Decentralized and Autonomous Key Management for Open Multi-Agent Systems of Embedded Agents*

Arthur Baudet[1,2][0000−0001−6811−3886], Oum-El-Kheir Aktouf[1][0000−0002−0493−9096], Annabelle Mercier[1][0000−0002−6729−5590], and Philippe Elbaz-Vincent[2][0000−0002−8629−3021]

[1] Univ. Grenoble Alpes, Grenoble INP, LCIS, 26000 Valence, France
{firstname.lastname}@lcis.grenoble-inp.fr
[2] Univ. Grenoble Alpes, CNRS, IF, 38000 Grenoble, France
{firstname.lastname}@univ-grenoble-aples.fr

Abstract. This paper presents a public key infrastructure for open multi-agent systems of embedded agents. Open multi-agent systems of embedded agents are a set of network embedded systems cooperating in real time to achieve their goal without a central server issuing commands. In this context, agents are very prone to attacks as they can be confronted with new agents with unknown goals and as they often rely on wireless ad hoc communications. The key infrastructure we propose allows the agents to communicate without the risk of their messages being tampered with. Thus, providing foundations for more advanced security solutions such as trust management systems. In order to do that, we leverage the agent's capabilities to establish and maintain the infrastructure by providing self-organization and trust management rules. Once established, the infrastructure provides a way for the agents to prove their right to take part in the systems with certificates and to secure their communications with asymmetric cryptography. As a result, agents can communicate securely without the risk of their identities being stolen and their communications being tampered with as well as being able to exclude intruders. The work proposed in this paper paves the way to build more secure open decentralized systems of autonomous embedded systems. To make our solution general and adaptable to many situations, we decoupled the cryptographic and trust management details from the infrastructure itself.

Keywords: Public key infrastructure · Multi-agent systems · Embedded agent · Decentralized security

1 Introduction

The multi-agent systems paradigm proposes a decentralized approach to software and system architectures. Distributed systems such as wireless sensor networks,

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

Internet-of-Things, autonomous vehicles, etc., can be deployed as a Multi-Agent System (MAS) with each embedded device being autonomous in its decision-making. This approach results in 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 MEAS, a class of systems that allow agents of different capabilities and origin to enter in and leave the system at run-time.

Recent studies [3, 7] show that MEAS and similar systems are particularly vulnerable to insider attacks, attacks coming from one or more malicious agent of the system, as well as attacks on communications, as they often rely on wireless ad hoc networks to communicate. Both types of attacks can have devastating effects on the communications integrity and the availability of the system. 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 [12, 15]. For example, those hypotheses may require the presence of a third party to provide a root of trust or to preload certificates in the agents, making them inapplicable in an open and decentralized context. Thus, a specific solution to provide agents with cryptographic keys to allow them to securely communicate is required here.

In the next section, we establish the threat model and review the security concerns related to our work. We introduce our contribution in Section 4 and discuss its validity through model checking in Section 5 and simulation in Section 6. Finally, we conclude the paper in Section 7.

2 Security Considerations and Threat Model

We aim to secure the communications in multi-agent systems of embedded agents so that any unauthorized modifications should be detectable (integrity checks), and that the source of each message can be verified (authenticity) and help accountable (accountability). As we add new behaviors and possibly increase the attack surface of, we also consider the vulnerabilities induced by our approach.

Regarding other security concerns, we consider the following assumptions:

- The cryptographic primitives that are used, the hardware they run on, and their implementations are secured.
- A suitable and robust TMS is running on each agent.
- Sybil attacks are mitigated by using either the TMS or by other means.

We define a threat model with a mote-class attacker, i.e., of similar resources as the agents of the system, which has complete control over the communication medium. It would be able to eavesdrop on, to replay and to 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.

3 Related Work

This is not the first attempt at providing a distributed and decentralized PKI, but most of the previous attempt do not go far enough in the context of open decentralized autonomy. For example, the works presented in [16] and [11] rely on threshold cryptography or Simple Distributed Security Infrastructure to provide a decentralized Public Key Infrastructure (PKI) but they still require either out-of-band verification or preloaded certificates to provide authentication. In [4], the authors propose an enhanced Distributed PKI for industrial control systems using an agent-based framework that requires an operator to add or remove systems from the PKI, which conflicts with our openness characteristic. Both the works in [1, 6] base their decentralized PKI on a distributed hash table to allow the signature, storage, and certification of certificates. While they solve the problem of consensus in managing certificates, they do not provide ways to autonomously filter out untrustworthy nodes. Lastly, the LocalPKI [10] seems to be a good candidate but requires user input.

Efforts toward designing a decentralized PKI also involves Blockchain Technology (BCT) [20, 22, 23]. 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, like the two previously cited works. 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.

Identity- and attribute-based approaches such as the ones presented in [9,19] could also yield are promising but they too strongly 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 three main requirements of the studied systems: decentralized autonomy, openness and heterogeneity. 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.

