MAKI: A Multi-Agent Public Key Infrastructure

Arthur Baudet[1,2], 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*

Keywords: Public Key Infrastructure, Multi-Agent Systems, Embedded Agent, Decentralized Security.

Abstract: This paper presents a public key infrastructure for open multi-agent systems of embedded agents. These systems rely on agents autonomy and cooperation between heterogeneous systems to achieve common goals. They are very prone to attacks as they are confronted with unknown systems with unknown goals. 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. The result of our approach is the Multi-Agent Key Infrastructure, a public key infrastructure leveraging and empowering a trust management system. Agents autonomously maintain a set of trusted certification authorities that deliver certificates to trusted agents and revoke intruders. This infrastructure 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 only provide recommendations related to the cryptographic primitives to use and the inner details of the trust management system.

1 INTRODUCTION

Multi-Agent Systems (MAS) bring a decentralized methodology to software and system architectures. As a system architecture, they are used 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., a class of systems that allow agents with different capabilities to enter in and leave the system as they need as long as they are running.

Recent meta-studies (Baudet et al., 2021; Boubiche et al., 2021) show that MEAS and similar systems are particularly vulnerable to insider attacks, a kind of attacks that come 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 (Jhaveri and Patel, 2017; Kukreja et al., 2018). For example, those hypotheses may require the presence of third-party to provide a root of trust or to pre-load 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 here.

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:

- The cryptographic primitives that are used, the hardware they run on, and their implementations are secured.
- A suitable TMS is executing and takes into account likely attacks on it.
- Sybil attacks are mitigated using either the TMS or by other means.

Baudet, A., Aktouf, O., Mercier, A. and Elbaz-Vincent, P.
MAKI: A Multi-Agent Public Key Infrastructure.
DOI: 10.5220/0011631800003393
In *Proceedings of the 15th International Conference on Agents and Artificial Intelligence (ICAART 2023) - Volume 3*, pages 177-184
ISBN: 978-989-758-623-1; ISSN: 2184-433X

- 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.

Related Work. The works presented in (Lesueur et al., 2009) and (Goffee et al., 2004) rely on respectively threshold cryptography and Simple Distributed Security Infrastructure to provide a decentralized Public Key Infrastructure (PKI). However, both solutions still require either off-band verification or pre-loaded certificates to provide authentication. In (Blanch-Torné et al., 2015), 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. This conflicts with the openness characteristic of the systems studied in this paper. Both the works in (Avramidis et al., 2012; Bonnaire et al., 2013) 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 filter out untrustworthy nodes. Overall, all the approaches above provide a way of decentralizing or distributing a PKI but rely too much on third parties or external control to apply to our problem.

For several years, most of the efforts toward designing a decentralized PKI have involved Blockchain Technology (BCT) (Singla and Bertino, 2018; Yakubov et al., 2018; Qin et al., 2020). The BCT is designed to provide a consensus on information in decentralized systems where no trust preexists, 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.

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 (Okamoto and Takashima, 2020; Cui and Deng, 2016). 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), an 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.

We introduce MAKI in Section 2 and discuss the result of a proof-of-concept of it in Section 3. Finally, we conclude the paper in Section 4.

2 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 linking an agent identity and leveraging the TMS to make inefficient to change identity (see Section 3 for an example). Then we empower the TMS by allowing the certificate to be revoked, thus its holder instead of just ignoring it.

2.1 Architecture

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 linked to the keys it uses to interact.

We designed MAKI as a lightweight decentralized Public Key Infrastructure, such as the X.509 PKI. From the PKI principles, we only kept the Certificate Authority (CA) function described below, the mandatory use of certificates, and the certificate revocation.

To enforce the use of cryptography, unsigned messages as well as messages with invalid signature, should be ignored. Moreover, agents should make sure that the source of a message they received holds a valid certificate, signed by a CA, for the key used to sign the message before processing it. Of course, requests to find CAs or certification requests can be signed with not-certified keys. These certificates are delivered and revoked by CAs elected through the use of a self-organization scheme, thus capitalizing on the autonomy of the agents. An agent can freely request a certificate and the CA autonomously to answer it or not. They 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. So, to mitigate this risk, we also use short-lived certificates, which will not be renewed by a CA that is aware of the revocation.

