'échanger leurs certificats de proche en proche, afin de les ajouter dans le bloc initial de la Blockchain. Une fois que le premier bloc est calculé, il est diffusé auprès des autres *Blockchain Agents*. Comme les *Applications Agents* reçoivent les messages s'ils sont à portée, ils reçoivent également le premier bloc de la Blockchain.

À ce moment, les *Application Agents* vont à leur tour transmettre leurs certificats aux agents autour d'eux. Parmi eux, quelques uns auront dans leur portée des *Blockchain Agents*, qui ajouteront leurs certificats dans le deuxième bloc de la Blockchain. Les *Blockchain Agents* s'échangent des informations et l'un d'eux créé un nouveau bloc qu'il diffuse dans le système. Lorsque les *Application Agents* reçoivent le nouveau bloc, ils regardent si leur certificat est inscrit dans la Blockchain. S'il est inscrit et qu'il correspond bien au certificat envoyé par l'agent, alors l'agent rentre dans un état sûr, dans lequel il a de bonnes raisons de penser qu'il n'a pas été attaqué. Si le certificat de l'agent n'est pas inscrit dans la Blockchain, alors celui-ci transmet à ses voisins une demande de suivi de certificat. Lorsque les voisins reçoivent la demande de suivi de certificat

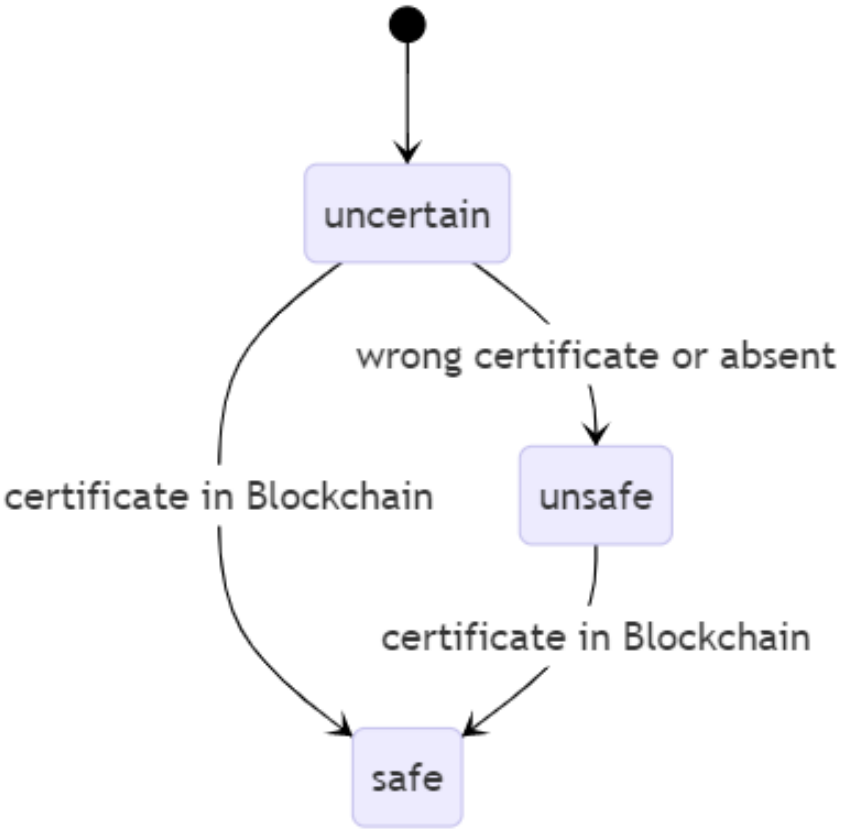

FIGURE 10 – Diagramme d'état des agents durant l'initialisation

et sous condition qu'ils aient leur propre certificat inscrit dans la Blockchain, alors ils signent eux-mêmes le certificat puis le transmettent à leur tour à leurs voisins.

De cette manière, de nouveaux certificats parviennent aux *Blockchain Agents*, qui peuvent les ajouter dans un nouveau bloc. Lorsque les *Applications Agents* dont le certificat n'était pas encore inscrit reçoivent le nouveau bloc, ils regardent si leur certificat a été ajouté. Tant que ce n'est pas le cas, ils continuent de diffuser à leurs voisins des demandes de suivi de certificat. De cette manière, il arrive un moment où tous les agents ont leur certificat inscrits dans la Blockchain, c'est-à-dire lorsque les *Blockchain Agents* ne reçoivent plus de nouvelles demandes d'ajout de certificat. Ils produisent alors un nouveau bloc vide, sans certificat à l'intérieur, pour signaler aux agents que l'initialisation du système est terminée.

