Contexte

Adrian Bonnet

24 02 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 établir le cadre de travail du stage, et décrire le fonctionnement de la couche de sécurisation des communication dans un système multi-agents embarqués.

Sommaire

1 Propriétés des systèmes multi-agents

Quel est le cadre de travail ?

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 connectés. 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[2], leur voisinage peut être amené à changer ;

sans-fil les agents communiquent entre eux sans fil[3], 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[4] ;

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], 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 SMA qui respecte les propriétés de ces systèmes.

[1] il est question ici de mobilité géographique

[2] simplification : on peut considérer que certains ou tous les agents sont immobiles

[3] simplification : on peut considérer que les agents immobiles possèdent une portée plus grande

[4] simplification : on peut considérer que les agents ne peuvent apparaître dans le système qu'à l'initialisation

2 Fonctionnement de la couche de sécurité

Qu'est-ce qui doit être fait ?

On cherche à assurer la **confidentialité** et l'**intégrité**, c'est-à-dire que personne à part les deux interlocuteurs ne connaît le contenu des communications et qu'il n'est pas possible que le contenu des communications 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. Il y a ici deux possibilités : les secrets partagés peuvent être injectés avant l'initialisation ou négociés après l'initialisation du système.

2.1 Injection avant l'initialisation

Pourquoi la solution la plus simple n'est pas réalisable ?

Une injection avant le démarrage du système implique que chaque agent possède à sa création les secrets nécessaires pour communiquer avec n'importe quel autre agent.

S'il existe un secret partagé pour chaque paire d'agents, alors chaque agent possède $n-1$ clés de sessions en mémoire. Il y a donc dans le système $\frac{n(n-1)}{2}$ secrets à injecter avant l'initialisation, soit un ordre de grandeur en $o(n^2)$. Sans compter la consommation de mémoire que cela implique, les fournisseurs d'agents doivent se mettre d'accord sur les secrets pour chaque paire d'agents, sans que ces informations ne soient divulguées. Un autre problème est qu'il est nécessaire que tous les fournisseurs communiquent entre eux, il faut donc qu'ils se connaissent et se reconnaissent mutuellement, le système n'est donc plus ouvert mais devient un consortium. Le dernier point est que, par conception, il n'est pas possible d'introduire de nouveaux agents dans le système, car il ne possèdera pas les secrets nécessaires pour communiquer avec d'autres agents.

Sinon, au lieu de créer un secret partagé pour chaque paire d'agent, il est possible de créer des secrets communs pour toutes les paires de fournisseurs d'agents. Cette idée part de la supposition que les agents issus d'un même fournisseur peuvent avoir confiance les uns dans les autres, et par extension qu'un agent donné peut avoir le même niveau de confiance en deux agents différents s'ils sont issus du même fournisseur. On ne peut faire cette supposition que si l'utilisateur final de l'agent ne peut pas rendre l'agent malveillant. Cette solution permet de réduire le problème du stockage en mémoire des secrets, puisqu'ils sont moins nombreux. De la même manière, il serait plus simple pour les fournisseurs de se mettre d'accord sur un nombre limité de clés. Cela résout également le problème de l'ajout de nouveaux agents dans le système, puisqu'il suffit de donner à ce nouvel agent les secrets issus de son fournisseur.

Cependant, le système reste un consortium entre des fournisseurs qui doivent se reconnaître mutuellement, et le nouveau problème que cela pose est le manque de sécurité : si un agent donné devient corrompu, alors il est en mesure d'espionner, voire de modifier, toutes les communications de tous les agents issus du même fournisseur, ce qui compromet la confidentialité et l'intégrité des données.

Ces solutions ne correspondent donc pas aux attentes liées à la sécurité.

2.2 Négociation après l'initialisation

Comment mettre en place un secret partagé ?

