Trusted Registration, Negotiation, and Service Evaluation in Multi-Agent Systems throughout the Blockchain Technology

Davide Calvaresi*†, Alevtina Dubovitskaya†§, Diego Retaggi‡, Aldo F. Dragoni‡ and Michael Schumacher§
*Scuola Superiore Sant'Anna, Pisa, Italy
†University of Applied Sciences and arts Western Switzerland, Sierre, Switzerland
‡Università Politecnica delle Marche, Ancona, Italy
§EPFL, Lausanne, Switzerland

***Abstract*—Some recent trends in distributed intelligent systems rely extensively on agent-based approaches. The so-called Multi-Agent Systems (MAS) are taking over the management of sensitive data on behalf of their producers and users (e.g., medical records, financial investment, energy market). Therefore, trusted interactions are needed more than ever, while accountability and transparency among the agents seem crucial characteristics to be achieved. To do so, recent trends advocate the use of blockchain technologies (BCT) in MAS. The blockchain is a distributed ledger technology that can execute programmable transaction logic, and provides a shared, immutable, and transparent append-only register of all the actions happening in the network.**

Although a few theoretical approaches have already been proposed, the quest for such a system consolidating BCT and MAS to guarantee privacy, scalability, transparency, and efficiency continues. This paper presents a reconciling system including BCT within the dynamics of a MAS. Such a system aims at *(i)* building a solid ground for trusted interactions and *(ii)* enabling more characterizing feature-based and trustworthy ways of computing agent reputation. The system has been tested in four scenarios with different configurations (regular executions and involving down-agents or malicious behaviors).

Finally, the paper summarizes and discusses the experience gained, argues about the strategic choice of binding MAS and BCT, and presents some future challenges.

I. Introduction

Nowadays many large and distributed information systems deal with sensitive data so that skepticism about data privacy, security, and integrity rised from both scientific and political circles. Recently some of these distributed systems have been implemented adopting agent-based approaches, namely "Multi-Agent Systems" (MAS), in different sensitive domains (e.g., e-health [1], [2], assisted living [3], tele-rehabilitation [4], manufacturing [5], and e-Commerce).

MAS dynamics and models emulate human social systems' behaviour, hence, when such systems are enabled to manage autonomously sensitive data the accountability and trust of the interactions and the agents themselves are mandatory factors. Over the years, a considerable number of technical and scientific challenges have been faced. Although many remarkable efforts to develop models and mechanisms guaranteeing secure communications and trust in MAS have been provided [6], [7], [8], a viable and sustainable solution meeting all the requirements has not been identified yet.

Blockchain is a peer-to-peer distributed ledger technology that provides a shared, immutable and transparent history of all the transactions in a network. This technology allows to execute arbitrary, programmable transaction logic in the form of smart contracts[1] to build applications with trust, accountability and transparency. It is secured using cryptographic primitives such as hash function, digital signature, and encryption [9].

This creates a unique opportunity to enhance transparency and trust for the reputation management within MAS (in particular if handled by a centralized entity). In *open* MAS, where the intentions of the agents can be unknown and driven by different interests, agents may exhibit malicious behavior.

The promising idea to address the aforementioned challenges, by integrating MAS with blockchain technologies (BCT) [10], [11], is gathering theoretical contributions from many fields. Yet a practical, efficient, scalable, and secure implementation evaluated in a real-world settings is still in demand [12].

Contributions:
This paper presents a first MAS tightly coupled with BCT, implemented and tested. Therefore, the contribution is three-fold:

- *(i)* a study about how and why integrating BCT and MAS.
- *(ii)* an implementation of a reconciling solution merging a Jade-based [13] MAS with Hyperledger Fabric v1.0 [14].
- *(iii)* a study and a development of dynamics enabling the computation of the agent reputation throughout smart contracts.

The paper is organized as follows: Section II present the current state of the art, Section III presents the objectives, design, and implementation of the developed system, Section IV presents evaluation scenarios, Section V discusses the obtained results. Finally, Section VI concludes the paper and presents ongoing and future works.

II. State of the Art

In MAS, decision-making, trading, and general interactions are often delegated to a broad set of cost-functions and a feature usually regarded as a discriminant is the "reliance". It can be computed directly or delegated to a specialized entity. Such an attribute can characterize a given entity and its behaviors [15].

[1]"smart contract" and chaincode logic concepts are quite similar. Therefore, we use the former when talking about chaincode, "programmable contrac" or "a set of rules" when discussing blockchain technology in general.

Reputation can be defined as *the collection of opinions received from other agents* [15]. Generally, it is used to frame the perception and expectation about someone's behavior based on previous interactions (presumably similar to future ones). On average, reputation can be considered as quite a reliable indicator, a trust building-block. Such a mechanism is an incentive for good behaviors. Hence, instilling positive effects on market quality is a constructive side-effect emerging in reputation-based systems.

Since the beginnings, reputation and trust mechanisms have been considered key elements in the design of MAS, in order to revise conflicting beliefs and choose among opinions received from different information sources [16], [17]. Recently a line that focuses on building the models to compute reputation appeared [18]. While in some models, the question about how reputation information is aggregated is not taken into account [19], other models can be classified depending on the information sources they use to infer trust or reputation, such as direct experience and witnesses information, to name a few [18]. Authenticity and integrity of this information, as well as transparency and correctness of the reputation calculation, are crucial requirements for building a reliable reputation mechanism.