4 Multi-Agent Key Infrastructure

The sole use of cryptographic signatures enables integrity and authenticity verification as well as accountability. This only requires agents to generate asymmetric keys and use them to sign all the messages they send. This mechanism alone meets the decentralized cryptographic requirements we set. However, doing so also 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

Algorithm 1 Algorithm describing how the decision of becoming a CA is taken.

```
T ∈ [0, 1) ▷ The probability that an agent decides to become a CA even if other trustworthy CAs are close.
Role ← None
CAs ← GetNeighborCAList()
TrustedCAs ← Filter(CAs, TrustLevel.Moderate)
if CanBecomeCA() and (TrustedCAs is empty or Random(0, 1) < T) then
    Role ← CA
return Role
```

linking an agent's identity to a key with certificates and by leveraging the TMS to make inefficient to change identity. Then we empower the TMS by allowing the certificate of malicious agents to be revoked, leading to their exclusion.

4.1 Architecture

As we are focusing on open heterogeneous MEAS, we do not expect new agents to enter with preloaded certificates nor do we want to enforce specific authentication protocols. Instead, we designed MAKI without requiring to authentication. In MAKI, agents are anonymous and are only categorized as friendly or malicious based on their actions. This is possible thanks to the accountability induced by the use of cryptographic signatures but it requires a strong coupling between an agent's identity and one pair of cryptographic keys. To enforce this coupling, MAKI uses a subset of the principles of standard PKIs, such as the X.509 PKI. It only includes the role of Certificate Authority (CA), the mandatory use of certificates and signatures, and the certificate revocation.

The CAs are designated using a self-organization algorithm (see Section 4.2), thus capitalizing on the autonomy of the agents. They have the responsibility to deliver certificates and revoke them when necessary. They are autonomous in their choice of delivering and revoking. To prevent abuses, we defined TMS rules (see Section 4.3) to sanction unproperly behaving CAs.

Revocation is carried out 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. So, to mitigate this issue, we also use short-lived certificates, which will not be renewed by CAs that are aware of the revocation.

4.2 Self-organization

The proper functioning of the self-organization depends on rules MAKI adds to the TMS. For example, the way we avoid the pitfalls of self-signed CAs, the likeliness a malicious agent will become CA or the rewards of being cross-certified is all explained in the next section.

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, otherwise they remain None, the default role which does not hold any responsibility toward the PKI. Algorithm 1 describes how the choice is made. 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, depending on `T`, the probability that an agent decides to become a CA, of CAs with one or more CAs by groups of agents. It is possible to adapt the definition of "close" to reduce the number of CAs. There will be more CAs if close means being in communication range than if it means being in three times the communication range. It is also possible to adapt the value of `T` to increase or lower the number of redundant CAs. If `T` is very high, almost all agents that can be CA will be, but if `T` is very low, only agents far from a CA will become one. Both the definition of close and the value of `T` should be tailored to the application, density and capabilities of the application MAKI runs on. Moreover, since we do not want to 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 trustworthy CAs.

Choosing a CA for a None agent is similar to deciding to become a CA. Once they know the identity of their neighbors CAs, they choose one of them. We do not recommend choosing the most trustworthy each time since it can create single-point-of-failure situations if all the agents have the same CA as their most trustworthy CAs. We rather recommend using an exploration scheme such a simple weighted sum or a multi-arm bandit strategy. This way, the most trusted CAs are still selected by less trusted CAs are given the opportunity to prove themselves, creating some redundancy and increase the resilience of the system.

Adding a new agent in MAKI is straightforward. The agent will first determine if it needs to become a CA and, if not, it requests a certificate from a CA. It may then decide to select a more trustworthy CA by requesting trust information from its neighbors or keep the CA it chose. In any case, it will advertise its certificate to ensure that its neighbors learn about it.

As self-signed CAs are not susceptible to the revocation mechanisms, the only way to exclude self-signed CAs is to ignore them and suggest new agents to avoid them. A way to diminish this advantage is to allow cross-certification. This means that CAs could request that other CAs sign their certificates which would make them susceptible to have their certificate revoked. This has no intrinsic benefit for the cross-certified CA as it will only make it harder to obtain and maintain a valid certificate, but it is considered as a show of good faith and is rewarded in the TMS.

4.3 Trust Management

MAKI is not designed for a specific TMS. Moreover, defining a trust model for each specific use case of MEAS is out of scope. Instead, we specify here how MAKI leverages the TMS to deploy its self-organization.

Table 1: Trade-offs between risk and benefit for each possible interaction between agents depending on their roles.

Interaction	Risk	Benefit	Required trust
Certificate Authority			
Delivering a certificate	Moderate. Allowing malicious agents to operate.	High. Having its own legitimacy increase since one more agent is trusting it to deliver trustworthy certificates.	Moderate or none[†]
Revoking a certificate	High. Decreasing the trust of agents not agreeing with the revocation. Excluding a benevolent agent.	High. Helps the overall system by excluding a malicious agent.	Moderate
Requesting a cross-certification	High. Having its reputation[*] decreased if the cross-certifier is distrusted. Giving more legitimacy to a malicious CA.	Moderate. Higher trust is given to cross-certified CA.	High
Accepting a cross-certification request	High. Giving more legitimacy to a malicious CA.	High. Having its own legitimacy increase since a CA is trusting it	High
None			
Requesting a certification	Moderate. If the CA is not trusted, having to renew the certificate with another one. Giving more legitimacy to a malicious CA.	High. Holding a valid certificate is mandatory to participate in the system.	Moderate or none[‡]

[*] The term "reputation" is used as a way to describe the trust other agents have in one agent.