Tous les certificats transmis doivent être auto-signés, afin d'augmenter le coût de création d'un certificat et de décourager les agents malveillants.

Sur le diagramme de séquence en figure 11, les encadrés en pointillé d'un même bloc *par* signifient que les actions contenues dans les encadrés se déroulent en parallèle de manière asynchrone, c'est-à-dire qu'elles se déclanchent approximativement au même moment et que leur ordre importe peu.

On pourrait également imaginer le cas d'un agent qui arriverait dans le système après l'initialisation, c'est-à-dire une fois que tous les certificats de tous les agents présents au démarrage du système sont inscrits dans la Blockchain, l'agent effectuerait alors un *entrée silencieuse*. Nous appelons ainsi la procédure qui décrit le comportement d'un tel agent : tant qu'il n'a reçu aucun bloc de la Blockchain, il ne signale sa présence à aucun de ses voisins. Lorsqu'il reçoit son premier bloc, il demande à ses voisins de faire suivre sa demande d'ajout de certificat aux *Blockchain Agents* desquels il a reçu le bloc. Il vérifie de la même manière dans le bloc suivant si son certificat a bien été ajouté.

Lorsqu'un agent constate que le certificat à son nom dans la Blockchain ne correspond pas au sien, ou si son certificat n'est toujours pas dans la Blockchain lorsque les *Blockchain Agents* considèrent avoir reçu tous les certificats, alors l'agent comprend qu'il subit une attaque et passe dans un état *non sûr*.

5.5 Comportements malveillants

Les agents malveillants du système auront différents modes d'attaques possibles, afin de parvenir à isoler les attaques entre elles et identifier pour chacune l'impact sur le système.

Les attaques suivantes sont données par ordre de priorité d'implémentation. Elles ne concernent que des agents malveillants avec le rôle de *Application Agent*, sauf le comportement **fork** qui sera le seul comportement malveillant possible pour les *Blockchain Agents*.

1. l'agent **Man-in-the-middle** aura pour but de transmettre des certificats erronés dans la Blockchain afin de se faire passer pour l'agent attaqué aux yeux du reste du système ;