Three approaches to manage reputation are *security-based, institutional*, and *social*. Security-based approaches focuses only on insuring integrity an authenticity of the data by means of cryptography. Institutional approaches assume a central authority that observes, controls, and enforces agents' actions, but they are vulnerable and prone to create single points of failure; if compromised, it can tamper the data and manipulate the computation of the reputation. Social approaches require an agent to be capable of modelling other agent behaviors, and to follow the similitude within human societies, which is not always applicable. Even combined, they cannot guarantee the requirements listed above.

"Blockchain" is a distributed technology that employs cryptographic primitives, and relies on a specific membership mechanism and consensus protocol [20] to maintain a shared, immutable, and transparent append-only register [9], [21]. Data, in the form of digitally signed transactions broadcasted by the participants, are grouped into blocks chronologically timestamped. A hash function is applied to the content of the block and forms a unique block identifier, which is stored in the subsequent block. A possible modification of the block content can be easily verified by hashing it again, and comparing it with the identifier from the subsequent block. The blockchain is replicated and maintained by every participant. A malicious attempt to tamper the information stored in the registry will be noticed by the participants, thus guaranteeing immutability of the ledger [12]. Many blockchains can execute arbitrary tasks, typically called smart contracts, thus allowing to implement desired functionality on top of this technology.

We can distinguish between *permissionless* and *permissioned* (*public* and *private*) blockchain systems. A system is permissionless when the identities of participants are either pseudonymous or anonymous [22], so that every user may participate in the consensus protocol, and, therefore, append a new block to the ledger. In contrast, in a permissioned blockchain identities of the users and rights to participate in the consensus (writing to the ledger and/or validating the transactions) are controlled by a membership service. A permissioned blockchain is *public* when anyone can read the ledger but only predefined set of users can participate in the consensus, and *private* when even the right to read ledger is controlled by the membership/identity service.

We aim at studying and experimenting all the advantages in the synergy of MAS and BCT. The need for enhancing the security in MAS (features such as integrity, identity management, provenance, transaction guarantees, and data security) affects applications in various domains. BCT can be the key to enhance MAS security and some attempts of merging these two technologies have already been done [12]. For example, Ferrer [23] discusses how a group of agents can reach an agreement on a particular state of affairs, and record that agreement without a controlling authority by combining peer-to-peer networks (e.g., communities of agents) with cryptographic algorithms. The author discusses potential advantages and limitations of the combination of blockchain technology and MAS, taking an example of swarm robotic systems.

Bottone et al. [24] presented a mathematical structure for a block-less, fee-less, distributed ledger technology employable in wireless-sensors networks, Internet of Things, and Cyber-Physical Systems that aims at overcoming the limitations implicated by the use of conventional blockchains. The confirmation time is reduced, avoiding the need for special computation. They mapped the nodes of the system (also associated with agents) on elements of a directed acyclic graph.

However, if the underlying infrastructure is blockchain-enabled, the auditability of information increases noticeably [25]. This is guaranteed by the BCT, which secures its repositories (ledgers) and the data posted on them [26]. In agent-based audit systems, an agent auditing specific transaction-related documentation interacts with the agents simultaneously auditing related documentation, and thus ensures the accuracy and comparability of the entire documentation under audit. Automatic verification, processing, storing, and reporting of the information in the blockchain-based triple-entry accounting information system could together form a self-sufficient accounting ecosystem [27]. MAS in an audit ecosystem can simultaneously confirm the accuracy of the blockchain-based verification, processing, storing, and reporting functions [25].

Both in research and commercial applications, reputation systems are gaining a stable share [28]. However, such a rapid growth resulted in generating diverging approaches implementing a broad set of solutions.

Following the recent trends mentioned above, a few solutions associated the reputation and trust in the framework of MAS and BCT. For example, Khaqqi et al. [29] proposed an Emission Trading Scheme with a double aim: to reduce emission production and stimulate adoption of long-term abatement technology. Such objectives have been pursued by uses of BCT and smart devices so as to improve the functionality, consistency, and credibility of the scheme. Their implementation gives more leverage to the participants adopting long-term solutions to determine the market price of the permits. Since these participants have more incentives to set a higher permit price to compensate their emission reduction strategy, the global price of a permit may increase even without a supply shortage or a price floor. A consequence of the reputation system is the financial incentive. Participants with

a better reputation are given the opportunity to choose a better trade offer and to conclude the trade faster. Two reputation-based mechanisms were added, facilitating these incentives: the market segmentation mechanism and the priority-value-order mechanism. The former is a filter based on reputation. In the case study, we observed two types of participation in the trading process: active and passive. Active buyers and sellers are benefited by the market segmentation mechanism while passive sellers benefit from the priority value order. Passive buyers, however, receive no benefit or restriction as a consequence of their reputation. Thus, it can be implemented more easily and quickly than other proposals that ask for complete alteration of the scheme. The evaluation using multi-criteria analysis shows that this scheme is feasible with its benefits outweighing its drawbacks.

Qayumi et al. [30] proposed the introduction of BCT in agentified computing systems dealing with potentially "very-large datasets". The multi-level principles of typical of crypto-currencies have also been employed in distributed master-less systems operating with reputation-based dynamics [31].

Although the solutions mentioned above positively testify the synergy of MAS and BCT, the lack of reliable, effective, and scalable practical implementations leave the following question still open: At which level and how can the blockchain technology be adequately integrated within a MAS?

III. BCT FOR MAS

This section presents the objectives specifically set to shape the design and implementation of the presented system.