[†] CAs have no way in checking the trustability of new agents at first so the required trust is set to none for them.

[‡] New agents have no way in checking the trustability of CAs at first so the required trust is set to none for them.

We present in Table 1 a risk assessment of the interactions in MAKI and recommended trust thresholds to reach to carry them out. These trust thresholds

are representations of the trust an agent should have in other agents to interact with them. Their values depend on the trust model. In addition to the presented interactions, we add that any agent should be able to request and share their certificates without risk nor required trust, as doing so allows checking or proving that the requirement of holding a valid certificate is met.

MAKI also leverages the TMS to mitigate the proliferation of malicious self-signed CAs by adding a cost to the role of CA. In MAKI, a CA can only be legitimate if it continually replies to certification requests, and illegitimate CAs should be ignored. This way, even a malicious CA must contribute to the system by delivering certificates to requesting agents. This can be translated by a small increase of the trust in a CA each time it answers a certification request. Moreover, the trust put in certified agent is weighted by the trust of the CA that signed its certificate. This is done to encourage agents to choose CAs they trust but that other agents also trust. This aims at making the CAs properly behave to every agent and not only to some of them.

While we explained how to mitigate the risk of malicious agents becoming CA and thus self-signed CAs, we can also provide a way to reduce the number of self-signed CA by adding a trust reward for cross-certified CA. Cross-certified CAs will end up more selected. This will force self-signed CAs to get cross-certified and take the risk of having their certificates revoked if they behave maliciously.

Overall, MAKI does not reduce the security brought by the TMS since, even though they cannot be revoked, self-signed CAs can still be ignored and their bad reputation shared to new agents. This means that the number of malicious agents MAKI can handle only depends on the TMS and the complexity of the attacks executed against it. We show in Section 6 how, with a simple TMS, MAKI handles malicious agents once the TMS detects their malicious behaviors.

4.4 Certificate Management

Certificate and CRL representations can be found in Figure. 1. Excluding the fields we did not keep from the X.509 format [5], the main differences are the inclusion of the public key of the issuer since it is part of its identity and the additional field, `subjectInfo`, which is let to be defined by the system designers. Using this format, with an empty `subjectInfo` field, 4-byte `time_t` for `UTCTime`, 193 bytes for the public key OpenSSH format, 105 bytes for the raw signature, 1-byte integer for `Version`, `Role` and `ReasonCode`, and 2-byte integer for `SerialNumber` and `Name`, and no padding, the size of a certificate is 507 bytes.

Since MAKI does not rely on registration authorities to deliver and distribute certificates, the distribution of certificates falls to the agents themselves. Agents may broadcast their certificates periodically and should attach them to the first messages of each exchange. An agent can also request the certificate of another agent. These distribution methods are less efficient than having a third-party gathering and sharing the certificates. But they remove any threat coming from this third party and any risk of it becoming a single-point-of-failure as well as enable better scalability and decentralization.

```
Identity ::= SEQUENCE { name INTEGER, publicKey BIT STRING }
```

(a) ASN.1 representation of the Identity field used in MAKI certificates and CRL.

```
Certificate ::= SEQUENCE {
  version [0]  INTEGER,
  serialNumber INTEGER,
  signature    BIT STRING,
  issuer       Identity,
  validity     SEQUENCE {
    notBefore UTCTime,
    notAfter  UTCTime
  },
  subject      Identity,
  subjectRole  Role,
  subjectInfo  SubjectInfo
}
Role ::= INTEGER {
  NONE(0),
  CA(1)
}
```

(b) ASN.1 representation of the MAKI certificates.

```
CertificateRevocationList ::=
      SEQUENCE {
  version [0]          INTEGER,
  signature            BIT STRING,
  holder               Identity,
  thisUpdate           UTCTime,
  certificates SEQUENCE OF
       SEQUENCE {
    serialNumber    INTEGER,
    issuer          Identity,
    subject         Identity,
    revocationDate UTCTime,
    reasonCode      ReasonCode
  }
}
ReasonCode ::= ENUMERATED {
  idComprise(0),
  cessationOfOperation(1)
}
```

(c) ASN.1 representation of the MAKI CRL.

Fig. 1: ASN.1 representation of the MAKI certificate and CRL formats.

Concerning the CRL format, we moved the `reasonCode` field from the CRL Extension to the mandatory fields so that CAs can be held accountable for each revocation. To keep the CRL format as small as possible, it is only meant to hold information related to agents exclusion. Hence, from the ten possible values, we only kept the `KeyCompromise` (renamed `IdCompromise`) and `CessationOfOperation`. Other reason codes could be added to indicate malicious behaviors specific to the application. Following the same memory size choices as in the certificate format, a CRL with $n \in \mathbb{N}$ certificates is $305 + n \times 397$ bytes. CAs should keep and distribute, gratuitously or on demand, their CRL.