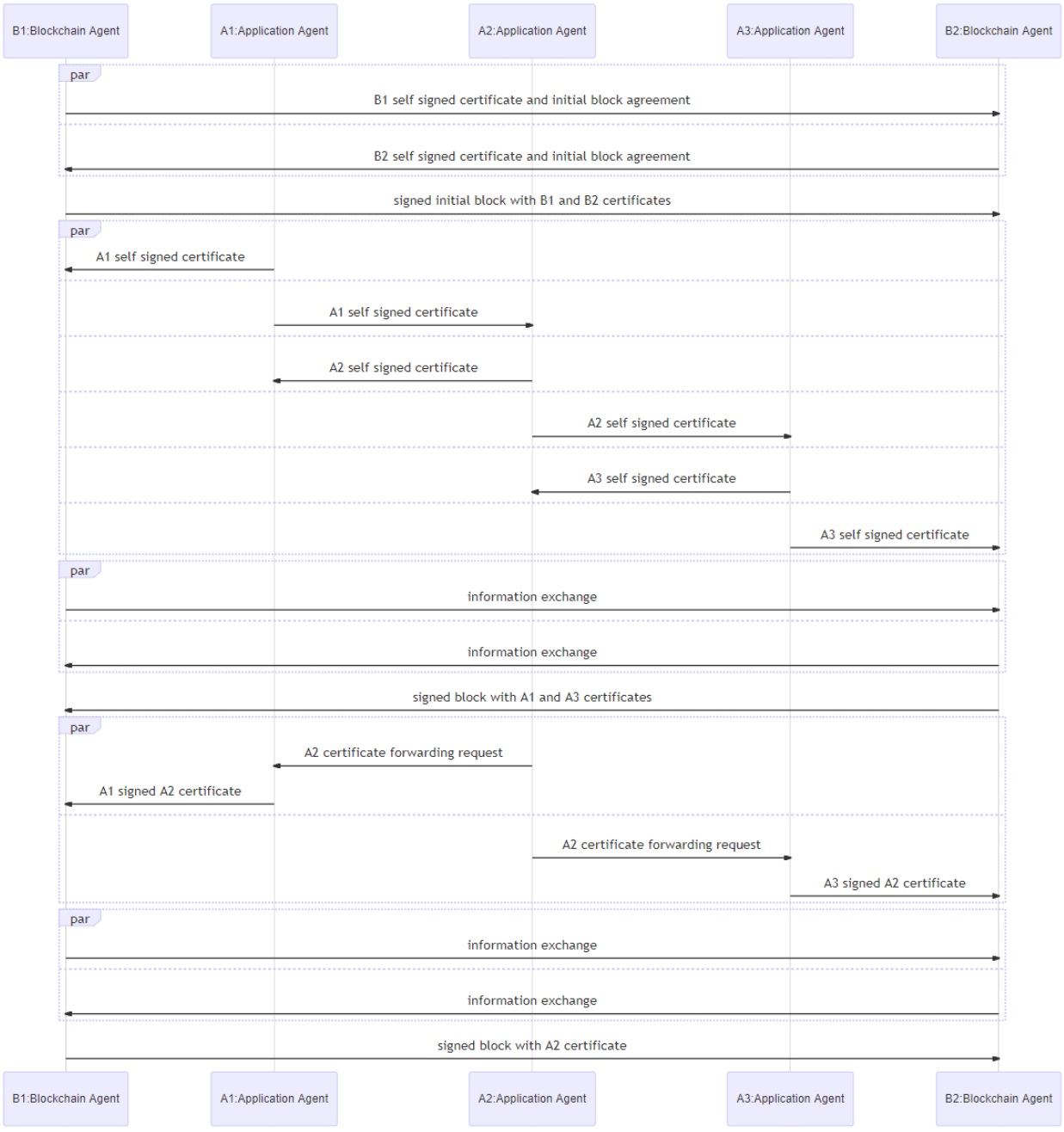

FIGURE 11 – Diagramme de séquence d'initialisation du système

2. les agents **Sybil** vont réaliser des attaques Man-in-the-Middle de manière coordonnées entre eux, afin de transmettre les mêmes certificats dans la Blockchain et ainsi isoler l'agent attaqué ;
3. l'agent **fork** va tenter de modifier les informations ajoutées dans la Blockchain, pour falsifier des certificats par exemple ;
4. le **faux *Blockchain Agent*** va se faire passer pour un *Blockchain Agent* pour transmettre une Blockchain erronée, les cibles de cet agent étant les *Application Agents* qui n'entendent qu'un seul véritable *Blockchain Agent*, voire qui sont hors de portée des *Blockchain Agents* ;
5. l'agent **menteur** va déclarer régulièrement de nouveaux certificats dans la Blockchain, sous diverses identités, en disant avoir été attaqué les fois précédentes, dans le but de réduire la réputation de ses voisins ;
6. l'agent **sniffer** va épier les communications de ses voisins et garder en mémoire les métadonnées, c'est-à-dire quel agent envoie un message à quel autre agent et à quel moment ;
7. l'agent **bavard** va initier de nombreuses communications dans le but de réduire la durée de vie de ses voisins.

Les attaques impliquant les comportements les moins prioritaires seront réalisées uniquement si la durée du stage le permet.

Plusieurs de ces comportements seront probabilistes, pour que les agents malveillants ne réalisent pas systématiquement leur attaque, afin de rester discret.

En réponse à ces attaques, un agent du système peut se considérer dans l'état non-sûr, il transmet alors régulièrement autour de lui un message de type *help*, jusqu'à ce que son message soit entendu. Si jamais il reçoit un message qui lui est destiné alors qu'il est dans cet état, l'agent refusera la communication.

La meilleure situation pour un agent malveillant serait que sa ou ses cibles pensent être dans l'état *sûr*.

6 Expérimentation

Durant la phase d'expérimentation du projet, nous espérons obtenir des résultats qui tendent à mettre en avant l'efficacité de la solution choisie. À partir du fonctionnement d'autorité de certificats basée sur la Blockchain et des comportements malveillants décrits précédemment, nous pensons que la solution choisie sera efficace pour une part non négligeable des communications. Nous ne nous attendons pas à des résultats de sécurité parfaits, car les dispositions aléatoires du système peuvent faire apparaître des cas de fonctionnement nouveaux et donc des failles dans la solution de sécurité.

Le second enjeu consiste en l'utilisabilité de la solution pour un système multi-agent embarqués. Nous espérons donc que la durée de vie des *Application Agents* lors de l'utilisation de la solution de sécurité aille dans ce sens-là, même face à des attaques.

