Decentralized Public Key Infrastructure for Autonomous Embedded Systems

Arthur Baudet[1], Oum-El-Kheir Aktouf[1], Annabelle Mercier[1] and Philippe Elbaz-Vincent[2]

[1]Univ. Grenoble Alpes, Grenoble INP, LCIS, Valence, France
[2]Univ. Grenoble Alpes, CNRS, Institut Fourrier, Grenoble, France

Abstract
In this paper, we tackle the issue of security of multi-agent systems of embedded agents. These systems provide scalable and flexible ways to control complex, distributed and interconnected systems of embedded components, which can connect to and disconnect from the system during runtime. The lack of central authority makes such systems more dynamic and adaptive. However, securing these systems is challenging and raises many issues. In this work, we aim at providing a public key infrastructure to enable agents to securely connect to the system while it runs and without the need to load certificates beforehand. To do so, we establish an infrastructure where agents generate their own keys and ask for certificate from certificate authorities. Those authorities act without the need to coordinate themselves and distribute certificates to requesters, following the rules of a trust management system. The infrastructure provides the ability for the agents to obtain certificates and establish secure communications between themselves without the need of an external, centralized system.

Keywords
Multi-Agent System, Public Key Infrastructure, Embedded System, Security

1. Introduction

Multi-agent system (MAS) is a paradigm for decentralized computing, which can be used in applications such as Wireless Sensor Networks, Internet-of-Things, Autonomous Vehicles, etc. In these systems, collaboration among agents is a key feature. To make this collaboration reliable and relevant, it is important for an agent to trust the agents with whom it interacts and to be confident that the information it receives is trustworthy.

In this context, we focus our effort on open multi-agent systems of embedded agents (MEAS). This kind of systems allows a high level of autonomy and adaptability as agents can enter and leave the system at any time. Agents are themselves autonomously deciding with whom they should collaborate. However, there are several restrictions on communication, energy, and storage as each agent is an embedded device. Such systems

C&ESAR'22: Computer & Electronics Security Application Rendezvous, Nov. 15-16, 2022, Rennes, France

arthur.baudet@lcis.grenoble-inp.fr (A. Baudet); oum-el-kheir.aktouf@lcis.grenoble-inp.fr (O. Aktouf); annabelle.mercier@lcis.grenoble-inp.fr (A. Mercier); philippe.elbaz-vincent@univ-grenoble-alpes.fr (P. Elbaz-Vincent)

can be seen as communities of agents autonomously cooperating to achieve their own goals and eventually achieve the global goal of the entire system. However, most of those MEAS rely on wireless ad hoc networks, which increases the likelihood of a wide range of attacks.

The traditional way of securing communications in an open system, including the Internet, requires the use of a Public Key Infrastructure (PKI). But, traditional PKIs are based on a central hierarchy of trusted third parties, which conflicts with the design of a decentralized MAS. This is why we propose Multi-Agent Key Infrastructure (MAKI), a minimalist PKI where trust is established without the need of third-parties, and which relies on the two most important features of MAS, the cooperation between agents and their autonomy.

The rest of the paper is organized as follows: the background and related works are presented in the next section; while Section 3 describes the infrastructure itself. We discuss security considerations and limitations in Section 4. An initial implementation in a MEAS simulator and obtained results are given in Section 5. Finally, we conclude the paper in Section 6.

2. Background and related work

MEAS are particularly vulnerable as they are affected by physical threats on their hardware and by software and network threats on their code and communication, but also threats targeting their collaboration algorithms and cognitive behavior. Meta-studies showed that many works focus on securing such systems using various tools such as trust management systems, intrusion detection systems, security norms enforcing or plain cryptography [1, 2, 3].

However, cryptographic constraints are not always considered in works deploying trust management systems (TMS) [4, 5]. This is problematic as communications are often required to propagate trust value among agents and that an attacker could thus easily tamper a message to change the trust value exchanged between two agents. As in [6], agents are often expected to enter the systems with already loaded certificates and keys. This assumption is hard to maintain in MEAS where evolution and adaptation are two main features.

Therefore, our work focuses on designing an infrastructure allowing cryptographic computation without the need for the agents to fetch and carry certificates or any cryptographic data before connecting to the MEAS.

To this end, Blockchain Technology (BCT) is a widely used solution [7, 8, 9]. It provides means to share information in a secure decentralized way and are used to share the necessary information to allow the deployment of a PKI. However, BCT are not perfectly adapted to MEAS, even if we deploy consensus algorithms less computationally intensive than the Proof-of-Work (PoW) one, they still suffer from a very high cost in storage and latency, especially in ad hoc networks. For example, blockchains using the Proof-of-Stake, which does not rely on computations to obtain a consensus as reliable as the one obtained with the PoW, still use an ever growing data structure to store the information.This is not suitable for MEAS.

BCT are not the first attempt at trying to decentralize PKI. The authors of [10] used threshold cryptography and those of [11] relied on Simple Distributed Security Infrastructure (SDSI) that provides a decentralized PKI. However, both solutions required either off-band verification or pre-loaded certificates to provide authentication. The work presented in [12] proposes an enhanced Distributed PKI for industrial control systems using an agent-based framework but it requires an operator to add or remove system from the PKI. Both systems in [13, 14] provide a way to decentralize certificates authorities (CAs) into a distributed hash table, which is used to keep track of the certificate distribution and revocation. While these works solve the problem of consensus when managing certificates, they lack a way to filter out untrustworthy nodes.

Another approach to solve this problem could be to remove the need for certificates. This can be done using identity- or attribute-based encryption [15, 16] but requires agents to be able to authenticate themselves. As a consequence, the systems should have prior knowledge of the agents connecting to the MEAS. This assumption is also hard to meet in our case.

