ΟΙΚΟΝΟΜΙΚΟ
ΠΑΝΕΠΙΣΤΗΜΙΟ
ΑΘΗΝΩΝ

ATHENS UNIVERSITY
OF ECONOMICS
AND BUSINESS

Thesis

Student: Lydia Athanasiou

November 16, 2022

Contents

1 Abstract

...

2 Introduction

It is undeniable that E-commerce is a fast-evolving market. Especially nowadays an immense majority of individuals shop via internet and coronavirus has contribute to that trend [36]. Obviously, the incentive for the buyers is the reputation scores and the feedback of other users that have already purchase from a specific e-shop. Most e-commerce markets used to store their reputation scores with a centralised manner which led to several drawbacks as mentioned below. Firstly, it is unconditionally easy for malicious users (insiders or outsiders) to alter the stored information. As a result, the integrity and simultaneously the reliability of the centralised technic is compromised. Secondly, we should not forget the intentional or unintentional malicious feedback under the products. Many attackers are trying to bombard platforms with fake comments to discourage users form buying a particular product. Thus, the centralised mechanism does not consider that parameter and computes the reputation score centrally. As a result, users may reject a good product unfairly just because they read negative comments. Last major disadvantage of the centralised mechanism is that does not provide incentive to users to evaluate a service of a platform. So, the reputation score is not representative due to the lack of summative comments by the buyers. Under those circumstances, many researchers have proposed a new generation of e-commerce environments that tries to take advantage of decentralised mechanisms such as the blockchain environments [42], [41], [35]. Last research revealed that about 2.14 billion consumers choose to shop online on platforms like amazon, wish, Shein etc [37], [16]. Those percentage give us the initiation to go through those technologies in order to evolve a trustworthy manner in online marketplaces.

3 Related work

This section is divided into 3 different subsections:

1. the first one studies papers related to reputation and trust mechanisms
2. the second one studies mathematical ways in order to transfer the reputation score from collective services to individual agents (counterparts of the hole service).
3. and the third one studies the blockchain mechanism and its' relation with the trust mechanism.

3.1 Reputation and trust mechanisms

Summary of paper [39]

The rapid development of Internet of Things (IoT) has contributed to a generation of smart objects. One of the main problems that IoT poses is that requires high interoperability for the heterogeneous objects-devices-networks and technologies. It is very common in such environments to use service composition. The users are unaware of this procedure. They only see the final service without having consensus of the background microservices that compose the final service. The final service appears as a unique service to the final users. In a service composition, the IoT middleware consists of a service-oriented architecture (SOA) as mentioned in [39]. Each connected device is a service provider that provides microservices. The middleware layer accepts service requests from the final users and perform multiple tasks (e.g., service discovery and aggregation, SP selection, network routing functions, and security and trust management). A sensing device for example can provide multiple different services such as temperature and simultaneously humidity measurements. For this reason, devices are belonged to service classes. All the objects of the same class have the same properties and the same abilities. The output of one device may pass as input to another like a **pipeline architecture**. **Obviously, to offer a user-requested service, the analogous microservices should be found that will compose the final service. This microservices are offered by service providers. But it is very common that many different service providers offer the same microservices**. In this case a way should be found to distinguish who SP is more reliable and who is malicious. Also, it is very important to mention that the middleware should consider the **compatibility** of the microservices-components to be easy to be mixed in a successful way. So, **trust is one of the most important security metrics in SOA IoT systems**.

The authors of this paper [39] mention that a trust management system (TMS) has integrated mechanisms and technics to evaluate the reputation and the trust of nodes of the networks, based on a specific trust model. **CTRUST focus on dynamic trust management for collaborative IoT applications**. The parametrization, weighting, maturity, and decay of trust between nodes are modelled based on = functional attributes of the nodes in the collaboration context. The work in [39] focused only on collaborative applications where the trustor can requests and receives immediately services from Service Providers and can also provide a direct trust rating of the nodes based on their performance.

When a user requests a final service, automatically triggered a collaboration of miscellaneous microservices. In addition, the final service will be composed by using at least 2 microservices which may be provided by multiple service providers. That specific architecture, according with the researchers, offers multiple advantages that are the key to the realisation of the Internet of things vision, such as **modularity, increased reliability and technology heterogeneity and interoperability**. According with this study, here are two categories of service compositions: homogenous and heterogeneous. In a homogenous composition, all the services are from the same class. In a heterogeneous composition, the underlying services are mixed and matched from different classes. In this scenario, the trust model must identify the most suitable SPs within each service class. Generally, there are three basic types of workflows, which may be used in any combination. First, it may be a simple case of selection, where the SR receives a list of SPs along with their trust ratings. Secondly, services may be composed in a sequential workflow where the results from one service are provided to the next and so on (the pipe-lined architecture that we mentioned before), and the results of the second service are then returned to the SR.

The authors of [39] provide a list of characteristics for an ideal TMS for IoT.

1. Platform Consideration
2. Trust as a Decision (TaaD): is the procedure in which you decide that you will choose the best option for collaboration depending on trust and by accepting the risk of wrong trust measures.
3. Contextual Trust Parametrization
4. Trust Persistence: continues to trust in a determined way.
5. Trust Decay
6. Risk Mitigation: the process of reducing risk exposure and minimizing the likelihood of an incident.
7. Trust Accuracy: is the degree of closeness between a measurement and its true value. Precision is the degree to which repeated measurements under the same conditions show the same results.
8. Trust Convergence: the act of converging and especially moving toward union or uniformity the convergence.
9. Trust Resilience: We have trust resilience when we can trust the nodes then we are to persist with the difficult tasks, realising that mutual assurance and confidence are likely to be rewarded with overall success.

10. Transparent Trust Composition: The trust model have to provide methods for estimating the trustworthiness of a composed service considering the trust scores of the SPs, the service context, and the workflows. This should be done transparently in order to conceal internal details of the composition from the SP.

11. Transparent Trust Decomposition: The SR would give a trust score after consuming the composed service. In addition, the TMS must have methods to decompose the trust score from the SR and utilize it to update the trust scores of the underlying services in an impartial and transparent manner.

Moreover, the authors propose SC-TRUST as a suitable trust model to guide compositions of IoT services. In this model the trust metric is calculated based on the **objective measurement** of a node's performance. The multiple services have some specific trust criteria relevant to their service class. The SR requests a middleware platform for composing the different services. The middleware composes a service workflow pattern to match the SR's requests. The prior partial trust scores on the parameters for each SP are aggregated according to the SR's weight assignment, to calculate a single trust score for each SP. In CTRUST project, functions are used to determine which SPs should be selected for the requested service composition. Such advanced functions are Trust Parameters, parameters weights (**The SR assigns weights to each parameter to reflect their relative importance based on the current subjective opinions of the SR**), partial Trust Scores and Aggregation, trust decay, trust recommendations and trust maturity.

Therefore, a novel, bottom-up approach is proposed by the authors in [39], to compose the trust value. The objective here is initially find a suitable composition which meets the SR's **threshold**, even if it is not the best possible composition. Then, based on the feedback from the SR, trust scores of the SPs can be updated and compose a better service with every iteration. In a selection workflow, the SR must select one Service Provider from a group of two or more Service Providers from the same or associated service classes and offer similar services. Once an Service Provider is chosen, the services of the others are not used and therefore do not impact on the composite trust score. Additionally, the authors determine the Service Provider with the maximum prior trust score based on the partial trust values on the same set of parameters, using weights assigned by the Service Rrovider.

In [39], [36], etc the composed service consists of several micro-services. Therefore, **a method is required to reliably decompose the partial trust scores given by the SR to the underlying services in a reasonable manner**. One method in order to improve the quality of group assessments is to assign both a group score and an individual score to every member. The individual component is based on a separate piece of work produced by each group member. Thus, the advantage of collaboration and group assessment may be achieved by **providing an incentive for improving individual performance**. Similarly, the individual component punishes bad or low-performing members, thus creating a fair distribution of the marks among the group members. Moreover, the group members are not required to provide feedback on their peers. Suppose that a SP adjusts its service provision such that it receives good ratings on its unique parameters but bad ratings on the others. **This is an opportunistic service attack**. However, the attack fails in this model because the SP destroys its own reputation as well with no overall gains.

To evaluate this state-of-the-art method, the researchers made several experiments by which is revealed that the trust risks are minimised. It may be argued that more trustworthy compositions would generally cost more because the price is an incentive for a Service Provider to produce better services.

Summary of paper [29]

In peer-to-peer environments with high probability a proportion of nodes may not be able to provide their services to other nodes in an efficient manner. Under those circumstances a metric should be take place like the reputation metric to distinguish low-performing peers and simultaneously maintain the characteristics of peer-to-peer environments. Examples of such characteristics are the anonymity and privacy. **But if all peers select only the high-performance peers to collaborate this may lead to unintended results such as low efficiency for high-performing peers**. This paper tries to justify experimentally that the computation of the reputation metric must be complemented by reputation-based policies that define the peers eligible to interact with each other. Two orthogonal dimensions of reputation-based policies are then introduced: **"provider selection" and "contention resolution"**. Moreover, in [29],[1] is justified experimentally that **this reputation cycle affects the speed and the accuracy of convergence of the reputation values to the real hidden information**. Peers according to their type, their inherent capabilities and/or their strategy, succeed or fail in offering services to other peers. After the finish of the transaction, client evaluate the peer for his performance. Then, **an aggregation is applied to all the history of peer's outcomes into a unique reputation value. Bayes' rule is an efficient aggregation function if there is defined an initial belief on the success probability of each type of peers and the proportions of the population of peers that belong to each type. If peers follow dynamic strategies over time and change their probability of success, then the fraction of the number of successful service provisions over the total service provisions of a peer could be used with more weight being placed to the recent history**.

$$R = s'/t'$$

where s=successful. t=total

The authors in [29] are posing a new revealed problem. Let's suppose that a peer is requesting a service. If all successfully provisioned services have the same value for the peer, one that has the maximum reputation value. However, following this strategy the high performing peer are punished in two ways:

1. the higher reputation a peer has, the more users he attracts to consume his own resources, and

2. such a peer receives equal benefit from the peer-to-peer system as other peers that have a low performance level.

Clearly, this behaviour of peers to exploit the reputation metrics provides wrong incentives to both high- and low-performing peers. A high-performing peer is motivated to lower his performance, while a low-performing peer is motivated to keep his performance to the same level. These incentives lead to a **market of "lemons"**, and possibly to the gradual decomposition of the peer-to-peer system. The authors are trying to classify the reputation-based policies into two dimensions: **"provider selection" and "contention resolution"**. Note that for a highly reputed peer that contents for the resources of another peer with a low-reputed peer the probability to be selected under this policy is close to 1, similarly to the highest reputation contention resolution policy. In the other end of the spectrum, under the present policy, peers with low reputation values have a small yet positive probability of receiving some services regardless of the contention level.

Reputation-based policies determine the pairs of peers that are eligible to interact. Recall that client peers rate the providing ones regarding the performance of the latter in their transactions with the former. This feedback is sent to the reputation system after several completed transactions. The reputation values of the respective peers are updated based on this feedback. However, these updated values determine the new pairs of peers that are eligible to interact. These interactions will result in additional ratings' feedback, etc. Thus, a vicious cycle is formed.

Next, the authors are dealing with the efficient aggregation of ratings in terms of communication overhead in a Peer-to-Peer system in the absence of a central authority. Reputation values are calculated using functions for the aggregation of votes. The votes sent by a peer are associated to his transacted peers. Thus, if λ is the mean rate according to which a peer is served, then the number of feedback messages per unit time that must be sent to the reputation holders is proportional to λ. Thus, the number of feedback messages sent is reduced to $p\lambda$. The proposed implementation are the following:

1. a vote to be sent to the corresponding reputation holder with probability p
2. send the votes aggregated in a time to the reputation system with probability p

In this P2P there are two types of peers: altruistic and egotistic. Each altruistic peer provides his service successfully with a high probability, while an egotistic one succeeds in each with a low probability. Time is assumed to be slotted and at each slot, each peer requests a service with a certain probability. The efficiency in this Peer-to-Peer system can be measured as the average ratio of successful transactions for an altruistic peer over either i) the average number of service requests or ii) the average number of initiated services. Only successful transactions provide value for client peers. **The total value provided to a peer should be in accordance with his performance, in order the right incentives for performance to be provided**.

To sum up, this paper reveals that a small randomly selected subset of the ratings' feedback is sufficient information for the fast and accurate calculation of the reputation values, even if the Peer-to-Peer population is renewed with a high rate.

Summary of paper [18]

This paper is the most-relevant published paper considering the trust management in IoT federations. The proposed model uses several methods to reward or punish things according to their behaviour in the network. The trust metric is calculated by using feedback from things and are integrated into game strategies. These strategies capture the dynamic nature of federations since the population of trustworthy versus untrustworthy things changes over time. To evaluate their approach, the researchers, made a series of experiments the results of whom show a better mitigation of attacks such as bad-mouthing and ballot-stuffing on trustworthy things.

The researchers suggested the gathering of things into federations. Federations' benefits include fostering things' collective over individual behaviours and enforcing cross-thing collaboration despite their nature. To discover answers in several questions they resort to the **Evolutionary Game Theory**, a way to run federations as games where things selects a strategy that reflects its trust in compliance with the federation's rules. Generally, things' trust behaviours are associated with payoffs that federations use to reward them when they complete the assigned operations as expected. On the contrary, the payoffs are limited and may lead to reject things from the federations. Because things are expected to frequently sign-up in and sign-off from federations, this could put the federations at the risk of instability leading to chaos. This risk is mitigated by tracking the trust behaviour of each thing in terms of what it did and what it is doing. This helps monitor federation evolution, which is a key characteristic of the EGT.

Requirements linked to IoT include; **quality, latency, trust, availability, reliability, and continuity** that should impact efficient access and use of IoT data and services. The former include distribution, interoperability, scalability, resource scarcity, and security. And, the latter include sensing, communication, and actuating that are mapped onto a 3-layer IoT architecture referred to as perception, network, and application, respectively. In IoT-Trust, the objectives are to mitigate the effect of malicious nodes on interaction quality and to minimize the number of collusion attacks. **IoT-Trust considers a simple case in which the direct user satisfaction experience f is a binary value (1/0 for satisfied/unsatisfied). Then, f is considered as an outcome of Bernoulli trial that is used as a weight for positive observations**.

Compared to the aforementioned non-game trust models, the future evolutionary game model of the researchers will provide an analytical study on the conditions that were satisfied using incentives. Moreover, their model will promote node stability that is built on top of the proposed trust model to detect peers' unstable trust behaviours.

Their trust-based IoT federations management framework puts focus on things' short and long-term trust behaviours and assigns incentives to strategies that things adopt throughout their course of actions. This management also relies on an evolutionary game-based trust-model that provides a formal setting for proving dominant strategies. It considers the dynamism of IoT environment, ensures the stability of things' behaviours, and mitigates the effects of malicious attacks such as bad-mouthing and ballot stuffing. This comparison's of the authors model illustrates with whom a thing interacts, how trust evolves over time, and who recommends a thing.

1. Direct trust relies on a thing's one-to-one interactions with peers to define its trustworthiness level.

2. Indirect trust proceeds differently to direct trust by relying on peers' recommendations about a thing.

3. Trust evolution tracks changes of a thing's trustworthiness level over time, which is normal due to the dynamic nature of federations.

4. Stability analysis is in line with trust evolution but focuses on the impact of trust changes on a federation's stability over time.

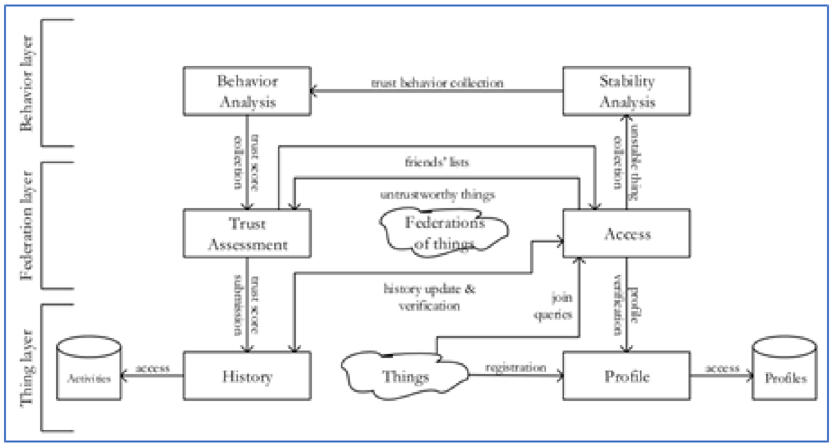
5. Provability provides a formal basis for finding the conditions to satisfy prior to reaching equilibrium states.

The architecture associated with the authors Trust-based Management of IoT Federations (TMIF) is built-upon three layers, thing, federation, and behaviour, hosting different modules and repositories.

1. The thing layer is concerned with defining things' profiles, tracking their ongoing operations, and forming their histories.

2. The federation layer is concerned with assembling and dismantling federations along with ensuring the stability of some.

3. the behaviour layer is concerned with assessing things' trust behaviours and analysing the stability of federations.



The modules involved in this assessment are presented thoroughly in the paper: trust assessment, behaviour analysis, stability analysis.

1. In the threat model, things are expected to be malicious by engaging in self-interest operations like over-using resources and delaying the release of services.
2. Federations provide incentives to things, so they trustfully collaborate. The authors endow each federation with an EGT model (IoT-EG) in which incentives depend on how trustworthy a thing is. Such incentives are part of the payoff of each thing and can increase or decrease depending on the strategy that the thing adopts.
    (a) Game strategies
    (b) Game analysis

To sum up, this paper presented a trust-based evolutionary game model for managing IoT federations. A federation acted as a platform for hosting things. Compared to existing trust models, the role of trust is fostered into the games that things play when they join federations based on factors like trustworthiness. These games capture the dynamic nature of federations that would like to attract, reward, and retain trustworthy things and penalize and expel untrustworthy ones. This exercise of retaining and expelling along with things' signing in and signing off has an impact on the stability of federations that wish to remain active for a longer period. The stability in the proposed model can be achieved by analysing the trust behaviours of things and retaining only those exhibiting stable behaviours. A set of experiments were conducted by simulating federations according to a specific network topology, things' capabilities, and interactions between things and federations. Results show that high incentives encourage things to follow trustworthy strategies and minimize bad-mouthing and ballot-stuffing attacks on their operations.

3.2 Transport of collective to individual reputation score

Summary of paper [9]

This paper submits a new mechanism in environments with noise that interferes the network. The proposed mechanism takes advantage of the properties of electronic markets as the ability to levy listing fees from sellers. The author shows that sellers with different cost functions, imperfect private monitoring of a seller's effort level and a simple feedback mechanism (e.g., 0 or 1) that asks buyers to rate a transaction as "good" or "bad". The mechanism distorts the resulting payoffs of individual sellers relative to the complete information case, transferring part of the payoffs of more efficient sellers to less efficient sellers. The magnitude of this distortion is proportional to the amount of noise associated with observing and reporting the quality of a good.

As I mention analytically in the annex, the feedback mechanisms are vitally important in the online marketplaces for building trust and enables cooperation among the nodes of the network. The network can aggregate past scores and produce statistics of his recent ratings that are available in the entire community. However, in environments with noise the maintenance of long-term reputations is impossible and that poses a serious problem. Once a seller has established a reputation for producing good quality, in environments with noisy observations of quality he will then be tempted to occasionally cheat because buyers will assume that occasional bad ratings are simply due to noise. Each period the mechanism charges a listing fee contingent on a seller's announced expected quality. **It subsequently pays the seller a reward contingent on both his announced quality and the rating posted for that seller by that period's winning bidder**. Furthermore, the auction mechanism leaves all seller types with positive rents.If these rents are sufficiently high, the center can design an individually rational schedule of listing fees and rewards that :

1. induce both truth-telling and first-best effort levels by all seller types and
2. maximize average social welfare for the entire community

The proposed mechanism in [9] is robust to a number of contingencies of particular concern in online environments, such as easy name changes (that tempt sellers to cheat, disappear and re-enter a community under a new online identity) and inept sellers (sellers who cannot produce at any quality and, hidden behind the relative anonymity of online environments, enter the market for the explicit purpose of cheating). Finally, **this mechanism demonstrates that simple binary ratings are sufficient for sustaining full cooperation and efficient effort levels in environments of multiple qualities**.