5 Model Checking

5.1 Tool and hypotheses

To validate our approach, we applied model checking with Model Checker for Multi-Agent Systems (MCMAS) [17]. MCMAS allowed us to describe our agents' behaviors directly and easily using a language designed for it: ISPL (Interpreted Systems Programming Language). Using algorithms based on ordered binary decision diagrams, MCMAS supports the verification of epistemic and temporal modalities. In particular, we used computation tree logic operators to describe the properties that interest us. However, MCMAS makes several hypotheses in its model:

A. The internal states are only known by their owner and actions are public.
B. There is no explicit ways to communicate between agents.
C. The number of agents is bound, meaning that each agent in the model is explicitly described. E.g., for 5 agents, we have to describe: `Agent1`, `Agent2`, `Agent3`, `Agent4` and `Agent5`, even if their behavior is the same.
D. The agents have no power or computational limits.

Due to hypotheses A and B, we modeled communications with agent's actions, e.g.: `request_certificate`. Hypothesis C leads to many repetitions and to the use of a templating tool and code generation to describe multi-target interactions. E.g.: with three agents, `Agent1`, `Agent2` and `Agent3`, the ISPL representation of one agent checking if the other two are waiting is:

- In `Agent1`: `Agent2.Action = wait and Agent3.Action = wait.`
- In `Agent2`: `Agent1.Action = wait and Agent3.Action = wait.`
- In `Agent3`: `Agent1.Action = wait and Agent2.Action = wait.`

Lastly, hypothesis D means that proper evaluation of the energy and memory consumption of the algorithm should be done elsewhere.

On top of the constraints due to the hypotheses, we also decided to not add security properties and trust as it would greatly increase the complexity of the models. We relied on simulations, presented below, to validate these aspects.

5.2 Model

Due to the complexity of representing security and trust characteristics in ISPL, we only validated the self-organization algorithm. To do so, we divided it in three sub-behaviors: a default behavior, a self-organization behavior and re-organization behavior. Theses behaviors were defined such as the whole algorithm could be described as a first self-organization behavior followed by a default behavior with a re-organization behavior repeated as needed. Validating these behaviors would thus mean validating the self-organization behavior.

5.3 Default behavior

In an organization including at least one CA, eventually, each agent will own a valid certificate. This is the nominal behavior and it use as ground for the two others. In this model, one or more None agent will ask to a CA to deliver a certificate and the CA will do so. ISPL does not support function or structures, so, we used the `Environment` to indicate the destination and type of a message. To successfully send a message, an agent will first send it then it will check if the environment choose it and it will retry if not. In turn, the environment state will change to indicate which message is chosen to be sent. Its default behavior consists of waiting for a message and changing its state to choose a message. As choosing a message and changing its state so it can be read by the agents takes two steps, it also makes sure to allow agent to read the state before its changes again.

5.4 Self-organization behavior

From a set of Nones, with at least one agent able to become CA, an organization with at least one CA will emerge. In this model, all agents start as None and, using the algorithm described in Algorithm 1, they will set their role. In this model, agents with the capacity of becoming CA (`ca_able`) will decide or not to become one. If not, they will wait for a CA. However, if no CAs advertise as they wait, they will change their role to make sure that at least one CA is available. This behavior is the one a new agent should consider when entering the system.

5.5 Re-organization behavior

From an organization including at least one CA, if all the CAs stop, a new organization with at least one CA will emerge. In this model, several agents start as self-signed CA and the rest starts with their certificate signed by one of the CA. Then, the CA change their role and the None have to drop their certificate since the signer are not CA anymore. Moreover, some agent (as the one we present the model here) can become CA and thus, do become CA to ensure that there is at least one CA in the system. At last, None agent will request a certificate to the new CA.

It is to note that, the self-organization behavior and re-organization behavior situations are similar: CAs emerge from a set of None agents, we differentiate them here to validate two distinct parts of the self-organization algorithm. The self-organization behavior illustrates the choice each agent has to make, including waiting for others to choose before them. The re-organization behavior illustrates the adaptation mechanism when an incident (the loss of CAs) disrupts the system.

5.6 Validity of the models and results

As there is no tool to check that the models we wrote are a reflection of the algorithm, we used an incremental, in the number of agents, approach and manually checked the output graphs, using the `-exportmodel` option of the MCMAS binary. The models wrote during the whole process can be found in [**?**].

For each of these models, the execution of MCMAS leads to an execution graph coherent with our expectations and a result of true for each of the properties we described as to be verified.

6 Simulations

In this section, we present results obtain using one implementation of MAKI in Yet Another Multi-Agent Systems Simulator (YAMASS), an in-house multi-agent simulator based on the Mesa framework [14]. This results shows that revocation is achieved when an agent is deemed untrustworthy. We also used this implementation to observe the MAKI overhead, for example, of the number of

exchanged messages or of the time required to obtain, and renew, a certificate. The source code and instructions to reproduce all the results presented below are given in [**?**].

6.1 Simulator overview

We built YAMASS to provide a straightforward way to describe embedded agents behavior with a focus on simulations setup and reproducibility. In YAMASS, agents have coordinates, range and a battery. As such, they can only communicate with the agents in their range and have to rely on ad hoc routing to communicate with the other agents. All their characteristics are described in an input file along with the seeds used by the different pseudorandom number generator. Moreover, for total reproducibility, the agents are not attributed a thread but are executed in randomized sequence. However, this solution removes the notion of time from the simulation. A step in time includes one step per agent, in random order. During one step, an agent processes the messages it receives, sends the message it needs to send and then adapt its state accordingly. This means that, in one step, an agent can react to the messages it received and adapt its states, once, afterward.

