Rapport d'état de l'art

Adrian Bonnet

09 03 2021

Sécurisation des communications dans un système multi-agents embarqués *encadré par* Oum-El-Kheir Aktouf*,* Arthur Baudet *et* Annabelle Mercier*, LCIS, année universitaire 2020-2021*

But du document

Dans ce document, nous allons d'abord revenir rapidement sur le contexte des systèmes multi-agents embarqués auxquels on s'intéresse, puis nous ferons un tour des solutions techniques existantes au problème de confidentialité et d'intégrité, notamment autour de la Blockchain, ensuite nous verrons plusieurs attaques possibles sur un système multi-agents embarqués, enfin nous comparerons les outils existants pour simuler un système multi-agents.

Sommaire

1 Propriétés des systèmes multi-agents

Le système multi-agent (SMA) que nous cherchons à modéliser est un système dans lequel les communications se font sans fil [JOL10]. Il peut être comparé à celui d'un réseau de véhicules communicants. 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 ;

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-agents possède déjà un système de gestion de confiance, ou Trust Management System (TMS) [DJM+19]. On cherche donc à réaliser une couche de sécurité sur laquelle pourra reposer le TMS.

Nous pouvons maintenant faire abstraction du cas particulier des véhicules communicants, pour appliquer une solution de sécurisation des communications à un SMA qui respecte ces critères.

[1] il est question ici de mobilité géographique

2 Solutions de sécurité

Afin de garantir la confidentialité et l'intégrité des communication on cherche à obtenir un secret partagé entre deux agents. En particulier, on souhaite se prémunir contre les tentatives d'usurpation d'identité et d'écoute intempestive. Pour cela, on cherche à s'assurer que la clé publique, ou le certificat, de l'interlocuteur n'a pas été modifiée.

2.1 Mécanismes de vérification partielle

On peut utiliser des méthodes de redondance pour la vérification de l'identité de l'interlocuteur. Ces mécanismes peuvent être mis en place tout au long de l'échange ou uniquement lors de l'échange de clés. Ces méthodes 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, comme dans une attaque Sybil par exemple, de telle sorte qu'il n'existe aucun chemin entre le ou les agents attaqués et le reste du système, alors la redondance ne permet 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 sera alors nécessaire d'avoir recours à un système de détection d'intrusion, ou Intrusion Detection System (IDS), tout en préservant la décentralisation. [Meh+18]

2.1.1 PGP TripleCheck

Le PGP TripleCheck est une adaptation à un SMA de la méthode DoubleCheck [AK09] qui peut être utilisée sur Internet pour faire la détection d'attaque. Le principe est de réaliser une requête via le réseau classique ainsi qu'une requête via le réseau TOR. Chaque requête récupère le certificat du serveur que l'on cherche à contacter : si les deux requêtes possèdent des résultats différents, c'est que l'une des deux est soumise à une attaque.

Puisqu'il n'existe pas de réseau parallèle de type TOR dans un SMA, l'idée du TripleCheck est de réaliser les requêtes par des chemins différents. Si l'agent A cherche à obtenir le certificat de l'agent B, alors il va faire une première requête directe, puis il va demander à un agent E de faire une autre requête vers B, puis récupérer le résultat. Afin d'éviter au maximum les nœuds en commun sur le chemin, A va choisir E tel que ABE forme (à peu près) un triangle rectangle isocèle en E.

De plus, pour ajouter de la correction d'erreur à la méthode, l'agent A va demander à un second agent G de faire une requête à B puis récupérer le résultat, afin de réaliser de la redondance modulaire triple : si l'une des deux requêtes diffère des deux autres, alors il y a de grandes chances qu'il y ait un attaquant sur le chemin associé.

Cette méthode implique que l'agent A est en mesure de trouver E et G dans le système, il faut donc que A possède une bonne connaissance du routage de tout le système.

2.1.2 Split and Merge

Le Split and Merge [Pim+04] consiste en une fragmentation des messages entre les agents qui souhaitent échanger. Cette fragmentation n'a pas lieu uniquement lors de l'échange de clés publiques, mais tout le long de la communication. L'agent émetteur va définir pour chaque fragment un nombre de sauts, Number of Hops to GO (NHG) et envoyer chaque fragment à un agent aléatoire du système. Chaque agent qui reçoit un fragment va décrémenter le NHG et transmettre à un autre agent aléatoire. Lorsque le NHG vaut zéro, l'agent doit transmettre le fragment à l'agent destinataire.

Le Split and Merge ne permet pas de s'assurer que le certificat reçu est authentique, mais il assure la confidentialité dans la mesure où seule la destination a connaissance de l'intégralité des données. Si le Split and Merge est utilisé sans chiffrement, chaque agent qui transmet un ou plusieurs fragments possède des connaissances non-nulles sur le contenu des messages. L'intégrité n'est pas assurée par le Split and Merge, et l'utilisation d'un mécanisme supplémentaire comme le hachage cryptographique ou la signature n'est pas évident car il n'est pas simple de transmettre cette valeur de contrôle sans risque de la fragmenter.

2.1.3 Forced-Latency Interlock

Interlock [Rah17] est un protocole de communication dont le but est de rendre plus difficile l'écoute intempestive. Le fonctionnement consiste à envoyer les messages chiffrés en deux fois : le premier agent envoie d'abord la première moitié de son message, lorsqu'il le reçoit le second agent envoie la première moitié de son message, et lorsqu'il le reçoit, le premier agent peut envoyer la seconde moitié de son message. Pour utiliser Interlock, il est nécessaire d'avoir réalisé un échange de clé au préalable.

Le Forced-Latency Interlock consiste à attendre un temps T connu des deux agents avant de répondre. Cette modification permet de rendre le protocole résistant à l'attaque Man-in-the-Middle (MitM). Le diagramme 1 page 6 illustre son fonctionnement.

Interlock permet d'assurer la confidentialité mais pas l'authenticité des données, il est donc nécessaire d'utiliser une autre méthode pour l'échange de clés.

2.2 Solutions purement cryptographiques

La cryptographie permet également de vérifier la provenance d'une clé publique, grâce à la participation à un secret partagé par exemple. Ces solutions pourraient être les plus adaptées car elles offrent la possibilité de décentraliser la vérification de certificats, mais les outils maniés doivent être développés pour pouvoir être utilisés.