The setting in [9] involves a monopolist seller who offers for sale a single unit of a product/service ("good") to one of multiple identical buyers. At the beginning of the game the seller announces the expected quality of his product. A mechanism is then used to allocate the good among the buyers by determining the buyer that receives the good and the price that has to be payed to the seller. Seller effort determines the quality of the resulting good. More specifically, if a seller of type ϑ exerts effort z, the quality q of the resulting good is characterized by a cumulative probability distribution Φ(ϑ, ϰ, z). The winning buyer privately observes the quality of the good delivered, but not the effort exerted by the seller. Moral hazard is introduced because high effort is costlier to the seller, who can reduce his costs by exerting lower effort, providing the buyer with a good of lower expected quality. Adverse selection is introduced because the type of the seller is unknown to buyers and therefore the seller can attempt to increase his gains by misrepresenting his true type. In most electronic marketplaces sellers cannot credibly pre-commit to a particular effort level. Sellers then have incentives to exert less effort than what they promise to buyers. Knowing this, buyers will place lower bids.

The author's objective is to design a mechanism that achieves the following two targets:

1. Induce sellers of all types to truthfully announce their production quality.
2. Maximize the average social welfare given the probability distribution of seller types on the market.

The mechanism in [9] asks the winning bidder to rate the seller based on the quality of the good she received. Buyers are given the option of reporting the outcome of a transaction as either "positive" (+) or "negative" (-), with positive ratings indicating that the quality of the good was equal to or superior to the promised quality and negative ratings indicating that the quality of the good was inferior to that promised by the seller.

Summary of bilateral exchange game studied in this paper:

1. Seller announces his expected quality y
2. Seller pays fee f(y) to center; center publishes the seller's expected quality y to buyers
3. Buyers bid their expected valuations for the good in a second price Vickrey auction; the winning bidder pays G, equal to the second-highest bid
4. Seller decides to exert effort corresponding to expected quality q [q, q], producing a good whose resulting quality ϰ follows the cumulative probability distribution φ(ϰ q)
5. Buyer receives the good, experiences its quality, and realizes the corresponding valuation w(ϰ); buyer rates the seller, reporting "+" if ϰ y and "-" otherwise
6. Center pays the seller f(x, y), where x +, is the buyer's report Under the assumption of truthful reporting, if a seller of type ϑ promises quality y but produces at an effort level associated with quality q, the probability of a negative rating is given by

$$\varphi(y-q)^5$$

.

An important consideration in mechanism design is the robustness of a mechanism to the assumed features of the underlying environment due to the sheer number and the heterogeneity of participants.

1. An attractive property of the mechanism is that it induces cooperation and achieves efficiency in a one-shot setting. This is a particularly desirable property in electronic markets where the relative ease with which players can change online identities reduces the efficiency of mechanisms that rely on repeated interaction.

2. The author defines an inept type as a seller who has no production facilities and enters an electronic market with the explicit intention to cheat. Given the relative ease with which players can anonymously enter/exit most online markets, inept types are plausible, and mechanisms ought to be robust to their existence. If the fees required to keep inept sellers out of the market are too high, the center can periodically redistribute part of those fees back to sellers who maintain average ratings at acceptable levels. If the ratings profile of inept sellers is substantially distinct from that of "normal" sellers this can still discourage inept sellers from entering while maintaining the payoffs of "normal" sellers at acceptable levels.

3. "Irrational" honest types who always perform what they promise are, perhaps, just as plausible in large-scale environments as inept types. Given that our mechanism induces strategic types to behave as if they were honest, assumption of honest types does not change the outcomes induced by the mechanism.

4. Implementation considerations. The mechanism is implemented in online auction marketplaces provided that (a) there is a simple way to communicate intended product quality y to the center and buyers, (b) the center has reliable estimates of the set of possible seller cost functions $c(\vartheta, y)$ and ratings distribution φ (c) the center can estimate the expected auction revenue function $G(y)$ (easy if the center keeps track of past auction results) and (d) for each product quality y, there is exactly one type ϑ for which y is the first best quality.

Under the above assumptions, the basic idea of the mechanism applies even to more complex marketplaces where types ϑ are multi-dimensional, y is communicated via a vector of attributes, and buyers supply separate ratings for each quality attribute.

To sum up, this paper shows how the combination of a simple feedback mechanism with the ability of a marketplace operator to levy listing fees from sellers can restore full average social efficiency in environments with both moral hazard and adverse selection. Finally, the mechanism is robust (or can be extended to become robust) to several contingencies of particular concern in online environments, such as easy name changes and inept sellers. The principal drawback of the mechanism is that it results in some distortion to individual seller payoffs, essentially transferring part of the payoffs of more efficient sellers to less efficient sellers.

Summary of paper [30]

In this paper the authors are trying to identify hidden information in grid systems. **According with the authors, the grid broker can observe only the collective outcome of groups of agents and not their individual efforts**. That circumstances lead to hard identify low-performance agents in the network. The authors of this paper are trying to identify hidden information in grid environments and in second place they are trying to explain why they cannot be handled satisfactory by the existing mechanisms. Finally, the researchers are trying to develop and evaluate, through several experiments, the reputation-based mechanism that deals effectively with hidden information. The proposed mechanism preserves a reputation metric for each agent and **the authors update this metric with the use of several approaches such as the observation of collective outcomes**.

A grid virtual organization allows the aggregation of computational resources in multiple administrative domains into a single pool. Thus, users can lease for a certain time-frame bundles of software, CPU cycles, bandwidth, and storage space to execute computationally intensive tasks, according to a service level agreement (SLA) between the user and the grid service broker. The user, however, is unaware of the resources that execute his task. Thus, the grid service broker must select an appropriate set of available resources of the pool and offer them to the user as a bundle, henceforth referred to as group.

The authors are proposing a complementary mechanism to aid the broker estimate individual performance. The contribution of this paper has two parts. First, the authors classify cases of information asymmetry in grids and secondly they propose and evaluate several approaches for estimating the individual performance of agents based on collective outcomes and in the presence of ratings or not. It is stated that **"one customer's poor behaviour can affect the reputation of the cloud as a whole"**.

Furthermore, the authors consider an inter-domain virtual organization (VO), where many agents can offer their resources to simple users. **Workflow tasks are assigned to groups of agents**. At any given time of a task request, an agent i is either available (i.e., has available resources), with probability ai, or unavailable with probability 1 - ai. An agent i has an inherit performance capability, i.e., quality qi, which is private information. The individual performance outcome of an agent i is given by xi(qi). For a group S with size —S— whose members' performance capabilities are given by the vector x = (xi, . . .), the collective outcome of the group is a function g(x). A simpler approach is for the broker to select the group S of nodes so that the expected performance for the present task is maximized yet in an economic way. To solve either of the above optimization problems, the broker should be aware of the performance capabilities or of the relevant strategy of each agent. An alternative approach would be to give all agents a chance to be selected, while favouring the ones with higher reputation. Thus, **it is beneficial for the broker to employ a reputation metric that reveals each agent's expected individual performance**.

In addition, the researchers study how individual expected performance of agents can be revealed by associating to each agent i a reputation metric ri that is properly updated based on collective outcomes. **According to Beta aggregation, each agent's reputation equals the fraction of the "weighted number" of her successful service provisions over the "weighted total number" of her service provisions, with the weight of each service provision being a negative exponential function of the elapsed time**. The authors study two different cases of available information in a group of agents:

1. intermediate outcomes of the computations may be exchanged among the nodes during the collective computation.
   (a) In this case, the agents participating in a group that performs a certain task may have information on the individual outcome of some others and on the lowest performance threshold φi that each member i of the group S should meet for the collective performance to meet threshold φ. Therefore, nodes can rate the performance of others. Two different rating approaches the applicability of which depends on the amount of information available to the members of a group about other members: RATE ALL and RATE ONE.
   (b) **RATE ALL approach**; each agent j submits a rating vector vj for all other members i of its group to the broker. Then, the broker sums the rating of each of the members and updates its reputation, according to Beta rule.
   (c) **RATE ONE approach**; each agent rates the performance of another randomly selected member of its group and submits this rating to the broker. Again, the broker sums the ratings for each member of the group and updates reputations according to a majority rule.
2. no intermediate outcomes are exchanged among individual nodes, e.g. when individual outcomes are sent to a coordinating agent that composes the final outcome delivered to the user. For simplicity, in each case of a successful collective outcome, the authors assume that all group members have exerted high effort and increase the reputation values of all group members. Then, they propose the following approaches for updating reputation of group members after a failed service provision:
   (a) **Punish all equally (PUNISH ALL)**: Decrease the reputation of each agent of the group
   (b) **Punish probabilistically fairly to reputation (PROP)**: The reputation of a member i is updated but with probability

$$ri/\sum jrj$$

, thus, the group members share the blame of the collective failure according to their expected performance.

(c) **Punish probabilistically “inversely” to reputation (INVERSE)**: The reputation of a member i now decreases with probability

$$(1 - ri)/\sum j(1 - rj)$$

.

3. Furthermore, if the grid broker has some idea of how many agents did not perform adequately, thus leading the whole group to a failure, then there are more possibilities.

   (a) **Punish the worst M (WORST M)**: Sort the reputation values in descending order and decrease the reputation of the M members with the lowest reputation values.

   (b) **Punish random M (RANDOM M)**: Decrease the reputation values of M random group members as above.

Next, the researchers of [30] employ Bayes’ rule for updating agents’ individual reputation values based on their collective outcome. Their setting can be considered as a special case of a Bayesian network with star topology, where the broker is the hub node, and the agents are one hop away. This approach is applicable when there are specific fixed performance types of nodes with known success probabilities. They assume that there are two performance types of High (H) and Low (L) of agents that have success probabilities pH, pL respectively. Then after a failure collective service outcome (F), the reputation ri of the individual agent i is computed according to the formulas below.

$$u_j[i] = \begin{cases} 1, & if\ x_i \geq \varphi_i \\ -1, & if\ \ x_i < \varphi_i \end{cases}$$

$$r'_i = \begin{cases} \dfrac{\beta r_i t_i + 1}{\beta t_i + 1}, & if \displaystyle\sum_j u_j[i] > d \\ \dfrac{\beta r_i t_i}{\beta t_i + 1}, & if \displaystyle\sum_j u_j[i] < -d \\ r_i, \quad if \left|\displaystyle\sum_j u_j[i]\right| \leq d & \end{cases}$$

$$r'_i = \Pr[A_i\ \varepsilon\ H|F] = \frac{\Pr[F|\ A_i\ \varepsilon\ H] * \Pr[A_i\ \varepsilon\ H]}{\Pr[F|\ A_i\ \varepsilon\ H] * \Pr[A_i\ \varepsilon\ H] + \Pr[F|\ A_i\ \varepsilon\ L] * \Pr[A_i\ \varepsilon\ L]}$$

In conclusion, the authors have proposed several reputation-based approaches. If intermediate outcomes are exchanged among truthful agents, rating-based approaches are very efficient in identifying low-performing agents. Yet, if severe collusion arises among agents that do not belong to known performance types, then the simple deterministic PUNISH ALL approach can provide an effective solution to this information asymmetry problem, **provided that a large enough tolerance threshold is employed**. According to researcher's results, rating based and PUNISH ALL approaches are the most effective stable ones for estimating the performance of individual agents and provide the right incentives for exerting high individual performance. However, between the rating-based approaches, RATE ONE involves less communication overhead, but only RATE ALL is efficient in case where the performance distribution of agents is unknown. Therefore, in an actual service paradigm, a grid service broker should apply both PUNISH ALL and RATE ALL approaches, but initially only employ reputation values calculated with the PUNISH ALL approach for agent selection. If they differ less than a threshold, then the grid service provider should employ RATE ALL for calculating reputation values of individual agents for maximizing the success ratio of the VO in service provision; otherwise, the broker should continue employing PUNISH ALL.

3.3 Blockchain-based reputation systems

Summary of [42]

In this paper the authors are trying to propose a blockchain-based reputation system to alleviate problems which provoked by centralised technics. With this state-of-the-art method, the information about a specific product saved in IPFS (interplanetary file system). The IPFS returns the specific address that this information has been stored. Finally, the reputation score of that product (that has been calculated by combining the ratings of the users) is stored on the blockchain with the returned address that I mentioned before. This mechanism overcome risks such as malicious misconfiguration, fraud, forgery, malicious modifications of reputation ratings of users, insertion of fake scores etc. This is happening since is very difficult to take advantage over the blockchain mechanism and modify the blocks unless you have more than 51 percent of nodes under your control, something practically infeasible. The researchers are using solidity language at Ethereum platform to compose the necessary smart contracts. Several blockchain-based reputation systems for e-commerce environments have already proposed [41], [35], [19]. However, **this research focus on differences between the centralised and the decentralised approaches for the reputation management**.

The authors have designed smart contracts to perform the evaluation of the reputation in the blockchain. This study offers added value to our community since the proposed method is the first one that use blockchain environment with the implementation of smart contracts and simultaneously they use the IPFS for the online marketplaces. Contrary of the traditional centralised methods, that store in the server both the product description and that reputations rating, the proposed mechanism saves the product description in the IPFS. The IPFS returns the address of the product and with the reputation score, all together they stored on chain. More added value has the fact that the evaluation technique of the reputation can overcame on common attacks such as the malicious and fake feedback-comments by the users. In the blockchain the reputation rations are calculated by all the ratings of the transactions weighted by the practical transaction factors including transaction time, transaction amount, and previous reputation scores of uses. Finally, this mechanism offers an incentive mechanism that is absent in the centralised approached as I mentioned before. The monetary initiative mechanism pushes the users to submit their feedback and their scores of evaluations, something very important for the future consumes who will be able to choose products value for money. The one aids the other and the consumers have finally incentive to contribute at the evaluation procedure.

This paper proposes a blockchain-based distributed reputation system and uses Ethereum platform for deploying smart contracts. It is also uses IPFS as descripted in the annex. The authors also propose a reputation evaluation mechanism that considers the transaction time, the transaction amount and the reputation values of the users. In addition, a reputation rating mechanism and simultaneously a monetary incentive technic are implemented via the implementation of a smart contract in the Blockchain environment.

System architecture:

There are six main entities in this paper. First, there are the sellers who are the product owners and can score the buyers to user Interface after each transaction. Second, there are the buyers who also can submit their feedback for the seller to user Interface after each transaction. Third entity is the UI itself. It is developed on IPFS and permits interaction among those different entities. It can present the product information and the reputation scores to consumers, to distribute data on IPFS or on chain and to trigger a smart contract. Fifth entity is the smart contract which is simply an executable program as described in the annex. It calculates the reputation and saves it on chain. The sixth and last entity in the blockchain.

The architecture has designed as follows; she consists of two stages that transaction stage and the evaluation stage. The transaction stage consists of two sub-stages: uploading products by sellers and purchasing products by buyers. Also, the evaluation stage consists of two sub-stages: evaluation by Sellers and evaluation by Buyers. The Sellers and Buyers submit their comments and ratings in the two sub-stages, respectively.

1. The buyers after a transaction submit their feedback for the products that purchased and the score for the sellers to UI.

2. User interface sends all this information to IPFS and the IPFS returns a unique address to UI.

3. After this, the UI triggers the Smart Contract.

4. Smart contract rewards the buyers for submitting their feedback in order to give them incentives for future evaluation of sellers. When both consumers and product owners have successfully submitted their comments and scores to UI, smart contract calculate and update the reputation metrics of consumers and product owners and as final step it broadcasts the reputation score on chain.

5. Blockchain saves the reputation and broadcasts the successful evaluation to UI.

Same procedure is for the evaluation by the sellers. The reputation evaluation scheme and monetary incentive mechanism are automatically implemented by the designed smart contract during the evaluation stage. Thus, in the proposed BC-DRS, the reputation score of a user is computed and updated by all the ratings given by the others who have transactions with him. The incentive mechanism here, is implemented to send a certain number of monetary rewards. Thus, in the proposed BC-DRS, the reputation score of a user is computed and updated by all the ratings given by the others who have transactions with him. After a transaction between a buyer and a seller, they will give a rating about the transaction to each other. Thus, for the user, **they weight all the received ratings by the three following factors to compute his reputation score**.

1. Transaction time: If the consumer byes the same product numerous times from the product owner in a short period of time, it is possible that they exchange grate scores to each other. So, for a current transaction, if the transaction has occurred previously a short time ago, the rating of current transaction should be set as a relatively low value to deter the collusion attacks. Thus, for the rating, the weighting factor of transaction time is defined by (T) = tanh (T)
2. Transaction amount: It is not costly for the buyer to purchase many products with low prices from the seller to unfairly submit low ratings to the seller. Therefore, to alleviate the problem of unfair ratings, the rating of a transaction should be related to the transaction amount. So, for the rating, the weighting factor of transaction amount is defined by ψ (A) = ln (1 + A)
3. Previous reputation scores of users; Since the honest sellers or buyers usually submit more fair ratings than the malicious ones to others, the rating of a transaction should be also related to the seller's or buyer's previous reputation score. Thus, the previous reputation score of the user is also used as the weighting factor of the rating. Where, it is initially set as 1 for all the users.

Positive rating r+ or negative rating r are calculated by:

$$e+ = \phi(T) \times \psi(A) \times RSpre \times r+$$

$$e- = \phi(T) \times \psi(A) \times RSpre \times r-$$

1. e+ and e are the weighted positive rating and negative rating, respectively.
2. Transaction time: for a current transaction, if the same transaction has occurred previously a short time ago, the rating of current transaction should be set as a relatively low value to deter the collusion attacks. Thus, for the rating, the weighting factor of transaction time is defined by (T) = tanh (T)
3. Transaction amount: it is not costly for the buyer to purchase many products with low prices from the seller to unfairly submit low ratings to the seller. To alleviate the problem of unfair ratings, the rating of a transaction should be related to the transaction amount.
4. positive rating r+ or negative rating r

Then, for the user, separately sum up all the weighted positive ratings and negative ratings given by others.Then, the reputation score of the user can be obtained.When the user has a new transaction and receives a new rating, its reputation score can be updated.

According to the computed reputation score of a seller, a buyer can make a trust decision to purchase the products from the seller or not. The incentive mechanism is calculated by the smart contract that sends an amount of reward from the seller to the buyer or vice versa. To encourage honest feedback, the monetary reward should be also related to the three factors:

$$MR = \phi(T) \times \psi(A) \times RSpre \times m0$$

The designed smart contract consists of three algorithms: product uploading algorithm, product purchase algorithm, and reputation evaluation and monetary incentive algorithm. To check their results, they perform some experiments by using the client application Geth to build the test private blockchain. They also deploy the proposed BC-DRS on the Ethereum blockchain and implement the experiment on the Ubuntu 18.0.4 operating system with Visual Studio Code and Solidity v0.5.0 language. To sum up, the results saw that the proposed model can protect the product information and reputation scores from modification. Finally, it can effectively prevent the unfair rating attack and resist the collusion attack.

Summary of paper [14]

The authors of this paper propose to group this agent in the miscellaneous IoT environments by considering their social capabilities. The researchers are deploying a reputation model for each agent, then they represent an algorithm that develops those groups of agents by considering their reputation capital and finally they utilise the blockchain technology to certify this reputation capital. In this approach a significant observation is that as the number of positive interactions is augmented among the nodes of a group of agents, is augmented in parallel the effectiveness of that group/ the social capital of that group. The researchers are trying to solve the problem of representation of trustworthiness metric in an IoT environment.