A simplified UML class diagram of the simulator and MAKI is given in Figure 2. It shows the main elements of a MAKI agent, its TMS, its network stack and the behavior which it selects depending on its capabilities and its neighbors. The basic block of YAMASS are also showed, an interface representing an embedded agent and its network stack as well as the time emulation through the Clock and the EMASModel making sure that agents cannot exceed their capabilities. We also indicate the components of Mesa we rely on, including the Visualization Server allowing for graphical representation when necessary and the DataCollector used to collect trust values during the simulations.

6.2 Setup

To use YAMASS, we first had to implement a dynamic routing algorithm. We implemented a lightweight version of [13] which provide a way to establish and maintain routes between agents in a mesh network without prior knowledge on the network topology.

Then, we also implemented a minimalistic TMS based on a trust model with a low initial trust, the common mitigation to whitewashing presented in [21]. The model was tailored to the duration of the simulation. The value of the trust is bound between 0 and 1 with a growth following the function defined in Equation 1. (The value 10 was chosen to set the slop of the curve.)

$$f : x \mapsto \frac{x}{x + 10} \tag{1}$$

The *Low*, *Moderate* and *High* trust thresholds are respectively set to 0.2, 0.5 and 0.8 and the initial value is set to *Low*. In this model, a trust value below the *Low* thresholds implies that the agent is to be ignored. The graph of

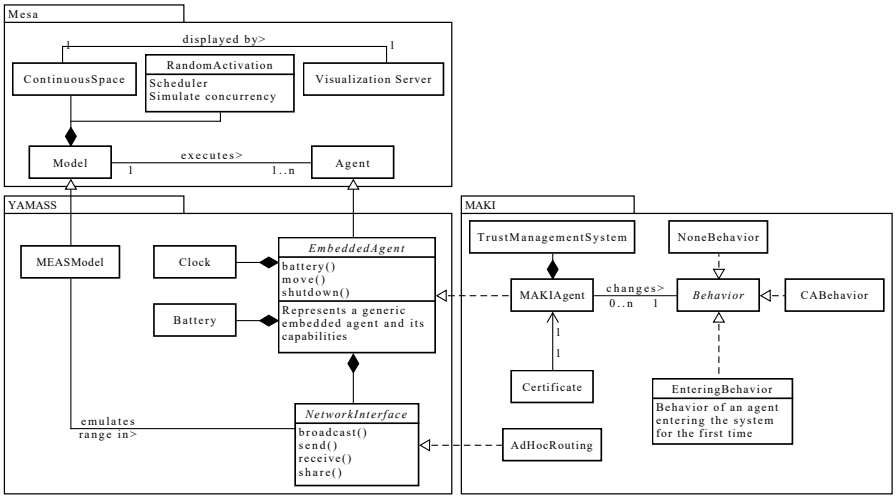

Fig. 2: Simplified UML class diagram of the simulator and MAKI

this trust growth function, with the thresholds, is given in Figure 3. The trust also slightly decay over time to compel agent to keep behaving correctly over time. Its value also depends on the duration of a simulation. 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. Trust from direct and indirect information is aggregated using the ratio of experience to recommendations, the more interactions an agent $\mathcal{A}_0$ will have with another agent $\mathcal{A}_1$, the less it will rely on recommendations to compute its trust in $\mathcal{A}_1$. Moreover, every agent eavesdrops on the communications to update in real time their trust model, in particular concerning the certificate deliveries. To cross-check the cooperation of the CA for example.

We followed the NIST recommendations [2] for our choice of cryptographic primitives. Thus, we used Elliptical Curve Cryptography with the Elliptic Curve Digital Signature Algorithm (ECDSA) and 256-bit keys using the P-256 Curve. The communications of MAKI are not meant to be encrypted, only signed, as all the information are public and might be eavesdropped for verification purpose. While ECDSA is the standard, it may not be adapted to embedded systems due to its high computational cost. The "NIST Report on Lightweight Cryptography" [18] can referred too for dealing with devices too constrained to run ECDSA.

We implemented a mock multi-agent application for the agents to cooperate and increase their trust in each other. This mock application consists of requests and replies between agents, each one increase the trust of the requester and requestee. Each step, each agent has a 0.7 chance to send a request to one of the agents it knows and the requestee will always correctly reply.

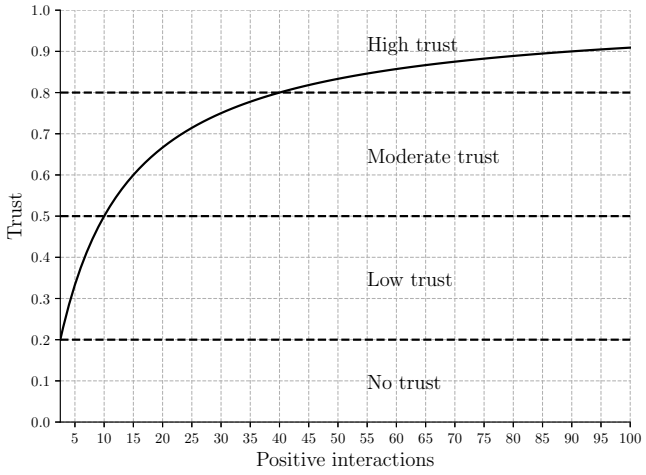

Fig. 3: Representation of the trust growth and trust thresholds in MAKI.