6.1 Mesures

Pour évaluer les résultats de notre simulation, nous devrons utiliser des mesures pour révéler l'état du système multi-agent. Ces résultats seront obtenus via un grand nombre de simulations, jusqu'à observer des résultats significatifs. Voici les grandeurs que nous pensons mesurer :

- le nombre total d'agents dans le système ;
- le nombre de *Blockchain Agents* ;
- la durée de vie des agents ;
- le temps pour ajouter tous les certificats dans la Blockchain ;
- le nombre d'opérations effectuées par les agents, parmi envoi de message, réception de message, et opération cryptographique symétrique ou asymétrique ;
- la réputation des agents ;
- le nombre d'agents hors de portée des *Blockchain Agents* ;
- le nombre de communications de la couche application ;
- le nombre de communications compromises ;
- le nombre de tentatives d'attaques ;
- le type d'attaque réalisé ;

Une fois les mesures réalisées, nous mettrons en compétition entre elles les grandeurs qui sont pertinentes, dans le but de mettre en avant la robustesse ou la faiblesse du système selon le contexte.

À partir des résultats précédents, nous tenterons ensuite de créer manuellement dans le système multi-agent des dispositions particulière, afin d'appuyer nos résultats quels qu'ils soient.

6.2 Résultats attendus

La solution de sécurité offerte par l'autorité de certificats basée sur la Blockchain ne peut offrir une couverture de sécurité de cent pourcents. Cependant, nous espérons avoir des résultats significativement meilleurs avec la solution, de

telles sortes à ce que la majorité des tentatives d'atteinte à la confidentialité ou à l'intégrité des données échouent. Aussi, nous pensons que la solution offrira une durée de vie suffisante aux *Application Agents*, malgré la solution, soit une réduction de la durée du vie globale inférieure à trente pourcents.

7 Bilan

7.1 Retour pour le laboratoire

Ce stage s'inscrit dans le cadre d'une collaboration sur le long terme entre le LCIS et l'Institut Fourier dans le domaine de la sécurité des systèmes embarqués intelligents. Mon travail sur la sécurité des systèmes multi-agents embarqués contribue à apporter de la visibilité au laboratoire, premièrement grâce à la contribution scientifique qui va être sousmise pour publication, ensuite grâce à la thèse d'Arthur BAUDET qui s'inspirera de mon travail.

Comme le stage s'inscrit dans la recherche fondamentale du laboratoire, il est difficile de calculer le retour sur investissement, à court ou à long terme. Mon travail contribue à la production de savoir et de connaissance du LCIS, ce qui offre au laboratoire de nouvelles compétences techniques sur les différentes solutions envisageables pour la sécurité des systèmes multi-agents embarqués. Les résultats de mon stage pourront être réutilisés par la suite afin par exemple de répondre à des appels à projets, qui sont la principale source de financement du laboratoire.

En plus de ces compétences, le stage a permis la production d'outils et de documents qui pourront être réutilisés suite à mon départ du laboratoire. Nous pouvons citer notamment :

- le rapport d'état de l'art qui comprend différentes solutions envisageables pour la sécurité des systèmes multi-agents, réparties en trois catégories, une présentation des attaques susceptibles de compromettre un système multi-agent embarqués, avec pour chacune une analyse sur la faiblesse exploitée, et une veille technologique sur les outils de simulation de systèmes multi-agents ;
- une notice d'utilisation de cinq simulateurs de systèmes multi-agents, qui comprend la procédure d'installation et l'utilisation du simulateur ainsi qu'un avis personnel sur les points forts et points faibles de chaque ;
- les spécifications et le développement de la simulation de système multi-agents embarqués ainsi que la documentation pour être réutilisé dans le laboratoire.

Ces outils et documents ont pour vocation à être réutilisés pour les travaux futurs dans le domaine.

Le financement de mon stage est détaillé dans le tableau 3. Les indemnités de stage correspondent au minimum légal de 3,90€ par heure, ce qui est la norme dans un laboratoire public de recherche, sur base de 35 heures par semaine. Le matériel informatique correspond à la mise en place ainsi que l'utilisation d'un ordinateur de bureau muni du nécessaire de travail, l'utilisation d'un bureau dans l'open-space n'est pas possible à calculer en raison de l'hébergement offert par Grenoble INP - Esisar. Enfin, l'encadrement correspond au salaire brut chargé des trois encadrants de ce stage lors de nos réunions hebdomadaires, à raison d'une heure de réunion par semaine.