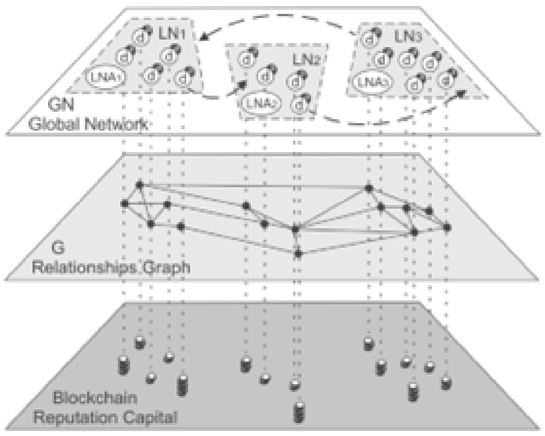
The problem that remains present is that agents must interact with other agents in other networks whose reliability is unknown. Global reputation does not appear as a solution to the above problem since is a metric difficult to obtain and spread without the use of centralised methods/repository as this metric needs to consider the previous/past interaction among agents and it should have high accessibility and reliability in all environments without distinction. The authors are trying to avoid the use of the central repository by providing a model reputation system **"reputation capital" that is occurred by aggregating feedback that received during the interactions between agents**. The decentralised approach that they use is based in blockchain protocols in order to validate the Reputation Capital of the agents. Finally, they present an algorithm for grouping the different agents in a trustworthy way. The information that is needed to group then it is stored in the blockchain. As a last step, the authors are testing their system in a mixed environment with both honest and malicious agents, and they prove that their framework is resilient to malicious behaviours. The combination of reputation metric, formation of groups, IoT environment and blockchain framework forms a valuable solution to the problem of migration of agents across domains based in their Reputation Capital for joining in groups.

In reputation systems some properties should be in order:

1. Entities must be long lived. This is very important as previous behaviours gives valuable information to provision the future behaviour of a node.
2. decisions about new interactions should be driven by past experiences.
3. ratings about current interactions need to be gathered by providing incentives to give initiative to agents to submit their feedback into community.

Nowadays, a large proportion of experts are starting to prefer blockchain technologies as they offer significant opportunities as easy sharing of resources among IoT devices [42], [41], [35], [19], [14]. It also, **guarantee data integrity, confidentiality, and availability thus, trust among unknown entities by using cryptographic technologies to identify source and sink of the data as mentioned and in the**

Annex. As it is mentioned analytically in the annex, the blockchain is composed of many blocks chronologically ordered and replicated on several distributed peer-to-peer nodes. All information is stored in ledger in an encrypted manner, verified by consensus mechanisms. When a block is validated by consensus is became integrated part of the chain.



The proposed framework is based on a large proportion of heterogeneous IoT devices that are assisted by software agents.

1. D = the set of devices
2. A = the set of software agents, each one associated with a unique, personal device.
3. GN = the global network composed by a number n ¿ 1 of federated local networks (LN)s.
4. A trusted and equipped agent called local network administrator (LNA) will correspond to each LN and provides all the agents temporarily living on its administrated domain of some basic services.
5. They represent the set of agents A registered on GN and their relationships by means of a graph G, where N represents the set of nodes belonging to G and each node of N is associated with a unique agent a A, while L is the set of oriented links, where each link of L represents a relationship occurring between two agents.
6. Within the single LN, each agent can form groups based on its RC witnessed by the blockchain. Agents can freely move from an LN to another LN and are free for joining with a group active on their current LN, based on their convenience.
7. each group of an LN is coordinated by the respective LNA that, to maximize the effectiveness of the group itself, can also contact other devices (i.e., agents) to join with or remove from the group of agents having an inadequate RC.

The use of blockchain helps the RC scored to be trusted over the GN. The RC is represented by a score obtained by the past interactions-behaviours of the with the other devices belonging to GN when it acts as a provider of qualified interactions. The RC of an agent is assumed to be a real positive number where "high" values indicate a "good" reputation. **Each new agent receives an initial RC, which should not penalize too much a newcomer, and at the same time, it represents a countermeasure over whitewashing strategies of malicious agents aimed to return into the system for receiving a new initial favourable RC**. On the other hand, when the feedback assumes a value equal or greater than 0.5, this contribution is considered as part of a qualified interaction, if and only if the value of the relevance is greater than the feedback itself. Each time a service is provided into a LN, it is committed by means of a smart contract, running on the blockchain platform. The smart contract must have some steps devoted to updating both the RC of the provider and a list of data.

In this paper the researchers considered a scenario comprising a wide IoT network federating several environments of heterogeneous, smart IoT devices. In the considered scenario, devices can move across local networks, and they cooperate to reach their own goals with their peers. Information as reputation can help in that choice, assuming that information about reputation is spread in a proper way. To this purpose, they designed a framework where every IoT device was associated with a software agent capable to exploit its social attitudes to cooperate as well as to form complex agent social structures, as groups. To support cooperation, they use the RC which is updated based on the devices' feedback by exploiting the support of the blockchain technology. Moreover, based on the RC values, each device can decide of asking the affiliation to a group of reliable agents with the expectancy of having satisfactory interactions and economic advantages. They designed, a suitable RC model that implemented some countermeasures against malicious behaviours aimed to gain unfair RC and a distributed group formation algorithm, driven by information about the RC of the agents that provided to divide the agents in groups based on their RC score. The experimental campaign of simulations carried out to verify efficiency and effectiveness of the proposed IoT framework highlighted that the synergy derived from the RC model, the group formation algorithm, and the blockchain allows the framework to work correctly.

Summary of paper [13]

In this research the authors target to find the best way for the agents to choose partners in a federation network. More precisely, they propose a well structure reputation model that also detect malicious nodes that are not reliable, or they reveal misleading behaviours. Last but not least, they developed a group formation algorithm that support's the aforementioned reputation model. Finally, according to numerous experiments it is revealed that their proposed model has a large proportion of advantages to Internet of things ecosystem. It is important to mention that any type of group formation must respect the characteristics of the Internet of things network. This work tries to extend the [14] which introduce a way to use reputation to give to devices the opportunity to migrate from one IoT environment to another and cooperate with reliable agents with high probability. The authors extend the reputation capital (RC) by adding the personal capital (PC) metric. The personal capital metric represents the aggregation of the feedback of the nodes during the interaction with other nodes in the network. As in [14], the PC is stored in the blockchain technology to be sure that the preservation of confidentiality, integrity, and availability of the information through decentralized-distributed peer-to-peer environments.

After a thorough analysis of [28] a reputation model should respect the following properties:

1. Entities must have a strong background of experience so that they can predict future behaviours and prevent whitewashing strategies.
2. New interactions must be based on past experiences.
3. Entities must grant and spread their feedback in the scenario.

Generally, the reputation scores in distributed environments can be propagated with several ways. First way is with distributed and synchronized repositories and second with data which are collected from feedbacks of nodes that have interactions with other nodes in the network. As it is described in the annex the Proof of Work initially adopted by the Bitcoin currency demands to resolve a computationally expansive hashing puzzle for making valid and adding a new block. The application of the Proof of Work in an Internet of Things context often requires the adoption of other technologies such as the cloud computing. In the literature, miscellaneous types of blockchains have been proposed that allow to propagate trust and reputation scores without the use of trusted and powerful third-party. For example, Islands of Trust, proposed in [32], that consents to spread trust across different IoT domains by using two blockchains. The proposal of the authors has three important differences with the paper [14]:

1. the adoption of a reputation system based on the reputation capital score
2. they use a single blockchain
3. they use smart contracts

The scenario is the same with the paper [14]. The researchers have N heterogeneous Internet of things objects, each of them supported by a software agent. The different agents interact with each other based on smart contracts. In the model the reliability of the reputation model is guaranteed by the blockchain, while the reputation capital of a provider witnesses its ability to provide quality service. The Reputation Capital represented by a numerical value computed based on the past interactions among agents on the basis of the following requirements:

1. the more recent the activities, the better the weight of the feedback
2. countermeasures should be taken against collusive behaviours among agents (i.e., behaviours aiming at mutually and incorrectly increasing their reputation capitals)
3. alternate behaviours should be hindered, by assigning a feedback value which indicates a "bad" behaviour
4. the behaviour of "habitual" complainers should be limited by measuring their "credibility"

An initial value of reputation capital is introduced as a countermeasure that aim at coming back in the system to take a new reputation. Moreover, the value should be chosen in order not to penalize a newcomer. When the provisioning of a service is required to an agent αi, this latter provides feedback representing the satisfaction of αj for the service received from αi. The authors put a threshold equal to 0.5 to distinguish "bad" and "good" feedbacks. To calculate RC, they consider only the interactions that are classified as qualified: to this end, whenever a provider agent αi releases a service s to a consumer agent αj the latter assigns feedback to express its own satisfaction. If the interaction is "qualified" then feedback is used to update the RCi of αi. Therefore, the Relevance R of an interaction is calculated as cs/C if cs ¡ C, otherwise it is set to 1. The parameter cs is the cost of a service s and the term C represents the price threshold at which the relevance is computed at its maximum value 1. The previous definition of relevance allows us to give the further definition of "qualified interaction": an interaction is qualified if (i) φ ¡ 0.5 or (ii) R φ 0.5. The ratio behind this last definition is to discern interactions having values φ 0.5 and, at the same time, a value of relevance R ¡ 0.5. Based on the last qualified interactions of the provider agent αi occurred with different agents (a measure aimed to contrast collusive behaviours).

Into a LN, the request for a service is equivalent to run a smart contract on the blockchain platform to check and implement all the contractual obligations. Even though the current proposal is independent from a blockchain. They propose to exploit the Ethereum platform for convenience: (i) the presence of documented API; (ii) the availability of an own cryptocurrency (i.e., Ether) to pay in GN and (iii) the availability of several simulation tools to create a personal local Ethereum blockchain for testing and development. The smart contract encloses the necessary steps to update both RC of the provider and the following information about:

1. the identifier of the agent who is associated the reputation capital and the identifier of its LN.
2. the number of interactions of the agent (i) successfully completed, (ii) aborted, (iii) with low feedback (i.e., φ ¡ 0.5), (iv) with no feedback
3. the list of the q transactions concurring to form the current RC, where each row is a tuple composed by: – the identifier of the counterpart agent and its LN
4. the price and the date of the transaction.
5. φ assigned to the interaction.

Each LNA administrates a private list of the agents currently living in the LN, and saving the actual RC, a timestamp and the reference on the blockchain that link to where the interesting information is stored. Also, when an agent starts an interaction, it receives a certificate signed by a private key that contains all the information mentioned above.

To this end, the authors consider that each of the N objects of the IoT community has its reputation capital RC, and thus the problem to solve can be viewed as that of partitioned a set of N objects into ng groups in the best way from the viewpoint of the homogeneity of each group. In their approach, to solve this problem they are applying the well-known K-means clustering algorithm, that has the goal of minimizing the total variance intra-cluster. Therefore, the input of this phase is the vector composed by the N values of reputation capital of the IoT objects, and the output will be the number of determined clusters, that they use as the value ng in the next phase, and the minimum reputation capital of each cluster, that they use as threshold value in the next phase. In our proposal, they have decided to adopt the classical approach of Calinski and Harabasz, that proposes to choose the value of k which maximizes an index.

The parameters computed in the first phase represents only a starting point of the designed strategy. Indeed, the reputation capital of the agents will change in time, therefore the procedure aimed at determining parameters ng. They choose to maintain the parameters computed in the phase 1 (ng and ni), and to periodically execute the second phase of the algorithm, which dynamically moves an object from a group to another based on the varying value of each individual reputation capital.

The level of satisfaction of the single device must be high at the end of interactions, therefore it is vitally important to select reliable collaborators. In this case, the reputation is a measure that can help in that choice. For this reason, the authors present an approach where a software agent is associated with an Internet of Things device to use its social attitudes to collaborate to perform groups. To ensure reliable partnerships, they define the reputation capital, a score that depends on the devices' feedback. The authors utilize the blockchain technology to propagate the reputation in a decentralized manner. Moreover, based on the reputation capital scores, every device can expect satisfactory interactions and economic advantages if it associates with groups of reliable agents. They introduce;

1. an appropriate reputation capital model that develops the countermeasures against collusive and malicious behaviours and
2. a distributed group formation algorithm that subdivides the agents in groups considering their reputation capital score.

Finally, they carry out experiments to check efficiency and effectiveness of the presented solution. The results highlight how the synergy obtained by the combined adoption of the reputation capital model, the group formation algorithm along with the blockchain provides benefits to the agents operating in the IoT environment.

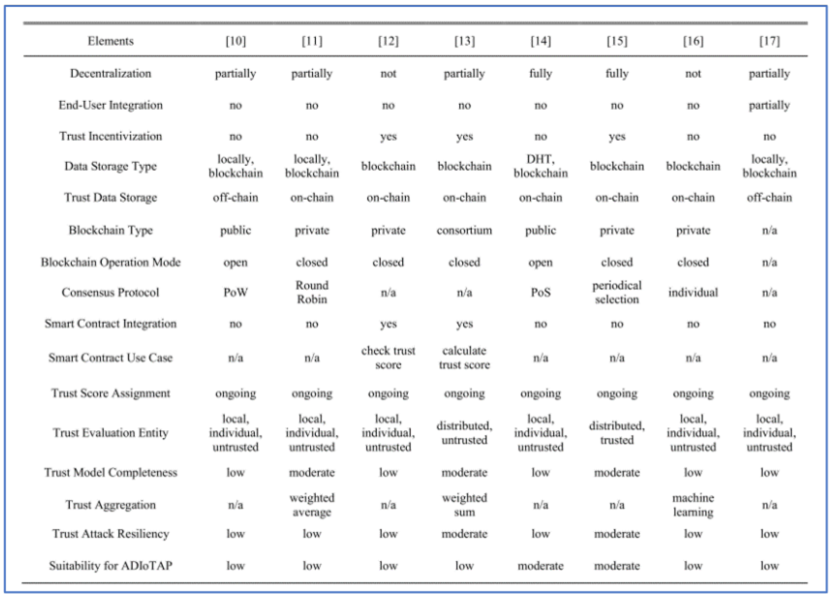
Summary of [5]

This paper is trying to illustrate the strengths and the weaknesses of various blockchain-based trust approaches in Internet of Things community. In addition, the authors proposed a new trust model by using a variety of trust metrics. This publication is a state of the art as it represents incentives in IoT marketplaces by using control loops and, simultaneously, smart contracts enhancing the participants to improve their behaviour inside the community.

Nowadays, the blockchain environment has stimulate the interest of the researchers to integrate its features in trust management processes within IoT environment. However, the blockchain architecture is not suitable to be integrated into IoT systems due to security issues, such as the overall trustworthiness of the participating nodes. In addition, limitations regarding the consensus-building in the network need optimization and adaptation to the strict specifications of the IoT community. Finally, decentralized issues should be considered, and trust-building processes should be merged in the blockchain processes to maximize the benefits of blockchain architecture.

The main contribution of this paper is the proposal of an optimized and blockchain-based trust approach considering strengths and limitations. Moreover, this publication presents a blockchain-based trust evaluation process using a lightweight and trust-based consensus protocol for decision making in the P2P network. More precisely, it presents multiple trust metrics and accordingly describes their mathematical models and finally, introduces a novel concept of combining control loops, blockchain and trust to motivate good participation of service providers in the network by using smart contracts.

Traditional trust management approaches have many drawbacks some of them are the following: insecure data storage, trust information can be manipulated and misused by malicious peers, centralized entity limits autocracy behaviour in a network and is vulnerable to single point-of-failure issues. The blockchain provides the possibility, through cryptographic principles and related consensus protocols, to securely store information in its ledgers and to increase the integrity level. Smart contracts in combination with blockchain enable the automation of processes in a network without the need of a coordinator. The combination of blockchain and TMS enhance the overall privacy, security, and trust level.



Elements	[10]	[11]	[12]	[13]	[14]	[15]	[16]	[17]
Decentralization	partially	partially	not	partially	fully	fully	not	partially
End-User Integration	no	no	no	no	no	no	no	partially
Trust Incentivization	no	no	yes	yes	no	yes	no	no
Data Storage Type	locally, blockchain	locally, blockchain	blockchain	blockchain	DHT, blockchain	blockchain	blockchain	locally, blockchain
Trust Data Storage	off-chain	on-chain	on-chain	on-chain	on-chain	on-chain	on-chain	off-chain
Blockchain Type	public	private	private	consortium	public	private	private	n/a
Blockchain Operation Mode	open	closed	closed	closed	open	closed	closed	n/a
Consensus Protocol	PoW	Round Robin	n/a	n/a	PoS	periodical selection	individual	n/a
Smart Contract Integration	no	no	yes	yes	no	no	no	no
Smart Contract Use Case	n/a	n/a	check trust score	calculate trust score	n/a	n/a	n/a	n/a
Trust Score Assignment	ongoing	ongoing	ongoing	ongoing	ongoing	ongoing	ongoing	ongoing
Trust Evaluation Entity	local, individual, untrusted	local, individual, untrusted	local, individual, untrusted	distributed, untrusted	local, individual, untrusted	distributed, trusted	local, individual, untrusted	local, individual, untrusted
Trust Model Completeness	low	moderate	low	moderate	low	moderate	low	low
Trust Aggregation	n/a	weighted average	n/a	weighted sum	n/a	n/a	machine learning	n/a
Trust Attack Resiliency	low	low	low	moderate	low	moderate	low	low
Suitability for ADIoTAP	low	low	low	low	moderate	moderate	low	low

The existing trust approaches in IoT [33], [26], [3], [7] provide facts regarding the combination of blockchain and trust to enhance the credibility of services. Blockchain-based factors start with the trust data storage, defining whether trust information is stored in the blockchain (on-chain) or outside (off-chain). Moreover, the consensus should consider the lightweight characteristics of IoT devices. Additionally, an emerging protocol in this context is a smart contract, used to automate processes based on a contract between participants without third parties.

The trust evaluation layer model is composed by 5 layers: Entity Layer, Evaluation Layer, Metric Layer, Score Layer and Storage Layer. One key aspect of the holistic trust model is the possibility to evaluate the trustworthiness of new services or new end-users joining the IoT community. Thus, the holistic trust model integrates blockchain for optimizing the storage system and to ensure tamper-proof trust data.

Trust Metric	Trust Sub-Metric	Symbol
Service Testing	Functional testing	S_{ft}
	Response time	S_{ptnrt}
	Service acceptance	S_{ptar}
Service Monitoring	Service uptime	S_{mtavt}
	Service online/offline actions	S_{mtava}
	Number of times a service is used	S_{mtacn}
	Ratio positive responses	S_{mtar}
Service Rating	Service satisfaction	S_{ratint}
Peer Task Participation	Participation in tasks	S_{tp}
Peer Integrity Checking	Service or Trust Information Integrity	S_{inch}

For each of the trust metrics there are specific trust sub-metrics defined and used for the trust evaluation. The table shows the different sub-metrics and their respective symbol used for the mathematical model.

1. Service testing – here the service capabilities including service functionality and service performance are tested.

2. Service monitoring – this is happening continuously during the lifetime of a service. The service monitoring is done by other end-users' part of the IoT community. Service monitoring includes some metrics from the performance such as the availability and the activity.

3. Service rating - other end-users have the possibility to rate a service based on their own experience.

4. Peer Task Participation – here the effort of end-users for participating in different community tasks is measured. Moreover, the participation in blockchain tasks is also part of this metric.

5. Peer Integrity Checking – considers checking the integrity of service and trust data by comparing information in the P2P overlay and the blockchain.

User in the Loop (UIL): where the user is part of a control loop and motivated to change the location to optimize the signal to interference noise ratio in wireless cellular networks. The basic idea is to incentivize/motivate the user towards a specific behaviour.