The atomic cells composing a MAS can be rationalized as an autonomous entity characterized by an expendable knowledge, driven by self-developed or induced objectives able to interact with each other [32]. Such loosely coupled entities (agents) are interconnected and organized in networks. Regardless distribution, dimensions, and nature (cooperative or competitive) of the agent community allows the possible generation and evolution of undesired behaviors, a matter of concern in the scientific community. *Trust* and *reliability* heavily affect the MAS pillars[2] [34].

A. Objectives

Merging BCT and MAS requires structural and functional studies. Therefore, concerning the design and implementation of the presented system, we set the following objectives:

(i) to identify which functionality in MAS need to be replaced, improved, or extended;
(ii) to establish where in the existing agent-framework, the BCT should be integrated;
(iii) to test and evaluate the effects of such interventions, discussing pros and cons.

B. Design

As we require authenticity of the data and assume that an agent may need to manage some sensitive information, in the model of our system, the agents can be conceptually associated to the peers of the permissioned private blockchain.

Designing this solution, we used Jade as the multi-agent framework; such a choice is due to its compliance with the FIPA[3] standard, and to its implementation of crucial elements such as the Directory Facilitator (DF) and the Agent Management System (AMS) [13]. Furthermore, we chose Fabric Hyperledger [14] as a permissioned BCT component due to its maturity (development and documentation) and its open-source nature.

Certificates and Identity Management

In the dynamics of the agent community, the two above-mentioned technologies get firstly connected in the *management of the identities of the agents*. In Jade, the AMS is responsible for the registration of the agents identity management. Traditionally, after a free or feed registration, an agent can request and/or provide the services, unless he demonstrates malicious behavior. This can affect his reputation, that in turn can become a reason of expelling the agent from the MAS. To ensure trust and authenticity in MAS, public key infrastructure (PKI) can be employed [18]. In this setting, it is assumed that every agent generates private and public keys, and that there exists a certification authority (CA) that can verify the identity of an entity and create a certificate of the entity's public key. Using the key pair and the certificate, the agent can be authenticated, and all the actions of the agent can be tracked.

An important component of the permissioned BCT is a membership service, (MS). MS is an entity that hosts a certification authority and manages the network identities of all the peers, maintains an access control list (ACL) based control over network activity, and guarantees that every transaction is ultimately traceable to a registered user. In the current design, we use a single certification authority. However, alternative implementations are envisaged for the next releases of Hyperledger Fabric, such as support of anonymous credentials with multiple certification authorities and the use of threshold signatures.

As we assume that an agent can "run" as BCT peer, for simplicity, we rely on one common certification authority that is hosted by the MS. A *CA-Agent* is an agent that encompasses the functionality of a standard AMS and connects BCT and MAS, by providing an interface for the other agents to interact with CA for enrollment and identity management.

The service ledger: an enhanced and distributed DF

Jade adopts the concept of DF from the FIPA standard[3] for agent management. DF is usually represented by one agent per community, but can also be represented by more than one agent. It is in charge of keeping track of which agent offers which service(s) and providing such an information to who may ask for it.

Although in our proposed design the agent DF is replaced by the BCT, the concept has been kept and enhanced. Indeed, the DF takes the form of a distributed ledger, so-called Service Ledger (SL). This design choice allows to:

- remove the possibility of a single point of failure (if unique in the community),
- provide a solution for the harmonization of the current state of DF (if DF consists of multiple agents)

[2] agent local scheduler, communication, and negotiation protocols [33]

[3] http://www.fipa.org/specs/fipa00023/

- reduce the response time when inquired by regular agents,
- ensure immutability and traceability of the information.

Therefore, features such as tracking the evolution of the services offered by given agent(s) over the time can be easily introduced.

Storing the Interactions and Computing the Reputation
As mentioned in Section II, the concept of reputation can be associated to an agent. Moreover, in our solution we can assume:

- an *overall value* - rating the general (average) reputation of an agent
- a *specific value* - associated to given services, both as provider or demander.

Recalling that the reputation is computed on previous behaviors, we assume that within an interaction both the agents (demander and executor) must be able to evaluate the output of the interaction. Then, based on the evaluation values provided by both demander and executor for this interaction, and their current reputation values, smart contract (incorporating the mechanism for computing the reputation), is executed by all the blockchain peers to update reputation values. In this way, the reputation of the agent is based on the immutable history of all the other previously provided evaluation values for this agent as a provider of the service, and as a requester and, therefore, an evaluator of the results of the previous interactions, the agent was involved in.

Differently from traditional MAS, where *assumed* trustworthy agents compute the reputation with *assumed* uniform and unbiased techniques, in the presented solution, reputation management is done using smart contracts. In such a way, the properties of BCT (i.e., data transparency, immutability, integrity) can be transparently transferred to the MAS dynamics.

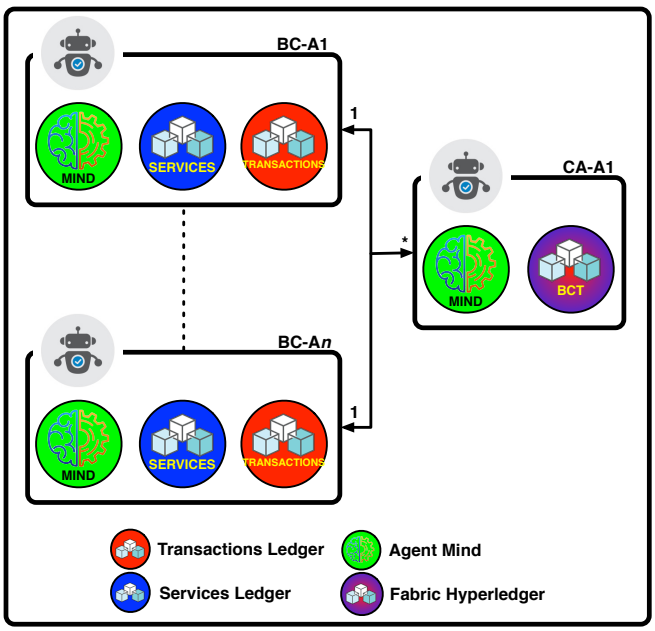

Fig. 1. Conceptual design of the system components.