2.2.1 Attribute Based Encryption

L'Attribute Based Encryption (ABE) est une solution cryptographique dans laquelle chaque agent possède des attributs qui lui sont propres par exemple

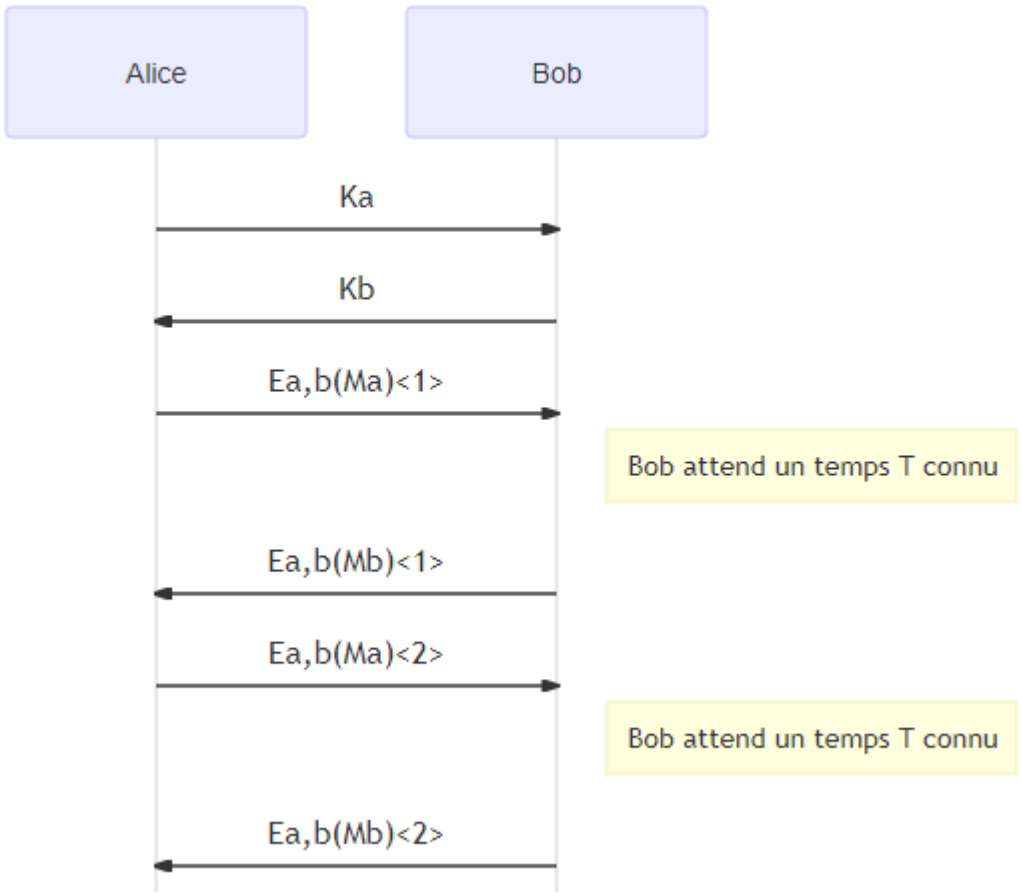

Figure 1: Diagramme d'échange avec Interlock en latence forcée.

l'emplacement, le fournisseur, le propriétaire, etc. La clé publique de chaque agent est calculable à partir de ses attributs par n'importe quel autre agent. Un tiers de confiance centralisé, Private Key Generator (PKG), publie sa clé publique (*public master key*) et garde sa clé privée (*private master key*) pour lui. Chaque agent doit faire une requête pour récupérer sa clé privée auprès du PKG. Le PKG forge la clé d'un agent à partir de la *private master key* et des attributs de l'agent. [LW11]

Le PKG est donc une autorité qui a connaissance des clés privées de tous les agents. Cette autorité peut cependant être détruite lorsque tous les agents ont reçu leur clé privée. Il n'est alors plus possible d'ajouter de nouveaux agents dans le système.

Pour révoquer les clés, une unité de temps (mois, année) doit être incluse dans les attributs des agents. En utilisant cet attribut, il n'est plus possible de détruire le PKG, car il est nécessaire pour attribuer de nouveaux certificats aux agents une fois les précédents ont expirés.

Pour décentraliser le PKG, il faut faire en sorte qu'aucun agent ne connaisse à lui seul la *private master key* et qu'aucun agent ne connaisse à lui seul la clé privée d'un autre agent, tout en gardant un PKG capable de forger et envoyer les clés privées, ce qui implique que les clés privées doivent être calculées et envoyées en plusieurs morceaux tout en restant fonctionnelles.

Afin d'assurer l'authenticité de la requête de récupération de clé privée, il faut que les attributs servent également de localisateur pour l'agent, comme une adresse IP ou mail. En revanche, il n'est pas possible de garantir la confidentialité lors de la récupération de la clé, car l'agent auquel elle doit être remise ne possède pas encore sa clé publique.

La solution pourrait être que la clé publique soit générée à partir des attributs de manière irréversible, c'est-à-dire de telle sorte qu'il soit impossible de calculer les attributs à partir de la clé publique. On pourrait alors avoir une méthode pour qu'un agent puisse calculer sa clé privée à partir de ses attributs sans jamais les divulguer. Ceci permettrait l'identification mais pas l'authentification.

2.2.2 Threshold and multi-party computing

Les cryptosystèmes à seuil (`threshold`) sont des solutions cryptographiques qui permettent de répartir un secret entre plusieurs participants, de manière à ce que pour reconstruire le secret, il soit nécessaire qu'un nombre minimum des participants se mettent d'accord.

Cette méthode peut être utilisée pour déchiffrer les messages reçus ou signer les messages émis avec une clé privée, mais elle n'est pas pensée pour établir une clé de session entre deux agents, puisqu'aucun agent ne connaît sa clé privée.

Un système à seuil pourrait être utilisé pour créer une *master secret key*, puis grâce au calcul multipartite sécurisé (*multi-party computing*) il serait possible de créer les couples de clés privée/publique [Fou11]. Cette approche nécessite la conception d'outils cryptographiques.

2.3 Autorités de certification distribuées

Pour réaliser une autorité de certification dans un environnement décentralisé, on peut chercher à la distribuer 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.

2.3.1 Distributed Hash Table Certificate Authority

Une table de hachage distribuée, ou Distributed Hash Table (DHT) est une structure de données répartie sur plusieurs nœuds dans un réseau pair-à-pair. Les données de la structure sont découpées en morceaux, dit chunks, de tel sorte qu'il soit possible de récupérer les données à partir de différents nœuds en même temps. Chaque agent du système possède un fichier, dit torrent, qui contient le hachage cryptographique de chaque chunk, afin de garantir l'intégrité des données de la DHT.