This publication proposes to integrate the UIL concept to the trust paradigm. The proposed control loop is called Trust in the Loop (TIL). The control loop contains a target trust score which must be achieved by the service provider and the service it is providing. To do so, the Trust Unit will look-up in the blockchain for the current trust score of the service provider or its service. The current trust score is compared with the target score and if it is below, the Trust Unit will set incentives for the service provider. The service provider then decides whether to provide a better service to get the benefits promised by the Trust Unit. The outcome of the revised service is the service behaviour which will trigger the initiation of a new trust evaluation of the service provider and the service in a feedback loop. The TIL concept can also be applied to service providers or to other relevant tasks in the IoT community. The activities of the trust loop are realized completely autonomously through smart contracts.

The evaluation results of the different trust metrics need to be aggregated to an overall trust score of the end-user. As mentioned, the overall trust score consists of the scores derived from service testing, service monitoring, service rating, peer integrity checking and peer task evaluation. Thus, it is important to assign a weighting system for the different trust metrics as they have a different impact under different conditions on the overall trust score of a service or service provider. Therefore, this publication proposes to combine different aspects to present a dynamic weighting system which enables efficient trust aggregation and automatically weighting adjustment based on the current situation in the community.

To compute the overall trust score, the current one should also be combined with the old one. Therefore, both scores are weighted according to the average peer trust score of each block (last block and current block). The following equation expresses the overall trust score of a peer:

$$\boxed{T_{total} = \frac{1}{n}\sum_{i=1}^{n} T_{p_i}^{old} \times T_{total}^{old} + \frac{1}{m}\sum_{i=1}^{m} T_{p_i}^{cur} \times T_{total}^{cur}}$$

To sum up, the paper [5] proposes to combine blockchain and trust with control loops to introduce a novel concept which optimizes the security of the ecosystem by incentivizing low-trust peers to improve their behaviour in the community. Finally, the performance of the trust model is demonstrated, and different experiments are carried out. The experimental results show that the proposed trust approach outperforms in terms of resiliency and reliability in comparison with existing ones.

4 Our mechanism

4.1 Transition

Somebody may start wandering why it is so important to have trust and reputation systems? Wouldn't be more wise to count the quality of a product based only on objective metrics that we capture from the platform? Isn't it more dangerous to trust reputation scores of the consumers? What if a consumer rates maliciously low a product?

The reputation mechanism, even if is more complicated, is solves the hidden quality problem. We cannot systematically and adequately count the quality of data products like the accuracy, the completeness, the consistency, the timeliness, the relevance, the usability, the validity, etc. As a result, We need the subjective feedback of consumers to capture the value of metrics like the ones that we mentioned before.

Although, there are some quality metrics that we can capture in an objective manner. We can assess to QoS of IoT devices and data delivery based on a set of technical KPIs like the availability, the load, the coverage, the latency, the reliability, etc. In order to capture those quality score we should make some assumptions:

1. each Provider (platform) maintains its own monitoring system
2. no trust that Providers truthfully report their performance
3. Internal Consumers falsely report low performance
4. External Consumers a cannot monitor the performance themselves

We need collective monitoring or other measures to assess the trustworthiness of measurements.

So, in final analysis the reputation of a product will be the combination of subjective score that is been given by the consumers (that gives us a picture of the quality metrics that we cannot capture in an objective way), and the objective score that is been collected during precise monitoring periods.

The first step was to understand which information is observable and which is unobservable. We should identify the hidden information problem. To start with, we have observable context if the data consumers can see the reputation quality score of the data sources. In our case, the data consumers cannot have atomic estimations, so it is unobservable context. As a result, we cannot use methods, in order to update the reputation scores, as the 'rate one' or the 'rate all' approaches. However, we can use other pioneer methods in order to update the reputation, such as the 'punish all' or the 'punish worst m' approaches that we will analyse in detail in the next subsection.

At this point, we should highlight, that the logic of the data providers is to promote the less good services that they provide, in parallel with the good services that they offer. So, if a provider has a good reputation score, and the users trust him and want to collaborate with him, then the provider can exploit that trust and ’sell’ the services of ambiguous quality simultaneously with the good quality ones. From the above analyzation, we can conclude that in this case the hidden information is the quality of the services. So, it is reasonable to try to desegregate the reputation of the products to the atomic reputation of each resource that participate in the formation of that particular product. In this way, the hidden quality problem will be alleviated as the user will be able to see the reputation score of each resource and understand if it has good reputation score or not in order to collaborate with it and trust it.

Moreover, in order to desegregate the reputation score of the product in each resource that participate in its’ formation we will use the mathematical approach and the conclusions of paper [30]. In the next chapter we analyse in detail all the mathematical approach and we fit them in our mechanism that is described in the previous subsection.

4.2 IoTFeds Reputation and Trust System Architecture

Reputation and trust systems are tools for assessing the quality of the services provided, when the relevant information is incomplete or asymmetric. They also provide incentives to Service Providers (SPs) and Consumers, so that they behave as expected and do not intend to abuse the system. In the IoTFeds platform we introduce and leverage the concept of reputation on multiple levels. However, in each of them, the reputation score aims to reveal a ”hidden” information about the provided service quality (hidden quality). The different reputation levels are summarized below:

1. Data Sources: Each data source is characterized by a degree of reputation. A data product essentially consists of multiple data sources. Thus, the degree of reputation of data sources can be leveraged when packaging a data product in a federated marketplace, in order to select data sources that achieve the required or best quality.

2. Data Products: Each data product is characterized by a reputation score, which results from the summation of the reputation scores of the data sources that make it up. The mechanism for calculating the reputation of a product based on the reputation of the sources composing it is studied next. A product's reputation may be leveraged for pre-built products provided on the IoTFeds global or federated marketplaces to facilitate the Consumer's product selection.

3. Data Providers: Each Data Provider is characterized by a reputation score, which results from the summation of the reputation scores of the data sources it provides. The mechanism for calculating the reputation of a Provider based on the reputation of the data sources it provides is studied next. A Provider's reputation score can be used as a criterion for accepting or rejecting their membership in a federation. Also, within a federation, the reputation of the Federation's member Providers could be leveraged for its governance. In particular, Providers with a higher degree of reputation than other members of the federation could have more influence on its decisions.

4. Provider Federations: Each federation is characterized by a reputation score which results from the aggregation of the data sources that the Data Providers have made available to them. The mechanism for calculating the reputation of a federation based on the reputation of the data sources subscribed to it is studied next. The reputation score of federations can be leveraged by a Provider/Consumer when choosing a federation to join.

Note that a common mechanism is likely to be followed to calculate the reputation of products, providers and federations based on the reputation of data sources.

To start with, the reputation of the Federation is produced by its members - the Data Providers. Generally, we want a total good reputation score that reflects the quality and the trustworthiness of the Federation. Instead, we would have to calculate the reputation of each member of the Federation to bring decisions, something very time-wasted and costly. That metric enhances us with many advantages. For example, the members with low reputation can be distinguished easily and fast and we can reject them from the Federation or, similarly, members may join the federation if they obey in specific rules (e.g., have reputation score ¿ 2). This idea can make our system more robust.

Another concern that we should consider is that a Data Provider wants with the good resources to promote and the lower quality ones. So, the hidden information in that case is the quality of the Data Sources that is provided by the Data Providers.

The first step in our research is to distinguish the different levels of scoring in IoTFeds Marketplace and to transfer those reputation scores from one level to another by updating the reputation metrics of each layer.

More precisely, as it is depicted in figure 1, the data consumer that interacts and purchase products from the marketplace must have the incentive to evaluate a product. It is undeniable that this rating will be **subjective** as the consumer has his own standards and expectations for the product that he selects to buy. However, the reputation score of a product does not consists only of the subjective rating of the consumer but also by **objective rating** from the data sources. To put it simply, the total reputation metric is the aggregation of both subjective evaluation of the consumer and the objective evaluation of the data sources. Additionally, it is important to mention that objective metrics may be the availability, the coverage, the latency, the accuracy, the completeness, the consistency, the timeliness, the uniqueness, the validity and the continuity. It is obvious that we have a variety of objective metrics and we should evaluate them and determine which of them are relevant in our case in order to be produced by the platform. Also, when we calculate the updated reputation score of the product, we should also consider the previous-old score of the specific product. Moreover, we should also have into our minds that to aggregate reputation scores, those scores should have same or similar meaning. Lastly, we should guarantee somehow the honesty of the subjective votes that evoked by the consumers. This could be done via the consensus by using the blockchain mechanism and the smart contracts (e.g., each vote to be a new transaction). All this considerations should be take into account when we will calculate the reputation metric of the product (steps 2a and 2b).

It is obvious now, that a product is composed of a variety of data sources. So, the collective reputation score of the product should be split to its' counterparts so as every data source acquire its own reputation score (step 3).

Additionally, each data source belongs to one Data Provider. Of course, a Data Provider may provide multiple Data Sources. So, as the reputation scores of the Data

Sources are updated, so has the reputation scores of the affected Data Providers (step 4a).

Now, each Data Provider belongs to a federation. So, as the reputation scores of the Data Providers are updated, so has the reputation scores of the affected Federations (step 4c).

We should not forget to mention that a data source that is used for the deployment of a product x, may be used also for a product y. So, as the reputation score of a specific data source is updated so has the reputation scores of the affected products that are composed of the same data source (step 4b).

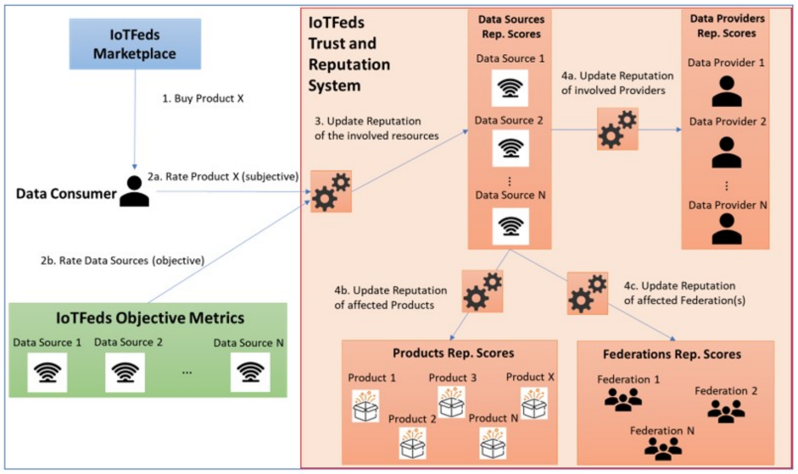

IoTFeds Marketplace
1. Buy Product X
Data Consumer
2a. Rate Product X (subjective)
2b. Rate Data Sources (objective)
IoTFeds Objective Metrics
Data Source 1
Data Source 2
...
Data Source N
IoTFeds Trust and Reputation System
3. Update Reputation of the involved resources
Data Sources Rep. Scores
Data Source 1
Data Source 2
Data Source N
4a. Update Reputation of involved Providers
Data Providers Rep. Scores
Data Provider 1
Data Provider 2
Data Provider N
4b. Update Reputation of affected Products
4c. Update Reputation of affected Federation(s)
Products Rep. Scores
Product 1
Product 2
Product 3
Product N
Product X
Federations Rep. Scores
Federation 1
Federation 2
Federation N

Figure 1: IoTFeds Reputation and Trust System Architecture

Figure 1 shows the general architecture of the reputation and trust system of the IoTFeds platform and the steps/processes performed by a Consumer rating a product:

1. A Consumer purchases a data product through IoTFeds' universal or federated marketplace. Through this transaction, the Consumer receives a token that can be used to evaluate the product.
2. Evaluation of the product
    (a) By using the coupon, the Consumer rates (subjectively) the level of service provided by the product.
    (b) Additionally, objective metrics (eg, availability, processability, etc.) from the data sources composing the product are forwarded to the reputation and trust system.
3. The mechanism of this step leverages the user's subjective evaluation and objective measurements on the part of the IoTFeds platform to calculate a rating per data source that composes the product. Afterwards, the scoring per data source

is used to update the reputation level of each source, taking into account their previous reputation level. This mechanism will be presented in detail below.

4. Refreshing the reputation scores of the involved data sources ”triggers” the refresh of the reputation scores of the Providers (4a), products (4b) and federations (4c) affected by the relevant data sources. The mechanism used for the three cases above is studied below.

4.3 Computation of Reputation as a whole

Having analysed the whole procedure of the sequentially updates, next, we must study how individual performance of data sources can be revealed by associating to each data source i to reputation metric ri that is properly updated based on collective outcomes of the products (figure 1 step 3). We recall that a product is composed of several data sources, so the collective reputation score of the product should be transferred individually to each data source. The whole study that is represented is based on [30].

More precisely, we study how update can be done under both the Bayes and the Beta aggregation rules, according to which ri expresses the probability of successful individual performance of data resource i (how good the reputation of a data source is) as in [30]. According to Beta aggregation, each data source's reputation equals the fraction of the "weighted number" of its successful service provisions over the "weighted total number" of its service provisions, with the weight of each service provision being a negative exponential function of the elapsed time. We also attempt to study other update approaches based on arbitrary scoring metrics that also lead to the ordering of data sources according to their performance. **We study the one (case 2) out of the two different cases of available information in a group of data sources as in [30]** :

1. intermediate outcomes of the computations may be exchanged among the nodes during the collective computation of the product.
2. no intermediate outcomes are exchanged among individual nodes (e.g., when individual outcomes are sent on chain that composes the outcome delivered to the user).

In the case [1], the data sources participating in the composition of a particular product that performs a certain task may have information on the individual outcome of some others and on the lowest performance threshold φi that each member i of the Product P should meet for the collective performance to meet threshold φ. Therefore, nodes can rate the performance of others. In our mechanism we do not have intermediate data as the rating of a product comes from the customers (Data consumers).

The exchange of intermediate outcomes among data sources cannot be assumed in certain cases like this one. The consumer rates the product externally. The smart contract takes subjective and objective scores as an input and calculates the reputation score update for each data source involved. The product reputation score is finalized and is calculated considering the data sources that participate in the formation of that particular product.

If no intermediate outcomes are available, case [2], the smart contract must infer individual performance based on collective outcomes only. For simplicity, in each case of a successful collective outcome (product), we assume that all group members (data sources) have exerted high effort and we increase the reputation values of all group members. We propose the following approaches for updating reputation of data sources after a failed service provision:

1. Punish all equally (PUNISH ALL): Decrease the reputation of each data source that participate in the composition of product P as in the middle case of (Equation 2).

2. Punish probabilistically fairly to reputation (PROP): The reputation of a data source i is updated according to the middle case of (Equation 2) but with probability $r_i / \sum_j r_j$, where the j's are all group members. Thus, the group members (the data sources) share the blame of the collective failure according to their expected performance.

3. Punish probabilistically "inversely" to reputation (INVERSE): The reputation of a member i now decreases according to the middle case of (Equation 2) with probability $(1 - r_i) / \sum_j (1 - r_j)$

$$r'_i = \begin{cases} \frac{\beta r_i t_i + 1}{\beta t_i + 1}, & if \ \sum_j u_j[i] > d \\ \frac{\beta r_i t_i}{\beta t_i + 1}, & if \ \sum_j u_j[i] < -d \\ r_i, & if \ \left|\sum_j u_j[i]\right| \leq d \end{cases} \qquad \text{(Equation 2)}$$

1. u_j is the rating vector

2. ti is the number of task computations of node i before the present task,

3. β (0, 1) is the factor that discounts the weight of past transactions,

4. and d is a confidence threshold; the higher the level of confidence required for the outcome of the rating procedure, the higher d.

5. the formula corresponds to a majority rule.

Furthermore, if we can assume that the system has some idea of how many data sources did not perform adequately, thus leading the whole product to a failure, then there are more possibilities. To illustrate them, while keeping our study simple, we adopt the following assumption: A product P fails in a service provision (i.e., has lower collective performance than acceptable) if at least M of its data sources do not exert the necessary effort. Then, the smart contract can punish only M members, by lowering their reputation, while leaving intact that of all other data sources of the group. We propose the following two punishing approaches as in [30]:

1. Punish the worst M (WORST M): Sort the reputation values in descending order and decrease the reputation of the M data sources with the lowest reputation values according to the middle case of (Equation 2).

2. Punish random M (RANDOM M): Decrease the reputation values of M random data sources as above.

Until now, we have only dealt with reputation in accordance with Beta rule. Next, we employ Bayes' rule for updating agents' individual reputation values based on their collective outcome. Bayes approach is a standard one for approximating hidden variables. This approach (BAYES) is applicable when there are specific fixed performance types of nodes with known success probabilities. For simplicity, we assume that there are two performance types of High (H) and Low (L) of agents that have success probabilities Ph , Pl respectively.

Then after a failure collective Product outcome (F), the reputation ri of the individual data source i is computed according to the formulas below (Equation 3) as in [30]:

$$r'_i = \Pr[A_i \, \varepsilon \, H|F] = \frac{\Pr[F|\, A_i \, \varepsilon \, H] * \Pr[A_i \, \varepsilon \, H]}{\Pr[F|\, A_i \, \varepsilon \, H] * \Pr[A_i \, \varepsilon \, H] + \Pr[F|\, A_i \, \varepsilon \, L] * \Pr[A_i \, \varepsilon \, L]} \quad (Equation\ 3)$$

- Where $\Pr[F|\, A_i \, \varepsilon \, H]$ = P_h * Pr[>= M+1 agents fail] + (1- P_h) * Pr[>= M agents fail]
- Where $\Pr[F|\, A_i \, \varepsilon \, L]$ = P_l * Pr[>= M+1 agents fail] + (1- P_l) * Pr[>= M agents fail]

- Where Pr[>= M agents fail] = $\sum_{m=M}^{N-1} \sum_{\sum_{j\,!=m} u_j=m} \Pi_{j\,!=i} Pj^{u_j} \, (1-Pj)^{1-u_j}$
- Where Pj = $r_j * P_h$ + (1-r_j)* P_l

Having finished with the employ of Bayes' rule for updating data source's individual reputation values based on their collective outcome of the Product in the case of a failed service provision, now we can see what happens in the case of successful collective outcome (S). We propose the following approaches for updating reputation of data sources after a succeeded service provision in the case that we do not have intermediate results:

1. Reward all equally (REWARD ALL): increase the reputation of each data source that participate in the composition of product P as in the first case of (Equation 2).

2. Reward probabilistically fairly to reputation (PROP): The reputation of a data source i is updated according to the first case of (Equation 2) but with probability $r_i / \sum_j r_j$, where the j's are all data sources group that compose the product P. Thus, the group members (the data sources) share the reward of the collective success according to their expected performance.

3. Reward probabilistically "inversely" to reputation (INVERSE): The reputation of a data source i now increases according to the first case of (Equation 2) with probability $(1 - r_i) / \sum_j (1 - r_j)$

Furthermore, if we can assume that the smart contract has some idea of how many data sources perform adequately in the composition of the product, thus leading the whole product to a success, then we have more possibilities. To illustrate them, while keeping our study simple, we adopt the following assumption: A product P success in a service provision (i.e., has higher collective performance than acceptable) if at least M of its data sources do exert the necessary effort. Then, the smart contract can reward only M members, by augmenting their reputation, while leaving intact that of all other data sources of the group. We propose the following two rewarding approaches:

1. Reward the best M (BEST M): Sort the reputation values in descending order and increase the reputation of the M data sources with the highest reputation values according to the first case of (Equation 2).

2. Reward random M (RANDOM M): Increase the reputation values of M random data sources as above.

Until now, we have only dealt with reputation in accordance with Beta rule. Next, we employ Bayes' rule for updating agents' individual reputation values based on their collective outcome. For simplicity, we assume that there are two performance types of High (H) and Low (L) of agents that have success probabilities Ph , Pl respectively.

