MAKI: A Multi-Agent Public Key Infrastructure

Arthur Baudet*, Oum-El-Kheir Aktouf*, Annabelle Mercier* and Philippe Elbaz-Vincent†
*Univ. Grenoble Alpes, Grenoble INP, LCIS, 26000 Valence, France
Email: {arthur.baudet, oum-el-kheir.aktouf, annabelle.mercier}@lcis.grenoble-inp.fr
†Univ. Grenoble Alpes, CNRS, IF, 38000 Grenoble, France
Email: philippe.elbaz-vincent@math.cnrs.fr

***Abstract*—This paper presents a public key infrastructure for open multi-agent systems of embedded agents. We aim at securing the communications between agents to provide foundations for more advanced security solutions, as well as allowing agents to communicate without the risk of their messages being tampered with. To do so, we deploy a key infrastructure taking advantage of the agents autonomy to allow authenticity and integrity checks and accountability of all interactions: Multi-Agent Key Infrastructure. Multi-Agent Key Infrastructure is a public key infrastructure leveraging and empowering a trust management system. This infrastructure paves the way to build more secure, open decentralized systems of autonomous embedded systems.**

I. Introduction

Multi-Agent Systems (MAS) are used as a system architecture in distributed systems such as wireless sensor networks, Internet-of-Things, autonomous vehicles, etc. This multi-agent approach to system design results in having a network of embedded devices, each one executing one agent, autonomously communicating and collaborating to reach a common goal. We define these systems as Multi-Embedded-Agent Systems (MEAS), i.e., multi-agent systems of embedded agents. In this context, we focus our study on heterogeneous open MAS, i.e., systems that allow agents with different capabilities to enter in and leave the system at runtime. Recent meta-studies [1] show that MEAS and similar systems are particularly vulnerable to insider attacks, attacks coming from one or more malicious agents of the system, as well as attacks on communications, as they often rely on wireless ad hoc networks to communicate. They also show that trust management systems (TMS) are a common way of mitigating those threats. However, these TMS often take strong hypotheses concerning the lower layers, especially the cryptographic layer. Those hypotheses may require a third party to provide a root of trust or to preload certificates in the agents, making them inapplicable in our open and decentralized context. Thus, a specific solution to provide agents with cryptographic keys to allow them to securely communicate is required.

II. Threat Model

We seek to secure the communications in multi-agent systems of embedded agents to allow the use of TMS without the risk of the exchanges being tampered. To this end, we consider the following assumptions: (i) the cryptographic primitives that are used, the hardware they run on, and their implementations are secured; (ii) a suitable TMS is running and takes into account likely attacks on it; (iii) Sybil attacks are mitigated by using either the TMS or by other means; (iv) an ad hoc routing protocol is used to allow the agents to communicate with each other. Following these assumptions, we define the attacker model as a mote-class attacker, i.e., an attacker with similar resources as the agents of the system, that would have complete control over the communication medium. It would be able to eavesdrop on, block and tamper with any message. Moreover, we make no assumption on the intentions or capabilities (inside the spectrum of the mote-class) of other agents, their behaviors are modeled as Byzantine behaviors.

III. Related Work

In [2], the authors propose an enhanced distributed Public Key Infrastructure (PKI) for industrial control systems using an agent-based framework that requires an operator to add or remove systems from the PKI. This conflicts with the openness characteristic of the systems studied in this paper. The work in [3] bases its decentralized PKI on a distributed hash table to allow the signature, storage, and certification of certificates. While it solves the problem of consensus in managing certificates, it does not provide ways to filter out untrustworthy nodes.

For several years, most of the efforts toward designing a decentralized PKI have involved Blockchain Technology (BCT) [4], [5]. The BCT is designed to provide a consensus on information in decentralized systems where no trust pre-exists, making it an ideal solution to deliver, store and revoke certificates. Yet, BCT is not adapted to our problem. Regardless of the used consensus algorithm, which can be highly power-consuming in the case of the proof-of-work, the security of a Blockchain partly relies on storing the history of all the exchanged information during the life of the system. This means that it will only grow and eventually reach a size too large to be stored in resource-constrained embedded systems.