Il est alors possible d'utiliser une DHT pour stocker les certificats des agents du système. De cette manière, il n'y aurait plus besoin de procéder à un échange de clé, car chaque agent peut accéder à la clé publique de n'importe quel autre agent, et vérifier grâce au hachage cryptographique qu'il connaît que la clé n'a pas été modifiée. [Tak+08]

Le problème de cette solution est qu'il n'est pas possible d'ajouter de nouveaux certificats dans la DHT sans modifier le fichier torrent sur tous les agents. De plus, le fichier torrent doit être créé avant l'initialisation du système, pour contenir toutes les clés publiques.

2.3.2 Blockchain Certificate Authority

La Blockchain 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. De cette manière, chaque agent peut accéder à la clé publique de tout autre agent sans faire de requête. [Cal+18] [SB18] [Won+18]

Pour ajouter des blocs dans la Blockchain, il est nécessaire de parvenir à un consensus sur la validité des informations que contiendra le bloc. Les méthodes existantes pour parvenir à un consensus dans la Blockchain sont diverses et font l'objet de la prochaine section.

2.4 Comparaison des solutions

Pour comparaiser ces solutions, on utilise les critères suivants :

- assure la **confidentialité** ;
- assure l'**intégrité** ;

- permet l'**ajout** de nouveaux agents ;
- permet la **révocation** ;
- **léger**.

Voici le tableau de comparaison obtenu :

Table 1: Comparaison des solutions de sécurité pour les SMAE.

Solution	Confidentialité	Intégrité	Ajout	Révocation	Léger
PGP TripleCheck	~	~	+	+	+
Split and Merge	~	−	+	+	+
Forced-Latency Interlock	−	−	+	+	+
ABE	~	~	+	+	+
Threshold + multi-party	+	+	+	+	+
CA DHT	+	+	−	−	~
CA Blockchain	+	+	+	+	~

Légende :

− le critère n'est pas respecté

~ le critère est partiellement respecté, ou respecté sous certaines conditions

+ le critère est entièrement respecté

3 Consensus

3.1 Méthodes de consensus

Le problème de la Blockchain est qu'il est nécessaire d'atteindre un consensus sur la branche à utiliser. Pour atteindre ce consensus, on propose aux agents qui le souhaitent de réaliser ce travail de vérification et d'ajouts de bloc, souvent en échange d'une récompense.

Les méthodes pour atteindre le consensus se répartissent en deux catégories : celles par **preuve**, dans lesquelles l'un des participants désigné vainqueur pour avoir résolu un challenge gagne le droit de valider l'un des blocs et d'empocher une récompense, en général une certaine somme dans une cryptomonnaie, et celles par **vote**, dans lesquelles pour chaque bloc, l'ensemble des participants contribuent à la validation ou au rejet d'un bloc.

Les approches par preuve sont :

Proof-of-Authority (PoA) [HM19] Les participants qui valident les blocs font partie d'une liste de confiance.

Proof-of-Capacity (PoC) [HM19] Les participants qui souhaitent valider un bloc résolvent un challenge qui nécessite peu de calcul mais beaucoup d'espace mémoire, le premier qui réalise le challenge valide le bloc (cf. PoW).

Proof-of-Hold (PoH) [PP21] Les participants qui souhaitent valider un bloc mettent en jeu une quantité d'argent, l'argent mis en jeu perd de la valeur s'il est échangé (cf. PoS).

Proof-of-Importance (PoI) [BMZ18] [HM19] [PP21] Les participants possèdent une réputation basée sur la valeur qu'ils mettent en jeu et la réputation de ceux avec lesquels ils échangent de l'argent, lorsqu'un bloc validé, la réputation de tous les participants autres que le validateur augmente (cf. PoS).

Proof-of-Luck (PoL) [BMZ18] La réputation des participants (ou *chance*) est répartie aléatoirement entre tous les participants, chaque participant peut être choisi pour valider un bloc proportionnellement à sa chance.

Proof-of-Minimum Aged Stake (PoMAS) [PP21] Les participants qui possèdent un enjeu supérieur au seuil minimum requis ont la même probabilité de valider un bloc (cf. PoS).

Proof-of-Personhood (PoP) Les participants prouvent qu'ils représentent un acteur humain, par exemple via un CAPTCHA.

Delegated Proof-of-Stake (dPoS) [BMZ18] [HM19] À chaque tour, c'est-à-dire pour chaque bloc à valider, les participants du système peuvent miser pour voter pour un *délégué*. Plus un délégué possède de mises sur lui, plus il a de chances de valider un bloc. Les délégués changent à chaque tour. Si un délégué valide un bloc, la récompense est partagée avec les participants qui ont voté pour lui au prorata des mises.

Proof-of-Stake (PoS) [BMZ18] [HM19] Les participants qui souhaitent valider un bloc mettent en jeu une partie de leur argent. Le validateur est choisi proportionnellement à la somme mise en jeu. Pendant une phase de vérification, si l'on se rend compte que le bloc n'est pas valide, le validateur perd sa mise. Sinon, il gagne une récompense de validation.

Proof-of-Stake/Time(PoST) [Pik+15] Les participants qui valident les blocs sont choisis alternativement parmi les utilisateurs qui possèdent une grande mise depuis longtemps, et ceux qui en possèdent peu (la probabilité en fonction de l'âge suit une loi du χ^2) (cf. PoS).

Proof-of-Stake-Velocity(PoSV) [Ren14] Les participants qui valident les blocs sont choisis parmi les utilisateurs avec la meilleure *vitesse de circulation* (velocity) de la monnaie (soit "le nombre de fois qu'une même unité de monnaie permet de régler des transactions") (cf. PoS).

Proof-of-Use(PoU) Les participants qui valident les blocs sont choisis parmi ceux qui font le plus de transactions.

Proof-of-Work(PoW) [PP21] [HM19] Le participant sélectionné pour valider un bloc est le premier à résoudre un challenge cryptographique nécessitant beaucoup de calculs, on appelle ces participants des mineurs.

Proof-of-eXercice(PoX) [BMZ18] Le participant sélectionné pour valider un bloc est le premier à résoudre un challenge utile à la communauté scientifique, par exemple le repliement des protéines (cf. PoW).