Figure 1 outlines the system design and its main components. In particular, it is possible to distinguish two main *types* of agents characterized by both MAS and BCT components.

A regular agent is characterized by a mind, performing the classic agent reasoning, an agent can be coupled with a peer of a Hyperledger Fabric network. Every peer ($BC - A_i$) maintains two independent ledgers:

(i) a Service Ledger (SL), storing the information about the service(s) that an agent provides in a form of a tuple: {*agent; service(s); additional info*};
(ii) a Transactions Ledger (TL), storing the information about the interactions that took place in the community and the related evaluation from both the service provider (executor) and the agent that requested a service (demander).

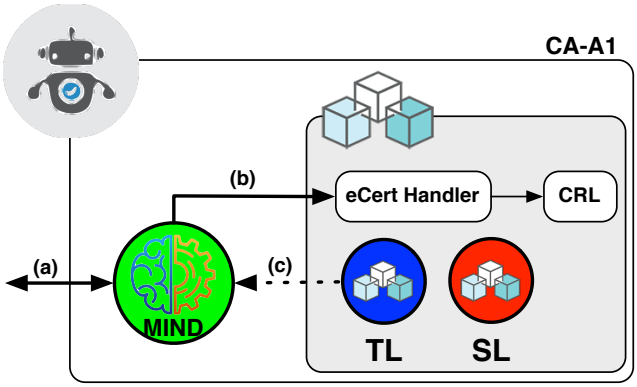

Fig. 2. Agent registration and certificate request.

The Certification Authority agent (CA-A1) is characterized by

(i) a mind, performing the classic agent reasoning;
(ii) an interface to interact with the BCT component.

C. Implementation

The CA-Agent
Th CA-Agent (CA-A), Figure 2, is in charge of interacting with the Fabric-MS server. Its behaviors are:

- forwarding messages from the agents to MS and distributing the certificates to interact with the BC (over a valid request) Figure 2(a);
- connecting to the Fabric network, Figure 2(b);
- triggering the revocation of a certificate (e.g., if a BC-Agent demonstrates malicious behavior and his reputation goes below the certain threshold), Figure 2(c).

As already mentioned in Section III-B, the CA-Agent is composed of a *mind* handling its agent dynamics and a node-js-based connector to interact with the BC network.

BC-Agents
The BC-Agents (BC-A$_i$) are regular agents populating the community and interacting with each other. Their behaviors are to:

- require a certificate to the CA-Agent;
- require the execution of a given service;
- read and write on the ledgers: SL and TL;

- load scenario (from an XML setting file[4]);
- write/receive messages.

As shown in Figure 1, two ledgers, SL and TL were used. To implement such ledgers (key-value-based structure), it has been chosen to employ the LevelDB[5]. The alternative would have been CouchDB, which unfortunately in the current version (v1.1.0-preview) presents consistent bugs.

Service Ledger

The SL contains the list of the agent associations - services offered by the agent. To represent such an association, the agent is represented as shown in Listing 1. Every service that

```
type Agent struct {
  IDA string 'json:"ida"'
  Address string 'json:"address"'
  Services []MyService 'json:"services"'
}
```

Listing 1: Agent representation in LS.

is offered by the agent has the structure shown in Listing 2, where weight is a generic parameter, that, depending on the scenario, can represent a cost or a time required for the service execution. The operations allowed on the SL are: *(i)*

```
type MyService struct {
   Name string 'json:"name"'
   Description string 'json:"descr"'
   Weight string 'json:"weight"'
}
```

Listing 2: Service(s) representation in LS.

add a new service, *(ii)* edit a service, *(iii)* remove a service, and *(iv)* search for a service.

Transactions Ledger

To keep track of the transactions that took place over the time and enabling the computation of the agent reputation, the transaction ledger has the following format as shown on the Figure 3:

```
type Transaction struct {
   Incr_id int 'json:"incremental_id"'
   AgentID string 'json:"writingAgId"'
   Tr_id string 'json:"transact_id"'
   Demander string 'json:"whodemands"'
   Executer string 'json:"whoexecutes"'
   Service string 'json:"?service(s)"'
   Outcome string 'json:"outcome"'
   Rating string 'json:"rating"'
   Timestamp string 'json:"timestamp"'
}
```

Listing 3: Transaction Ledger Structure.

According to the proposed design, both executor and demander have to write on the TL at the completion of a transaction. This is necessary to identify a possible misalignment among the evaluations and the perception of the outcomes.

[4]The file settings contains: services, needs, and heuristics distribution.
[5]a simple key-value store, http://leveldb.org

The operations allowed on the TL are: *(i)* add a new transaction, *(ii)* search for a transaction (using different parameters as filters), *(iii)* search for an agent (using different parameters as filters), and *(iv)* computing the reputation.

The Fabric-Network has been configured using configuration file *docker-compose.yml* stored at the MS. It contains the settings of:

- orderer.example.com (orderer);
- ca.example.com (certificate authority);
- peer0.org1.example.com;
- peer1.org1.example.com;
- peer2.org1.example.com:
- cli[6].