Alternatively, identity- and attribute-based PKIs, PKIs that do not rely on certificates but on the characteristics of the members of the network to distinguish them, can be adapted to decentralized contexts [6]. But this means that they rely on prior knowledge about the agents, an assumption difficult to meet in open heterogeneous systems.

Consequently, we could not find any existing infrastructure satisfying the requirements of the studied systems: decentralization, openness and autonomy. This is why we provide in our work the foundations of such a PKI through Multi-Agent Key Infrastructure (MAKI), one infrastructure designed

for open MEAS. MAKI empowers TMS by enabling secure communications and enforcing exclusions. It also leverages it to deploy a resilient self-organization against insiders' attacks.

IV. Multi-Agent Key Infrastructure

The use of cryptographic signatures provides integrity and authenticity verification as well as accountability. It only requires agents to generate asymmetric keys and use them to sign the messages they send. This mechanism alone meets the decentralized cryptographic requirement we set. However, doing so enables abusive behaviors such as agents using multiple pairs of keys at the same time or changing their keys over time. We prevent those behaviors by leveraging the TMS to make it inefficient to change identity. Then we empower the TMS by allowing the certificates to be revoked, instead of just ignoring their holders.

A. Architecture

We designed MAKI as a lightweight decentralized Public Key Infrastructure. From the PKI principles, we only kept the Certificate Authority (CA) function described below, the mandatory use of certificates, and the certificate revocation.

As we are focusing on open heterogeneous systems, we do not expect agents entering a system with preloaded certificates nor do we want to enforce specific authentication protocols. Instead, we designed MAKI not to require authentication. In MAKI, agents are anonymous and are only deemed friendly or malicious based solely on their actions. This is possible thanks to the accountability brought by the use of cryptographic signatures. This means that the identity of an agent is defined as the keys it uses to interact with other agents.

To enforce the use of cryptography, it is mandatory to sign all communications with a key for which a valid certificate is held by the sender. Only requests to find CAs or certification requests can be signed with not-certified keys.

Certificates are delivered and revoked by CAs elected through the use of a self-organization scheme, thus capitalizing on the autonomy of the agents. CAs are also autonomous in choosing to revoke an agent but they are expected to do so when the majority of the agents they trust asks for it. The revocation is done using two mechanisms. First, the CAs will add the revoked certificates to their Certificate Revocation Lists (CRLs) and broadcast them. This method is direct and instantaneous but, depending on the network capabilities, the CRL updates may take time to reach every agent. Then, to mitigate this risk, we also use short-lived certificates, which will not be renewed by a CA that is aware of the revocation.

B. Self-organization

Agents are either CA or None. None is the default role and does not hold any responsibility toward the PKI. The CAs are responsible for delivering certificates to None agents and other CAs, revoking certificates, storing a list of the delivered certificates and a CRL. As MAKI does not rely on third parties to establish a root of trust, CAs are all initially self-signed and can later use cross-certification to create a network of trusted CAs.

CAs are self-elected. Agents capable of being CA (from a resource management point of view) decide for themselves if they will become a CA following a self-organization algorithm.

Algorithm 1 Role selection.

```
▷ T ∈ [0, 1), the probability an agent becomes CA in any case ◁
Role ← None
CAs ← BROADCAST(CAListingRequest)
TrustedCAs ← FILTERBYTRUST(CAs, Moderate)
if TrustedCAs is empty or RAND(0, 1) < T then
    Role ← CA
```

Algorithm 1 describes how role selection is done. This algorithm was designed with two modular goals: (i) every agent should be close to a CA and (ii) CAs will not become a single-point-of-failure. This will lead to a uniform distribution of CAs with one or more (depending on `T`, the probability that an agent decides to become a CA) CAs by groups of agents. Agents keeping the None role, because they don't have the resources or because there are CAs in their neighborhood, will simply have to choose one of the most trusted CAs they know to ask for a certificate. Any new agent will follow the same process: first determining if there is a need to become a CA and, if not, requesting a certificate from a CA. Afterward, it may decide to select a more trustworthy CA by requesting trust information from its neighbors or keep the CA it chose.