Les approches par vote sont :

Delegated Byzantine Fault Tolerance (DBFT) [BMZ18] [HM19] Chaque participant vote pour un *délégué*, les délégués votent pour valider un bloc, le bloc est validé si plus de la majorité des délégués votent pour. C'est une légère amélioration de la Practical Byzantine Fault Tolerance (PBFT).

Ripple Consensus Protocol Algorithm (RPCA) [BMZ18] Les blocs qui sont votés vrais à 80% par des serveurs définis sont validés.

Stellar Consensus Protocol (SCP) [BMZ18] La validation des blocs se fait en deux temps : pendant la *nomination*, les participants votent pour choisir un bloc parmi la liste des blocs à valider, ensuite pendant le *scrutin*, les participants votent pour valider ou rejeter le bloc en question.

3.2 Méthodes applicables

La plupart de ces méthodes sont basées sur l'utilisation de cryptomonnaies : la récompense lors de la validation d'un bloc et la mise en jeu pour participer sont des valeurs de la monnaie. Dans le cadre du système multi-agents, nous utiliserons une autre valeur : la **réputation**.

La réputation que l'on définit, dans le cadre de la Blockchain dans un SMA, est une valeur non-cessible, c'est-à-dire qu'on ne peut pas la transférer 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.

Nous devons donc sélectionner parmi les méthodes listées ci-dessus celles qui pourraient convenir à ce fonctionnement, grâce à deux critères :

- La **quantité de ressources** nécessaire sur les nœuds du système pour mettre en place la méthode ;
- La manière utilisée pour **sélectionner les votants**.

Ces critères sont utilisés dans le tableau 2 ci-dessous. Si le critère n'est pas rempli, le contenu de la case justifie pourquoi.

Table 2: Classification des méthodes de consensus.

Consensus	Quantité de ressources	Sélection des votants	Les deux
PoA	OK	Liste de confiance	
PoC	Beaucoup de mémoire	OK	
PoH	OK	Temps de possession	
PoI	OK	Echange de valeurs	
PoL	OK	OK	OK
PoMAS	OK	OK	OK
PoP	OK	Validateur humain	
dPoS	OK	OK	OK
PoS	OK	OK	OK
PoST	OK	Temps de possession	
PoSV	OK	Echange de valeurs	
PoU	OK	Echange de valeurs	
PoW	Beaucoup de calcul	OK	
PoX	Beaucoup de calcul	OK	
DBFT	OK	OK	OK
RPCA	OK	Serveurs validateurs	
SCP	OK	OK	OK

Termes utilisés :

Beaucoup de calcul il est nécessaire que les agents réalisent un calcul nécessitant beaucoup de ressources (énergie et mémoire vive) ;

Beaucoup de mémoire il est nécessaire que les agents possèdent une grande quantité de mémoire ;

Liste de confiance certains agents font partie d'une liste blanche de confiance alors que d'autres non ;

Validateur humain pour valider chaque bloc dans la blockchain, un humain doit confirmer ;

Serveurs validateurs les validateurs sont des serveurs centralisés et non des agents du système ;

Echange de valeurs les agents ont besoin d'échanger entre eux une valeur, généralement une monnaie (incompatible avec de la réputation non échangeable) ;

Temps de possession le temps de possession d'une valeur, généralement une monnaie, est calculé et réinitialisé lors de l'échange de cette valeur (incompatible avec de la réputation non échangeable).

3.3 Fonctionnement modifié

Les méthodes de consensus par preuve qui répondent à ces critères doivent être adaptées au fonctionnement avec réputation. Voici la description des méthodes qui pourraient convenir dans le cadre d'un système multi-agents embarqués :

Proof of Stake (PoS) Les agents qui souhaitent valider un bloc mettent en jeu une partie de leur réputation. Le validateur est choisi proportionnellement à la réputation mise en jeu. Pendant une phase de vérification, si l'on se rend compte que le bloc n'est pas valide, le validateur perd sa mise. Sinon, il gagne une récompense de validation.

Delegated Proof of Stake (dPoS) À chaque tour, c'est-à-dire pour chaque bloc à valider, les agents du système peuvent miser de la réputation pour voter pour un délégué. Plus un délégué possède de mise sur lui, plus il a de chances de valider un bloc. Les délégués changent à chaque tour. Si un délégué valide un bloc, la récompense est partagée avec les agents qui ont voté pour lui au prorata des mises.

Proof of Minimum Aged Stake (PoMAS) Tous les agents qui possèdent une réputation supérieure au seuil minimum requis peuvent être choisis pour valider un bloc avec la même probabilité.

Proof of Luck (PoL) La réputation des agents (ou chance) est répartie aléatoirement entre tous les agents, chaque agent peut être choisi pour valider un bloc proportionnellement à sa chance.

Les méthodes de consensus par vote n'ont pas besoin d'être adaptées à un fonctionnement avec réputation. Les méthodes qui répondent aux critères précédents sont :

Delegated Byzantine Fault Tolerance (DBFT) Chaque agent vote pour un délégué, les délégués votent pour valider un bloc, le bloc est validé si plus de la majorité des délégués votent pour.

Stellar Consensus Protocol (SCP) La validation des blocs se fait en deux temps : pendant la nomination, les agents votent pour choisir un bloc parmi la liste des blocs à valider, ensuite pendant le scrutin, les agents votent pour valider ou rejeter le bloc en question.

4 Modèle d'attaque

4.1 Classes de modèles

Dans un système multi-agents, on peut répartir les modèles d'attaque en deux classes selon la capacité de calcul dont on suppose que l'attaquant est doté. On distingue alors les **mote-class attackers** qui possèdent des agents d'une capacité de calcul comparable aux agents normaux du système, et les **laptop-class attackers** qui parviennent à simuler des agents du système à partir de machines telles que des ordinateurs personnels ou plus [KW03].

4.2 Attaques possibles

Plusieurs attaques peuvent se produire dans un système multi-agents tel que nous l'avons défini. Ces attaques ont pour but l'écoute intempestive pour récolter des informations, la modification des communications pour modifier le comportement des agents ou la mise hors-service des communications du système.

51% attack (ou attaque par majorité) attaque sur une Blockchain, le ou les attaquants possèdent plus de la majorité des agents ou des ressources et peuvent prendre le contrôle du système. Pour s'en prémunir, on cherche à limiter les possibilités de création d'un nouvel agent et limiter les possibilités d'agréger des ressources.