In this implementation, to simplify design and development, the orderer is configured in a stable *SOLO* mode offered by the currently available version of Hyperledger Fabric (v1.0), keeping the default values for *Batch Timeout* (2 seconds) and *MaxMessageCount* (10 messages). However, distributing the ordering service and employing a consensuses mechanism to order the transactions are envisaged in the upcoming versions of the Hyperledger.

Framework Dynamics

The architecture we have implemented respect the design presented in Figure 1. The system's components (see Figure 3) are the following components:

- *n* regular agents (BC-A);
- a single CA-Agent (CA-A1), whose functionality can be distributed among multiple agents to eliminate the risk of a single point of failure;
- membership service (ms.example.org);
- three BCT peers (peer*i*.org1.example.com);
- ordering service (orderer.example.com).

Figure 3 also shows the dynamics and possible interactions between the agents and the elements of the blockchain composing the system.

When the agent container (hosting the agents' community) and the ledgers are running, the system is ready to work.

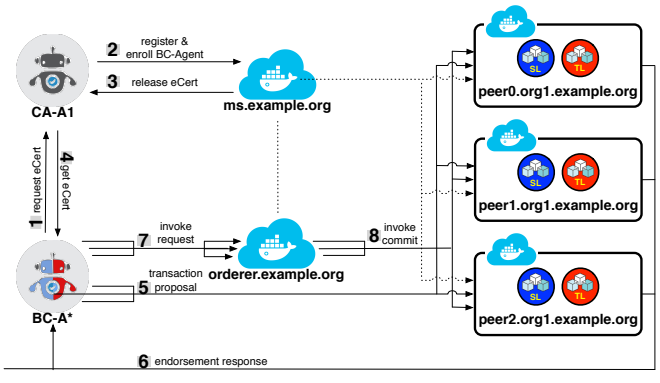

Fig. 3. Architecture of the implemented system.

[6]special node enabling the management of the peer and the installation of the chaincode.

Referring to Figure 3, to be part of such a community, a regular agent (BC-A) has to be enrolled to it. Thus, it sends a message to the CA-Agent to require the registration in the community in order to obtain credentials and the certificates to operate on the SL and TL *(1)*. The AC-Agent controls the credentials of the demander and, if it is satisfied, it requires a certificate to the BCT-CA *(2)*. Once the certificate is released *(3)* the CA-Agent sends a message containing the related eCert to the BC-A *(4)*.

At this point, concerning LS, the BC-A is able to *(i)* publish a service(s) it is willing to offer and *(ii)* search for a service(s) it is willing to demand.

BC-A can also host the peer. Alternatively, the peers can be deployed only on the selected agent, or on the independent entities. In the current implementation, we decouple an agent and the peer to clarify all the communication dynamics.

In order publish the service, the BC-A has to issue an `invoke` transaction that updates the Service Ledger, in order to search for the available services, the agent sends a `query` transaction. To execute a transaction, first, a transaction proposal is broadcasted to the peers *(5)*. The peers verify (i) that the transaction proposal is well formed, (ii) it has not been submitted already in the past (replay-attack protection), (iii) the signature is valid (using MS), and (iv) that the BC-A is properly authorized (that a certificate of the BC-A is valid and ACL policy is satisfied). The endorsing peers take the transaction proposal inputs as arguments to the invoked function of the smart contract. The smart contract is then executed against the current state database to produce transaction results that is sent in a form of the endorsement response *(6)*. Then, BC-A verifies the endorsing peer signatures and compares the proposal responses to determine if the endorsement responses are the same before submitting the transaction (together with received responses) to Ordering Service that will also verify the corresponding endorsement responses *(7)*.

When the orderer receives transactions, it orders them chronologically and creates blocks of transactions, that are then "delivered" to all peers on the channel *(8)*, to update the ledger. As we assume having two ledgers, the transactions issued for each of them, are sent through separate channels. For instance, further in the paper, we discuss the updates of the Transaction Ledger that happens in a similar way as described above, just by using a separate channel.

IV. Tested Scenarios

The realized system has been tested and studied in four different scenarios. The main actors (CA-A1, BC-A1, BC-A2, and BC-A3) have been kept for all the tests[7]. However, their capabilities and needs have been changed, thus generating different and arbitrary dynamics.

The interactions 1 to 4 in Figure 3 are common to all the tests. The related Agent-UML (AUML) diagram is shown in Figure 4.

Table I summarizes the four configurations used in the following scenarios. The four agents offer and may request the execution of a given set of services. All the services are indicated by $S_n(t_i)$, $n \in \overline{1, N}$, where N is a total number of the services and t_i is a release time.

[7]The executions of the four scenarios have been recorded and are available at the following link: https://retis.sssup.it/ d.calvaresi/scenarios4wi/

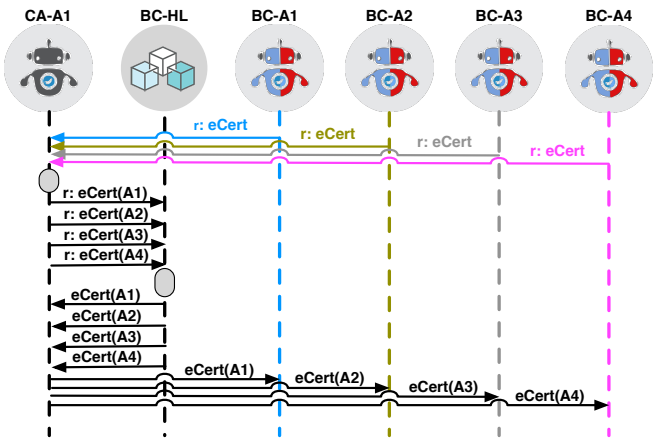

Fig. 4. Agent registration and certificate request.