Table 3 : Coût du stage

Indemnités de stage	3 000 €
Matériel informatique	1 600 €
Encadrement	8 500 €
Total	13 100 €

Une partie de cet investissement est accordé au LCIS par l'UGA au travers du financement par projet IDEX. Le retour sur investissement du stage est perçu par le LCIS uniquement.

7.2 Retour personnel

Ce stage m'aura beaucoup apporté sur plusieurs niveaux. Le premier point est l'obtention de la labellisation SecNumEdu délivrée par l'ANSSI, en collaboration avec Grenoble INP - Esisar, que mon stage devrait me permettre d'obtenir. C'était l'un de mes objectifs dès ma recherche de stage, car je souhaite me garder la possibilité de me spécialiser dans la cybersécurité.

Le sujet sur lequel j'ai travaillé m'a permis de profiter de la légère expérience de l'environnement embarqué que j'ai pu acquérir lors de mon projet PX510 sur le chiffrement symétrique léger, et la transformer en expérience sur les systèmes intelligents, les environnements décentralisés ainsi que la technologie Blockchain. Ceci constitue pour moi une réelle opportunité, car je ne souhaite pas me spécialiser dans la sécurité des systèmes embarqués.

Ce stage m'a également permis de fixer ma position sur la poursuite d'études en thèse : le métier d'enseignant m'a toujours beaucoup intéressé, et je me suis longtemps demandé si un poste d'enseignant-chercheur pourrait me convenir. Cette expérience dans un laboratoire de recherche m'a fait comprendre que cela implique une spécialisation dans un domaine précis de l'informatique, or je souhaite dans un premier temps acquérir des compétences transversales. Je ne réaliserai donc pas de thèse à la suite de mes études, mais je conserve l'idée d'un jour enseigner à des étudiants, peut-être en qualité d'intervenant, ce qui me conviendrait également.

Le dernier point qui m'a séduit dès le départ dans ce stage est l'opportunité de publier une contribution scientifique dans un workshop. L'idée de contribuer au monde de la recherche scientifique est quelque chose de très parlant pour moi, et étant donné que je ne poursuivrai pas dans le secteur de la recherche, je suis ravi d'avoir eu la chance d'y participer.

Bibliographie

[AK09] M. Alicherry et A. D. Keromytis. “DoubleCheck : Multi-path verification against man-in-the-middle attacks”. In : *2009 IEEE Symposium on Computers and Communications*. 2009, p. 557-563. doi : 10.1109/ISCC.2009.5202224. url : https://ieeexplore.ieee.org/document/5202224 (visité le 22/03/2021).

[Alv+17] Rafael Alvarez et al. “Algorithms for Lightweight Key Exchange”. In : *Sensors* 17.7 (2017). issn : 1424-8220. doi : 10.3390/s17071517. url : https://www.mdpi.com/1424-8220/17/7/1517 (visité le 22/03/2021).

[BC11] Tapolina Bhattasali et Rituparna Chaki. “A survey of recent intrusion detection systems for wireless sensor network”. In : *International conference on network security and applications*. Springer. 2011, p. 268-280. doi : 10.1007/978-3-642-22540-6_27. url : https://link.springer.com/chapter/10.1007%2F978-3-642-22540-6_27 (visité le 22/03/2021).

[BMZ18] LM Bach, Branko Mihaljevic et Mario Zagar. “Comparative analysis of blockchain consensus algorithms”. In : *2018 41st International Convention on Information and Communication Technology, Electronics and Microelectronics (MIPRO)*. IEEE. 2018, p. 1545-1550. doi : 10.23919/MIPRO.2018.8400278. url : https://ieeexplore.ieee.org/document/8400278 (visité le 22/03/2021).

[BPR01] Fabio Bellifemine, Agostino Poggi et Giovanni Rimassa. “Developing multi-agent systems with a FIPA-compliant agent framework”. In : *Software : Practice and Experience* 31.2 (2001), p. 103-128. doi : 10.1002/1097-024X(200102)31:2<103::AID-SPE358>3.0.CO;2-O. url : https://onlinelibrary.wiley.com/doi/10.1002/1097-024X(200102)31:2%3C103::AID-SPE358%3E3.0.CO;2-O (visité le 22/03/2021).

[Bri07] Robert Bridson. “Fast Poisson disk sampling in arbitrary dimensions.” In : *SIGGRAPH sketches* 10 (2007), p. 1. url : https://www.cs.ubc.ca/~rbridson/docs/bridson-siggraph07-poissondisk.pdf (visité le 08/04/2021).