Thus, we develop a specific infrastructure, designed for MEAS and leveraging the autonomy and cooperative behavior of the agents that do not require neither pre-loaded certificates nor off-band verification and no high volume of memory or computational power.

3. Infrastructure

3.1. Goal and threat model

The objective of our work is to develop an infrastructure allowing all the agents to send messages and interact with the assurance of the authenticity and the integrity of their exchanges, including the messages exchanged when applying a trust management system.

To do so, we consider the following hypotheses: (H1) Authentication of an agent is not required when connecting to the system. This means that the names of the agents are only a way to differentiate them. The cooperation between agents will be based on the capabilities announced by each agent. (H2) Agents are capable to sign the messages they send using asymmetric keys. (H3) Private keys can not be compromised, agents never emit their key and their hardware is either hardened enough or inaccessible enough to prevent physical attacks.

In this work, we focus on communication-related attacks with mote-class attackers, i.e., attackers with similar resources as the agents of the system. As MEAS often rely on ad hoc, wireless networks, we assume that attackers are able to eavesdrop on the communication media and manipulate it by tampering with messages.

3.2. General concept

The core concept of MAKI is to autonomously distribute the role of CA to the agents. CAs are responsible for generating, signing, and revoking certificates used by the other members of the PKI. By doing so, MAKI does not require external third-parties and does not rely on a single agent. Its most fundamental rule is that every agent should hold a valid certificate to be authorized to take part in the system. The role of CA is thus fundamental and it must be considered with care when defining algorithms for assigning this role to agents. To allow as most flexibility and resilience as possible, we intend to allow every agent to decide for itself whether or not it should endorse the role of CA leveraging an existing TMS, or a new TMS if there is none, to guide its decision to trust or not a CA, or to become one if none are available. We consider the autonomy of the agents and decentralized feature of MAS which prevents the risk of single point of failure and increases the scalability of the system.

MAKI is threefold. The use of cryptographic primitives allows agents to sign their messages. That alone is enough to satisfy the need of integrity and authenticity when agents are exchanging messages, especially when using the TMS. Moreover, MAKI also provides a way to enforce the decision taken by the TMS by adding the requirement of holding a valid certificate and thus most likely globally excluding an agent by revoking its certificate. However, the use of certificates requires an authority to deliver them, authority that should not impede the autonomy of the MEAS. Lastly, MAKI provides a way to decide which agents should be authority and a way to filter out malicious authorities by leveraging and extending the TMS in use.

3.3. Identity definition

Only knowing the name, an identifier such as a number or a string, of an agent, is not enough to establish a communication with it as it can very easily be spoofed. To prevent identity spoofing or stealing, we use a name and a public key, or only a public key, to define the identity of an agent. This prevents any attempt of identity spoofing but not Sybil attacks [17] since we have no way to prevent a malicious agent from generating a new name and a new set of keys. Moreover, the use of TMS also reduces the impact of Sybil attacks as the attacker would have to earn trust for all its identities before using them.

3.4. The Certificate Authority in MAKI

CA is a role, a behavior, one agent can decide to endorse if it considers it necessary. Once an agent becomes CA, it is allowed to self-sign its certificate and will inherit responsibilities. Those responsibilities are (i) delivering, and updating, certificates to other agents, (ii) maintaining the list of certificates it delivered and (iii) maintaining the list of certificates it revoked.

Since CAs are self-signed, it is impossible to revoke their certificates. However, they can still be ignored if they do not behave properly. For example, a CA that, after a given amount of time, has not delivered certificates when new agents are joining the system will be judged suspicious since it does not behave as a CA as it should. Then, its legitimacy may decrease. And, even though its certificate can only be revoked by itself, if it is not legitimate, it will be considered as revoked and thus agents will not interact

Algorithm 1 How an agent chooses its role

```
   T ∈ [0, 1)
1: CAs ← SendToInRange(CAPresenceRequest)
2: TrustedCAs ← FilterByTrust(CAs, TrustLevel.Moderate)
3: if can become CA and (TrustedCAs is empty or Random(0, 1) > T) then
4:     Role ← CA
5: else
6:     Role ← NotCA
```

with its holder. This rule of checking CAs behavior applies for the three responsibilities listed above. By doing so, we decrease the risk of malicious agents simply taking on the role of CA to prevent any exclusion attempt and ensure that the privilege of being a CA always come with the responsibility to serve other agents.

Moreover, we consider adding cross-certification for CAs that would want to share their trust and distrust with another CA. By simply doing so, they earn more legitimacy since they now are liable to be revoked by another CA.

The number of CAs needed for a system to be efficient dependents on run-time parameters such as the number, the position of the benevolent agents and malicious ones but also on the capabilities of the agents. For example, to maintain low latency when delivering certificates, it can be suitable to have each agent in range of a CA. This means that the proportion of CAs to the total number can be expressed as a variation of:

$$P = \frac{N}{D \times R} + O \quad \text{with}$$

P	the proportion of benevolent CAs
R	the average range of agents
D	the density of benevolent agents
N	the total number of CAs
O	a positive offset, proportional to N to prevent single-point-of-failure situations

However, if N and D varies at run-time, P will too. This is why we let the agents choose for themselves if they deem necessary to become a CA. A simple algorithm describing a way to make this choice is given in Algorithm 1. The O is implemented as a probability of one agent becoming CA even though there already is one in its range.

Even if this is for an example with an objective of having at least one CA in range of all agents, we could simply tailor the algorithm by changing the definition of “in range” (`SentToInRange`) by the maximum of agents necessary to route a message from its source to a CA.