Acknowledgement spoofing [KW03] attaque sur un réseau de capteurs, l'attaquant envoie systématiquement des Acknowledgment afin de faire croire que des routes mortes sont de bonne qualité et aussi provoquer de la congestion. Pour s'en prémunir, on peut utiliser un IDS.

DoS (déni de service) attaque sur n'importe quelle architecture, l'attaquant inonde l'architecture ou s'en prend à un Single-Point-Of-Failure (SPOF), ou point de défaillance unique. Il peut aussi utiliser une armée d'agents corrompus, on parle alors de DDos. Pour s'en prémunir, on cherche à limiter le nombre et l'impact de SPOF.

Eavesdropping (écoute intempestive) attaque sur n'importe quel réseau, 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. Pour s'en prémunir, on cherche à chiffrer les communications.

`HELLO` flood attack [KW03] [TV09] attaque sur un réseau de capteurs, l'attaquant augmente momentanément sa portée afin de s'annoncer comme étant accessible de plus d'agents que dans la réalité, puis induire les agents à envoyer des paquets à une destination hors de portée. Pour s'en prémunir, on peut utiliser un IDS.

Jamming attack (brouillage) [Kik+17] [TV09] attaque une communication sans fil, l'attaquant émet un signal brouillé sur la porteuse pour empêcher les agents autour de lui de communiquer. Pour s'en prémunir, on peut essayer de trouver un autre chemin si c'est possible.

Long-range attack [Li+17] attaque sur une Blockchain avec preuve d'enjeu, l'attaquant parvient à compromettre un grand nombre de comptes qui ne sont plus utilisés et réalise un fork sur la Blockchain à un moment dans le passé où la somme de ces comptes représentait une part conséquente de l'enjeu.

Man-in-the-Middle [Qin+20] attaque sur n'importe quel réseau, 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 aussi déchiffrer les communications. Pour s'en prémunir, on cherche à authentifier les communications, on cherche notamment à éviter le TOFU.

Nothing-at-Stake (N@S) [Li+17] attaque sur une Blockchain avec preuve d'enjeu, l'agent malveillant qui a l'occasion de valider un bloc a tout intérêt à miser sur différents forks, afin de gagner la mise quel que soit le fork choisi par le consensus, ce qui ralentit le temps de consensus.

Routing information attack [KW03] [TV09] attaque sur un réseau de capteurs, l'attaquant modifie les informations de routage pour induire un problème de disponibilité. Pour s'en prémunir, on peut utiliser un IDS.

Selective forwarding [KW03] [TV09] attaque sur un réseau de capteurs, l'attaquant transmet certains paquets et droppe les autres, pour empêcher les communications sans passer pour un agent mort. Pour s'en prévenir, on peut utiliser un TMS.

Side channel attack (attaque par canaux cachés) attaque sur un composant matériel, l'attaquant mesure une grandeur physique (e.g. courant électrique) liée à l'activité et cherche à établir un lien entre les mesures et les opérations effectuées, afin de déterminer la valeur d'une clé par exemple. L'attaquant peut aussi chercher à injecter des fautes dans le composant, par un laser ou en changeant le signal de l'horloge par exemple, on parle alors d'attaque en faute. Pour s'en prémunir, on peut chercher à concevoir des algorithmes robustes, ou à limiter l'accès aux composants matériels.

Sinkhole attack [TV09] attaque sur un réseau de capteurs, l'agent malveillant augmente sa sphère d'influence et attire le trafic de ses voisins, pour avoir plus de connaissances et user les ressources de ses voisins. Pour s'en prémunir, on peut utiliser la réputation.

Sleep deprivation [BC11] attaque sur un SMA embarqué, l'attaquant va envoyer régulièrement des messages à d'autres agents pour les empêcher de rentrer dans un mode d'économie d'énergie.

Sybil attack [TV09] 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. Pour s'en prémunir, on cherche à limiter les possibilités de création d'un nouvel agent et authentifier les communications.

White washing [DJM+19] attaque sur un système de confiance, l'agent malveillant quitte le système lorsqu'il pense avoir été découvert, puis le rejoint avec un nouvel identifiant. Pour s'en prémunir, on peut limiter les possibilités de création d'un nouvel agent.

Wormhole attack [TV09] attaque sur un réseau de capteurs, l'agent malveillant transmet les paquets qu'il reçoit sur un autre support de communication que celui utilisé par le réseau. Pour s'en prémunir, on cherche à définir une distance maximale que peut parcourir un paquet (une *laisse*) ou mesurer le nombre de paquets émis et reçus par les agents sur le réseau.

4.3 Classification des attaques

Pour différencier ces attaques, nous utiliserons les critères suivants :

niveau À quel niveau se situe l'attaque ? Dans l'ordre du plus bas au plus haut : *physique*, *hardware*, *communication* et *software*

cible Quel est le type d'architecture visé par l'attaque ? Par exemple : *SMA*, *Blockchain*, *réseau de capteurs*

propriété de sécurité Quel aspect de sécurité est-il compromis par l'attaque ? *confidentialité*, *intégrité*, *disponibilité*, *authentification*

échelle L'attaque est-elle ciblée, c'est-à-dire qu'elle vise un ou plusieurs agents en particuliers, ne fait-elle pas de distinction entre les agents attaqués ?

La classification selon le **niveau** donne le résultat suivant :

physique La **Jamming** attack est une attaque physique. Les attaques physiques se font généralement sur les réseaux de capteurs, et portent atteinte à la disponibilité. Elles ne peuvent pas être ciblées.

hardware Les **Side channel** attacks sont des attaques hardware. Les attaques hardware se font forcément sur les composants matériels, et portent atteinte à la confidentialité ou à l'intégrité. Elles sont nécessairement ciblées puisqu'il faut avoir accès aux composants physiques.

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 et ne sont pas ciblées.

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, la Long-range attack et le Nothing-at-Stake sont exclusives à la Blockchain basée sur une PoS, la 51% attack peut se faire sur tout type de Blockchain et sur les SMA, la Sybil attack et le White washing sont exclusives à un système de confiance, la Sleep deprivation est spécifique aux SMA embarqués, le DoS, l'eavesdropping et enfin le Man-in-the-Middle s'appliquent à tout type d'architecture réseau. Elles peuvent porter atteinte à la disponibilité comme le DoS, le Nothing-at-Stake ou le Sleep deprivation, ou à l'intégrité et la confidentialité comme la Long-range attack, le Man-in-the-Middle, la Sybil attack, le White washing et la Wormhole attack. Le Sleep deprivation, le Man-in-the-Middle, l'eavesdropping et la Sybil attack sont des attaques ciblées.