[Bui+17] Duy-Hieu Bui et al. “AES datapath optimization strategies for low-power low-energy multisecurity-level internet-of-things applications”. In : *IEEE Transactions on Very Large Scale Integration (VLSI) Systems* 25.12 (2017), p. 3281-3290. doi : 10.1109/TVLSI.2017.2716386. url : https://ieeexplore-ieee-org.gaelnomade-1.grenet.fr/abstract/document/7970126 (visité le 26/05/2021).

[Cal+18] Davide Calvaresi et al. “Multi-Agent Systems and Blockchain : Results from a Systematic Literature Review”. In : juin 2018. doi : 10.1007/978-3-319-94580-4_9. url : https://link.springer.com/chapter/10.1007%2F978-3-319-94580-4_9 (visité le 22/03/2021).

[DJM+19] Arthur Darroux, Jean-Paul Jamont, Annabelle Mercier et al. “An Energy Aware Approach to Trust Management Systems for Embedded Multi-Agent Systems”. In : *International Workshop on Software Engineering for Resilient Systems*. Springer. 2019, p. 121-137. doi : 10.1007/978-3-030-30856-8_9. url : https://link.springer.com/chapter/10.1007%2F978-3-030-30856-8_9 (visité le 22/03/2021).

[Fal+19] Sara Falcone et al. “Blockchain Design for an Embedded System”. In : *Ledger* (2019). doi : 10.5195/ledger.2019.172. url : http://ledger.pitt.edu/ojs/ledger/article/view/172 (visité le 22/03/2021).

[Fou11] Apostolos P Fournaris. “Distributed threshold cryptography certification with no trusted dealer”. In : *Proceedings of the International Conference on Security and Cryptography*. IEEE. 2011, p. 400-404. url : https://ieeexplore-ieee-org.gaelnomade-1.grenet.fr/abstract/document/6732422 (visité le 22/03/2021).

[GCB06] Miguel Escrivá Gregori, Javier Palanca Cámara et Gustavo Aranda Bada. “A jabber-based multi-agent system platform”. In : *Proceedings of the fifth international joint conference on Autonomous agents and multiagent systems*. 2006, p. 1282-1284. doi : 10.1145/1160633.1160866. url : https://dl.acm.org/doi/10.1145/1160633.1160866 (visité le 22/03/2021).

[GF00] Olivier Gutknecht et Jacques Ferber. “Madkit : A generic multi-agent platform”. In : *Proceedings of the fourth international conference on Autonomous agents*. 2000, p. 78-79. doi : 10.1145/336595.337048. url : https://dl.acm.org/doi/10.1145/336595.337048 (visité le 22/03/2021).

[HGN20] Mohammadreza Hazhirpasand, Mohammad Ghafari et Oscar Nierstrasz. “Java Cryptography Uses in the Wild”. In : *Proceedings of the 14th ACM/IEEE International Symposium on Empirical Software Engineering and Measurement (ESEM)*. 2020, p. 1-6. doi : 10.1145/3382494.3422166. url : https://dl.acm.org/doi/10.1145/3382494.3422166 (visité le 22/03/2021).

[HM19] Shihab S Hazari et Qusay H Mahmoud. “Comparative evaluation of consensus mechanisms in cryptocurrencies”. In : *Internet Technology Letters* 2.3 (2019), e100. doi : 10.1002/itl2.100. url : https://onlinelibrary.wiley.com/doi/full/10.1002/itl2.100 (visité le 22/03/2021).

[HS19] Delphi Hanggoro et Riri Fitri Sari. "A Review of Lightweight Blockchain Technology Implementation to the Internet of Things". In : *2019 IEEE R10 Humanitarian Technology Conference (R10-HTC)(47129)*. IEEE. 2019, p. 275-280. doi : 10.1109/R10-HTC47129.2019.9042431. url : https://ieeexplore.ieee.org/document/9042431 (visité le 22/03/2021).

[Jam05] Jean-Paul Jamont. "DIAMOND : Une approche pour la conception de systèmes multi-agents embarqués". Thèse de doct. Institut National Polytechnique de Grenoble-INPG, 2005. url : https://tel.archives-ouvertes.fr/tel-00189046 (visité le 22/03/2021).

[JOL10] J.-P. Jamont, M. Occello et A. Lagrèze. "A multiagent approach to manage communication in wireless instrumentation systems". In : *Measurement* 43.4 (2010), p. 489-503. issn : 0263-2241. doi : 10.1016/j.measurement.2009.12.018. url : https://www.sciencedirect.com/science/article/pii/S0263224109002668 (visité le 22/03/2021).