3.5. Certificate management

The certificate and Certificate Revocation List (CRL) formats were adapted to fit the MAKI needs. The certificate format is much simpler than an X.509 one [18]. However, communicating with, and so obtaining the certificate for, a specific agent is very difficult without a trusted third party or without knowing its public key. So, each certificate includes the public key of the deliverer. We added one field to allow systems designers add relevant information on the agent, the agent role in the application for example. The main difference with an X.509 certificate is that we include the public key of the issuer since it is part of its identity. 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. From the ten possible values of this field, only the values `KeyCompromise (1)`, `CACompromise (2)`, `Superseded (4)` and `CessationOfOperation (5)` are used. The `unspecified (0)` is forbidden so that giving a reason for a revocation is mandatory.

Certificates propagation relies on a combination of gratuitous broadcast and adding them to the exchanged messages, depending on the network throughput and the acceptable overhead on the messages.

Certificates are revoked through two mechanisms. The first one is the CRL; when necessary, a CA will update its CRL and broadcast it. This will allow an immediate revocation but, as the communications are ad hoc, it does not assure that the revocation will reach immediately each agent. To complete the usage of CRL, we also consider using short-lived certificates. By doing so, revocation orders only need to reach the CAs to ensure that the targeted agent will eventually be revoked since its certificate will expire and the CAs will not renew it.

3.6. Trust management

MAKI is originally intended to be deployed in coordination with a TMS. We provide in this section the rules and criteria used to extend the TMS to include MAKI prerogatives. If there is no TMS, we also provide some essential explanation on TMS as well as guidelines to deploy a minimalistic TMS satifying the requirements of MAKI. As stated in [1], there are currently many researchers working on that topic.

Trust and reputation are very common mechanisms to increase resilience in MAS [19, 20]. When using a TMS, each agent has to compute the trust it has in the other agents it interacts with, including the results of its previous interactions or the trust that other agents have computed themselves. Moreover, the level of trust an agent requires from another agent to interact with also depends on a trade-off between the risk and the need for this interaction. Contacting a CA to request a certificate is mandatory and with limited risk since no information is disclosed by doing so. However, if the CA is revoked, the certificate loses its relevance.

TMS are commonly split in three parts: trust management, trust modelling and decision making. The first part describes how the information is gathered, the second describes how the trust is represented and information aggregated and the last one describes how the decisions are made depending on the trust values.

In our case, trust gathering includes direct information, i.e., the information from experiencing the interaction with the agent and indirect information, i.e., the information collected by asking other agents. Since the communications are wireless and not encrypted, for the exchanges to establish MAKI at least, we consider using eavesdropping on communications to corroborate the declarations of an agent. For example, if a CA refutes

Table 1
Trade-off between risk and benefit for each possible interaction between agents depending on their role.

Interaction	Risk	Benefit	Required trust level
		Certificate Authority	
Requesting or accepting cross-certification requests	High. Having its reputation decreased if the cross-certifier is distrusted, giving more legitimacy to a malicious CA.	Low. But higher trust is given to cross-certified CA.	High
Delivering a certificate to an agent	Moderate. Providing legitimacy to a malicious agent.	High. Having its own legitimacy increase since one more agent would be trusting it to provide a trustworthy certificate	High
Revoking a certificate on demand of an agent	Moderate. Having its reputation decreased if all the agents agree with the revocation.	High. Helps the overall system by excluding a malicious agent.	High
		Not CA	
Requesting certification delivery	Moderate. If the CA is distrusted, having its reputation decreased and having to renew the certificate with another CA	High. Holding a valid certificate is mandatory to take part in the system.	Moderate
		Any	
Requesting an agent certificate or sharing its certificate	None	High. Each identity should be linked to a valid certificate in order to be used.	None

The term “reputation” is used solely as a way to describe the trust other agents may have in one agent.

having delivered a certificate to a malicious agent, an accuser could provide the signed exchange between the CA and the malicious agent in which the delivery is done. However, eavesdropping requires the agents to stay in promiscuous mode which is very energy consuming. Depending on the role taken by an agent, we can establish criteria for the TMS. These are indicators of the implication of the agent in MAKI, to prevent selfish behaviors or indicators of the concordance of the behavior with the other agents. For the CA role, the criteria are as follows:

- A high number of delivered certificates means that it is implicated in MAKI at the cost of its own computation and its energy and that a large number of agents trust it to deliver their certificates.
- A cross-certified CA share the trust with its cross-certifier and it shows that it

is ready to take the risk of being cross-certified with a malicious CA to be more trustworthy and useful.

- Refusing to deliver any certificate, hence only being a CA to be able to self-sign its certificate is a red flag. It means that, at best, the agent is selfish but most-likely, it wants to make it harder to be excluded.
- Revoking a certificate with no good reason or proof is also a red flag. CAs should not be allowed to act unilaterally.

Overall, any agent caught lying, or undertaking an attack such as a Sybil attack should lose the trust of the system and have its certificate revoked. For all other intents, the certificates delivered by an agent $\mathscr{A}$, a CA, will have a value for an agent $\mathscr{B}$ that is related to the mutual trust between $\mathscr{A}$ and $\mathscr{B}$. The trust $\mathscr{B}$ will have in $\mathscr{A}$ is related to the trust $\mathscr{B}$ has in the agents holding a certificate delivered by $\mathscr{A}$.

We established guidelines for the decision making part. They are presented in Table 1. They are based on the trade-off between the risk and the benefit of undertaking an interaction with other agents depending on their needs and their roles. We defined three level of trust, `Low`, `Moderate` and `High` that the target agent should satisfy in order to allow an interaction. Although, a CA has no information about newcomers when they first ask for a certificate so we choose not to hold them responsible if an agent using a certificate it delivered is excluded. This also removes the incentive of refusing to revoke the certificate since that would correspond to admitting a fault. This can be moderate in the case of a CA blindly delivering certificate to agents that were revoked when it seems obvious that the CA could have known about the revocation.