4.4 Modèle retenu

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, afin d'échanger des informations par exemple. Ces communications forment alors une vulnérabilité vis-à-vis de la sécurité des agents et du système dans sa globalité.

Pour sécuriser ces communications, nous nous intéressons en particuliers aux attaques portant sur la confidentialité et l'intégrité des échanges entre les agents du système. Nous cherchons entre autres à nous prémunir de l'écoute intempestive et de la modification de ces communications. On considère donc que l'attaquant cible certaines communications 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.

Comme l'étude aura recours à la simulation pour le système multi-agents, les attaques physiques et hardware ne pourront donc pas être prises en compte.

Étant donné le délai assez court du projet, nous nous intéresserons à un modèle **mote-class**, avec une attention particulière sur les attaques de fork sur la Blockchain, l'eavesdropping, le Man-in-the-Middle et l'attaque Sybil.

5 Outils de simulation

5.1 Comparaison

Dans cette section, nous comparons des outils de simulation de systèmes multi-agents. Pour réaliser cette comparaison, nous utiliserons les critères suivants :

plateforme il possible d'outiliser l'outil pour simuler les agents

embarqué il existe des notions utiles pour les agents embarqués, comme la portée, l'énergie ou la sérialisation ;

open source le code source est ouvert ;

spécifications les communications répondent à des spécifications existantes ;

langage le langage de programmation utilisé ;

publications l'outil a été utilisé dans la communauté scientifique pour faire des recherches ;

documentation la documentation permet d'utiliser l'outil ;

popularité le nombre d'étoiles sur GitHub ;

date de sortie la date de publication de la première version.

Table 3: Tableaux de comparaison des outils de simulation de systèmes multi-agents.

Outil	**Plate-forme**	**Em-bar-qué**	**Open Source**	**Spécifica-tions**	**Langage**	**Publica-tions**	**Docu-menta-tion**	**Popu-larité** (stars)	**Date de sortie**
JADE	**oui**	non	oui	FIPA-ACL	Java	++	fournie		2000
PADE	**oui**	sériali-sation	oui	FIPA-ACL	Python	++	assez fournie	48	2015
SPADE	**oui**	non	oui	FIPA-ACL	Python	++	fournie	174	2012
KQML class	non	non	oui	KQML	Java	−	courte		
PyKQML	non	non	oui	KQML	Python	−	absente	0	2016
AgentPy	non	non	oui	non	Python	−−	fournie	39	2020
JACK	**oui**	non	non	non	Java	−	existante		< 1999
Janus	**oui**	portée	oui	non	SARL (Java/Python)	+	existante	110	2008
Jason	non	non	oui	non	Interpré-teur Java	++	existante	119	2014
GAMA	**oui**	non	oui	non	GAML (JDK)	++	fournie	194	2009
Mad-Kit	**oui**	non	oui	non	Java	++	existante	38	2011
MESA	**oui**	portée	oui	non	Python	++	fournie	1.4k	2016
NetL-ogo	**oui**	non	oui	non	NetLogo	++	existante	759	1999

Légende :

−− aucune publication

− moins de 10 publications

\+ légèrement plus de 10 publications

++ plusieurs pages de résultats dans un moteur de recherche[2]

fournie documentation existante, détaillée et facile à utiliser

existante documentation difficile à utiliser, liste exhaustive des classes et des méthodes existantes

5.2 Résultats

Il ressort de cette comparaison que JADE, PADE, SPADE, MadKit et MESA offrent tous une plateforme de simulation des agents en plus d'un framework pour le développement.

- Aucun ne supporte de fonctionnalités avancées liées aux contraintes de l'embarqué.
- Ils ont servi d'outils pour de nombreux travaux dans la communauté scientifique.
- Ils possèdent tous une documentation dense, mais pas toujours synthétique.
- JADE, PADE et SPADE ont une communication basée sur la standardisation FIPA-ACL.

Afin de comparer la facilité d'utilisation de ces solutions et de vérifier que celle qui sera retenue colle bien à ce qui est attendu pour le projet, ces cinq outils seront soumis à un test qui consiste en la réalisation d'un programme multi-agent simple.

Les solutions seront testées par ordre de vraisemblance :

1. MadKit [GF00] (déjà vu)
2. MESA [MK15] (absence d'interface de message)
3. PADE [Mel+19] (même si pas d'interface graphique)
4. JADE [BPR01] (pas de tutoriel)
5. SPADE [GCB06] (besoin de créer un serveur XMPP)

[2] le moteur de recherche utilisé est Google Scholar

5.3 Outil de simulation retenu

Les outils de simulation ont été testés sur un problème d'enchère, après installation et essai sur un programme Hello World ou Ping-Pong. Le but était d'évaluer ces outils sur la facilité de mise en place et d'utilisation et la pertinence de la documentation.

Le déroulement de l'enchère est le suivant : un agent Vendeur annonce le premier article à son prix de départ auprès de tous les agents Acheteurs. Lorsqu'il reçoit une offre dont le montant est supérieur à la dernière valeur proposée, le Vendeur met à jour la valeur de l'article puis diffuse cette offre après 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 annonce ensuite le deuxième article, ainsi de suite pour tous les articles. Lorsqu'ils reçoivent une offre, les agents Acheteurs attendent une durée aléatoire entre 0 et 10 secondes, puis si la dernière offre est issue d'un autre acheteur et qu'ils peuvent surenchérir, alors ils font une nouvelle offredont le montant est choisi aléatoirement entre le montant actuel et la somme qu'ils ont en possession, tirée au hasard entre 1000 et 10000 à l'initialisation des agents. Lorsqu'un agent Acheteur reçoit l'annonce d'un gagnant et s'il est l'agent qui a remporté l'article, alors il soustrait le montant dans l'annonce à la somme qu'il possède.

Un exemple d'exécution du problème des enchères avec MadKit est disponible à l'annexe A.

Voici les résultats des tests des outils de simulation :

MadKit facile à utiliser, documentation complète, pratique ;

MESA facile à utiliser, pratique, transmission du code simplifiée par la légèreté, traitement des données intégré ;

PADE facile à utiliser, encore en développement, faible communauté et documentation peu fournie ;

JADE vieux et non maintenu, documentation et aide difficiles à trouver, peu pratique ;

SPADE facile à utiliser, nécessite l'installation et la configuration d'un serveur XMPP local donc forte dépendance de l'environnement.