2.2 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 are all explained in Section 2.3.

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 trustworthy 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. 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 of CAs with one or more (depending on $\mathtt{T}$, the probability that an agent decides to become a CA) CAs by group 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 $\mathtt{T}$ to increase or lower the number of redundant CAs. If $\mathtt{T}$ is very high, almost all agents that can be CA will be, but if $\mathtt{T}$ is very low, only agents far from a CA will become one. Both the definition of close and the value of $\mathtt{T}$ should be tailored to the application, density and capabilities of the application MAKI runs on.

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 ← BROADCAST(CAListingRequest)
TrustedCAs ← FILTER(CAs, TrustLevel.Moderate)
if can become CA and (TrustedCAs is empty or RANDOM(0, 1) < T) then
    Role ← CA
```

Choosing a CA for a None agent is similar to deciding to become a CA. Singe-point-of-failure situations can arise if the agents choose the most trustworthy CA to deliver their certificates and that one CA is designated the most trustworthy by most of them. To avoid such cases, an agent will choose one from the ones it trusts the most and specially the most trusted one. This translates into a weighted random choice using the trust values as weights. This way of choosing ensures that most of the time a highly trusted CA will be chosen but with a non-zero probability that a less trusted CA will also have the possibility to show that it can be trusted.

Adding a new agent in MAKI is straightforward. The agent will first determine if it needs to become a CA and, if not, will request a certificate from a CA. Once done, it may first 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 the certificate to ensure that its neighbors learn about it.

An agent can only be revoked by the CA that signed its certificate. However, self-signed CA 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 remove this advantage is to propose 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 harder to obtain and maintain a valid certificate, but can be considered as a demonstation of good faith and be rewarded in the TMS.

Overall, MAKI does not reduce the security brought by the TMS since, even though they can not

Table 1: Trade-off 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.

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 3 how, with a simple TMS, MAKI handles malicious agents once the TMS detects their malicious behaviors.

2.3 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.

We present in Table 1 a risk assessement 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 to check or prove 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 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-

```
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,
  revokedCertificates 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.

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

certified and take the risk of having their certificates revoked if they behave maliciously.

2.4 Certificate Management

Certificate and CRL representations can be found in Figure. 1. Excluding the fields we did not keep from the X.509 format, 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 byte for the public key OpenSSH format, 105-byte 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 distributions 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.

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 code 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.

3 PROOF-OF-CONCEPT

3.1 Setup

We developed a proof-of-concept (PoC) using the Yet Another Multi-Agent System Simulator (YAMASS), an in-house multi-agent simulator based on the Mesa framework (Kazil et al., 2020). The code used for this PoC and the results we obtained are available at (Baudet et al., 2022) to allow complete replicability of the results and transparency on the implementation choices. YAMASS allowed us to implement the agent behavior described above without constraints while making sure that agents can not share states or information outside of sent and received messages.

We followed the NIST recommendations (Barker

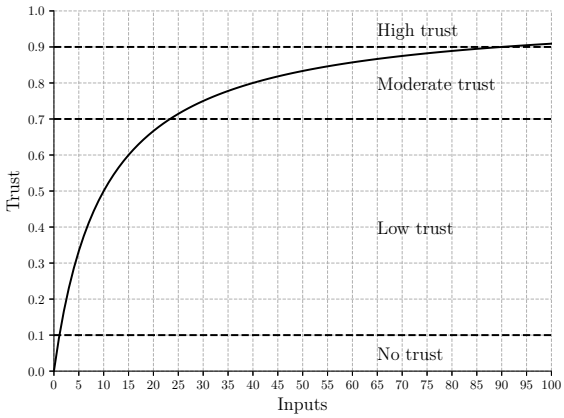


Figure 2: Representation of the trust growth and trust thresholds in MAKI.

and Dang, 2015) for our choice of cryptographic primitives. Thus, we used Elliptical Curve Cryptography with the Elliptic Curve Digital Signature Algorithm and 256-bit keys using the P-256 Curve.