Lastly, we do not assume to be able to choose one model to fit all requirements for all kinds of MEAS as there are many trust models in the literature ranging from linear sums [21], to more complex mathematical systems [22] or even sociological models [23]. But, if there is no TMS originally, we would recommend to use a liner model or equivalent to reduce the overhead of MAKI.

One difficulty about TMS is to decide whether or not to trust newcomers, especially in systems where an agent could easily change its identity when it is identified as malevolent or even plain excluded. One way to mitigate this behavior named “whitewashing” is to trust newcomers very little. This way, a whitewasher would have to behave properly in order to earn the trust of the other agents before trying to exploit them.

3.7. Bootstrapping and adding new agents

Starting the systems with unknown and possibly malicious agents, could lead to malicious agents obtaining disproportionate weight in the system, which would be disastrous. With the hypothesis that the first benevolent agents of MAKI are deployed by the same entity, for example, operators deploy the first agents and open the system to external agents later on, we propose two ways of bootstrapping MAKI. The first way would be to hard-code some initial trust values for certain agents. It could be acceptable since the operators are in control of the initial conditions so that they know that all agents are benevolent. The second way would be to run MAKI in a controlled environment long enough so that some

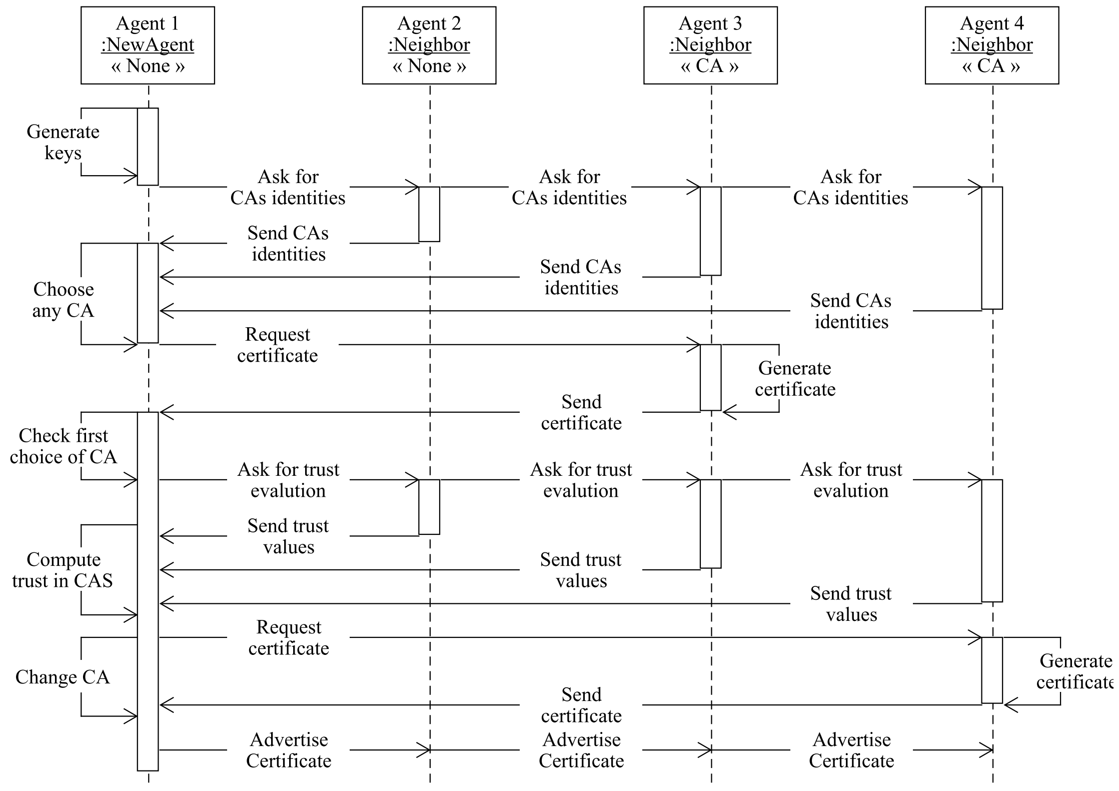


Figure 1: An agent joining the system and choosing a trustworthy CA

agents obtain sufficiently high trust to steer the whole system in the right direction. Both ways can also be complementary, we could hard-code some values and let the system organize around them. The main point to consider is the time taken by the system to stabilize which also depends on the population and the density of agents and should be studied on a case-by-case basis.

Adding a new agent is quite simple from an insider point of view, one CA should deliver to it a certificate and it would only be trusted with lower importance tasks until it earns trust from the agents it interacts with.

From the new agents point of view, connecting to MAKI is straightforward. It has to find out the identity of one CA and request a certificate. To do so, it can either eavesdrop on the communications or broadcast a request. Once it has a certificate, it can probe the state of the system and decide whether or not it made the right choice or if it should change its CA or become a CA. Figure 1 illustrate such an example.

4. Security considerations and limitations

By signing every interaction, we ensure that the source of each interaction can be identified and held accountable to fuel the TMS, leading to the detection and possible exclusion of untrustworthy agents. Signing also allows the detection of any attempt in tampering with a message during its transmission. However, detection and exclusion do not apply to an agent but to an identity, i.e., a pair of name and public key. Nevertheless, we can reduce the impacts of agents continually generating new identities using detection at network level as described in Section 3.3 and by not trusting newcomers right away.