Table 2: Details on the number of simulations.

Number of agents	10	20	30	40
Density per number of agents	1–4	2–4	2–4	2–4
Configurations per density	10	10	10	10
Repetitions per configuration	5	5	5	5
Total	200	150	150	150
	650 simulations			

Then, we needed to define physical configurations representing different formation agents could take and ran enough simulations to make sure that no unexpected behavior would emerge and do gather execution traces to process. We used the fast Poisson disk sampling function [8] to populate the space given to the agents. This way, the agents were randomly placed will remaining near enough to communicate with each other but not close enough to form clusters. Using this method, we generate configuration of 10, 20, 30, 40, and ~ 200 agents. In each of the configuration, about 50% of the agents are able to become CA, independently to their position. To create diversity, we defined a "density" as the average number of neighbors (agents in communication range) an agent range (the higher the density, the more the agents will be packed) and try to obtain configurations with different density. Then, for each configuration and each density we ran 5 simulations of 500 steps each, with a certificate duration of 100 steps. The details on the exact number of simulations are given in Table 2.

Lastly, we defined scenarios describing the general behavior of the agents. In the next session, we focus on two scenarios:

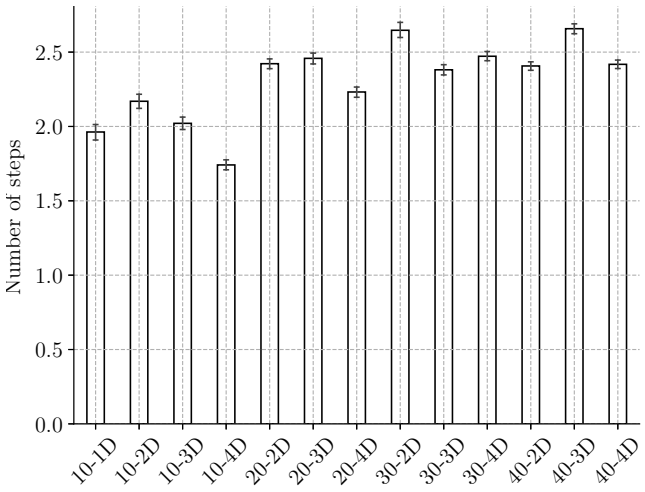

Fig. 4: Average time to obtain a certificate in systems with different numbers of agents and densities.

Scenario 0 The agents cooperate and all follows the rules. This scenario provides a baseline for the behavior of the agents. The results concerning the average time to obtain and renew a certificate and the general overhead, in messages exchange, of MAKI are computed from this scenario.

Scenario 1 After some time (300 steps), one randomly chosen agent will be deemed untrustworthy by $\sim$ 60% of the agent cooperating with it and those agents will request its revocation. This scenario aims at showing the efficiency of MAKI in excluding untrustworthy agent and the benefit compared to only using the TMS.

6.3 Results

Average time to obtain and renew a certificate While MCMAS allowed us to be confident in the self-organization algorithm validity, we also wanted to ensure its efficiency. In particular, we wanted to make sure that agents could easily and quickly obtain or renew their certificates. Especially since the CA able agents are not always homogeneously located in the systems which lead to situations where, even when using the self-organization algorithm we described, not all agents have a CA in their neighborhood, as it might simply not be possible. Using Scenario 0 on the 650 simulations described in Table 2 and measuring the average time between a certificate request and the advertisement of the newly received certificate, we obtained an average between 1.5 and 2.5 steps, as presented in Figure 4. These values seem reasonable as they show that, even without a CA in the neighborhood, obtaining or renewing a certificate is almost instantaneous.

Number of sent messages As the simulation only include a mock application, we do not try to compute an overhead but we rather estimate the number of necessary sent messages. A first situation where exchanges are required in MAKI is for an agent to obtain a certificate, it must send a request and the CA must send the certificate: 1 request and 1 reply, plus the number of messages required to route them, plus the routing messages to establish and maintain the route between the two agents if it wasn't done before, the cost of establishing and maintaining the route is also shared with the application since it also requires routing. Then, the newly certified agent may share its certificate. This process repeats each time an agent must renew a certificate. The total number of sent messages thus also depends on the duration of a certificate, the longest the duration is, the lesser the number of messages to send, but the harder it is to exclude an agent, this trade-off is to be considered in very constrains network with low latency. The overall number of necessary sent messages, for one certification is:

$$N_{certification} = 2 \times RouteLength + N_{sharing}$$

with $N_{sharing}$ the number of messages sent to share the certificate. $N_{certification}$ is then to be multiplied by the number of certificates to renew and the frequency of renewal. It is to note that the lower the number of CA, the higher the number of messages to send to renew all the not self-signed certificates and the longer the average route length.