During the negotiation, the services are exchanged in the form of a tuple characterized by

$$< serviceID >< cost >< timeToComplete > \quad (1)$$

TABLE I
Services and Needs distribution over the scenarios.

Snr	Services	BC-A1	BC-A2	BC-A3	BC-A4
1	Ofr Dmd	S1(t_0) S2(t_3)	S2(t_0)	S2(t_0), S3(t_0), S5(t_0)	
2	Ofr Dmd	S1(t_0) S5(t_0)	S2(t_0)	S2(t_0), S3(t_0), s5(t_6)	
3	Ofr Dmd	S1(t_0) S2(t_3)	S2(t_0)	S2(t_0), S3(t_0), S5(t_0)	S2(t_0)
4	Ofr Dmd	S1(t_8)	S2(t_0)	S2(t_0), S3(t_0), S5(t_0)	

Legend: *Snr* - scenario, *Ofr* - offered, *Dmd* - demanded

A. Scenario 1

After the registration phase (see Figure 4), the agents publish the services they offer on SL, Figure 5 (p1,p2,p3).

In this scenario, BC-A1 needs the execution of S2. Querying SL, it finds two available executors (BC-A2 and BC-A3). Therefore, it proceeds engaging in a negotiation with them. The proposals received are *BC-A2:* cost = 4 and time = 9 and *BC-A3:* cost = 3 and time = 10. In this case, BC-A1 implements a cost-oriented heuristic. Thus, it awards for the execution of S2 the agent BC-A3, rejecting BC-A2. At the completion of S2, BC-A3 writes on TL that a given transaction (identified by a unique ID) is concluded[8] and notifies BC-A1. Accordingly, BC-A1 evaluates the outcome received and writes as well about the transaction completed[8].

We introduced this two-folded evaluation to enable the investigation of possible discrepancy among the evaluation provided by the agents (demander and executor). Such a misalignment can be due to several factors: *(i)* diverse evaluation metrics, *(ii)* honest mistakes, and *(iii)* possible strategic lies. Concerning the possibility of lying, it might be due to mis-, dis-, mal-information, and intentional lies.

By employing the two-folded evaluation within TL, such ambiguous situation can be easily identified. Moreover, it is

[8] according to the format shown in Listing 3

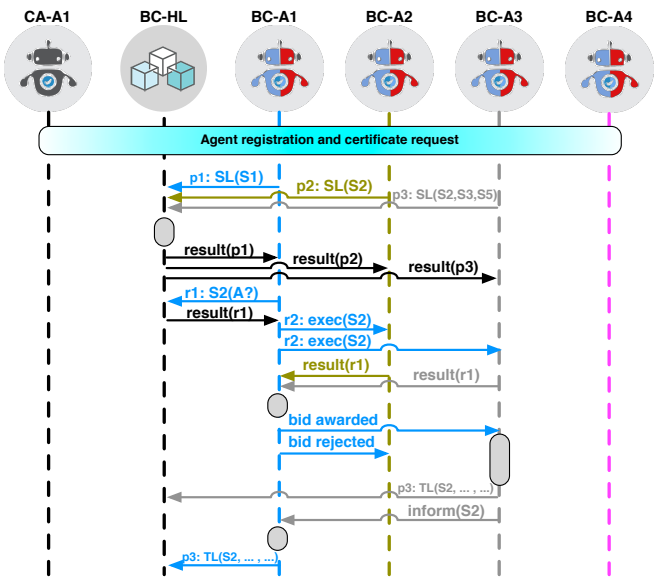

Fig. 5. Agent registration and certificate request.

possible to identify recidivist, intentional, and honest behaviors. Hence, based on transparency, consistency, and persistence provided by the BCT, accurate investigation can be conducted.

Finally, almost instantaneous controls on the agents behaviors and reputation calculation can be performed directly in the TL throughout the use of smart contract without relying on a single trusted entity.

Once tested the basic functionality, it was necessary to study the corner cases. The next sections analyze the cases when:

- a given agent is looking for a service not yet available in the ledger (Scenario 2);
- an agent that is currently down was chosen to provide a service (Scenario 3);
- an agent banned from the community (e.g., due to his malicious behavior his certificate was revoked) tries to interact with other agents in the community (Scenario 4).

B. Scenario 2

In this configuration, BC-A1 needs the execution of S5 (released at t_0). The only agent able to perform it is BC-A3 (who publishes it only at t_6). In this case, we implemented two possible solutions:

- forced the agent to check periodically SL;
- allowed the agent to register for events (e.g, a block is committed to a peer's ledger or a given smart-contract-event is executed) so that they can be notified when a given ledger changes[9].

C. Scenario 3

In this configuration, BC-A1 needs the execution of S2 (released at t_3). In this scenario, we simulated a down-fall of BC-A2 at t_2 (right after the registration of its services on SL). BC-A1 opens the negotiation with BC-A2, BC-A3, and BC-A4. It gets back the bid from the first two but does not receive an answer from BC-A4. Therefore, at the end of the bidding window BC-A1 proceed to evaluate the only bid received neglecting BC-A4 for that transaction.

D. Scenario 4

In this last scenario, BC-A1 wants to publish its service S1 (released at t_8) on TS. However, earlier than t_8, we imposed to such an agent to demonstrate malicious behaviors. Consequently, its certificate gets revoked (and appears in the Certificate Revocation List (CRL[10]) issued by CA) before the time to publish his service occurred. Therefore, the agents that will try to negotiate with BC-A1 will be notified that the certificate of BC-A1 was revoked, and BC-A1 will not be able to issue any more transactions to interact with the ledgers (for both query and update).