[KAK17] Olaide O Kazeem, Olubiyi O Akintade et Lawrence O Kehinde. "Comparative study of communication interfaces for sensors and actuators in the cloud of internet of things". In : *Int. J. Internet Things* 6.1 (2017), p. 9-13. url : https://www.researchgate.net/publication/318054538_Comparative_Study_of_Communication_Interfaces_for_Sensors_and_Actuators_in_the_Cloud_of_Internet_of_Things (visité le 02/06/2021).

[Kap+17] Aleksandr Kapitonov et al. "Blockchain-based protocol of autonomous business activity for multi-agent systems consisting of UAVs". In : *2017 Workshop on Research, Education and Development of Unmanned Aerial Systems (RED-UAS)*. IEEE. 2017, p. 84-89. doi : 10.1109/RED-UAS.2017.8101648. url : https://ieeexplore.ieee.org/document/8101648 (visité le 22/03/2021).

[Kik+17] Kaito Kikuchi et al. "Stochastic communication protocols for multi-agent consensus under jamming attacks". In : *2017 IEEE 56th Annual Conference on Decision and Control (CDC)*. IEEE. 2017, p. 1657-1662. doi : 10.1109/CDC.2017.8263888. url : https://ieeexplore.ieee.org/document/8263888 (visité le 22/03/2021).

[KW03] Chris Karlof et David Wagner. "Secure routing in wireless sensor networks : Attacks and countermeasures". In : *Ad hoc networks* 1.2-3 (2003), p. 293-315. doi : 10.1016/S1570-8705(03)00008-8. url : https://www.sciencedirect.com/science/article/abs/pii/S1570870503000088?via%3Dihub (visité le 22/03/2021).

[Li+17] Wenting Li et al. "Securing proof-of-stake blockchain protocols". In : *Data Privacy Management, Cryptocurrencies and Blockchain Technology*. Springer, 2017, p. 297-315. doi : 10.1007/978-3-319-67816-0_17. url : https://link.springer.com/chapter/10.1007%2F978-3-319-67816-0_17 (visité le 22/03/2021).

[LW11] Allison Lewko et Brent Waters. “Decentralizing attribute-based encryption”. In : *Annual international conference on the theory and applications of cryptographic techniques*. Springer. 2011, p. 568-588. doi : 10.1007/978-3-642-20465-4_31. url : https://link.springer.com/chapter/10.1007%2F978-3-642-20465-4_31 (visité le 22/03/2021).

[Meh+18] Amjad Mehmood et al. “NBC-MAIDS : Naïve Bayesian classification technique in multi-agent system-enriched IDS for securing IoT against DDoS attacks”. In : *The Journal of Supercomputing* 74.10 (2018), p. 5156-5170. doi : 10.1007/s11227-018-2413-7. url : https://link.springer.com/article/10.1007%2Fs11227-018-2413-7 (visité le 22/03/2021).

[Mel+19] Lucas Silveira Melo et al. “Python-based multi-agent platform for application on power grids”. In : *International Transactions on Electrical Energy Systems* 29.6 (2019), e12012. doi : 10.1002/2050-7038.12012. url : https://onlinelibrary.wiley.com/doi/abs/10.1002/2050-7038.12012 (visité le 22/03/2021).

[MK15] David Masad et Jacqueline Kazil. “MESA : an agent-based modeling framework”. In : *14th PYTHON in Science Conference*. Citeseer. 2015, p. 53-60. url : https://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.966.1392&rep=rep1&type=pdf (visité le 19/05/2021).

[Mor+06] Ruggero Morselli et al. *Keychains : A decentralized public-key infrastructure*. Rapp. tech. University of Maryland, College Park College Park United States, 2006. url : https://apps.dtic.mil/sti/citations/AD1006909 (visité le 22/03/2021).

[Nad+16] Sarah Nadi et al. “Jumping through hoops : Why do Java developers struggle with cryptography APIs ?” In : *Proceedings of the 38th International Conference on Software Engineering*. 2016, p. 935-946. doi : 10.1145/2884781.2884790. url : https://dl.acm.org/doi/10.1145/2884781.2884790 (visité le 22/03/2021).

[Nak+08] Satoshi Nakamoto et al. *Bitcoin : a peer-to-peer electronic cash system (2008)*. 2008. url : https://www.bitcoinpaper.info/bitcoinpaper-html/ (visité le 10/06/2021).