In terms of the intrusion detection, MAKI relies on a TMS and, thus, is at least as efficient as the TMS. Moreover, MAKI does not create new ways for malicious agents to circumvent the TMS as, even though they can not be revoked, CAs can still be excluded by being ignored and given a very bad trust review when asked. The simple model we proposed is vulnerable to some attacks such as on-off attacks, selective-lying attacks and such, but it is only given a simple placeholder until a more rigorous model is implemented.

The number of CAs also impacts the security MAKI provides, too few CAs implies a high risk of having to rely on a malicious CA; too many makes the choice harder as, most likely, none of the CAs will be able to meet the minimal number of distributed certificates.

The ratio of malicious to benevolent agents is only interesting to look at the scale the agents can collaborate. At this scale, the ratio can not exceed fifty percent since it would mean that the malicious agents can collude to exclude any agent arbitrarily. Under this limit, as it is the responsibility of the TMS to detect the malicious agents, the ratio can be as high as the TMS allows it.

MAKI adds a necessary overhead in terms of the number and the size of exchanged messages, storage, computation and time. First, messages need to be exchanged to request and deliver certificates and to find trustworthy CAs. At least one request and one reply are necessary to get a signed certificate. The size of messages also increases since they now include a signature and possibly a certificate. Second, it is necessary for CAs to store the certificates they have delivered as well as a CRL. NotCA agents only have to store their current certificate and are advised to store the certificates of the agents they interact with the most. Third, the computation requirements are increased due to the use of cryptographic primitives (see Section 5 for the details on an example of such primitives). Last, due to the above mentioned overhead, the time before an agent can start collaborate after joining the system is increased since it first has to get a certificate.

5. Proof-of-Concept

The code of the proof-of-concept and the totality of the execution traces and results are freely available at [24].

5.1. Technical choices

As stated, MAKI is agnostic of cryptographic choices and most of the trust models. Cryptographic choices and TMS should be tailored to the real application of the MEAS and capabilities of the agents.

For our experiment, we followed the NIST recommendations [25] on key size and signing algorithm for non-repudiation and CA signing keys and used Elliptical Curve

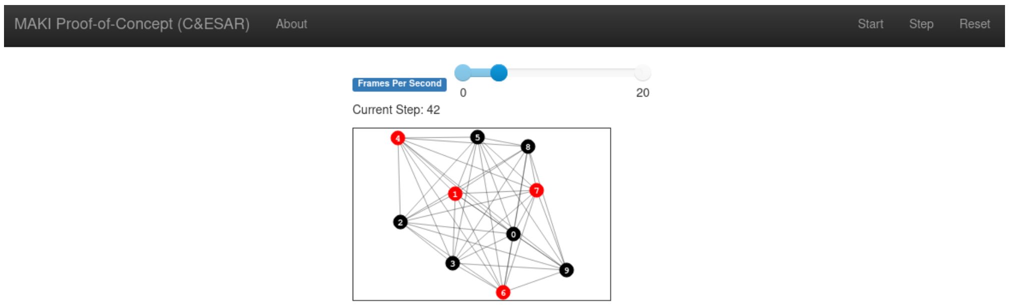


Figure 2: Agents execution inside our simulator, seen using the Mesa interface. Red agents are CAs, blacks agents are NotCA

Cryptography (ECC) with the Elliptic Curve Digital Signature Algorithm (ECDSA) and 256-bits keys using the P-256 Curve.

For trust modelling, we defined the trust as a value starting at 0 and increasing infinitely. We set the increments to small values such as 1 or 2 and defined the function of the trust growth rate to: $Trust : x \mapsto x + A$ with A, the `Low` trust level.

We fixed the trust level according to trust growth rate, `Low` is `1` so that `0` trust means being untrustworthy, `Moderate` is set to `30` and `High` to `90`. This prevents free whitewashing attacks without being too harsh and preventing agents to earn their place in the system. We used without-compromise approach for the trust loss, as of now, every malicious behavior is punished by a total loss of trust, effectively setting the trust in the misbehaving agent to `0` independently from its previous trust. This particular model was designed for demonstration purpose and may no fit all the modelling needs of real-world MEAS. More advanced and tailored models are available depending on the deployed MEAS. As of now, the trust model is only fed with four kinds of information:

- "Agent $\mathscr{X}$ collaborated successfully." This is used for bootstrapping and is discuss in the next section.
- "CA $\mathscr{X}$ delivered a new certificate." This increases the trust in $\mathscr{X}$ which is used to weight the trust of the certificate holder. Thus, a CA has a real incentive to deliver certificate since it earns more trust each time.
- "The certificate of agent $\mathscr{X}$ is revoked." The trust in $\mathscr{X}$ is nullified.
- "The agent $\mathscr{X}$ is malicious." The trust in $\mathscr{X}$ is nullified and a certificate revocation request will be sent to the CA that delivered the certificate $\mathscr{X}$ is using.

5.2. Experimentation

This proof-of-concept was implemented using an in-house multi-embedded-agents simulator built on top of the Mesa agent modeling framework [26]. A screenshot of an execution of MAKI is given in Figure 2. We uses the configuration of ten agents, four of which are CAs, as shown in the figure for the rest of the experimentation. It is worth noting that we only studied the behavior at a local scale in the sense that all agents are in the range of one another. This greatly reduced the complexity of the ad-hoc routing, which is out of the scope of our work, and helped us to precisely show the changes in trust at the scale multi-agent systems are meant to operate. The number of CAs we choose enables us to describe more possible events within the same configuration and allowed us to implement five scenarios describing different behaviors of agents. Furthermore, we added, at each round, for each agent, a gain of trust in one of its neighbors chosen randomly. This is our implementation of bootstrapping. It is equivalent to have the system running in a controlled environment where all the agents are behaving correctly. Since trust can only increase by collaborating and collaboration requires a certificate, we also jump-started bootstrapping by adding an acceptable risk of choosing a malicious CA, which should not happen in a controlled environment. Thus, when there is no good option, an agent not being able to become a CA may choose to temporary trust a CA instead of waiting for a better CA to appear.