Then after a success collective Product outcome (S), the reputation ri of the individual data source i is computed according to the formulas below (Equation 4):

$$r'_i = \Pr[A_i\ \varepsilon\ H|S] = \frac{\Pr[S|\ A_i\ \varepsilon\ H] * \Pr[A_i\ \varepsilon\ H]}{\Pr[S|\ A_i\ \varepsilon\ H] * \Pr[A_i\ \varepsilon\ H] + \Pr[S|\ A_i\ \varepsilon\ L] * \Pr[A_i\ \varepsilon\ L]} \quad (Equation\ 4)$$

- Where $\Pr[S|\ A_i\ \varepsilon\ H]$ = P_h * Pr[>= M+1 data sources succeeded] + (1- P_h) * Pr[>= M data sources succeeded]
- Where $\Pr[S|\ A_i\ \varepsilon\ L]$ = P_l * Pr[>= M+1 data source succeeded] + (1- P_l) * Pr[>= M data source succeeded]

In paper [30] the researchers n order to examine the effectiveness of the various heuristic approaches for estimating the hidden performance of an individual agent did an extensive simulation in order to identify which of the previous cases (PUNISH ALL, PROP, INVERSE, REWARD ALL, etc) works better. From their experimental results we can identify that RATE ALL and PUNISH ALL are consistently very effective and stable choices in a competitive environment. However, in large groups with high collusion the last resort would be the PUNISH ALL approach, which achieves fair enough effectiveness in all cases, provided that a large enough tolerance threshold is employed.

We recall that the more effective is an approach in estimating individual performance the stronger are the incentives for higher performance to individual agents. This is why we discussed before about the hidden quality problem. So, before the user just knew the total reputation score of a product. The user did not know the atomic reputation score of each data source. Now, that the reputation score is split to each data source, the data source should have a good reputation score in order to have chances to be selected and in order to avoid its' exclusion from the formation of the product.The user employs a reputation-based selection policy, so it is obvious that he will select the nodes with the higher reputation score.

Nevertheless, the main conclusions drawn for the effectiveness of the various approaches are intuitively expected to apply for any static behavioral model. The estimation of dynamic behavior is in general more difficult and it depends on the percentage of agents that follow rational strategies and the speed of convergence of the various reputation update approaches. However, it should be noted that if rational agents are aware of the specific reputation update approach for the estimation of their individual performance, then there may exist performance strategies for them to avoid detection of low performance from that particular approach.

To sum up, in order case, as we discussed before, the information is unobservable so we do not use the RATE ALL approach. As a result, in our experimental simulation we will use the PUNISH ALL approach that works better according the experimental results that we retrieve from the research [30]. So, between the aforementioned two approaches, Beta rule versus Bayes' rule, we will use for our reputation update the Beta rule and more precisely the PUNISH ALL subcategory. Like this we will calculate the data sources' individual reputation values based on the collective outcome of the product that it participate.

Having finished with the case of 3 of Figure 1, we should now see how we will update the reputation score of cases 4a, 4b and 4c.

1. Case 4a: We recall that if the reputation score of a data source is updated, since it belong to a Data Provider, so has the reputation value of the affected Data Provider that this data source belongs to (Figure 2). The reputation score of that federation must be updated with the simplest way:

   $$AffectedDataProviderReputationScore_i = \sum_j^p \frac{w_{j*}r_j}{w_j}, \mathrm{J} \; \varepsilon \; \mathbb{R}_p$$

   Where p is the number of resources that belong to the data Provider, and j each resource. W represents the weight of each Data Source in the Data Provider. The more important the Data Source is, the more weight has in this equation. For example, some Data Sources may be used more frequently than others or have better performance than others. Then their contribution in the calculation of the reputation of the Data Provider should be higher than other Data Sources. We do not know a priori the number of resources that belong to Provider. This number of resources is depicted as j and it belongs to real numbers system from where we can also take the different weights w_j for each resource.

   This calculation has O(n) complexity, and it is not complicated. Instead, we could use other reputation computation methods such as: Weighting based of normal distribution, Beta distribution, Bayes distribution model etc.

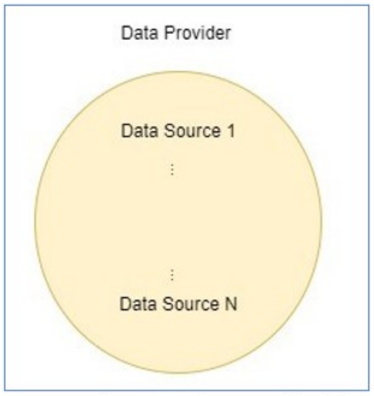
   

   Figure 2: Each Data Provider consists of multiple Data Sources.

2. Case 4b: We recall that if the reputation score of a data provider is updated, since it provides a data source to the procedure of deployment of a product and the reputation value of this data source has been updated, so has the reputation

value of the affected federation that this provider belongs to (Figure 3). The reputation score of that federation must be updated with the simplest way:

$$AffectedFederationReputationScore_i = \sum_j^f \frac{W_{j*r_j}}{W_j}, J \in \mathbb{R}_f$$

Where f is the number of data sources that belongs to the Data Provider that belongs to the particular federation, and j each Data Source. W represents the weight of each Data Source in the federation. The more important the Data Source is, the more weight has in this equation. For example, some Data Sources may be used more frequently than others or have better performance than others. Then their contribution in the calculation of the reputation of the federation should be higher than other Data Sources. We do not know a priori the number of resources that belong to Federation. This number of resources is depicted as j and it belongs to real numbers system from where we can also take the different weights w_j for each resource.

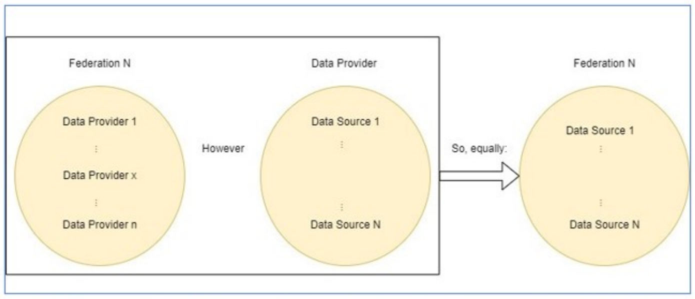


Figure 3: A Federation N has its' Data Providers. However, each Data Provider has a variety of Data Sources. As a result, each federation is composed by a variety of Data Sources that belong to the Data Providers of that specific Federation.

3. Case 4c: We should also recall that a data source is used to form products. It is very common a data source to be used in the formulation of variety of products. So, if a data source's reputation score is updated, so has the reputation score of the affected products that have been deployed with the use of that data source (Figure 4). The reputation score of the affected products will be updated according to the Equation below:

$$AffectedDataProviderReputationScore_i = \sum_j^p \frac{W_{j*r_j}}{W_j}, J \in \mathbb{R}_p$$

Where p is the number of data sources that contribute to the deployment of the affected product. W represents the weight of each Data Source in the formation of that product. The more important the Data Source is, the more weight has in this equation. For example, some Data Sources have more contribution in the final product-service. Then their contribution in the calculation of the reputation of the federation should be higher than other Data Providers. We do not know a priori the number of resources that participate in the formation of a Product. This number of resources is depicted as j and it belongs to real numbers system from where we can also take the different weights w_j for each resource.

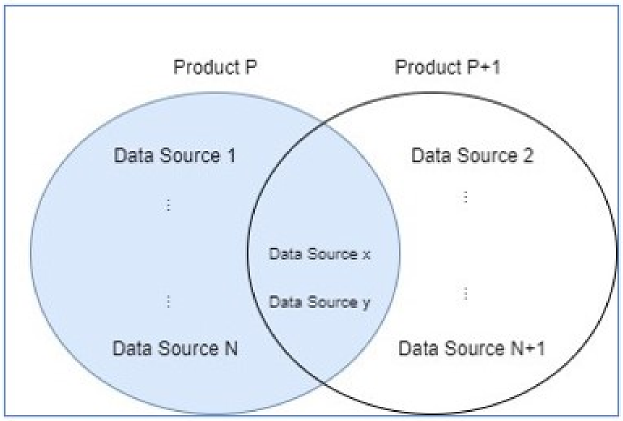


Figure 3: Representation of Products with the Data Sources that are composed of.

Finally, having finished with the cases 3, 4a, 4b and 4c of Figure 1, we have to study the cases 2a and 2b. This cases are referring to the initial calculation of the reputation score of the Product. This calculation is the aggregation of both subjective reputation score that is submitted by the consumer and the objective reputation score that submitted by the system.

The reputation score of the initial product is calculated as the aggregation of both subjective and objective scores:

1. The subjective score represents the targets of the user. It express the pleasure or displeasure of the user experience and takes values **from 0 to 1**. For example, if we have an application that drives the user to the nearest parking slot and in the end the user cannot find the particular parking, the user can score the system with 0 that express his displeasure.

2. The objective score is different for each data source and it is provided by the system. It express the quality of experience instead of the user experience that we had before. Those metrics can be the latency, the availability, the noise, the coverage, the accuracy, the completeness, the consistency, the timeliness, the uniqueness, the validity, the continuity etc. For example, in the previous scenario that a user calls for a taxi, the latency and the availability are very important in the user experience. Another useful example that demonstrates our case is a weather application. In order to provide the temperature the application is using a resource e.g. a sensor that captures the temperature. The objective metric here is the accuracy and we want an accuracy grater than 0.95. So, if that resource has good reputation score e.g. grater than 0.9 that means that the objective metric is covered.

3. The final reputation score of the initial product is calculated as the aggregation of the subjective and objective reputation scores.

4. In equation 5, k express different objective metrics that have value in the functionality of the final product.

5. In equation 6, we can identify that the two sets are alien to each other. Their aggregation add up to the unit.

6. In the last equation we can see that the final reputation score is the is the weighted sum of both subjective and objective scores.

$$Objective\ quality\ of\ a\ product = \sum_k \frac{w_k * k}{w_k}, \quad \text{Equation 5}$$

$$Subjective\ score\ of\ a\ product = 1 - Objective\ score\ of\ a\ product = \sum_i \frac{w_i * i}{w_i}, \quad \text{Equation 6}$$

$$Reputation\ of\ the\ product = \sum \frac{w_1 * Objective\ score\ of\ a\ product + w_2 * Subjective\ score\ of\ a\ product)}{w_1 + w_2}$$

Now, that all the reputation scores have been calculated and the procedures of update have been represented we should focus on when the update will be taking place.

More precisely, we propose to update the reputation values periodically instead of each time that we have a new submission of reputation. In the same way, the objective metrics will be captured periodically.

4.4 Computation of Reputation as a whole

Having analysed the whole procedure of the sequentially updates, next, we must study how individual performance of data sources can be revealed by associating to each data source i to reputation metric ri that is properly updated based on collective outcomes of the products (figure 1 step 3). We recall that a product is composed of several data sources, so the collective reputation score of the product should be transferred individually to each data source.

First of all, we should remind that in each transaction, the data consumer should submit a subjective score for each product of the transaction (case 2a). This score depends on the private preferences of the consumer and helps us identify the hidden quality problem in metrics for which we cannot capture an objective value from the platform. The subjective score takes values in [0,1].

Secondly, in each transaction period we capture values for the objective metrics of each quality class. We suppose that periodically, in specific monitoring periods we collect values of objective metrics from the platform in order to be able to calculate the objective score of a datasource (case 2b). If a datasource belongs to multiple quality classes then it will have multiple different objective score, one per quality class. That is happening because a datasource may contribute in the formation of products that may belong to different vertical category. Each vertical category has different objective metrics, or/and different target values for those objective metrics or/and different weights for those objective metrics. In this way the system is context-specific.

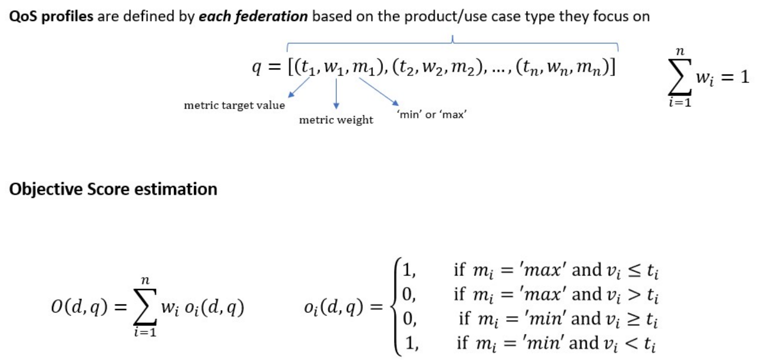
Finally, the updated reputation score of the transacted product is the aggregation of objective score of the resources that participate in the formation of that product, and the subjective score of that product that is submitted by the data consumer.

Calculation of the objective score per datasource per Quality Class.

1. QoS profiles are defined by each federation based on the product/use case type they focus on (vertical distinction of the products based on their use case). In this way our mechanism is context specific. Products may belong to different federations have different set of QoS profiles, so different objective metrics.

2. each time that we get new values for those objective metrics (in each monitoring period) we should calculate the new datasource objective value based on those new-captured values.

3. in order to do calculate the new objective score of this datasource we follow the below formula.



QoS profiles are defined by ***each federation*** based on the product/use case type they focus on

$$q = [\underbrace{(t_1, w_1, m_1), (t_2, w_2, m_2), \ldots, (t_n, w_n, m_n)}] \qquad \sum_{i=1}^{n} w_i = 1$$

Objective Score estimation

$$O(d,q) = \sum_{i=1}^{n} w_i \, o_i(d,q) \qquad o_i(d,q) = \begin{cases} 1, & \text{if } m_i = 'max' \text{ and } v_i \le t_i \\ 0, & \text{if } m_i = 'max' \text{ and } v_i > t_i \\ 0, & \text{if } m_i = 'min' \text{ and } v_i \ge t_i \\ 1, & \text{if } m_i = 'min' \text{ and } v_i < t_i \end{cases}$$

4. in this way, we managed to capture a value for each datasource in a monitoring period.

Now, we will update the reputation score of all datasources in each monitoring period. As we said in the previous chapter the reputation update procedure is happening periodically after n transactions. The procedure that we follow in order to update the reputation score of a datasource after n transaction is the following:

1. Calculate the average subjective score of data source d over subjective scores per products submitted for number of product transactions, where data source d is part of.

2. Calculate the average objective score of data source d over objective scores collected per datasource for each quality class q (e.g, the different quality classes of products in which data source d contributes) from —T— monitoring periods. In fact this is the average of the datasource objective scores that we calculate previously. But, previously those scores were saved and now that it is the time for the reputation update we will calculate their average in order to calculate the new total objective reputation score of each datasource of each quality class (if we have one vertical context of products, we have one quality class. If we have two vertical product categories, we have two quality class). A datasource that participate in the formation of products that belong to different vertical categories will have different objective score per category. So, in that step we will aggregate all those different objective values (per QoS) to one total per datasource.

3. in order to calculate the new objective score of a datasource we follow the below formula.

   1. **Avg. subjective score of data source $\boldsymbol{d}$** over subjective scores per products
      - submitted for $|P_d|$ **product transactions**, where data source d is part of

   $$\bar{S}(d) = \frac{\sum_{p \in P_d} S(p)}{|P_d|}$$

   2. **Avg. objective score of data source $\boldsymbol{d}$** over objective scores collected per datasource
      - for each **quality class** q $\in Q_d$, *(i.e., the different quality classes of products in which data source d contributes = 2)*
      - from $|\mathrm{T}|$ *monitoring periods (5 transactions = 5 monitoring periods)*

   $$\bar{O}(d) = \frac{\sum_{\tau \in \mathrm{T}} \sum_{q \in Q_d} \mathrm{O}_\tau(d, q)}{|\mathrm{T}||Q_d|}$$

   Objective score of monitoring period τ for data source d and for QoS profiled q

We should not forget to take into account the previous (old) reputation score of a datasource. So, in order to complete the procedure of reputation update we should implement the following formulas:

- Reputation score in the current time period

$$R^{cur}(d) = \bar{O}(d)\, w + \bar{S}(d)\,(1 - w)$$

Weight – trust in objective score

- Reputation score update

$$R^{new}(d) = R^{cur}(d)\, \lambda + R^{old}(d)\,(1 - \lambda)$$

Decay - Exponential weighted moving average

Having available all the updated values of the datasources (case 2b) by following the previous procedure, and the subjective score for that product from the client (case 2a) we can calculate very easily the new reputation score of the product.

- Reputation score in the current time period

$$R^{cur}(p) = \frac{\sum_{i=1}^{n} R^{new}(d)}{|\mathrm{n}|}$$

- Reputation score update

$$R^{new}(p) = R^{cur}(p)\,\lambda + R^{old}(p)\,(1-\lambda)$$

we know the datasources from which the product is composed of. So, having updated the reputation scores of the datasources that is the combination of both subjective and objective variables, we can easy update the reputation score of that product.

Having finished with the case of 2a, 2b and 3 of Figure 1, we should now see how we will update the reputation score of cases 4a, 4b and 4c.

1. Case 4a: We recall that if the reputation score of a data source is updated, since it belong to a Data Provider, so has the reputation value of the affected Data Provider that this data source belongs to (Figure 2). The reputation score of that federation must be updated with the simplest way:

   - Reputation score in the current time period

$$R^{cur}(pr) = \frac{\sum_{i=1}^{n} R^{new}(d)}{|\mathrm{n}|}$$

   - Reputation score update

$$R^{new}(pr) = R^{cur}(pr)\,\lambda + R^{old}(pr)\,(1-\lambda)$$

In this type we could also use weights W that represent the weight of each Data Source in the Data Provider. The more important the Data Source is, the more weight has in this equation. For example, some Data Sources may be used more frequently than others or have better performance than others. Then their contribution in the calculation of the reputation of the Data Provider should be higher than other Data Sources.

This calculation has O(n) complexity, and it is not complicated. Instead, we could use other reputation computation methods such as: Weighting based of normal distribution, Beta distribution, Bayes distribution model etc.

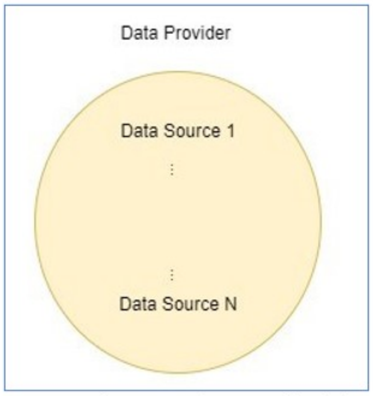


Figure 2: Each Data Provider consists of multiple Data Sources.

2. Case 4b: We recall that if the reputation score of a data provider is updated, since it provides a data source to the procedure of deployment of a product and the reputation value of this data source has been updated, so has the reputation value of the affected federation that this provider belongs to (Figure 3). The reputation score of that federation must be updated with the simplest way:

   - Reputation score in the current time period

$$R^{cur}(f) = \frac{\sum_{i=1}^{n} R^{new}(d)}{|n|}$$

   - Reputation score update

$$R^{new}(f) = R^{cur}(f)\,\lambda + R^{old}(f)\,(1-\lambda)$$

In this type we could also use weights W that represent the weight of each Data Source in the federation. The more important the Data Source is, the more weight has in this equation. For example, some Data Sources may be used more frequently than others or have better performance than others. Then their contribution in the calculation of the reputation of the federation should be higher than other Data Sources.

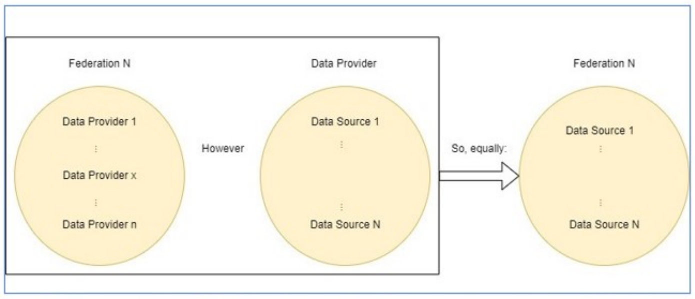


Figure 3: A Federation N has its' Data Providers. However, each Data Provider has a variety of Data Sources. As a result, each federation is composed by a variety of Data Sources that belong to the Data Providers of that specific Federation.