The self-organization process assumes that agents are honest, we rely on the TMS to detect abusive behaviors and prevent malicious agents from overtaking the PKI.

C. Trust Management

MAKI is not designed for a specific TMS. We simply assume that we can add new rules to it and request trust values. The new rules are meant to (i) define which agents to cooperate with in MAKI; (ii) add responsibilities to the role of CA to reduce the probability of malicious agents abusing it.

The first set of rules includes the required trust level for each interaction between agents. For example, an agent should, at least, moderately trust a CA before requesting it a certificate. A CA should also moderately trust a agent to accept delivering a certificate. Cross-certification should only be done between agents highly trusting each other. And a certificate should only be revoked if moderately trusted agents ask for it.

The second set of rules describes the expected behavior of agents, the TMS is used to reward agents acting on the role they chose. Hence, even malicious agents first have to behave properly before trying to attack the system, making the attack more costly and less impactful. For example, any agent ignoring a request will lose the trust of the requester. A CA will earn trust by delivering certificates, cross-certification especially increases the trust of other agents.

Moreover, we promote benevolent CA by enticing agents to choose a trustworthy CA to sign their certificate by weighting the trust of its certifier when computing the trust of an agent.

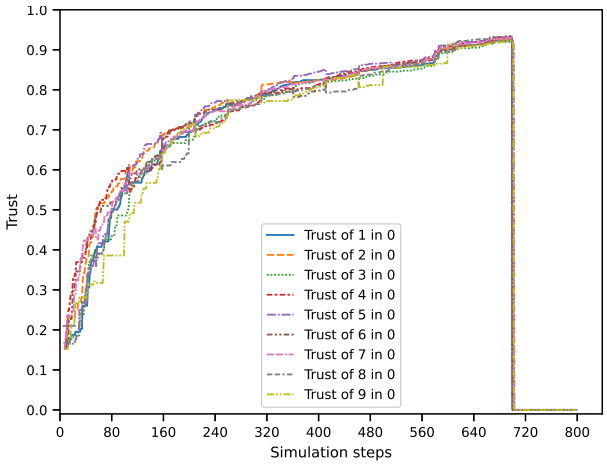

Figure 1. Trust fluctuations with the certificate of agent 0 being revoked.

Thus, an agent holding a certificate of a known malicious CA will not be trusted and the malicious CA will end up completely ignored.

V. Proof-of-Concept

We developed a proof-of-concept (PoC) using an in-house multi-agent simulator. As a placeholder, we designed a trust model with a low initial trust, a common way to prevent agents changing identity to reset their trust. The value of the trust is bound between 0 and 1 with a growth following the function $f : x \mapsto \frac{x}{x+10}$, where the value 10 was experimentally chosen to set the slop of the curve. Three trust thresholds, *Low*, *Moderate* and *High*, are respectively set to 0.3, 0.7 and 0.9 and the initial value is set to *Low*. In this model, a trust value below the *Low* threshold implies that the agent is to be ignored. The TMS also includes indirect information. An agent can ask other agents how much they trust a certain agent and add it to its model. Moreover, agents eavesdrop on the communications to update in real time their trust model. Agents' trust increases randomly over time to emulate the successful interactions among them and feed the TMS. This PoC focuses on the behavior at a local scale, which is defined as an area in which all agents are in range of each other. This allowed us to deploy a very minimalistic routing protocol. As we assumed that a proper routing protocol is deployed, we did not emulate packet loss in the simulator but agents do not assume that every request will be answered in due time, or at all. We show two results from the experiments we realized using the PoC. Both of them were done using a configuration of ten agents ($\mathcal{A}_0$ to $\mathcal{A}_9$), with only the agents $\mathcal{A}_6$ to $\mathcal{A}_9$ having the resources to become CA. Agents that initially become CA during our experiments are $\mathcal{A}_6$, $\mathcal{A}_8$, and $\mathcal{A}_9$.