V. Discussion

The nature of MAS can be both cooperative and competitive. However, in both cases agents have to rely on the quality of the information and resources exchanged. The risk of being exploited or deceived can hamper the dynamics of a single agent and the whole community. Therefore, to enforce the concept of trust, the developed system aims at providing a unambiguous and immutable means to punctually compute and update the agent reputation.

Our ongoing work focuses on realizing smart contracts able to unequivocally update the reputation on TL. By doing so, we can avoid relying on a trusted entity (agent) who can be subject to manipulation or possibly biased judgment, especially if unique. One can argue that by employing PKI and a single certification authority for the current implementation we keep a risk of having a single point of failure. However, as we decouple the functionalities of the identity management (MS) and computing reputation (smart contracts) we can provide stronger security. Moreover, single MS can be substituted by Collective Authority servers, an example of such is presented in [35]. CA (that is a component of MS) can also be decentralized (thus eliminating a risk of having a single point of failure) by plugging a chain of CAs, and using threshold signatures as announced for the later releases of Hyperledger Fabric.

BCT also provides a solution to the problem of reaching a distributed consensus on the reputation value among the peers in the community: every peer executes the same smart contract and the transactions are aligned by the orderer. The latter can also be distributed and then can employ a consensus protocol such as PBFT – a partially synchronous protocol for Byzantine state machine replication that was presented in [36] and has been used in the design of the previous version of Hyperledger Fabric (v0.6). As mentioned in Section III, this framework enables the possibility of providing both overall- and specific-values for the agent reputation only by modifying the logic of smart contracts. Enabling specific-values-based reputation means being able to evaluate all the possible features characterizing the agents' behaviors. In particular, a given agent can be evaluated according to its capability of *(i)* performing and *(ii)* evaluating *a given task*.

Currently, every evaluation is written on the ledger by both the participant of a given transaction (demander and executor)

[9]http://hyperledger-fabric.readthedocs.io/en/latest/

[10]https://tools.ietf.org/html/rfc5280

according to their judgment. When both the values have been written on TL, a smart contract can be triggered. Base on the values provided, it can compute and update the reputations. Moreover, analyzing previous behaviors, further investigations can be conducted.

Thanks to the blockchain technology, evaluation and reputations values can also be tracked and monitored over the time. Thus, the timely understanding of malicious and suspicious activities/intention can be easily achieved. Such investigation can be possible by extending the structures of SL and TL (see Listings 1, 2, and 3) and implementing the related smart contracts.

VI. Conclusion

This paper addressed the challenge of enabling trusted interactions in MAS. We studied the binding of BCT and MAS, and proposed the design, development, and evaluation of a MAS based on Jade and Fabric Hyperledger v1.0. In the design phase, we identified the MAS functionalities to be improved in *(i)* agent identity management, *(ii)* distribution of the DF, and *(iii)* tamper-proof storage of the agent interactions to enable the trustworthy computation of the agent reputation.

Such a system has been tested in different configurations and scenarios, both in regular and corner cases.

The results confirmed that the implementation of such system constitutes a solid ground for having trusted interactions in MAS. Moreover, it enables the possibility of computing both overall and specific values for the agent reputation by storing immutable evaluations and, based on them, updating the reputation via smart contracts in a transparent way. Developing the smart contracts for computing the reputation is an ongoing work that is unveiling imperative challenges for both agent and blockchain communities. The application primarily targeted are startup assessment and tourism-orient platforms

References

[1] D. Calvaresi, A. Claudi, A. Dragoni, E. Yu, D. Accattoli, and P. Sernani, "A goal-oriented requirements engineering approach for the ambient assisted living domain," in *Proceedings of the 7th International Conference on PErvasive Technologies Related to Assistive Environments*, 2014, pp. 20:1–20:4.

[2] A. Dubovitskaya, V. Urovi, I. Barba, K. Aberer, and M. I. Schumacher, "A multiagent system for dynamic data aggregation in medical research," *BioMed Research International*, vol. 2016, 2016.

[3] D. Calvaresi, D. Cesarini, P. Sernani, M. Marinoni, A. Dragoni, and A. Sturm, "Exploring the ambient assisted living domain: a systematic review," *Journal of Ambient Intelligence and Humanized Computing*, pp. 1–19, 2016.

[4] D. Calvaresi, M. Schumacher, M. Marinoni, R. Hilfiker, A. Dragoni, and G. Buttazzo, "Agent-based systems for telerehabilitation: strengths, limitations and future challenges," in *Proc. of X Workshop on Agents Applied in Health Care*, 2017.

[5] F.-S. Hsieh, "Modeling and control of holonic manufacturing systems based on extended contract net protocol," in *American Control Conference, 2002. Proceedings of the 2002*, vol. 6, 2002, pp. 5037–5042.

[6] B. Yu and M. P. Singh, "An evidential model of distributed reputation management," in *Proceedings of 1st international conference on Autonomous Agents and Multiagent Systems*. ACM, 2002, pp. 294–301.

[7] S. D. RAMCHURN, D. HUYNH, and N. R. JENNINGS, "Trust in multi-agent systems," *The Knowledge Engineering Review*, p. 1–25, 2004.