We designed a trust model with a slow growth and a low initial trust, the common mitigation to whitewashing presented in (Ruan and Durresi, 2016). The model was tailored to the number of agents and 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 experimentally 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 0.3, 0.7 and 0.9 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 this trust growth function, with the thresholds, is given in Figure 2.

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, every agent eavesdrops on the communications to update in real-time their trust model, in particular concerning the certificate deliveries.

This PoC focuses on the behavior at a local scale, the way agents are meant to be designed for. We define this local scale by an area in which all agents are in range of each other. This allowed us to deploy a very minimalistic routing protocol in which all messages are broadcasted and no acknowledgements are expected. 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.

Lastly, agents' trust increases randomly over time to emulate the successful interactions among them and feed the TMS.

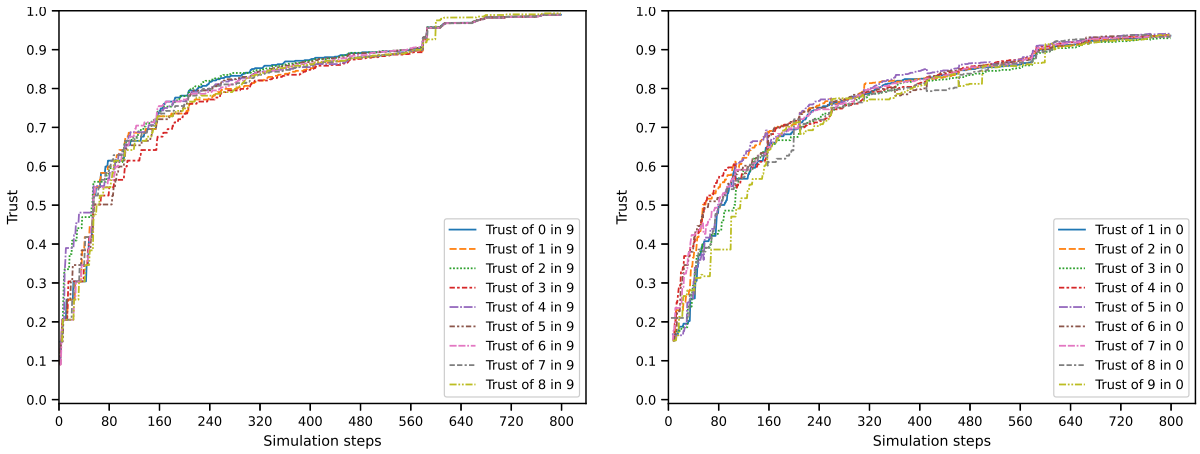
3.2 Experimentations

We show two results from the experiments we realized using the PoC. Both of them were done using a configuration of ten agents. The agents are numbered from 0 to 9 and only the four agents 6 to 9 have the resources to become CA. Agents that initially become CA during our experiments are agents 6, 8, and 9 (fixed by the seed used for the agents random number generator).

Figure 3 presents the normal fluctuation of trust in an agent. Overall, the trust increases steadily in both cases and fluctuates as agents change CA as long as no CA really stands out. By the 600^{th} step, the CAs have enough trust in each other to request cross-certification and see their trust increase accordingly. Based on this execution, we designed two scenarios to show how MAKI leverages and empowers TMS.

In the first scenario, we aim to compare the use of CRL against the use of indirect information to exclude an agent. To do so, we established a situation where the trusts of agents 1 to 6 in agent 0 drop below the *Low* threshold at step 700 and expect that the agents 7 to 9 also distrust agent 0 (the choice of the agents does not matter as long as the malicious agent in not a CA and that the revocation requesters represent the majority.) From this situation, we executed MAKI with and without certificate revocations. The results in trust fluctuations are shown in Figures 4. For conciseness, we do not present the graph of the trust fluctuations with certificate revocations and no indirect information as it is the same as the one with indirect information (it is available in the repository (Baudet et al., 2022)). From these figures we can see that the revocation leads to all the agents distrusting agent 0. However, the trust decreases only a little but the indirect information is not enough to really change the trust value of agents 7 to 9. The last result may be improved with a TMS giving more weight to indirect information. However, direct information, such as correct interactions or revocation will often be more important than indirect information to prevent agents being swayed too easily by their neighbors.