In the first scenario, we aim to show the impact of certificate revocation on the TMS. We established a situation where the trust values of agents $\mathcal{A}_1$ to $\mathcal{A}_6$ in $\mathcal{A}_0$ drop below the *Low* threshold at step 700, leading to the revocation of $\mathcal{A}_0$, and expect that the agents $\mathcal{A}_7$ to $\mathcal{A}_9$ also distrust $\mathcal{A}_0$. The result in trust fluctuations are shown in Figure 1. We can see that

```
step:701
agent:7:<>:CertAdvert(Certificate(issuer: 7, subject: 7,
      ...))
agent:0:=>:CertReq(src: 0, dest: 7)
agent:1:=>:CertReq(src: 1, dest: 7)
agent:3:=>:CertReq(src: 3, dest: 7)
agent:2:=>:CertReq(src: 2, dest: 7)
agent:4:=>:CertReq(src: 4, dest: 7)
Agent:5:=>:CertReq(src: 5, dest: 7)
```

Figure 2. Simplified excerpt of the execution trace showing that agent 7 is now a CA and agents 0 to 5 requesting that it delivers them a certificate.

the revocation leads to all the agents distrusting agent $\mathcal{A}_0$ as expected.

In the second scenario, we show how MAKI leverages the TMS to adapt to a coalition of malicious CAs. To do so, the three CAs are considered as malicious starting from step 700. Then, we expect the system to adapt accordingly by no longer relying on them and finding another CA. An excerpt of the execution trace at step 701 in shown in Figure 2. It shows that $\mathcal{A}_7$ changes its role, broadcasts its new certificate and agents $\mathcal{A}_0$ to $\mathcal{A}_6$, which lost their trust in the other CAs, request a certificate from $\mathcal{A}_7$. The self-organization algorithm led to the promotion of a benevolent CA and the malicious CAs being ignored.

VI. Conclusion

We introduced a decentralized public key infrastructure, coined MAKI, adapted to open multi-agent systems of embedded agents. This infrastructure secures the communications to allow agents to securely exchange information to detect intruders. Those intruders are then excluded thanks to the use of certificates delivered and revoked by a subset of trusted certification authorities maintained with no third parties involved. We are now focusing our efforts on further validating MAKI through simulations and model checking.

Acknowledgements

This work is supported by the French National Research Agency in the framework of the "Investissements d'avenir" program (ANR-15-IDEX-02).

References

[1] A. Baudet, O.-E.-K. Aktouf, A. Mercier, and P. Elbaz-Vincent, "Systematic Mapping Study of Security in Multi-Embedded-Agent Systems," *IEEE Access*, vol. 9, pp. 154 902–154 913, 2021.

[2] S. Blanch-Torné, F. Cores, and R. M. Chiral, "Agent-based PKI for Distributed Control System," in *2015 World Congress on Industrial Control Systems Security (WCICSS)*, 2015, pp. 28–35.

[3] X. Bonnaire, R. Cortés, F. Kordon, and O. Marin, "A Scalable Architecture for Highly Reliable Certification," in *2013 12th IEEE International Conference on Trust, Security and Privacy in Computing and Communications*, 2013, pp. 328–335.

[4] A. Yakubov, W. M. Shbair, A. Wallbom, D. Sanda, and R. State, "A Blockchain-Based PKI Management Framework," in *2018 IEEE/IFIP Network Operations and Management Symposium*, 2018, pp. 1–6.

[5] B. Qin, J. Huang, Q. Wang, X. Luo, B. Liang, and W. Shi, "Cecoin: A decentralized PKI mitigating MitM attacks," *Future Generation Computer Systems*, vol. 107, pp. 805–815, 2020.

[6] T. Okamoto and K. Takashima, "Decentralized Attribute-Based Encryption and Signatures," *IEICE Transactions on Fundamentals of Electronics, Communications and Computer Sciences*, vol. E103.A, no. 1, pp. 41–73, 2020.