Nous avons d'abord utilisé MadKit pour la suite du projet, avant de changer pour MESA qui offre plus de légèreté.

6 Bibliographie

[AK09] M. Alicherry and A. D. Keromytis. "DoubleCheck: Multi-path verification against man-in-the-middle attacks". In: *2009 IEEE Symposium on Computers and Communications*. 2009, pp. 557–563. DOI: 10.1109/ISCC.2009.5202224. URL: https://ieeexplore.ieee.org/document/5202224 (visited on 03/22/2021).

[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 (visited on 03/22/2021).

[BC11] Tapolina Bhattasali and Rituparna Chaki. "A survey of recent intrusion detection systems for wireless sensor network". In: *International conference on network security and applications*. Springer. 2011, pp. 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 (visited on 03/22/2021).

[BMZ18] LM Bach, Branko Mihaljevic, and Mario Zagar. "Comparative analysis of blockchain consensus algorithms". In: *2018 41st International Convention on Information and Communication Technology, Electronics and Microelectronics (MIPRO)*. IEEE. 2018, pp. 1545–1550. DOI: 10.23919/MIPRO.2018.8400278. URL: https://ieeexplore.ieee.org/document/8400278 (visited on 03/22/2021).

[BPR01] Fabio Bellifemine, Agostino Poggi, and Giovanni Rimassa. "Developing multi-agent systems with a FIPA-compliant agent framework". In: *Software: Practice and Experience* 31.2 (2001), pp. 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 (visited on 03/22/2021).

[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 (visited on 04/08/2021).

[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), pp. 3281–3290. DOI: 10.1109/TVLSI.2017.2716386. URL: https://ieeexplore-ieee-org.gaelnomade-1.grenet.fr/abstract/document/7970126 (visited on 05/26/2021).

[Cal+18] Davide Calvaresi et al. "Multi-Agent Systems and Blockchain: Results from a Systematic Literature Review". In: June 2018. DOI: 10.1007/978-3-319-94580-4_9. URL: https://link.springer.com/chapter/10.1007%2F978-3-319-94580-4_9 (visited on 03/22/2021).

[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, pp. 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 (visited on 03/22/2021).

[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 (visited on 03/22/2021).

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

[GCB06] Miguel Escrivá Gregori, Javier Palanca Cámara, and 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, pp. 1282–1284. DOI: 10.1145/1160633.1160866. URL: https://dl.acm.org/doi/10.1145/1160633.1160866 (visited on 03/22/2021).

[GF00] Olivier Gutknecht and Jacques Ferber. "Madkit: A generic multi-agent platform". In: *Proceedings of the fourth international conference on Autonomous agents*. 2000, pp. 78–79. DOI: 10.1145/336595.337048. URL: https://dl.acm.org/doi/10.1145/336595.337048 (visited on 03/22/2021).

[HGN20] Mohammadreza Hazhirpasand, Mohammad Ghafari, and 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, pp. 1–6. DOI: 10.1145/3382494.3422166. URL: https://dl.acm.org/doi/10.1145/3382494.3422166 (visited on 03/22/2021).

[HM19] Shihab S Hazari and 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 (visited on 03/22/2021).

[HS19] Delphi Hanggoro and 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, pp. 275–280. DOI: 10.1109/R10-HTC47129.2019.9042431. URL: https://ieeexplore.ieee.org/document/9042431 (visited on 03/22/2021).

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

[JOL10] J.-P. Jamont, M. Occello, and A. Lagrèze. "A multiagent approach to manage communication in wireless instrumentation systems". In: *Measurement* 43.4 (2010), pp. 489–503. ISSN: 0263-2241. DOI: 10.1016/j.measurement.2009.12.018. URL: https://www.sciencedirect.com/science/article/pii/S0263224109002668 (visited on 03/22/2021).

[KAK17] Olaide O Kazeem, Olubiyi O Akintade, and 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), pp. 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 (visited on 06/02/2021).

[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, pp. 84–89. DOI: 10.1109/RED-UAS.2017.8101648. URL: https://ieeexplore.ieee.org/document/8101648 (visited on 03/22/2021).

[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, pp. 1657–1662. DOI: 10.1109/CDC.2017.8263888. URL: https://ieeexplore.ieee.org/document/8263888 (visited on 03/22/2021).

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

[Li+17] Wenting Li et al. "Securing proof-of-stake blockchain protocols". In: *Data Privacy Management, Cryptocurrencies and Blockchain Technology*. Springer, 2017, pp. 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 (visited on 03/22/2021).

[LW11] Allison Lewko and Brent Waters. "Decentralizing attribute-based encryption". In: *Annual international conference on the theory and applications of cryptographic techniques*. Springer. 2011, pp. 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 (visited on 03/22/2021).

[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), pp. 5156–5170. DOI: 10.1007/s11227-018-2413-7. URL: https://link.springer.com/article/10.1007%2Fs11227-018-2413-7 (visited on 03/22/2021).

[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 (visited on 03/22/2021).

[MK15] David Masad and Jacqueline Kazil. "MESA: an agent-based modeling framework". In: *14th PYTHON in Science Conference*. Citeseer. 2015, pp. 53–60. URL: https://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.966.1392&rep=rep1&type=pdf (visited on 05/19/2021).

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

[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, pp. 935–946. DOI: 10.1145/2884781.2884790. URL: https://dl.acm.org/doi/10.1145/2884781.2884790 (visited on 03/22/2021).

[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 (visited on 03/22/2021).

[Pim+04] João Paulo Pimentão et al. "Agent-based communication security". In: *German Conference on Multiagent System Technologies*. Springer. 2004, pp. 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 (visited on 03/22/2021).

[PP21] Ochchhav Patel and Hiren Patel. "Blockchain-Based Mechanisms to Address IoT Security Issues: A Review". In: *Data Science and Intelligent Applications* (2021), pp. 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 (visited on 03/22/2021).

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

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

[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 (visited on 03/22/2021).

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

[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, pp. 483–488. DOI: 10.1109/WAINA.2008.203. URL: https://ieeexplore.ieee.org/document/4482962 (visited on 03/22/2021).

[TV09] Chanatip Tumrongwittayapak and 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, pp. 1–5. DOI: 10.1109/ICICS.2009.5397594. URL: https://ieeexplore.ieee.org/document/5397594 (visited on 03/22/2021).