[8] Y. Hedin and E. Moradian, "Security in multi-agent systems," *Procedia Computer Science*, vol. 60, pp. 1604 – 1612, 2015, knowledge-Based and Intelligent Information and Engineering Systems 19th Annual Conference, KES-2015, Singapore, September 2015 Proceedings. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S1877050915023972

[9] S. Nakamoto, "Bitcoin: A peer-to-peer electronic cash system," 2008.

[10] M. Swan, *Blockchain: Blueprint for a new economy*, 2015.

[11] D. Tapscott and A. Tapscott, *Blockchain Revolution: How the technology behind Bitcoin is changing money, business, and the world*, 2016.

[12] D. Calvaresi, A. Dubovitskaya, J. P. Calbimonte, K. Taveter, and M. Schumacher, "Multi-agent systems and blockchain: Results from a systematic literature review," 2018.

[13] F. L. Bellifemine, G. Caire, and D. Greenwood, *Developing multi-agent systems with JADE*. John Wiley & Sons, 2007, vol. 7.

[14] C. Cachin, "Architecture of the hyperledger blockchain fabric," in *Proc. of Distributed Cryptocurrencies and Consensus Ledgers*, 2016.

[15] J. Granatyr, V. Botelho, O. R. Lessing, E. E. Scalabrin, J.-P. Barthès, and F. Enembreck, "Trust and reputation models for multiagent systems," *ACM Computing Surveys (CSUR)*, vol. 48, no. 2, p. 27, 2015.

[16] A. Dragoni, F. Mascaretti, and P. Puliti, "A generalized approach to consistency based belief revision," *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, vol. 992, pp. 231–236, 1995.

[17] A. Dragoni and P. Giorgini, "Distributed belief revision," *Autonomous Agents and Multi-Agent Systems*, vol. 6, no. 2, pp. 115–143, 2003.

[18] I. Pinyol and J. Sabater-Mir, "Computational trust and reputation models for open multi-agent systems: a review," *Artificial Intelligence Review*, vol. 40, no. 1, pp. 1–25, Jun 2013. [Online]. Available: https://doi.org/10.1007/s10462-011-9277-z

[19] J. Sabater and C. Sierra, "Regret: A reputation model for gregarious societies," in *Fourth workshop on deception fraud and trust in agent societies*, vol. 70, 2001, pp. 61–69.

[20] C. Cachin and M. Vukolić, "Blockchains consensus protocols in the wild," *arXiv preprint arXiv:1707.01873*, 2017.

[21] A. Kiayias, A. Russell, B. David, and R. Oliynykov, "Ouroboros: A provably secure proof-of-stake blockchain protocol," in *Annual International Cryptology Conference*. Springer, 2017, pp. 357–388.

[22] T. Swanson, "Consensus-as-a-service: a brief report on the emergence of permissioned, distributed ledger systems," 2015.

[23] E. C. Ferrer, "The blockchain: a new framework for robotic swarm systems," *arXiv preprint arXiv:1608.00695*, 2016.

[24] M. Bottone, F. Raimondi, and G. Primiero, "Multi-agent based simulations of block-free distributed ledgers," 2018.

[25] S. Kozlowski, "An audit ecosystem to support blockchain-based accounting and assurance," in *Continuous Auditing: Theory and Application*. Emerald Publishing Limited, 2018, pp. 299–313.

[26] A. Zuiderwijk, M. Janssen, and C. Davis, "Innovation with open data: Essential elements of open data ecosystems," *Information Polity*, vol. 19, no. 1, 2, pp. 17–33, 2014.

[27] J. Dai and M. A. Vasarhelyi, "Toward blockchain-based accounting and assurance," *Journal of Information Systems*, vol. 31, pp. 5–21, 2017.

[28] A. Jøsang, R. Ismail, and C. Boyd, "A survey of trust and reputation systems for online service provision," *Decision support systems*, vol. 43, no. 2, pp. 618–644, 2007.

[29] K. N. Khaqqi, J. J. Sikorski, K. Hadinoto, and M. Kraft, "Incorporating seller/buyer reputation-based system in blockchain-enabled emission trading application," *Applied Energy*, vol. 209, pp. 8–19, 2018.

[30] K. Qayumi, "Multi-agent based intelligence generation from very large datasets," in *Cloud Engineering (IC2E), 2015 IEEE International Conference on*. IEEE, 2015, pp. 502–504.

[31] J. Gattermayer and P. Tvrdik, "Blockchain-based multi-level scoring system for p2p clusters," in *Int Conf Parallel Processing Workshops ICPPW*. IEEE, 2017, pp. 301–308.

[32] S. J. Russell and P. Norving, "Norvig," *Artificial Intelligence: A Modern Approach*, pp. 111–114, 2003.

[33] D. Calvaresi, M. Marinoni, A. Sturm, M. Schumacher, and G. Buttazzo, "The challenge of real-time multi-agent systems for enabling iot and cps," *International Conference on Web Intelligence*, 2017.

[34] S. D. Ramchurn, D. Huynh, and N. R. Jennings, "Trust in multi-agent systems," *The Knowledge Engineering Review*, vol. 19, pp. 1–25, 2004.

[35] E. Syta, I. Tamas, D. Visher, D. I. Wolinsky, P. Jovanovic, L. Gasser, N. Gailly, I. Khoffi, and B. Ford, "Keeping authorities "honest or bust" with decentralized witness cosigning," in *Security and Privacy (SP), 2016 IEEE Symposium on*. Ieee, 2016, pp. 526–545.

[36] M. Castro and B. Liskov, "Practical byzantine fault tolerance and proactive recovery," *ACM Transactions on Computer Systems (TOCS)*, vol. 20, no. 4, pp. 398–461, 2002.