3. Case 4c: We should also recall that a data source is used to form products. It is very common a data source to be used in the formulation of variety of products. So, if a data source's reputation score is updated, so has the reputation score of the affected products that have been deployed with the use of that data source (Figure 4). The reputation score of the affected products will be updated according to the Equation below:

- Reputation score in the current time period

$$R^{cur}(p) = \frac{\sum_{i=1}^{n} R^{new}(d)}{|\mathrm{n}|}$$

- Reputation score update

$$R^{new}(p) = R^{cur}(p)\,\lambda + R^{old}(p)\,(1-\lambda)$$

In this type we could also use weights W that represent the weight of each Data Source in the formation of that product. The more important the Data Source is, the more weight has in this equation. For example, some Data Sources have more contribution in the final product-service. Then their contribution in the calculation of the reputation of the federation should be higher than other Data Providers.

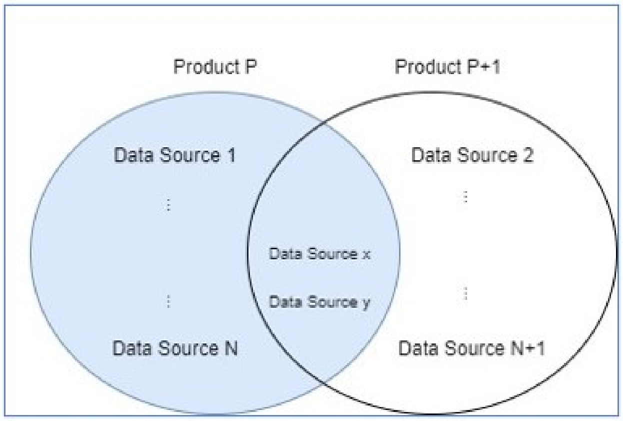


Figure 3: Representation of Products with the Data Sources that are composed of.

The reputation score of the initial product is calculated as the aggregation of both subjective and objective scores:

1. The subjective score represents the targets of the user. It express the pleasure or displeasure of the user experience and takes values **from 0 to 1**. For example, if we have an application that drives the user to the nearest parking slot and in the end the user cannot find the particular parking, the user can score the system with 0 that express his displeasure.

2. The objective score is different for each data source and it is provided by the system. It express the quality of experience instead of the user experience that we had before. Those metrics can be the latency, the availability, the noise, the coverage, the accuracy, the completeness, the consistency, the timeliness, the uniqueness, the validity, the continuity etc. For example, in the previous scenario that a user calls for a taxi, the latency and the availability are very important in the user experience. Another useful example that demonstrates our case is a weather application. In order to provide the temperature the application is using a resource e.g. a sensor that captures the temperature. The objective metric here is the accuracy and we want an accuracy grater than 0.95. So, if that resource has good reputation score e.g. grater than 0.9 that means that the objective metric is covered.

3. The final reputation score of the initial product is calculated as the aggregation of the subjective and objective reputation scores.

4. In equation 5, k express different objective metrics that have value in the functionality of the final product.

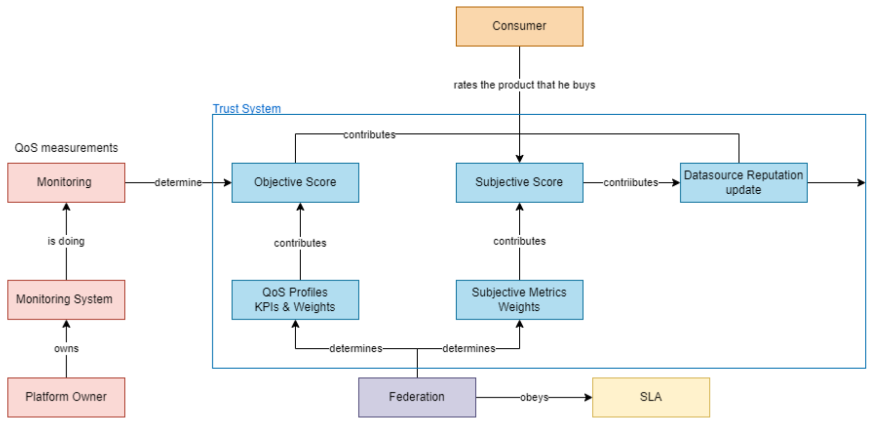
5. In equation 6, we can identify that the two sets are alien to each other. Their aggregation add up to the unit.

6. In the last equation we can see that the final reputation score is the is the weighted sum of both subjective and objective scores.

Now, that all the reputation scores have been calculated and the procedures of update have been represented we should focus on when the update will be taking place. More precisely, we propose to update the reputation values periodically instead of each time that we have a new submission of reputation. In the same way, the objective metrics will be captured periodically.

4.5 Overview

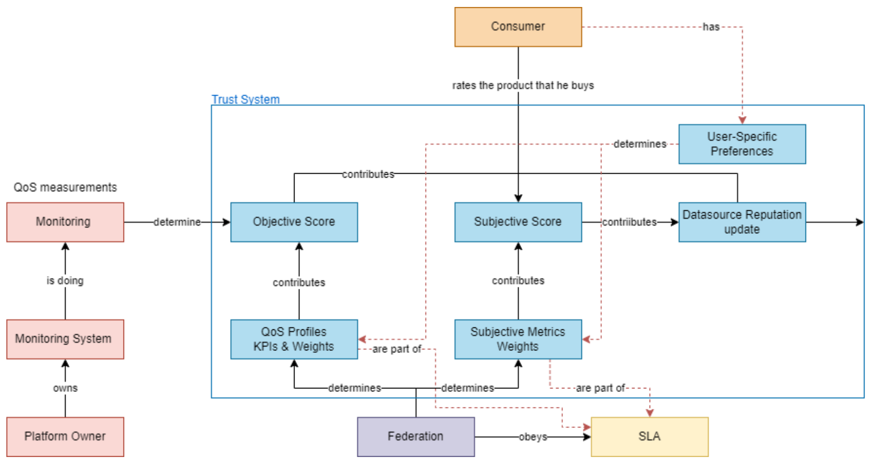
As we saw in the previous chapters, the only interaction between the data consumers and the system is when the consumer submits the subjective score for the product of the transaction that he made. The objective metrics, the weights of those KPIs, snd the weights of the subjective scores are determine by the federation that this product belongs to. As you can see in the following figure, the federation is following a Service Level Agreement (SLA) and determines both the objective quality metrics and their weights and the weights for the objective scores. The scores of the quality metrics are collected by the platform in precise monitoring periods periodically.



Now, lets suppose that we want those objective metrics to be context/profile specific. For example to divide the products in vertical categories and each category to have her own quality metrics based on the important characteristics and services that this product offers. In this way, products that belong to same vertical category will have the same quality metrics. However, they may have different target values or weights.

Now lets suppose that except from context-specific, we want or system to be also user specific. That means to give the opportunity to the data consumer, to select which objective metrics are more important for him, so to give him the opportunity to select the weights or the quality metrics of the product category that this product belongs to. The data consumer according to his preferences he can select also subjective objective metrics as we illustrate in the following figure.

If we update our system with the following ideas we can create a context specific and user friendly environment. Each data consumer will choose the quality metrics that he wants in order to match his preferences (determines both KPIs and Weights). He can also determine the subjective metrics weights according to his preferences. In this way the system becomes transparent to the client and it becomes also scalable as we can add more quality metrics per vertical product category or even to add more vertical categories in the future if new products with new specifications launch the market.

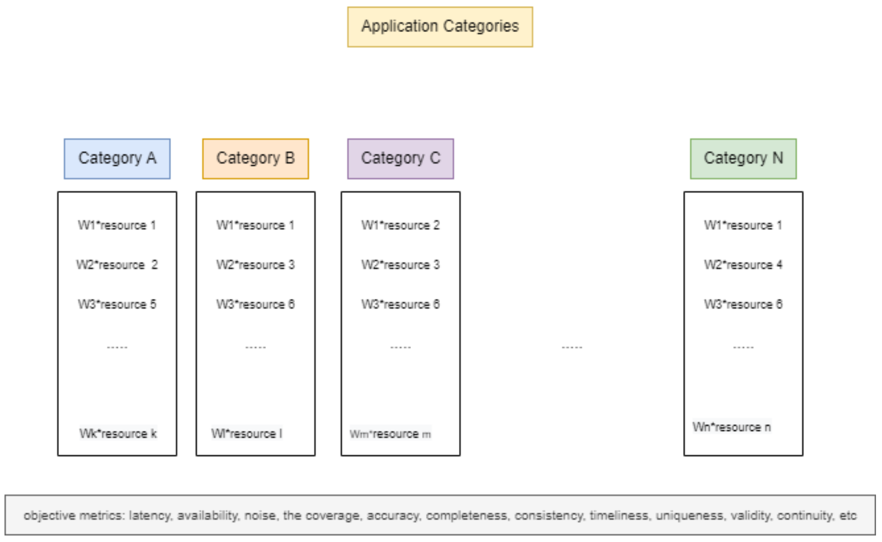


4.6 Computation of reputation per application

In the previous subsection we saw that the reputation of a product is calculated as the aggregation of both objective and subjective scores. The subjective score takes values from zero to one and it is submitted by the consumer. The objective score is derived by the system per each data source that participate in the formation of the final product.

However, when we are implementing aggregation we have hidden information from the client. The client is unaware of how this reputation score is occurred and is able to see only the final reputation of a product and is not able to recognise the atomic score of each resource/objective metric. For that reason, we propose a new transparent way to keep our reputation score. More precisely, we suggest the objective scores and the subjective scores to be transparent and available to the clients. In that way each client, according to his requirements and needs he can calculate the reputation score of a product on his own with a total transparent way.

To be more specific, lets suppose that we have 10 different types of application. It is obvious that each application has its' own specifications, its' requirements and its' characteristics. In that case, each application has its' own objective metrics that counts differently in the calculation of the reputation of each application. **So, the objective score should be application specific**. It is impossible to have same way of calculation of the reputation for all the different types of application as the reputation is the aggregation of subjective and objective scores but each application has her owns objective scores with different weights per category.



This pioneer way of calculation of reputation offers transparency in the way that the calculation is made and simultaneously it offers new opportunities of scalability in the case that new applications are launched in the market or if the users are particular expectations and requirements. In other words, a new application category may demands different set of objective/quality metrics, but from the time that we keep in memory all the objective scores, it will be very effective and efficient to calculate instantaneously the new reputation score for that new type of application/product. In addition, if a new application demands new objective requirements to reach a threshold it will be very each to scale our system by simple adding new objective metrics and new vertical categories of applications.

Last but not least, this design is user-centralised. The user can see in a transparent way all the reputation scores of the resources and select the ones that match better his preferences. Also, the weights of each resource are selected by the user. As a result the reputation score is calculated in a user-specific way by aggregating the history of the reputation scores of the resources and by using the weights that the client wants.

We categorize the application in N vertical categories. It is obvious that, the applications with the same set of resources belong to the same category. **Each category has both specific objective metrics and a set of weights**. Apparently, they may exist similar applications with same objective requirements - resources. However, each resource may contributes in different manner in the calculation of the final reputation of each application. That's why we will use different weighs per objective metric per application that they will be user-specific. For example, if I have 10 different application categories, then I have 10 different values for one resource, plus one the subjective score that the client submits. That means that each time I have 11 reputation scores for each resource. So, that we have a different reputation score per each resource for each different application context.

It is obvious now that instead of the previous way of calculation of reputation, now each objective metric is application specific and takes different value with different weigh per application category. So, it derives that we have different reputation score per application and that's the added value of that particular idea, the transparency.

In that new context, if a new application join the network, or if a client has specific requirements, different from the existing ones, for example he wants availability higher than 90 percent and at the same time he demands the noise to be less than 5 percent, if we have saved the history of all the objective and subjective metrics we can calculate the personalised based-ranking for that new application category, or for that specific requirements of the client. So, our system remain scalable for future additions.

In order to keep the flow from the previous subsection, it is important to mention that we will use the PUNISH ALL or the REWARD ALL approaches in order to transfer the reputation score from the collective reputation score of the product to each atomic reputation score of the resources. Also, we will update the reputation score of the federation, of the data provider and of the affected products with the equations that we also described in detail in the previous chapter. The difference in that chapter is that we introduce a new way to calculate the initial reputation of a product in a vertical manner, in an application specific and personalised way. As a result the reputation score of the affected providers, federations and products with be also context and user specific. For example, a provider will have different reputation score for the category A, different for the category B etc. This is happening because the reputation score of the provider is the aggregation of the reputation scores of the resources that this provider provides. So, from the time that the resources have different reputation scores per category so will be the reputation scores of the provider. Same logic are following the affected federations and the affected products.

To sum up, the applications are divided into vertical categories. Each category has specific resources/objective metrics that counts different in the calculation of the reputation score depending on the client. So, the weights are specific for each client and in the end we calculate a weighted subjective plus objective average for each vertical application by using the equation that we analyse in the previous subsection. Then, the update of the reputation is calculated as in the previous subsection. So, the innovation here is that the reputation score is calculated as the summarisation of the history that we keep in memory. That is useful for transparency reasons. In this way, the user will be able to compare and select not only products but he can also compare data sources, data providers or even federations.This is also solves the problem of hidden quality that we discussed in previous chapters.

5 Experimental results

The innovative idea that we described in the previous section undeniably adds extra complexity. Instead of the calculation of the reputation with a simple aggregation in a uniformly manner for all the types of applications, now we should keep in the memory all the history of the reputation scores of each objective metric of each category and all the weights. So, it is vitally important to prove that this new way of summarisation of the history is more efficient and more precise than the previous one.

So, in this chapter we will make our experiments in order to see how the reputation score of a product is fluctuating after some transactions.We suppose that we have 1 vertical category of applications, so all the products have same quality metrics, same weights and follow the same distribution. We will try to:

1. calculate the new products reputation score for this vertical category
2. update the reputation score of datasources
3. update the reputation score of the affected federations, providers and products
4. examine how the reputation score of a product is fluctuating in order to conclude if it worth's' to adopt the vertical application specific mechanism in order to calculate the reputation with the formulas that we discussed in the previous sections. We should also take into the account the fact with this way if calculation we promote a more user friendly and user specific way of computation of reputation. This is leading the data consumers in more wise choices of products based on their special needs and preferences.

With some probability the user chooses one product that belongs to a single application category. So, this product has specific objective/quality metrics. The steps that we will follow are the following:

1. calculate the new reputation score of the product
2. update the reputation score of the data sources, of the provider and of federation
3. compare the results
4. wherever we will need weights we will take them by a random generator formula. All the weights will be produced by the same formula, that means that all the products belongs to the same vertical category and have the same quality metrics, with same targets and weights.
5. the above calculation will be done by scripting in python programming language

6 Smart Contract Implementation

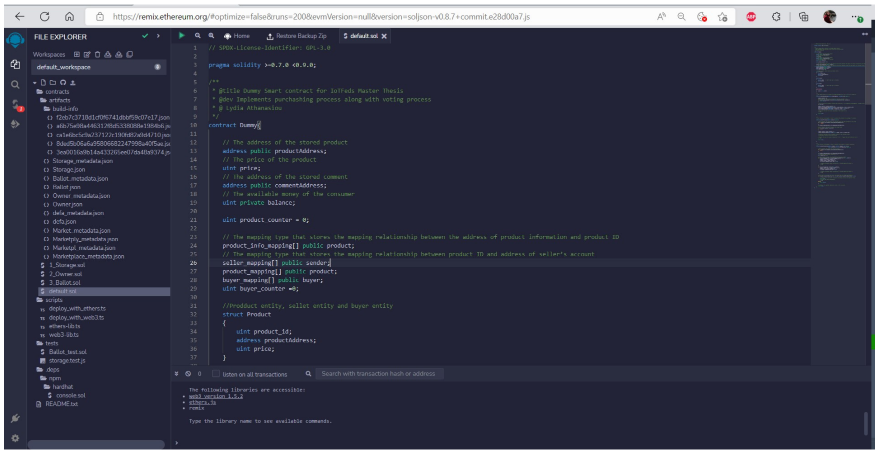
In the previous sections we made an introduction to reputation systems and to the blockchain environment. In addition, we analyze and explained in detail the IoTFeds architecture and the way that the reputation score is calculated in each layer of this architecture. Now, in order to acquire a whole picture of the IoTFeds mechanism we should examine how the blockchain environment will be used in this state-of-the-art concept. We should determine which information is saved on-chain and which information is saved off-chain and how the procedures of uploading a product, of purchasing a product and of evaluation of a product can be implemented with the use of smart contract technology.

So, in this section we provide a smart contract in Solidity language for Ethereum blockchain development. The code deployment is made with the use of IDE 'Remix' via the web portal interface. The following algorithm is used to implement the evaluation and the incentive process. Once both a buyer and a seller have submitted their comments and ratings to User Interface in an online marketplace, the smart contract computes and updates the reputation scores of the buyer and the seller, and then publishes the reputation scores to the blockchain by following the below steps.

1. The smart contract verifies whether the product has been purchased or not.
2. If the product information has been uploaded, the smart contract checks whether the buyer has submitted the rating and comment which are necessary in the evaluation procedure of a product only after the successful purchasing.
3. The smart contract checks whether both the buyer and the seller have submitted the comments and the ratings.
4. If they have submitted, the smart contract computes and updates the reputation scores of the buyer and the seller accordingly to the above equations that presented in detail in section 4.
5. Finally, the smart contract broadcasts the updated reputation scores to the blockchain.

It is highly important to mention that the following code is complied successfully without any grammar issues, however, it is not connected with a graphical interface. Under those circumstances it represents a dummy structure of a smart contract that could be used in an environment like the one we analyse on this thesis but at this point it is not neither fully developed nor totally completed. Thus, it is helpful in order to understand how a smart contract can be used in our scenario and to depict the hole mechanism in practise.

In the following figure we can see the environment in which the smart contract has been developed. The environment is called REMIX and is an IDE for scripting in Solidity language of Ethereum.



```
// SPDX-License-Identifier: GPL-3.0

pragma solidity >=0.7.0 <0.9.0;

/**
 * @title Dummy Smart contract for IoTFeds Master Thesis
 * @dev Implements purchashing process along with voting process
 * @ Lydia Athanasiou
 */
contract Dummy{

    // The address of the stored product
    address public productAddress;
    // The price of the product
    uint price;
    // The address of the stored comment
    address public commentAddress;
    // The available money of the consumer
    uint private balance;

    uint product_counter = 0;

    // The mapping type that stores the mapping relationship between the address of product information and product ID
    product_info_mapping[] public product;
    // The mapping type that stores the mapping relationship between product ID and address of seller's account
    seller_mapping[] public sender;
    product_mapping[] public product;
    buyer_mapping[] public buyer;
    uint buyer_counter =0;

    //Prodduct entity, sellet entity and buyer entity
    struct Product
    {
        uint product_id;
        address productAddress;
        uint price;
    }
```

```
The following libraries are accessible:
• web3 version 1.5.2
• ethers.js
• remix

Type the library name to see available commands.
```

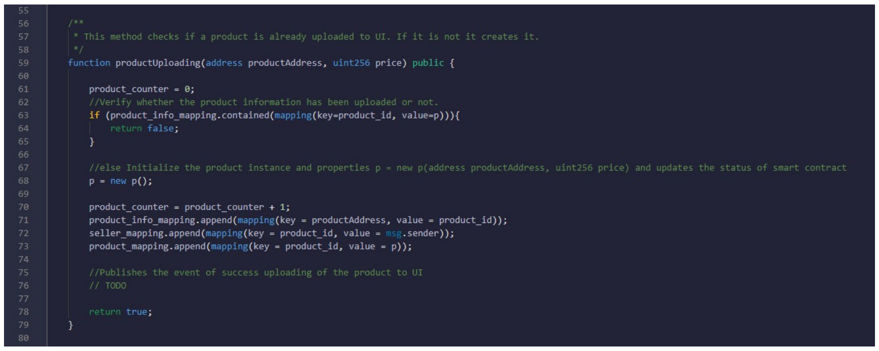
In the beginning of our implementation as we can see in the previous figure, we declare all the global variables that we will use during the implementation of the smart contract. After that, we declare all the hash-maps and the tables that are used in order to keep an eye on the available products, on the sellers and on the buyers. Before the implementation of the different functions (uploading of a product, purchasing of a product, evaluation of a product) we should definitely create some structs that are will represent 'objects' that are necessary to be represented. For example the product entity, the seller entity, the buyer entity etc.