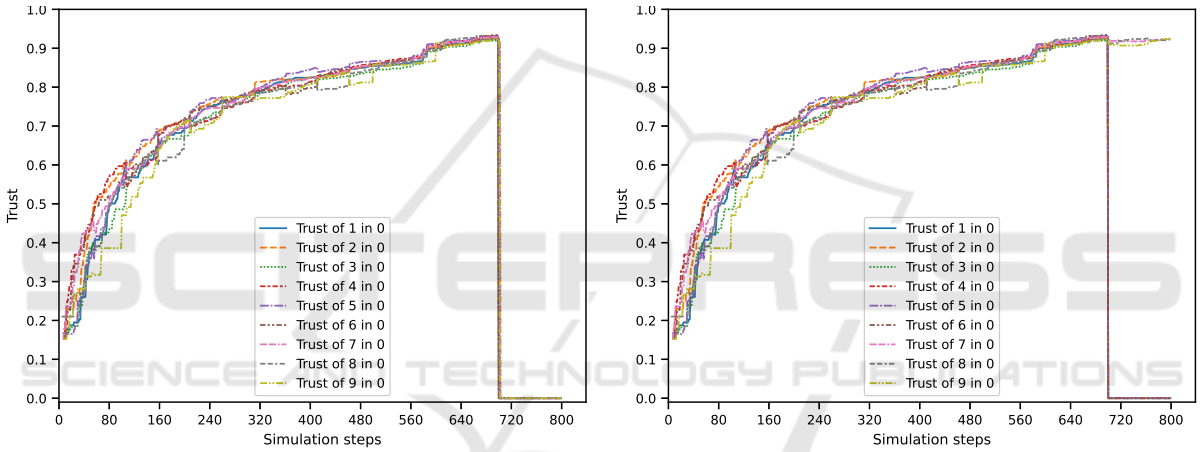
In the second scenario, we show how MAKI leverages the TMS to adapt to a coalition of malicious CAs. To do so, from the four agents with the capabilities to become CAs, the three which decided to become CA, agents 6, 8, and 9 are detected malicious at step 700. Then, we expect the system to adapt accordingly by stopping to rely on them and find another



(a) Trust fluctuations of a CA agent over time.

(b) Trust fluctuations of a None agent over time.

Figure 3: Trust fluctuation of two agents, one CA one None, over time.



(a) Trust fluctuations with the certificate of agent 0 being revoked.

(b) Trust fluctuations with the certificate of agent 0 not being revoked.

Figure 4: Graphs showing the fluctuations of trust in agent 0 as the other agents detect its malicious behavior.

```
1098 INFO:main:step:701
1099 DEBUG:agent:7:net:<>:Data(src: Identity(7, <public_key>), dest: -1, payload:
       CertAdvert(Certificate(serial_number: 0, issuer: Identity(7, <public_key>)
       , subject: Identity(7, <public_key>), subject_role: Role.CA, not_before:
       702, not_after: 4702, version: 1)))
1100 DEBUG:agent:0:net:=>:Data(src: Identity(0, <public_key>), dest: Identity(7, <
       public_key>), payload: CertReq())
1101 DEBUG:agent:1:net:=>:Data(src: Identity(1, <public_key>), dest: Identity(7, <
       public_key>), payload: CertReq())
1102 DEBUG:agent:3:net:=>:Data(src: Identity(3, <public_key>), dest: Identity(7, <
       public_key>), payload: CertReq())
1103 DEBUG:agent:2:net:=>:Data(src: Identity(2, <public_key>), dest: Identity(7, <
       public_key>), payload: CertReq())
1104 DEBUG:agent:4:net:=>:Data(src: Identity(4, <public_key>), dest: Identity(7, <
       public_key>), payload: CertReq())
1105 DEBUG:agent:5:net:=>:Data(src: Identity(5, <public_key>), dest: Identity(7, <
       public_key>), payload: CertReq())
```

Figure 5: 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.

CA. An excerpt of the execution trace at step 701 in shown in Figure 5. It shows that agent 7 changes its role, broadcasts its new certificate and agents 0 to 6, which lost their trust in the other CAs, request a certificate from agent 7. The self-organization algorithm led to the promotion of a friendly CA and the malicious CAs being ignored.