We designed five scenarios that are derivations of the six following steps:

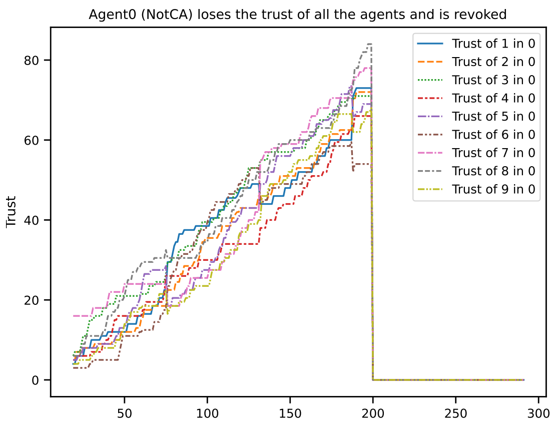
1. N agents detect malicious behavior from an agent $\mathscr{E}$.
2. The trust in $\mathscr{E}$ of these N agents drops to 0.
3. If $\mathscr{E}$ certificate is not self-signed, the N agents request to the issuer a revocation of $\mathscr{E}$ certificate.
4. If the certifier agrees to it, it revokes the certificate and advertises it.
5. All the agents learning about the revocation will lose their trust in $\mathscr{E}$ if not already done.
6. $\mathscr{E}$ will not receive any new certificate.

In the first scenario, N is the totality of the agents (except $\mathscr{E}$). In this case, everything happens as described above with the step 5 skipped since all the agents already lost their trust in $\mathscr{E}$. The result is presented in Figure 3a.

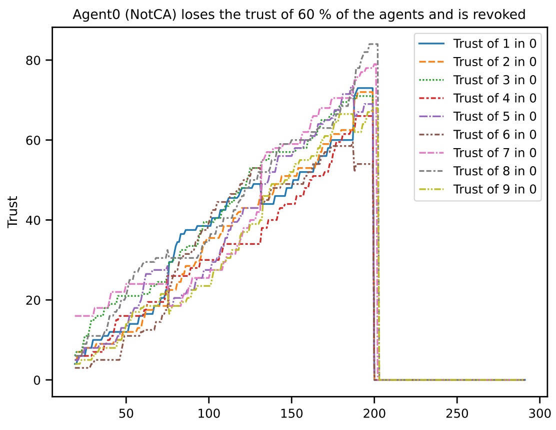
In the second scenario, N is 60% of the agents, including or not the certifier. In this case, if more than half of the agents the certifier trust are in the 60%, which is the case in the configuration we chose, everything happens exactly as described above. The result is presented in Figure 3b.

In the third scenario, N is only 40% of the agents, including or not the certifier. In this case, it is less likely that the 40% of the agents are more than half of the agents trusted by the certifier. It is not the case in the configuration we chose, and steps 4 to 6 do not happen. The rest of the agents continue to trust $\mathscr{E}$. The result is presented in Figure 3c.

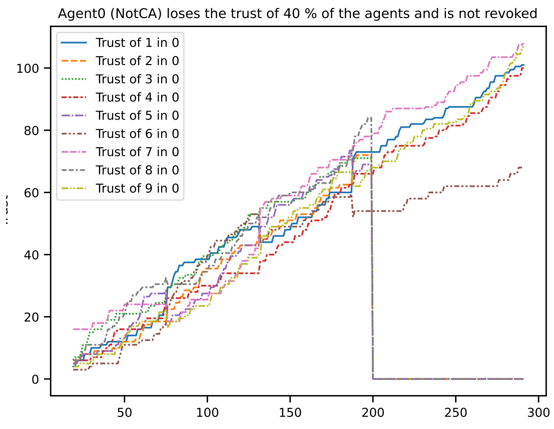
For the last two scenarios, $\mathscr{E}$ is a CA and its certificate is self-signed and thus can not be revoked. In the first scenario, N is all the agents making their trust in $\mathscr{E}$ drops to 0, and agents certified by $\mathscr{E}$ will change their CA. However, some agents may not have time to change CA and will be revoked and excluded. Future implementations will account for this particular timing problem by waiting a little before requesting the revocation of certificates delivered by a malicious CA. As in the first two scenarios, every agent will ignore $\mathscr{E}$ and new agents may reach the same conclusion after checking the number of certificates signed by $\mathscr{E}$ in use. It is less direct than being able to revoke $\mathscr{E}$ but using the number of delivered certificates rule and indirect information will eventually lead to the same result. The result is presented in Figures 3d.



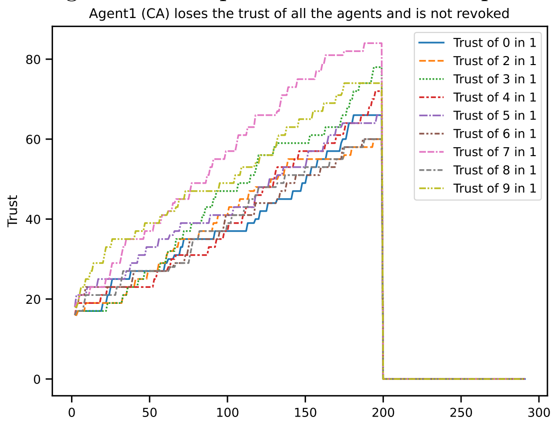
(a) Scenario 1. Trust in agent 0 drops as all agents detect its malicious behavior.