[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, pp. 907–913. DOI: 10.1109/MILCOM.2018.8599710. URL: https://ieeexplore.ieee.org/document/8599710 (visited on 03/22/2021).

[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, pp. 1–4. DOI: 10.1109/ASICON.2015.7517207. URL: https://ieeexplore-ieee-org.gaelnomade-1.grenet.fr/abstract/document/7517207 (visited on 05/26/2021).

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

A Problème des enchères

Exemple d'exécution du problème des enchères avec MadKit :

```
[Acheteur-3] INFO : encheres.Acheteur entered the room with 8832
[Acheteur-7] INFO : encheres.Acheteur entered the room with 8326
[Acheteur-5] INFO : encheres.Acheteur entered the room with 8689
[Acheteur-10] INFO : encheres.Acheteur entered the room with 9769
[Acheteur-11] INFO : encheres.Acheteur entered the room with 5700
[Acheteur-6] INFO : encheres.Acheteur entered the room with 3118
[Acheteur-12] INFO : encheres.Acheteur entered the room with 7335
[Acheteur-9] INFO : encheres.Acheteur entered the room with 7265
[Acheteur-4] INFO : encheres.Acheteur entered the room with 4856
[Acheteur-8] INFO : encheres.Acheteur entered the room with 4138
[Vendeur-2] INFO : encheres.Vendeur entered the room
[Vendeur-2] INFO : New:{name:Vase;price:200;}
[Acheteur-5] INFO : Offer:{name:Acheteur-5;price:4646;}
[Vendeur-2] INFO : Offer:{winner:Acheteur-5;price:4646;}
[Acheteur-12] INFO : Offer:{name:Acheteur-12;price:5925;}
[Vendeur-2] INFO : Offer:{winner:Acheteur-12;price:5925;}
[Acheteur-7] INFO : Offer:{name:Acheteur-7;price:8199;}
[Vendeur-2] INFO : Offer:{winner:Acheteur-7;price:8199;}
[Acheteur-11] INFO : Offer:{name:Acheteur-11;price:334;}
[Acheteur-10] INFO : Offer:{name:Acheteur-10;price:4483;}
[Acheteur-4] INFO : Offer:{name:Acheteur-4;price:3439;}
[Acheteur-6] INFO : Offer:{name:Acheteur-6;price:2131;}
[Acheteur-12] INFO : Offer:{name:Acheteur-12;price:6736;}
[Acheteur-9] INFO : Offer:{name:Acheteur-9;price:576;}
[Acheteur-3] INFO : Offer:{name:Acheteur-3;price:5396;}
[Acheteur-10] INFO : Offer:{name:Acheteur-10;price:5356;}
[Acheteur-5] INFO : Offer:{name:Acheteur-5;price:7587;}
[Acheteur-8] INFO : Offer:{name:Acheteur-8;price:2379;}
[Acheteur-11] INFO : Offer:{name:Acheteur-11;price:5117;}
[Acheteur-5] INFO : Offer:{name:Acheteur-5;price:8589;}
[Vendeur-2] INFO : Offer:{winner:Acheteur-5;price:8589;}
[Acheteur-7] INFO : Offer:{name:Acheteur-7;price:5328;}
[Acheteur-9] INFO : Offer:{name:Acheteur-9;price:5966;}
[Acheteur-4] INFO : Offer:{name:Acheteur-4;price:4666;}
[Acheteur-7] INFO : Offer:{name:Acheteur-7;price:7876;}
[Acheteur-3] INFO : Offer:{name:Acheteur-3;price:8455;}
[Acheteur-10] INFO : Offer:{name:Acheteur-10;price:9456;}
[Vendeur-2] INFO : Offer:{winner:Acheteur-10;price:9456;}
[Acheteur-3] INFO : Offer:{name:Acheteur-3;price:6596;}
[Acheteur-9] INFO : Offer:{name:Acheteur-9;price:6968;}
[Acheteur-3] INFO : Offer:{name:Acheteur-3;price:8306;}
[Acheteur-10] INFO : Offer:{name:Acheteur-10;price:9451;}
```

```
[Acheteur-3] INFO : Offer:{name:Acheteur-3;price:8735;}
[Acheteur-10] INFO : Offer:{name:Acheteur-10;price:9735;}
[Vendeur-2] INFO : Offer:{winner:Acheteur-10;price:9735;}
[Vendeur-2] INFO : Winner:{winner:Acheteur-10;name:Vase;price:9735;}
[Vendeur-2] INFO : New:{name:Tableau;price:1000;}
[Acheteur-9] INFO : Offer:{name:Acheteur-9;price:6034;}
[Vendeur-2] INFO : Offer:{winner:Acheteur-9;price:6034;}
[Acheteur-4] INFO : Offer:{name:Acheteur-4;price:1134;}
[Acheteur-6] INFO : Offer:{name:Acheteur-6;price:1904;}
[Acheteur-5] INFO : Offer:{name:Acheteur-5;price:1537;}
[Acheteur-12] INFO : Offer:{name:Acheteur-12;price:5752;}
[Acheteur-8] INFO : Offer:{name:Acheteur-8;price:2254;}
[Acheteur-3] INFO : Offer:{name:Acheteur-3;price:7145;}
[Vendeur-2] INFO : Offer:{winner:Acheteur-3;price:7145;}
[Acheteur-11] INFO : Offer:{name:Acheteur-11;price:4922;}
[Acheteur-7] INFO : Offer:{name:Acheteur-7;price:1759;}
[Acheteur-5] INFO : Offer:{name:Acheteur-5;price:8464;}
[Vendeur-2] INFO : Offer:{winner:Acheteur-5;price:8464;}
[Acheteur-5] INFO : Offer:{name:Acheteur-5;price:8446;}
[Acheteur-12] INFO : Offer:{name:Acheteur-12;price:6755;}
[Acheteur-12] INFO : Offer:{name:Acheteur-12;price:7309;}
[Acheteur-7] INFO : Offer:{name:Acheteur-7;price:6623;}
[Acheteur-3] INFO : Offer:{name:Acheteur-3;price:6673;}
[Acheteur-9] INFO : Offer:{name:Acheteur-9;price:7146;}
[Acheteur-7] INFO : Offer:{name:Acheteur-7;price:7634;}
[Vendeur-2] INFO : Winner:{winner:Acheteur-5;name:Tableau;price:8464;}
[Vendeur-2] INFO : Closing:{}
```