Si le secret partagé n'est pas injecté avant l'initialisation du système, les deux agents qui veulent communiquer vont chercher à **négocier** ce secret. La méthode communément utilisée pour cette négociation repose sur le **chiffrement asymétrique** : chaque agent possède une clé publique et une clé privée (ou secrète), de telle manière que tout ce qui est chiffré avec l'une des deux clés ne peut être déchiffré qu'avec l'autre clé. Grâce à cette paire de clés, il est possible d'assurer la **confidentialité** (en chiffrant les données avec la clé publique de l'interlocuteur) et l'**authentification** (en chiffrant avec sa propre clé privée pour prouver son identité).

Une fois que les deux agents qui souhaitent communiquer ont échangé leurs clés publiques, ils peuvent utiliser ce chiffrement afin de négocier la valeur du secret partagé, ici une clé de session. La méthode naïve repose sur la suggestion puis la validation de la valeur de la clé : le premier propose une clé, le second l'accepte ou la rejette pour en proposer une autre, le premier peut à son tour accepter ou rejeter la clé. Or le chiffrement asymétrique est plus coûteux en calcul que le chiffrement symétrique. Pour limiter les échanges chiffrés de manière asymétrique, on utilise la méthode largement répandue d'**échange de clé Diffie-Hellman** [Alv+17], pour laquelle la clé de session est forgée à partir des clés publiques des deux agents, sans que cette clé ne soit explicitement proposée.

2.3 Partage des clés publiques

Comment s'assurer que la clé reçue est la bonne ?

Lorsque les deux agents s'échangent leurs clés publiques, ils ne sont pas nécessairement à portée l'un de l'autre, ils vont donc passer par des intermédiaires, en l'occurrence les autres agents du système. Mais tant que ces deux agents ne se rencontrent pas pour échanger ou vérifier leur secret partagé, ils n'ont aucun moyen de s'assurer qu'il n'existe pas sur le chemin qui les relie de tiers qui altère leurs communications. Ceci implique que les agents ont besoin de faire confiance lors du premier échange de clés publiques, c'est le TOFU (*Trust On First Use*) [AK09]. Un agent malveillant qui se trouve alors sur le chemin dès le premier échange peut se faire passer pour l'interlocuteur auprès de chacun des deux agents : il s'agit de l'attaque de l'homme du milieu, **Man-in-the-Middle** (MitM).

Ce que l'on cherche à faire, c'est donc d'authentifier les communications dès le premier échange. Pour résoudre ce problème, sur Internet, on utilise une solution centralisée hiérarchique de certification. Lorsqu'un navigateur web établit une connexion avec un serveur web, le serveur renvoie un certificat qui contient sa clé publique ainsi que le nom de l'**autorité de certification** (CA) qui certifie que telle clé publique correspond bien à tel serveur. Le navigateur fait alors une requête au CA, qui lui envoie sa réponse signée. Si le navigateur connaît la clé publique du CA, alors il sait qu'il peut lui faire confiance, sinon par hiérarchie, il peut s'adresser à un second CA qui certifie le premier, et remonter ainsi jusqu'à trouver un CA en lequel il ait confiance. L'utilisation d'autorités de certification n'est pas possible dans un SMA car cela implique une architecture centralisée.

Il faut donc chercher des alternatives à l'autorité de certification hiérarchique centralisée, soit en mettant en place des mécanismes de vérification partielle pour palier l'absence de CA, soit en décentralisant une autorité de certification.

2.4 Mécanismes de vérification partielle

Comment limiter les risques d'attaque sans CA ?

Sans autorité de certification, afin d'éviter le TOFU, on cherche à vérifier plusieurs fois l'information que l'agent cherche à obtenir, c'est-à-dire la clé publique de l'agent qu'il cherche à contacter. Cette redondance peut être temporelle [Rah17], en dupliquant les messages dans le temps ou en retardant volontairement certaines communications, ou spatiale [AK09] [Pim+04], en empruntant plusieurs chemins dans le réseau.

L'avantage de ces solutions est qu'elles sont relativement simples à mettre en œuvre. La sécurisation des communications s'apparente alors au protocole **PGP** [Mor+06] auquel on ajoute quelques mécanismes pour palier le besoin d'échange de clés en personne.