In the following three figures we can see the deployment of the smart contract in which:

1. we initialize the basic variables that we will use during the deployment of the smart contract as presented before in detail.

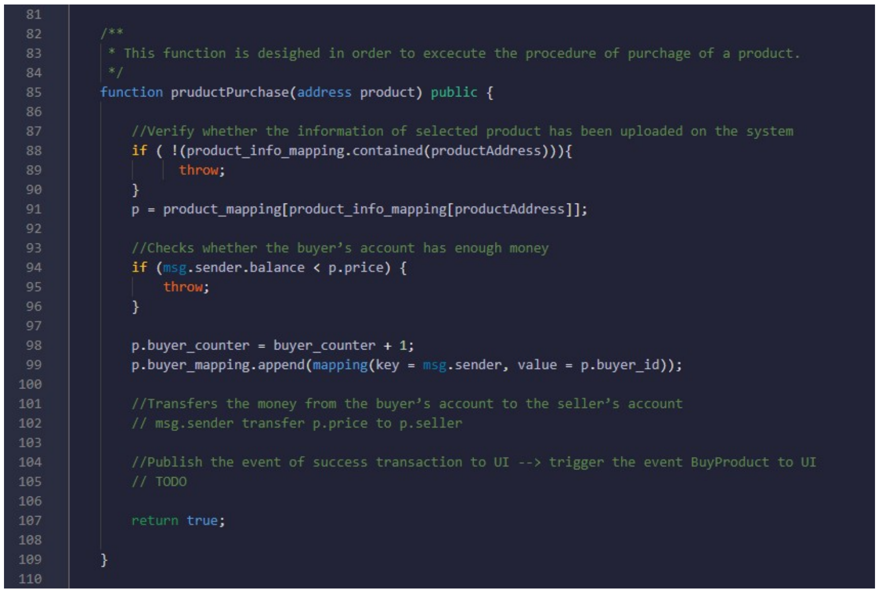
2. we declare the structures - entities of the smart contract that interacts with each other as presented before in detail.

3. The algorythm ”function productUploading(address productAddress, uint256 price) public” which checks if a product is already saved in the system and if it is not, it creates it, saved it and revealed-uploaded it in the User Interface (UI). More precisely, this function takes as input the address of the product and its’ corresponding price. Then it checks if that address exists in the available addresses that are stored in a table (product-info-mapping). If the address exists there, that means that the product is already uploaded and the procedure is stopped by returning ’false’. Else, the smart contract creates the product with the corresponding price and assign it a new address.The product counter, of course, is augmented by one (product-counter + 1), the product is added to the products table (its id and its address) and the seller of that particular product is also added to the list of the corresponding sellers (seller-mapping).



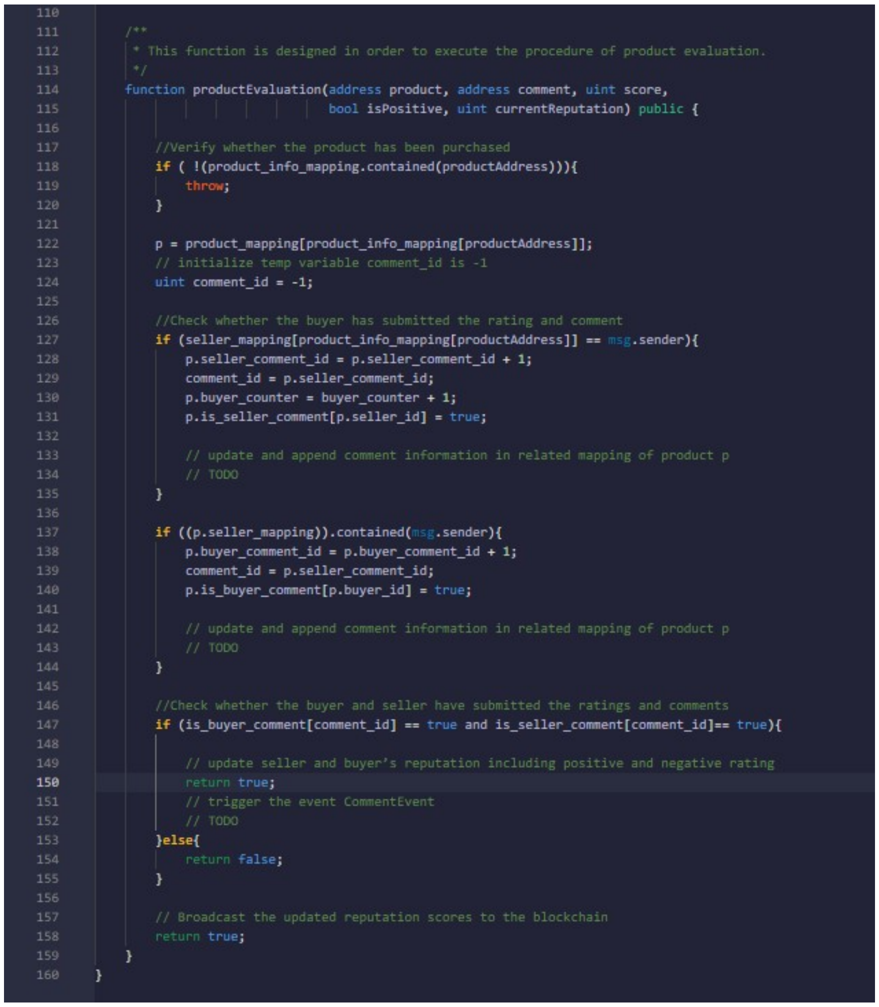
```

    /**
     * This method checks if a product is already uploaded to UI. If it is not it creates it.
     */
    function productUploading(address productAddress, uint256 price) public {

        product_counter = 0;
        //Verify whether the product information has been uploaded or not.
        if (product_info_mapping.contained(mapping(key=product_id, value=p))){
            return false;
        }

        //else Initialize the product instance and properties p = new p(address productAddress, uint256 price) and updates the status of smart contract
        p = new p();

        product_counter = product_counter + 1;
        product_info_mapping.append(mapping(key = productAddress, value = product_id));
        seller_mapping.append(mapping(key = product_id, value = msg.sender));
        product_mapping.append(mapping(key = product_id, value = p));

        //Publishes the event of success uploading of the product to UI
        // TODO

        return true;
    }

```

4. The function ”function pruductPurchase(address product) public” checks if a product exists in the stored list of products. If it is not exists that means that the procedure of purchase is stopped as the corresponding product is not available, leading to an exception. Else, the implementation stores in a variable p the particular object that it finds it in available stored tables with the use of the product address that is the input variable of that function (p = product-mapping[product-info-mapping[productAddress]]). Now, we have the product stored in variable p that will be purchased. Moreover, the smart contract should check if the consumer has the necessary money in order to buy the specific product. If both of the above conditions are true then the algorithm removes this product from the list of available products and takes the according money from the balance of the buyer. In the end, the algorithm add those money to the balance of the seller. The buyers variable is augmented by one quantity as a new consumer just bought successfully a product.



```

    /**
     * This function is desighed in order to excecute the procedure of purchage of a product.
     */
    function pruductPurchase(address product) public {

        //Verify whether the information of selected product has been uploaded on the system
        if ( !(product_info_mapping.contained(productAddress))){
                throw;
        }
        p = product_mapping[product_info_mapping[productAddress]];

        //Checks whether the buyer’s account has enough money
        if (msg.sender.balance < p.price) {
            throw;
        }

        p.buyer_counter = buyer_counter + 1;
        p.buyer_mapping.append(mapping(key = msg.sender, value = p.buyer_id));

        //Transfers the money from the buyer’s account to the seller’s account
        // msg.sender transfer p.price to p.seller

        //Publish the event of success transaction to UI --> trigger the event BuyProduct to UI
        // TODO

        return true;

    }

```

5. The function ”function productEvaluation(address product, address comment, uint score, bool isPositive, uint currentReputation) public ” takes as input the address of the product that should be evaluated, the score that submitted for that product, and helping Boolean variable and the current reputation that that product has before the submission of the new reputation score by the consumer. First of all the function checks if that product exists.If so, we assigned in a variable the corresponding product with that particular address that we have as input. Then we check whether the buyer has submitted the rating for that particular product. Finally, the function checks whether the buyer and seller have submitted the ratings and comments.



```

    /**
     * This function is designed in order to execute the procedure of product evaluation.
     */
    function productEvaluation(address product, address comment, uint score,
                               bool isPositive, uint currentReputation) public {

        //Verify whether the product has been purchased
        if ( !(product_info_mapping.contained(productAddress))){
            throw;
        }

        p = product_mapping[product_info_mapping[productAddress]];
        // initialize temp variable comment_id is -1
        uint comment_id = -1;

        //Check whether the buyer has submitted the rating and comment
        if (seller_mapping[product_info_mapping[productAddress]] == msg.sender){
            p.seller_comment_id = p.seller_comment_id + 1;
            comment_id = p.seller_comment_id;
            p.buyer_counter = buyer_counter + 1;
            p.is_seller_comment[p.seller_id] = true;

            // update and append comment information in related mapping of product p
            // TODO
        }

        if ((p.seller_mapping)).contained(msg.sender){
            p.buyer_comment_id = p.buyer_comment_id + 1;
            comment_id = p.seller_comment_id;
            p.is_buyer_comment[p.buyer_id] = true;

            // update and append comment information in related mapping of product p
            // TODO
        }

        //Check whether the buyer and seller have submitted the ratings and comments
        if (is_buyer_comment[comment_id] == true and is_seller_comment[comment_id]== true){

            // update seller and buyer’s reputation including positive and negative rating
            return true;
            // trigger the event CommentEvent
            // TODO
        }else{
            return false;
        }

        // Broadcast the updated reputation scores to the blockchain
        return true;
    }
}
```

7 The drawbacks and the attacks in the blockchain mechanism

Nowadays, we notice that the majority of scientists is using the Blockchain mechanism. It is undeniable that the blockchain architecture is very famous and it has became a famous tread. The advantages that offers are uncountable such as trust, safe transactions and incentives. However, the scientific public should be aware of the disadvantages of such a mechanism in order to weight both sides and distinguish if the mechanism will, indeed, be helpful in the deployment of a system or if it poses new problems, high complexity and new threats. Through the study of bibliography I identified that the majority of the authors declare the large proportion of advantages without mentioning the possible drawbacks. Taking this as incentive I decided to make a deep research and present the possible disadvantages analytically in this paper.

This section consists two subsections. In the first subsection we will describe in an analytical manner all the disadvantages of the blockchain mechanism that are identified in the present thesis. In the second subsection we will mention some of the most common attacks that are affect the blockchain mechanism because the attacks also reveal vulnerabilities in the architecture and as a result reveal drawbacks of the mechanism.

7.1 Drawbacks

To begin with, blockchain is not a DCS (Distributed Computing System). To be more precise, blockchain is a network that relies on properly functionality of nodes. The quality of the nodes determines the quality of the hole blockchain. For example, Bitcoin's blockchain gives the required incentive to the nodes to participate in the network. However, this is not happening for all blockchain networks. From the above we can identify that the blockchain is not a distributed computing system, in where the network does not depend on the involvement and on the participation of the nodes. In comparison, a distributed computing system in order ensure the record the transactions, preserves the transactional history. It is obvious now that the blockchain mechanism might be a distributed network, but it lacks the features that make a distributed computing system beneficial for the corporations.

Secondly, the blockchain is not as scalable as their counterpart centralized system. In Bitcoin network, the transactions are only depending on the network congestion. This problem is related to scalability issues with blockchain networks. To be more clear, the more nodes join the network, the more is slowing down. However, with the right evolution of the technology, scalability options are being integrated in the Bitcoin network. The solution is to do transactions off-blockchain and use the blockchain mechanism only for storing and accessing the related information.

Thirdly, some blockchain solutions consume much energy. Bitcoin uses the Proof-of-Work consensus algorithm that relied on the miners to do the difficult work. At the same time, the miners have the incentive to solve complex mathematical problems. The high energy consumption is what makes these complex mathematical problems not ideal for the real-world. Every time the ledger is updated with a new transaction, the miners need to solve the problems which means spending a lot of energy. However, not all blockchain solutions work in the same manner. There are other consensus algorithms that have solved the problem. For example, private networks do not have these problems as the number of nodes within the network is limited. Also, as there is no need for global consensus, they use efficient consensus methods to reach consensus. But, if you take the most popular blockchain network, like the Bitcoin, the problem still persists and needs to be solved.

We should also take into account the energy crisis that plague our society. We live in 2022 where a war in Ukraine is under progress. Under those circumstances is impossible to select and promote solutions that consumes so much energy, because we have both luck of energy and also the energy consumption is very expensive. A company wont afford so expensive solution if a more economical alternative exists.

Forth, the blockchain cannot go back, in other words, data is immutable. Data immutability has always been one of the biggest disadvantages of the blockchain. It is clear that multiple systems benefit from it, such as supply chain, financial systems etc. However, if we consider the networks functionality, we should understand that this immutability can only be present if the network nodes are distributed fairly. What I am trying to say is that a blockchain network can be controlled by an entity if he owns 50/100 or more of the nodes. In addition, the data once written cannot be removed. However, every person has the right to privacy especially now that the GDPR (General Data Protection Regulation) is primarily importance for more of the companies an a law has been set. However, if the same person utilizes a digital platform that runs on blockchain technology, then he will be unable to remove its trace from the system. In simple words, there is no way, he can remove his trace, something that indeed violated the right to privacy.

Fifth, the blockchains is inefficient. Nowadays, there are multiple blockchain technologies out there. If we pick up the most popular ones including the blockchain technology used by Bitcoin, we will find a lot of inefficiencies within the system. This is one of the big disadvantages of blockchain.First of all, when I tried to set up the bitcoin miner on myself, I quickly found out that the ledger can easily cross 100's of GBs. It was not efficient in data storage which can lead to storage problems for multiple nodes who want to become part of the network.

Clearly, there needs to be a better way to handle this as whenever the data is updated, nodes need to replicate it and update it. Moreover, the size of the blockchain grows unconditionally with more transactions and nodes. If it continues to grow, then the whole network is slowed down. This is not ideal for commercial blockchains where it is essential for the network to be fast and secure at the same time.

Sixth, the blockchain mechanism is not completely secure. In general, blockchain technology is more secure than other platforms. However, this does not mean that it is completely secure. There are different ways the blockchain network can be compromised. We will analyse some of the biggest attacks in blockchain architecture in the following subsection of the current section like the 51/100 attack, the double spending attack, the Denial of Service attack and the Cryprographic cracking.

Seventh, the users have their own bank - private keys. To make blockchain decentralized, it is important to give individuals the ability to act as their own bank. However, this also leads to another problem. In order to access the assets or the information stored by the user in the blockchain, they need private keys. The private keys are generated during the wallet creation process, and it is the responsibility of the user to take proper use of it. They also need to make sure that they do not share it with anyone else and keep them secretly. If they fail to do so, their wallet is in danger. Also, if they lose the private key, they will lose access to the wallet without distinction.

So, if we, as users who forget our private key, we are eventually logged out of our wallet and no one can get it back. This is a serious drawback as not all users are tech-savvy and have more chances to make mistakes. If there is a centralized authority that takes care of it, then it defeats the purpose of decentralization.

Eighth, the blockchain mechanism face cost and implementation problems. The underlying cost of implementing blockchain technology is huge. Even though most of the blockchain solutions including 'Hyperledger' are open source, they require a lot of investment from the organization that is willing to pursue it.There are costs associated with hiring developers, managing a team that excels at different aspects of blockchain technology, licensing costs if you opt for a paid blockchain solution etc. We also need to take care of the maintenance cost associated with the solution. For enterprise blockchain projects, the cost can go over a million dollars as well. So for businesses who like the idea of blockchain, but do not have the funds or budget to carry out, might need to wait more before they can jump into the blockchain bandwagon.

Ninth, the blockchain mechanism in order to be implemented it requires expertise knowledge. Implementing and managing a blockchain project is hard. It requires thorough knowledge from the business to go through the whole process. They need to hire multiple experts in the blockchain field that leads to the problem and hence it is counted as one of the disadvantages of blockchain. Not only that they also need to train their existing professionals on how to utilize blockchain but also they should ensure that the management team can understand the complexities and outcomes of a blockchain-powered business.

This way, they can understand their requirements and help transform their business processes to utilize blockchain. It is highly important to mention that the blockchain developers and specialists are harder to be find and they will cost more compared to traditional developers due to their demand and supply ratio.

Tenth, the blockchain lacks of interoperability. Another disadvantage that blockchain technology suffers from is interoperability. As mentioned in the last point, there are multiple types of blockchain networks which work differently, trying to solve the DLT problem in their own unique way. This leads to interoperability issues where these chains are not able to communicate effectively. The interoperability issue also persists when it comes to traditional systems and systems using blockchain technology.

Eleventh, the blockchain mechanism is difficult of Development. Most startups need to implement some complex protocols for achieving consensus and scaling off from the beginning. We cannot hastily implement an idea without deciding the features we need with the envision of expanding the application later. Once it is implemented, it is unlikely to add new features to it without the redeployment of the entire network, as blockchains cannot be edited.

Twelfth, The blockchain architecture is Time-Consuming. To add the next block in the chain miners need to compute nonce values many times so this is a time-consuming process and needs to be speed up to be used for industrial purposes.

Thirteenth, the blockchain mechanism face storage difficulties. Blockchain databases are stored on all the nodes of the network and that poses an issue with the storage as we have an increasing number of transactions that will require more storage over the time.

Fourteenth, the blockchain gives the opportunity to split the chain. The nodes, which are operating to the old software, won't accept the transactions in the new chain. This chain is creating with the same history as the chain, which is based on the old software. This second chain it is called fork. There are two fork's kinds – the soft fork and the hard fork.

The soft fork establishes the new rule-set to the blocks in the protocol. The nodes are updated to enforce the soft fork's rules. If the block, which was considered valid before, does violate the new soft fork rules, the block won't consider after the soft fork activation.

The hard fork is loosed the ruleset to the blocks in the protocol. This process is the same with the soft fork process, but the value and result of it is the opposite. If the block is gone through all the rules of the hard fork, the block will be accepted, even if the block was not in the chain before.

7.2 Attacks

Taking as reference the paper 'The Attacks and Problems of the Blockchain' [15] we will try to investigate the most common attacks in the blockchain mechanism. The Blockchain can be attacked by the different threats, which are connecting with the PoW (Proof of Work) and PoS (Proof of Stake) protocols. Most of the following attacks are very difficult to be implemented as they need to compromise the average computation ability.

1. Attack of 51/100 . It will happen when two miners are calculating the hash of the block at the same time and get the same results. In this case the Blockchain will split and as the result, users have two different chains, and both are considered as true. In the 51/100 attack, if an entity can control 51/100 or more of the network nodes, then it can result in control of the network. By doing so, they can modify the data in the ledger and also do double-spending. This is possible on networks where the control of miners or nodes are possible. This means that private networks are more likely to be safe from 51/100 attacks, whereas public ones are more vulnerable to this.

2. Double-spending . Princip of this attack is the same with the previous attack, but here can be used the split of the chain to spend the money again. Double-spending is yet another problem with the current blockchain technology. To prevent double-spending the blockchain network deploys different consensus algorithms including Proof-of-Stake, Proof-of-Work, and so on. Double spending is only possible on networks with a vulnerability to the 51/100 attack.

3. Sybil's attack . Its possible when one node accepts several essences, because the network can't authentically distinguish the physical machines. The Sybil's attack can help to fill the Blockchain with users under its control. It can lead to the previous two attacks and the ability to see all transactions with special programs.