A second situation where messages are to be sent is to request a revocation and share a CRL. As for the certificate sharing, the number of sent messages depends on the implemented strategy as discussed in Section 4.

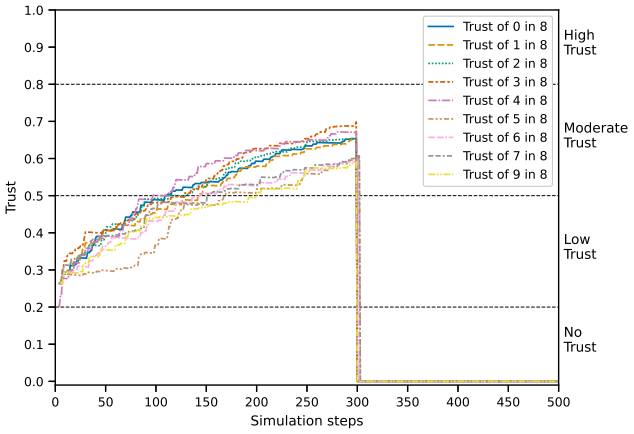

Fig. 5: Trust fluctuations in the malicious agent over time.

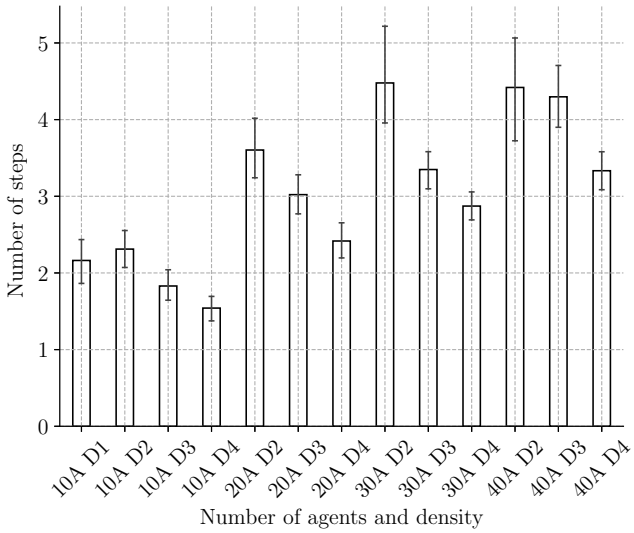

Fig. 6: Average time for the CRL to reach all the agents of the system, with different numbers of agents and densities.

Revocation of untrusted agents Using the same setup but with the second scenario, we could see that the (deemed) malicious agent was excluded in 500 of the 650 simulations. The exclusions happened in two steps: (i) the first aware agents aware lose all trust in the malicious agent and then request a revocation; (ii) when the CRL reach the other agents, their trust in the malicious agents drop to zero. An example of the trust fluctuation in the malicious agent in Figure 5. Just like for the time to obtain and renew a certificate, we estimate the time for the CRL to reach all the agents. The Figure 6 presents these results. We can see that, in most of the configuration, the revocation only takes from 2 to 5 steps, depending on the number of agents and the density, to be received by all the agents. The missing 150 (22%) exclusions are due to the lack of trust of the CA in the revocation requesters. This lack of trust can happen as CAs can cooperate with an agent (leading to earning enough trust for certification) but not with the agents the agent cooperate with. So, when the other agents ask for revocation, the trust the CA puts in them is not high enough to take the revocation into account. However, this is mainly due to the absence of a real organization at the multi-agent application level which is only lacking because we emulate it. We expect real applications to be less prone to this problem as the cooperation between agents may be more consistent, and less random, than in our simulations.

7 Conclusion

We introduced a decentralized public key infrastructure, coined MAKI, designed for open heterogeneous multi-agent systems of embedded agents and with a focus on autonomy on its management. This infrastructure allows agent to securely communicate and enforce the exclusion of intruders. This revocation is possible thanks to a subset of trusted certification authority agents maintained with no third parties involved.

Future works include a deployment of MAKI on embedded systems to confront our approach to real-time problematics. We are also exploring a solution to provide a way to easily share their identity and the lists of excluded agents, a minimalistic blockchain-based solution is currently being studied.