(b) Scenario 2. Trust in agent 0 drops for the agents that initially detected its malicious behavior. Then, the trust of all the other agents in it drops as the revocation is spread.



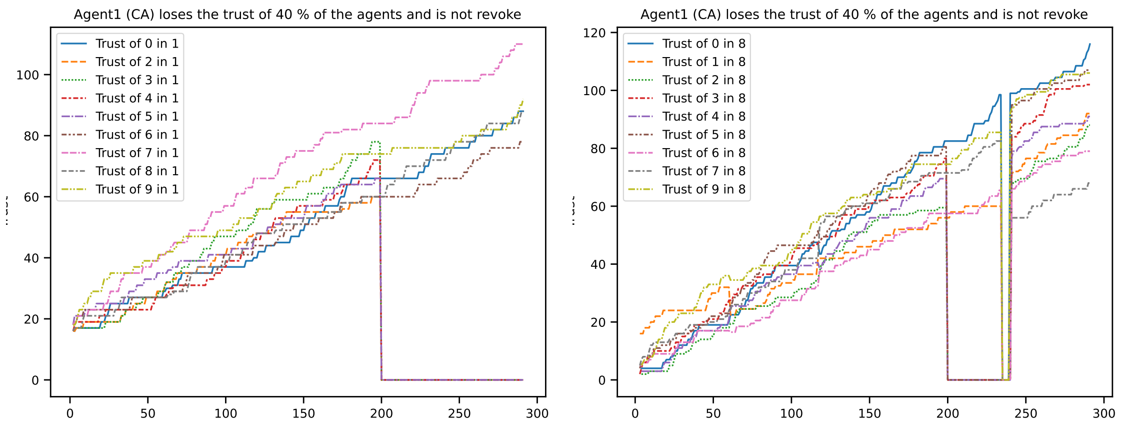
(c) Scenario 3. Trust in agent 0 drops for the agents that initially detected its malicious behavior. The trust of all the other agents remains the same as the revocation did not happen for not enough agents requested it.



(d) Scenario 4. Trust in agent 1 drops as all agents detect its malicious behavior.

Figure 3: Graphs showing the variations of trust in agent $\mathscr{E}$ as the other agents detect its malicious behavior

In the last scenario, N is only 40% of the agents, including agents certified by $\mathscr{E}$ and others that are not. Only the agents that detected the malicious behavior will lose their trust in $\mathscr{E}$ but, since the trust in an agent is weighted by the trust in its certifier they will also lose their trust in the agents certified by $\mathscr{E}$. Some of those agents will change certificates since they are part of the 40% and some will not since they have no reason to do so. The result is presented in Figures 4a and 4b. We can see in Figure 4b that the agent 8 is temporarily distrusted by the rest of the system. At first, it lose the trust of the agents not trusting agent 1 since since the its certificate was signed by agent 1, this



(a) Scenario 5. Trust in agent 1 drops for the agents that detected it malicious behavior. The trust of all the other agents remains the same as no revocation is possible since agent 1 is self-signed.

(b) Scenario 5. Trust in agent 8 drops when its CA falls out of grace. The trust lost is then regained when the agent changes its CA.

Figure 4: Graphs showing the variations of trust in agent $\mathcal{E}$ as the other agents detect its malicious behavior (continued)

includes the agent 4. Then, it loses the trust of the whole system as it certificate expires. Its certificate expires because it choose the agent 4 as a CA which was not replying. Eventually, agent 8 asks another CA while holding its expired certificate and regains the trust of the whole system.

Lastly, there is one pattern appearing in Figures 3 and 4 that surprised us. We can see the trust occasionally dropping, but not to 0, or increasing a lot at once while we did not implement any rules in our TMS leading to this behavior. This is due to a change of CA, the agent may choose a new CA based on its view of the system but chooses a CA that is less trusted by the other agents. On the other hand, an agent can choose a more trusted CA and increases its trustability quite quickly.

6. Conclusion

This paper presented a new public key infrastructure, coined MAKI, aiming at enabling public key encryption in open multi-agent systems of embedded agents with no third-party nor central server. We discussed several security concerns depending on the threat model and hypotheses on the use case of the multi-agent system. As with any cryptographic system, MAKI comes with additional cost in computing, energy, and storage but we tried to limit its cost while not impinging on the autonomy of the agents. A proof-of-concept was developed using the Mesa framework and used five scenarios to demonstrate how MAKI handles malicious behavior and certificate distribution. We are currently working on finalizing the proof-of-concept by adding indirect information to the TMS, describing more malicious behavior, coalition of malicious agents for example, and computing the energy cost of MAKI. Furthermore, we intend to provide formal security proof using a security protocol verification tool.

Acknowledgments

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, P. Elbaz-Vincent, Systematic Mapping Study of Security in Multi-Embedded-Agent Systems, IEEE Access 9 (2021) 154902–154913. doi:10.1109/ACCESS.2021.3128287.

[2] D. E. Boubiche, S. Athmani, S. Boubiche, H. Toral-Cruz, Cybersecurity issues in wireless sensor networks: Current challenges and solutions, Wireless Personal Communications 117 (2021) 177–213. doi:10.1007/s11277-020-07213-5.

[3] R. Ande, B. Adebisi, M. Hammoudeh, J. Saleem, Internet of things: Evolution and technologies from a security perspective, Sustainable Cities and Society 54 (2020) 101728. doi:10.1016/j.scs.2019.101728.