[Pik+15] Douglas Pike et al. “PoST White Paper”. In : *online] https://cdn.vericonomy.com/documents/VeriCoin-Proof-of-Stake-Time-Whitepaper.pdf* (2015). url : https://www.vericoin.info/downloads/VeriCoinPoSTWhitePaper10May2015.pdf (visité le 22/03/2021).

[Pim+04] João Paulo Pimentão et al. “Agent-based communication security”. In : *German Conference on Multiagent System Technologies*. Springer. 2004, p. 73-84. doi : 10.1007/978-3-540-30082-3_6. url : https://link.springer.com/chapter/10.1007%2F978-3-540-30082-3_6 (visité le 22/03/2021).

[PP21] Ochchhav Patel et Hiren Patel. "Blockchain-Based Mechanisms to Address IoT Security Issues : A Review". In : *Data Science and Intelligent Applications* (2021), p. 325-337. doi : 10.1007/978-981-15-4474-3_36. url : https://link.springer.com/chapter/10.1007%2F978-981-15-4474-3_36 (visité le 22/03/2021).

[Qin+20] Bo Qin et al. "Cecoin : A decentralized PKI mitigating MitM attacks". In : *Future Generation Computer Systems* 107 (2020), p. 805-815. doi : 10.1016/j.future.2017.08.025. url : https://www.sciencedirect.com/science/article/abs/pii/S0167739X17318381?via%3Dihub (visité le 22/03/2021).

[Rah17] Robbi Rahim. "Man-in-the-middle-attack prevention using interlock protocol method". In : *ARPN J. Eng. Appl. Sci* 12.22 (2017), p. 6483-6487. url : http://www.arpnjournals.org/jeas/research_papers/rp_2017/jeas_1117_6511.pdf (visité le 22/03/2021).

[Ren14] Larry Ren. "Proof of stake velocity : Building the social currency of the digital age". In : *Self-published white paper* (2014). url : https://www.cryptoground.com/storage/files/1528454215-cannacoin.pdf (visité le 22/03/2021).

[SB18] Ankush Singla et Elisa Bertino. "Blockchain-based PKI solutions for IoT". In : *2018 IEEE 4th International Conference on Collaboration and Internet Computing (CIC)*. IEEE. 2018, p. 9-15. doi : 10.1109/CIC.2018.00-45. url : https://ieeexplore.ieee.org/document/8537812 (visité le 22/03/2021).

[Tak+08] Atushi Takeda et al. "A new authentication method with distributed hash table for p2p network". In : *22nd International Conference on Advanced Information Networking and Applications-Workshops (aina workshops 2008)*. IEEE. 2008, p. 483-488. doi : 10.1109/WAINA.2008.203. url : https://ieeexplore.ieee.org/document/4482962 (visité le 22/03/2021).

[TV09] Chanatip Tumrongwittayapak et Ruttikorn Varakulsiripunth. "Detecting sinkhole attack and selective forwarding attack in wireless sensor networks". In : *2009 7th International Conference on Information, Communications and Signal Processing (ICICS)*. IEEE. 2009, p. 1-5. doi : 10.1109/ICICS.2009.5397594. url : https://ieeexplore.ieee.org/document/5397594 (visité le 22/03/2021).

[Won+18] Jongho Won et al. "Decentralized public key infrastructure for internet-of-things". In : *MILCOM 2018-2018 IEEE Military Communications Conference (MILCOM)*. IEEE. 2018, p. 907-913. doi : 10.1109/MILCOM.2018.8599710. url : https://ieeexplore.ieee.org/document/8599710 (visité le 22/03/2021).

[Wu+15] Cheng Wu et al. “An ECC crypto engine based on binary edwards elliptic curve for low-cost RFID tag chip”. In : *2015 IEEE 11th International Conference on ASIC (ASICON)*. IEEE. 2015, p. 1-4. DOI : 10.1109/ASICON.2015.7517207. URL : https://ieeexplore-ieee-org.gaelnomade-1.grenet.fr/abstract/document/7517207 (visité le 26/05/2021).

[YSS18] Alexander Yakubov, Wazen Shbair et Radu State. “BlockPGP : A blockchain-based framework for PGP key servers”. In : *2018 Sixth International Symposium on Computing and Networking Workshops (CANDARW)*. IEEE. 2018, p. 316-322. DOI : 10.1109/CANDARW.2018.00065. URL : https://ieeexplore.ieee.org/document/8590919 (visité le 22/03/2021).