References

1. Avramidis, A., Kotzanikolaou, P., Douligeris, C., Burmester, M.: Chord-PKI: A distributed trust infrastructure based on P2P networks. Computer Networks **56**(1), 378–398 (2012). https://doi.org/10.1016/j.comnet.2011.09.015
2. Barker, E., Dang, Q.: Recommendation for Key Management Part 3: Application-Specific Key Management Guidance (2015). https://doi.org/10.6028/NIST.SP.800-57pt3r1
3. Baudet, A., Aktouf, O.E.K., Mercier, A., Elbaz-Vincent, P.: Systematic Mapping Study of Security in Multi-Embedded-Agent Systems. IEEE Access **9**, 154902–154913 (2021). https://doi.org/10.1109/ACCESS.2021.3128287
4. Blanch-Torné, S., Cores, F., Chiral, R.M.: Agent-based PKI for Distributed Control System. In: 2015 World Congress on Industrial Control Systems Security (WICICSS). pp. 28–35 (2015). https://doi.org/10.1109/WCICSS.2015.7420319
5. Boeyen, S., Santesson, S., Polk, T., Housley, R., Farrell, S., Cooper, D.: Internet X.509 Public Key Infrastructure Certificate and Certificate Revocation List (CRL) Profile. RFC 5280 (2008). https://doi.org/10.17487/RFC5280
6. Bonnaire, X., Cortés, R., Kordon, F., Marin, O.: A Scalable Architecture for Highly Reliable Certification. In: 2013 12th IEEE International Conference on Trust, Security and Privacy in Computing and Communications. pp. 328–335 (2013). https://doi.org/10.1109/TrustCom.2013.44
7. Boubiche, D.E., Athmani, S., Boubiche, S., Toral-Cruz, H.: Cybersecurity Issues in Wireless Sensor Networks: Current Challenges and Solutions. Wireless Personal Communications **117**(1), 177–213 (2021). https://doi.org/10.1007/s11277-020-07213-5
8. Bridson, R.: Fast Poisson Disk Sampling in Arbitrary Dimensions. In: ACM SIGGRAPH 2007 Sketches. pp. 22–es (2007). https://doi.org/10.1145/1278780.1278807
9. Cui, H., Deng, R.H.: Revocable and Decentralized Attribute-Based Encryption. The Computer Journal **59**(8), 1220–1235 (2016). https://doi.org/10.1093/comjnl/bxw007
10. Dumas, J.G., Lafourcade, P., Melemedjian, F., Orfila, J.B., Thoniel, P.: LocalPKI: An Interoperable and IoT Friendly PKI. In: E-Business and Telecommunications. pp. 224–252 (2019). https://doi.org/10.1007/978-3-030-11039-0_11

11. Goffee, N.C., Kim, S.H., Smith, S., Taylor, P., Zhao, M., Marchesini, J.: Greenpass: Decentralized, PKI-based Authorization for Wireless LANs. In: In 3rd Annual PKI Research and Development Workshop. pp. 26–41 (2004)
12. Jhaveri, R.H., Patel, N.M.: Attack-pattern discovery based enhanced trust model for secure routing in mobile ad-hoc networks. International Journal of Communication Systems **30**(7), e3148 (2017). https://doi.org/10.1002/dac.3148
13. Johnson, D., Hu, Y., Maltz, D.: The Dynamic Source Routing Protocol (DSR) for Mobile Ad Hoc Networks for IPv4. RFC 4728 (2007). https://doi.org/10.17487/RFC4728
14. Kazil, J., Masad, D., Crooks, A.: Utilizing Python for Agent-Based Modeling: The Mesa Framework. In: Social, Cultural, and Behavioral Modeling. vol. 12268, pp. 308–317 (2020). https://doi.org/10.1007/978-3-030-61255-9_30
15. Kukreja, D., Dhurandher, S.K., Reddy, B.V.R.: Power aware malicious nodes detection for securing MANETs against packet forwarding misbehavior attack. Journal of Ambient Intelligence and Humanized Computing **9**(4), 941–956 (2018). https://doi.org/10.1007/s12652-017-0496-2
16. Lesueur, F., Me, L., Tong, V.V.T.: An efficient distributed PKI for structured P2P networks. In: 2009 IEEE Ninth International Conference on Peer-to-Peer Computing. pp. 1–10 (2009). https://doi.org/10.1109/P2P.2009.5284491
17. Lomuscio, A., Qu, H., Raimondi, F.: MCMAS: an open-source model checker for the verification of multi-agent systems. International Journal on Software Tools for Technology Transfer **19**(1), 9–30 (2017). https://doi.org/10.1007/s10009-015-0378-x
18. McKay, K., Bassham, L., Turan, M.S., Mouha, N.: Report on Lightweight Cryptography (2017). https://doi.org/https://doi.org/10.6028/NIST.IR.8114, https://tsapps.nist.gov/publication/get_pdf.cfm?pub_id=922743
19. Okamoto, T., Takashima, K.: Decentralized Attribute-Based Encryption and Signatures. IEICE Transactions on Fundamentals of Electronics, Communications and Computer Sciences **E103.A**(1), 41–73 (2020). https://doi.org/10.1587/transfun.2019CIP0008
20. Qin, B., Huang, J., Wang, Q., Luo, X., Liang, B., Shi, W.: Cecoin: A decentralized PKI mitigating MitM attacks. Future Generation Computer Systems **107**, 805–815 (2020). https://doi.org/10.1016/j.future.2017.08.025
21. Ruan, Y., Durresi, A.: A survey of trust management systems for online social communities – Trust modeling, trust inference and attacks. Knowledge-Based Systems **106**, 150–163 (2016). https://doi.org/10.1016/j.knosys.2016.05.042
22. Singla, A., Bertino, E.: Blockchain-Based PKI Solutions for IoT. In: 2018 IEEE 4th International Conference on Collaboration and Internet Computing (CIC). pp. 9–15 (2018). https://doi.org/10.1109/CIC.2018.00-45
23. Yakubov, A., Shbair, W.M., Wallbom, A., Sanda, D., State, R.: A Blockchain-Based PKI Management Framework. In: NOMS 2018 - 2018 IEEE/IFIP Network Operations and Management Symposium. pp. 1–6 (2018). https://doi.org/10.1109/NOMS.2018.8406325