4 CONCLUSION

In this paper, we introduced a decentralized public key infrastructure, named 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 revoked thanks to the use of certificates delivered and revoked by a subset of trusted certification authorities agents maintained with no third parties involved. A proof-of-concept of MAKI is also presented, with the source available to reviews, to demonstrate how revocations are obtained once one or more intruders are detected. We are now focusing our efforts on further validating MAKI using formal methods such as model checking. We are also exploring a blockhain-based solution to provide a way to easily share certificates and allow agents to better audit certification authorities.

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

Avramidis, A., Kotzanikolaou, P., Douligeris, C., and Burmester, M. (2012). Chord-PKI: A distributed trust infrastructure based on P2P networks. *Computer Networks*, 56(1):378–398.

Barker, E. and Dang, Q. (2015). Recommendation for Key Management Part 3: Application-Specific Key Management Guidance.

Baudet, A., Aktouf, O.-E.-K., Mercier, A., and Elbaz-Vincent, P. (2021). Systematic Mapping Study of Security in Multi-Embedded-Agent Systems. *IEEE Access*, 9:154902–154913.

Baudet, A., Aktouf, O.-E.-K., Mercier, A., and Elbaz-Vincent, P. (2022). Code and data presented in ICAART. https://zenodo.org/record/7180985.

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

Bonnaire, X., Cortés, R., Kordon, F., and Marin, O. (2013). A Scalable Architecture for Highly Reliable Certification. In *2013 12th IEEE International Conference on Trust, Security and Privacy in Computing and Communications*, pages 328–335.

Boubiche, D. E., Athmani, S., Boubiche, S., and Toral-Cruz, H. (2021). Cybersecurity Issues in Wireless Sensor Networks: Current Challenges and Solutions. *Wireless Personal Communications*, 117(1):177–213.

Cui, H. and Deng, R. H. (2016). Revocable and Decentralized Attribute-Based Encryption. *The Computer Journal*, 59(8):1220–1235.

Goffee, N. C., Kim, S. H., Smith, S., Taylor, P., Zhao, M., and Marchesini, J. (2004). Greenpass: Decentralized, PKI-based Authorization for Wireless LANs. In *In 3rd Annual PKI Research and Development Workshop*, pages 26–41.

Jhaveri, R. H. and Patel, N. M. (2017). Attack-pattern discovery based enhanced trust model for secure routing in mobile ad-hoc networks. *International Journal of Communication Systems*, 30(7):e3148.

Kazil, J., Masad, D., and Crooks, A. (2020). Utilizing Python for Agent-Based Modeling: The Mesa Framework. In *Social, Cultural, and Behavioral Modeling*, volume 12268, pages 308–317.

Kukreja, D., Dhurandher, S. K., and Reddy, B. V. R. (2018). Power aware malicious nodes detection for securing MANETs against packet forwarding misbehavior attack. *Journal of Ambient Intelligence and Humanized Computing*, 9(4):941–956.

Lesueur, F., Me, L., and Tong, V. V. T. (2009). An efficient distributed PKI for structured P2P networks. In *2009 IEEE Ninth International Conference on Peer-to-Peer Computing*, pages 1–10.

Okamoto, T. and Takashima, K. (2020). Decentralized Attribute-Based Encryption and Signatures. *IEICE Transactions on Fundamentals of Electronics, Communications and Computer Sciences*, E103.A(1):41–73.

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

Ruan, Y. and Durresi, A. (2016). A survey of trust management systems for online social communities – Trust modeling, trust inference and attacks. *Knowledge-Based Systems*, 106:150–163.

Singla, A. and Bertino, E. (2018). Blockchain-Based PKI Solutions for IoT. In *2018 IEEE 4th International Conference on Collaboration and Internet Computing (CIC)*, pages 9–15.

Yakubov, A., Shbair, W. M., Wallbom, A., Sanda, D., and State, R. (2018). A Blockchain-Based PKI Management Framework. In *NOMS 2018 - 2018 IEEE/IFIP Network Operations and Management Symposium*, pages 1–6.