Cependant, ces mécanismes de redondance ne permettront pas de mettre en place un échange sécurisé entre deux agents si l'un des deux se situe "derrière" un agent malveillant par rapport au reste du système. De manière plus générale, si des agents malveillants qui collaborent entre-eux parviennent à isoler une partie du réseau, alors deux agents de part et d'autre de cette séparation ne pourront jamais établir une communication sécurisée, puisque toutes les tentatives de vérification passeront par le ou les attaquants. Une telle séparation est possible lors d'une attaque **Sybil** par exemple.

Pour détecter une telle attaque, il serait nécessaire de mettre en place un système de détection d'intrusion (**IDS**). Ce système doit être décentralisé, et suffisamment léger pour répondre au critère embarqué du SMA étudié dans ce travail. Cet IDS sort du cadre du travail, car il implique un système de gestion de la confiance et d'analyses de métriques décentralié, et mériterait sa propre étude.

2.5 Solutions purement cryptographiques

Existe-t-il des solutions à notre problématique apportées par la cryptographie ?

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.

Par exemple, il serait possible de faire en sorte que la clé publique d'un agent soit générée à partir d'attributs uniques de cet agent et ce 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.

Une alternative serait d'utiliser un cryptosystème à seuil 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 assez avancés.

2.6 Autorités de certification décentralisées

Comment répartir les informations sur le système ?

Si l'on souhaite utiliser un CA malgré l'architecture décentralisée, il faut trouver un moyen de le répartir en garantissant l'intégrité des informations qu'il contient. Si l'information qu'un agent recherche, c'est-à-dire une clé publique, est détenue par un agent malveillant, on souhaite qu'un agent qui a fait la requête puisse détecter des éventuelles modifications de l'information.

Pour garantir cette intégrité des données, il est possible d'utiliser une table de hachage distribuée (**DHT**) [Tak+08], qui permet de diffuser une information statique au sein d'un réseau de clients qui souhaitent obtenir l'information. La table de hachage contient alors les clés publiques de tous les agents du système. La clé publique que l'on cherche est une valeur dans la DHT, la clé qui permet de retrouver cette clé est le hash de l'identifiant de l'agent dont on cherche la clé publique. Ce hash sert d'identifiant unique. Toutes les clés sont réparties sur l'ensemble des agents. Afin de garantir l'intégrité, tous les agents possèdent un fichier, l'équivalent d'un torrent, qui contient le hachage cryptographique de chaque clé publique, afin de pouvoir vérifier que les informations reçues n'ont pas été altérées. Ce fichier torrent est injecté avant l'initialisation du système. Avec cette solution, il n'est pas possible d'ajouter de nouveaux agents dans le système sans modifier le fichier torrent sur tous les agents. De plus, toutes les fournisseurs d'agents doivent mettre en commun leurs connaissances avant l'initialisation. De plus, chaque agent doit stocker les valeurs de hachage cryptographique de chaque clé publique, bien que cela représente une quantité de mémoire plus faible que le stockage de toutes les clés publiques.

Pour répartir un CA sur le système multi-agents, on cherche une structure de données qui garantisse l'intégrité des données et à laquelle il est possible d'ajouter des informations pendant l'utilisation. On peut utiliser pour cela la technologie Blockchain, (**BCT**) [Cal+18] [SB18] [Won+18], dans laquelle chaque bloc contiendra les clés publiques des agents, de manière à ce que chaque agent puisse vérifier grâce à la Blockchain les informations qu'il reçoit. Cette Blockchain ne grandira pas indéfiniment comme dans le cadre des cryptomonnaies, car une fois que les clés publiques de tous les agents vivants dans le système se trouvent dans la Blockchain, il n'y a plus de bloc à ajouter tant qu'on n'ajoute pas d'agent.

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).

[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).

[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).

[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).

[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).

[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).

[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).

[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).

[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).