[4] R. H. Jhaveri, N. M. Patel, Attack-pattern discovery based enhanced trust model for secure routing in mobile ad-hoc networks, International Journal of Communication Systems 30 (2017) e3148. doi:10.1002/dac.3148.

[5] D. Kukreja, S. K. Dhurandher, B. V. R. Reddy, Power aware malicious nodes detection for securing manets against packet forwarding misbehavior attack, Journal of Ambient Intelligence and Humanized Computing 9 (2018) 941–956. doi:10.1007/s12652-017-0496-2.

[6] B. E. Sabir, M. Youssfi, O. Bouattane, H. Allali, Authentication and load balancing scheme based on JSON Token For Multi-Agent Systems, Procedia Computer Science 148 (2019) 562–570. doi:10.1016/j.procs.2019.01.029.

[7] A. Singla, E. Bertino, Blockchain-Based PKI Solutions for IoT, in: 2018 IEEE 4th International Conference on Collaboration and Internet Computing (CIC), 2018, pp. 9–15. doi:10.1109/CIC.2018.00-45.

[8] A. Yakubov, W. M. Shbair, A. Wallbom, D. Sanda, R. State, A Blockchain-Based PKI Management Framework, in: NOMS 2018 - 2018 IEEE/IFIP Network Operations and Management Symposium, 2018, pp. 1–6. doi:10.1109/NOMS.2018.8406325.

[9] B. Qin, J. Huang, Q. Wang, X. Luo, B. Liang, W. Shi, Cecoin: A decentralized PKI mitigating MitM attacks, Future Generation Computer Systems 107 (2020) 805–815. doi:10.1016/j.future.2017.08.025.

[10] F. Lesueur, L. Me, V. V. T. Tong, An efficient distributed PKI for structured P2P networks, in: 2009 IEEE Ninth International Conference on Peer-to-Peer Computing, 2009, pp. 1–10. doi:10.1109/P2P.2009.5284491.

[11] N. C. Goffee, S. H. Kim, S. Smith, P. Taylor, M. Zhao, J. Marchesini, Greenpass:

Decentralized, PKI-based Authorization for Wireless LANs, in: In 3rd Annual PKI Research and Development Workshop, 2004, pp. 26–41.

[12] S. Blanch-Torné, F. Cores, R. M. Chiral, Agent-based PKI for Distributed Control System, in: 2015 World Congress on Industrial Control Systems Security (WCICSS), 2015, pp. 28–35. doi:10.1109/WCICSS.2015.7420319.

[13] A. Avramidis, P. Kotzanikolaou, C. Douligeris, M. Burmester, Chord-PKI: A distributed trust infrastructure based on P2P networks, Computer Networks 56 (2012) 378–398. doi:10.1016/j.comnet.2011.09.015.

[14] X. Bonnaire, R. Cortés, F. Kordon, 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. doi:10.1109/TrustCom.2013.44.

[15] T. Okamoto, K. Takashima, Decentralized Attribute-Based Encryption and Signatures, IEICE Transactions on Fundamentals of Electronics, Communications and Computer Sciences E103.A (2020) 41–73. doi:10.1587/transfun.2019CIP0008.

[16] H. Cui, R. H. Deng, Revocable and Decentralized Attribute-Based Encryption, The Computer Journal 59 (2016) 1220–1235. doi:10.1093/comjnl/bxw007.

[17] J. R. Douceur, The Sybil Attack, in: Peer-to-Peer Systems, 2002, pp. 251–260. doi:10.1007/3-540-45748-8_24.

[18] S. Boeyen, S. Santesson, T. Polk, R. Housley, S. Farrell, D. Cooper, Internet X.509 Public Key Infrastructure Certificate and Certificate Revocation List (CRL) Profile, RFC 5280, RFC Editor, 2008. URL: https://www.rfc-editor.org/info/rfc5280. doi:10.17487/RFC5280.

[19] I. Pinyol, J. Sabater-Mir, Computational trust and reputation models for open multi-agent systems: a review, Artificial Intelligence Review 40 (2013) 1–25. doi:10.1007/s10462-011-9277-z.

[20] X. Fan, L. Liu, R. Zhang, Q. Jing, J. Bi, Decentralized Trust Management: Risk Analysis and Trust Aggregation, ACM Comput. Surv. 53 (2020). doi:10.1145/3362168.

[21] W. Widyawan, S. Sulistyo, V. Suryani, Trust-Based Privacy for Internet of Things, Institute of Advanced Engineering and Science 6 (2016) 2396–2402.

[22] S. Liu, A. C. Kot, C. Miao, Y.-L. Theng, A Dempster-Shafer Theory Based Witness Trustworthiness Model to Cope with Unfair Ratings in e-Marketplace, in: Proceedings of the 14th Annual International Conference on Electronic Commerce, ICEC '12, 2012, pp. 99–106. doi:10.1145/2346536.2346555.

[23] J. Sabater, EVALUATING THE ReGreT SYSTEM, Applied Artificial Intelligence 18 (2004) 797–813. doi:10.1080/08839510490509027.

[24] Anonymous, Code and data presented in C&ESAR 2022, 2022. URL: https://doi.org/10.5281/zenodo.7079281. doi:10.5281/zenodo.7079281.

[25] E. Barker, Q. Dang, Recommendation for Key Management Part 3: Application-Specific Key Management Guidance, 2015. doi:10.6028/NIST.SP.800-57pt3r1.

[26] J. Kazil, D. Masad, A. Crooks, Utilizing Python for Agent-Based Modeling: The Mesa Framework, in: Social, Cultural, and Behavioral Modeling, volume 12268, 2020, pp. 308–317. doi:10.1007/978-3-030-61255-9_30.