4. DDos's attack . The attack consists of a large amount of the similar requests. There is the protection in the DDos's attack – size of the block up to 1 MB, size of each script up to 10000 bytes, up to 20000 of the signatures can check and maximums of the multiple signature is 20 keys. In a DDoS attack, the nodes are bombarded with similar requests, congesting the network and bringing it down.

5. Cracking of the cryptographic . It is possible if use the quantum algorithms such as 'Shora' which can break the RSA encryption. The scientists work on the cryptographical algorithms, which based on the hash functions. Another way the blockchain technology is not secure is that the cryptographic solution that it utilizes. Quantum algorithms or computing are more than capable of breaking cryptographic cracking. However, blockchain solutions are now implementing quantum-proof cryptographic algorithms.

8 Added Value

9 Conclusion

10 Future work

11 Annex

11.1 Blockchain mechanism

The blockchain mechanism [42], [41], [35], [6] is basically a distributed data structure that does not include a main repository (as in centralised systems). There exist multiple nodes of the network that can download locally all the blocks of the blockchain and validate their existence to be sure for the validity of the blockchain. This can happen with the validation of the transactions. The first leader block of the blockchain is called "genesis block". Every block, except from the genesis block, consider the cryptographic hash value of the previous block on the chain. Except form the simple nodes that simply download and validate the blockchain, there are and enhanced nodes that are called miners, that they are trying to produce a new valid block and increase the size of the blockchain. The new blocks are verified by the proof of work mechanism. When a miner finally succeeds to produce to new block, they rewarded by receiving an amount of cryptocurrency e.g., bitcoin or ethers. To alter the information of a block, for example to modify the transactions that compose the block, we need to compromise simultaneously 51 percent of the nodes, something extremely rare and in practise infeasible.

Having analyse the main mechanism of the blockchain we can go one step further and discus the numerous advantages of this decentralised idea. First, as we mentioned before, this mechanism can resist to alterations in the blocks. Users can not modify any block unless the 51 percent of nodes cooperate simultaneously. Secondly, it has the ability of tolerance to faulty nodes. Thirdly, each node trusts one another, and they collaborate harmonically without the need of a central coordinator or a third-party certificatory agency (CA). Fourthly, the blockchain gives the opportunity to every single node to download and verify locally all the blocks.

Having as initiative the blockchain mechanism researchers have developed numerous decentralized storage systems. That means that there is not a central repository, a server in the cloud. Instead, the information is stored in nodes. This is called peer-to-peer technology. IPFS (InterPlanetary File System) is a distributed file system take advantage of this technology and uses blockchain as file structure. IPFS's main ambition is to correlate all computing devices, that are distributed in the network, with the same file system. When a node saves data to IPFS, the IPFS returns a unique data's address. This address is a unique cryptographic hash string that is saved on chain.

Smart contracts are scripts of code and are executed on Ethereum. Ethereum Virtual machine is a platform that provides a distributed user interface via browser, and we can compile and run our smart contracts on the blockchain. Each smart contract has a unique id in the blockchain, and it is triggered by a new transaction that is been sent. Therefore, it reads the data that contains the input transaction and runs on every node in the network. The output that produces the smart contract is broadcasted and all the nodes of the network that participate in the transaction can verify the result of the smart contract. Expansive consensus protocols, such as the Proof of Work (PoW) first adopted by the Bitcoin currency, requires solving a computationally hard hashing problem for making valid and adding a new block and its adoption in an Internet of Things context requires support of other technologies, such as cloud computing. The long latency, low scalability, and poor environmentally friendly of Proof of Work have led to the development of other consensus mechanisms. Among the alternatives, the Proof of Stake protocols are the more known; they are based on the idea that there is something at stake and include different strategies.

11.2 Reputation-based trust management system

As defined in [39], [29], [18], [38], trust is a subjective criterion of what is expected by an individual. If the expected behaviour of the node matches with the real behaviour, then the individual trusts the node. It is highly mentioned that the trust metric can take binary value. The main ambition of a reputation system is to ensure that behaviours of the parties in a system reflect their reputations, and to prevent these reputations from being modified by unauthorized-malicious parties. In the system, a party with higher reputation value means that it is more trustworthy and has more honest behaviours. There are two kinds of reputation systems; centralized reputation systems (CRSs) and decentralized/distributed reputation systems (DRSs).

Centralized reputation systems (CRSs) [34]: To control the reputation of a network we can employed a centralised- third party authority. In the centralised environments all the scores are calculated and saved on central servers. With this mechanism the management of trust and the management of reputation of nodes it is an easy procedure. In centralised environments all nodes need to trust the central main authority, however this authority is not unfaulty. The server could make mistakes, forgery, and collusion. That can unfairly affect the nodes reputation. Moreover, there is lack of effective incentive mechanism, resulting in many default comments and ratings in the CRSs. These offer little reference for users to make decisions and the products that are selected by the buyers are not value for money/quality.

Decentralized/Distributed Reputation Systems (DRSs): The main ambitions of reputation systems in Peer-to-Peer environments include the identification of reliable resources, promoting behaviours of nodes to be honest and finally rating the quality of shared content. These Distributed Reputation Systems can be categorized as local reputation-based, global reputation-based, and combination of global and local reputation-based systems. In addition, the Distributed Reputation Systems in Peer-to-peer environment have revealed some of the known problems of the reputation systems, they still have a variety of problems. First, compared to the Centralised Reputation Systems, the reputation systems will cause high computational complexity for the Peer-to-Peer networks. Second, it is very hard to keep the reputation data up to data and accurate and distribute the reputation data to many dynamic peers. Finally, the Distributed Reputation Systems cannot resist the reputation modification as the common attacks such as unfair rating and collusion.

Advantages of reputation metric in an IoT P2P environment [17]:

1. Trust
2. Incentive for improvement to be more competitive, to augment our performance = our reputation and be more selectable by the buyers.
3. Diminished of cheating low reputation = bad seller
4. Makes easier the decision of buyers, which seller to trust, which product to purchase

5. In a federation the broker node can select nodes with high reputation to establish a safe environment for valid transactions without frauds
6. An incentive compatible reputation mechanism for ubiquitous computing environments
7. stimulate reputation information sharing and enforce honest recommendation elicitation.
8. Metric for trustworthiness evaluation of entities
9. reputation mechanism should not only show robustness against lies, but also stimulates honest and active recommendations.
10. ensuring that active and honest recommenders, compared to inactive or dishonest ones, can obtain the greatest number of honest (helpful) recommendations and thus suffer the least number of wrong trust decisions, as validated by simulation-based evaluation.
11. QoS-aware
12. reputation-based mechanism for isolating selfish nodes in ad hoc networks

11.3 List of known algorithms and references

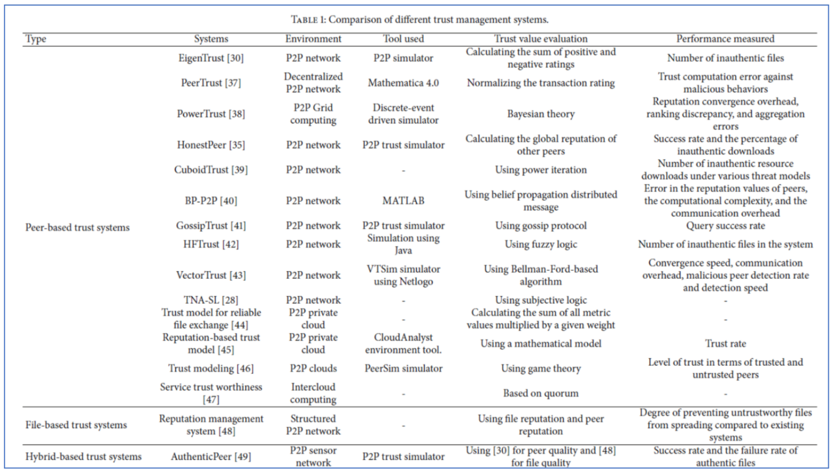


TABLE 1: Comparison of different trust management systems.

Type	Systems	Environment	Tool used	Trust value evaluation	Performance measured
Peer-based trust systems	EigenTrust [30]	P2P network	P2P simulator	Calculating the sum of positive and negative ratings	Number of inauthentic files
	PeerTrust [37]	Decentralized P2P network	Mathematica 4.0	Normalizing the transaction rating	Trust computation error against malicious behaviors
	PowerTrust [38]	P2P Grid computing	Discrete-event driven simulator	Bayesian theory	Reputation convergence overhead, ranking discrepancy, and aggregation errors
	HonestPeer [35]	P2P network	P2P trust simulator	Calculating the global reputation of other peers	Success rate and the percentage of inauthentic downloads
	CuboidTrust [39]	P2P network	-	Using power iteration	Number of inauthentic resource downloads under various threat models
	BP-P2P [40]	P2P network	MATLAB	Using belief propagation distributed message	Error in the reputation values of peers, the computational complexity, and the communication overhead
	GossipTrust [41]	P2P network	P2P trust simulator	Using gossip protocol	Query success rate
	HFTrust [42]	P2P network	Simulation using Java	Using fuzzy logic	Number of inauthentic files in the system
	VectorTrust [43]	P2P network	VTSim simulator using Netlogo	Using Bellman-Ford-based algorithm	Convergence speed, communication overhead, malicious peer detection rate and detection speed
	TNA-SL [28]	P2P network	-	Using subjective logic	-
	Trust model for reliable file exchange [44]	P2P private cloud	-	Calculating the sum of all metric values multiplied by a given weight	-
	Reputation-based trust model [45]	P2P private cloud	CloudAnalyst environment tool.	Using a mathematical model	Trust rate
	Trust modeling [46]	P2P clouds	PeerSim simulator	Using game theory	Level of trust in terms of trusted and untrusted peers
	Service trust worthiness [47]	Intercloud computing	-	Based on quorum	-
File-based trust systems	Reputation management system [48]	Structured P2P network	-	Using file reputation and peer reputation	Degree of preventing untrustworthy files from spreading compared to existing systems
Hybrid-based trust systems	AuthenticPeer [49]	P2P sensor network	P2P trust simulator	Using [30] for peer quality and [48] for file quality	Success rate and the failure rate of authentic files

Most popular (Short summary):

In EigenTrust [22], the global reputation of a peer in the system is calculated using the left principal eigenvector of a matrix of normalized local reputation values. In addition, an overall history of the system is available, and each peer is known and considered in the calculation of the reputation values, which is performed in a distributed and node-symmetric manner with minimal overhead on the network.

IoT Trust [10]

TNA-SL [21] This algorithm is used to discover trust networks between two parties and derive trust measures from such networks. Trust in TNA-SL is stored as an opinion, and each opinion consists of four tuples. These four tuples represent belief, disbelief, uncertainty, and a base rate, respectively. Subjective logic offers different types of operators frombinary logic and probability calculus aswell as specific operators for combining and merging different opinions. This variety of operators makes it possible to support a wide range of different applications and systems. Details regarding the opinion formulation and operators are addressed in the following sections.

[40] In peer trust factors such as the feedback and its scope, the credibility of the source, the context of transactions, and the community context. The metric of general trust combines the above factors and then significantly decreases ordinary threats, such as man-in-themiddle attacks, nodes in compromised bases, and tainted information being spread within the decentralized environment of the P2P. However, in this scheme, the underlying presumption is that the trust value of a peer is a measure of its reliability. Therefore, peers with higher trust values always provide more reliable feedback, but this is not always correct.

[4] The proposed BP/P2P system computes reputation and trustworthiness values by using a belief propagation based distributed message, passing algorithm between peers on a factor graph representation of a P2P network. Using BP/P2P, the reputations of peers are determined based on the quality of service a peer receives, and trustworthiness is determined based on the ratings provided by each peer after successful transactions. A comprehensive evaluation showed that the BP/P2P is efficient in calculating trust values, filtering malicious ratings, and reducing errors in the reputation values of peers. Moreover, compared with EigenTrust and PowerTrust, BP/P2P is more efficient in detecting and eliminating malicious nodes.

[44] The system leverages a gossip-based protocol to aggregate the score of a global reputation. Every peer randomly contacts the others and exchanges information regarding the reputation of the data in a periodic manner. The proposed gossip-based protocol is simple, does not require error recoveries, and provides controlled overheads compared with optimum deterministic protocols, which include information building of a data dissemination tree. In addition, GossipTrust provides a fast aggregation module of local trust scores, a new efficient scheme for storing reputation information, and secure communication using identity-based cryptography

[20] Hierarchical fuzzy trust management (HF Trust) makes use of fuzzy logic to model trusts. Every peer keeps records of all local exchanges, to determine if the peer has fulfilled the requirements or not.The factors related to trust are subjected to fuzzy inference by peers to produce a local trust index. All the data from every peer regarding local transactions is compiled by the HF Trust system, and every peer's global reputation is prepared. The seven significant parameters for evaluating trust are explained using an application that allows file sharing between peers. A considerable improvement in the performance of the P2P system has been demonstrated using this trust model, as it brings about a substantial reduction in the number of nongenuine files in the network.

[43] A Bellman–Ford-based algorithm is utilized for the quick compilation of trust scores. To analyze and compile trust values, the trust vector aggregation (TVAA) is proposed. Every single trust path in TVAA is collected, and the highest trust rating is given to a target peer. VectorTrust can be employed in decentralized and distributed networks, where no global trust data is available. As the complexity of both topology and P2P networks is increasing, VectorTrust scales effectively owing to its high speed of convergence and manageable costs.

[11] Three trust parameters, namely, contribution, trustworthiness, and quality of resource, are used to build four relations, and the global trust value of every single peer is calculated using the power iteration. CuboidTrust provided good results and brought about a substantial reduction in the number of inauthentic resource downloads in different threatmodels.Theparameter of trustworthiness was considered in

CuboidTrust and PeerTrust, and not in EigenTrust. Therefore, both CuboidTrust and PeerTrust perform efficiently, even in presence of various malicious peers in the system

[25] The set of honest peers with high reputation values are given greater roles in calculating the global reputations of other peers. HonestPeer dynamically selects the honest peers based on the quality of the provided files, rather than depending only on the static group of pretrusted peers, as in EigenTrust. Compared with the EigenTrust algorithm, HonestPeer has a better success rate and a minimal percentage of inauthentic downloads

[23] Trust model is another upgraded version of the PeerTrust model. This model has corrected certain flaws in the PeerTrust model and has also created a place for PeerTrust within the environment of the grid.This is achieved by modifying the definition of satisfaction criteria and introducing a decay function in the algorithm.The trust factor in the satisfaction criteria is responsible for handling all the requirements that can be satisfied by the resource source. Theprovider of resources is chosen from a performing grid by taking into account the basic requirements that the resource consumer wants to fulfill. The addition of a decay function, which is upgraded via a feedback trust calculation algorithm, forms the basic concept of the algorithm.

[45] It tilizes a TON to simulate the interrelationships of trust present between nodes. This scheme utilizes the power-law feedback properties of P2P networks and uses random selection power nodes, which are those with the best reputations. PowerTrust achievedmuch better results in terms of the precision of global reputation and the speed of aggregation. However, PowerTrust is prone to threats by malicious pretrusted peers.This is because, in such a model, the status of completely reliable peers (pretrusted peers are considered to be in EigenTrust) is given to power nodes [53]. Therefore, there is a high probability of severe system damage if power peers turn malicious

[8] [44] E. D. Canedo, R. Junior, and R. Albuquerque, "Trust model for reliable file exchange in cloud computing," International Journal of Computer Science Information Technology (IJCSIT), vol. 4, no. 1, 2012.

[12] [45] N. Dladlu and O. O. Ekabua, "Implementation of a novel peerto peer reputation-based trust management model in a cloud service provisioning environment," in The International Conference on Digital Information Processing, E-Business and Cloud Computing, Kuala Lumpur, Malaysia, 2016.

[31] [46] I. Petri,O. F. Rana, Y. Rezgui, andG. C. Silaghi, "Trustmodelling and analysis in peer-to-peer clouds," International Journal of Cloud Computing, vol. 1, no. 2/3,

p. 221, 2012.

[2] [47] J. Abawajy, “Determining service trustworthiness in intercloud computing environments,” in Proceedings of the 10th International Symposium on Pervasive Systems, Algorithms, and Networks, I-SPAN 2009, pp. 784–788, Taiwan, December 2009.

[27] [48] S. Y. Lee,O.-H.Kwon, J.Kim, and S. J.Hong, “A reputation management system in structured peer-to-peer networks,” in Proceedings of the 14th IEEE International Workshops on Enabling Technologies: Infrastructure for Collaborative Enterprises (WET ICE ’05), pp. 362–367, June 2005.

[24] [49] H. Kurdi, S. Alnasser, and M. Alhelal, “AuthenticPeer: a reputation management system for peer-to-peer wireless sensor networks,” International Journal of Distributed Sensor Networks, vol. 11, no. 11, Article ID 637831, 2015.

12 Bibliography

References

[1] Mousa A, Bentahar J, and Alam O. "Multi-dimensional trust for context-aware services computing". In: *Expert Systems with Applications* 172 (2021). DOI: 10.1016/j.eswa.2021.114592.

[2] J Abawajy. "Determining service trustworthiness in intercloud computing environments". In: *in Proceedings of the 10th International Symposium on Pervasive Systems, Algorithms, and Networks* (2009), pp. 784–788.

[3] S Asiri and A Miri. "A sybil resistant IoT trust model using blockchains". In: *IEEE International Conference Internet Things* (2018), pp. 1017–1026. DOI: 10.1109/Cybermatics2018.2018.00190.

[4] E Ayday and F Fekri. "BP-P2P: Belief propagation-based trust and reputation management for P2P networks". In: *in Proceedings of the 2012 9th Annual IEEE Communications Society Conference on Sensor* (2012), pp. 578–586.

[5] Shala B et al. "Blockchain and Trust for Secure, End-User-Based and Decentralized IoT Service Provision". In: *IEEE Access* 8 (2020), pp. 119961–119979. DOI: 10.1109/access.2020.3005541.

[6] Shala B et al. "Novel trust consensus protocol and blockchain-based trust evaluation system for M2M application services". In: *Internet of Things* 7 (2019). DOI: 10.1016/j.iot.2019.100058.

[7] M Boussard et al. "STewARD: SDN and blockchain-based trust evaluation for automated risk management on IoT devices". In: *INFOCOM WKSHPS* (2019), pp. 841–846. DOI: 10.1109/INFCOMW.2019.8845126.

[8] E D Canedo, R Junior, and R Albuquerque. "Trust model for reliable file exchange in cloud computing". In: *International Journal of Computer Science Information Technology* 4.1 (2012).

[9] Dellarocas Ch. "Efficiency through Feedback-contingent Fees and Rewards in Auction Marketplaces with Adverse Selection and Moral Hazard". In: *Sloan School of Management Massachusetts Institute of Technology Cambridge* ().

[10] I Chen, J Guoa, and F Bao. "Trust management for SOA-based IoT and its application to service composition". In: *IEEE Transactions on Services Computing* 9.3 (2016), pp. 3444–3449.

[11] R Chen et al. "CuboidTrust: a global reputation-based trust model in peer-topeer networks". In: *in in Autonomic and Trusted Computing* 4610 (2007), pp. 203–215.

[12] N Dladlu and O O Ekabua. "Implementation of a novel peerto peer reputation-based trust management model in a cloud service provisioning environment". In: *in The International Conference on Digital Information Processing* (2016).

[13] Fortino G et al. "A blockchain-based group formation strategy for optimizing the social reputation capital of an IoT scenario". In: *Simulation Modelling Practice and Theory* 108 (2021). DOI: 10.1016/j.simpat.2020.102261.

[14] Fortino G et al. "Using Blockchain in a Reputation-Based Model for Grouping Agents in the Internet of Things". In: *IEEE Transactions on Engineering Management* 67.4 (2020), pp. 1231–1253. DOI: 10.1109/TEM.2019.2918162.

[15] J. Golosova and A. Romanovs. "The Advantages and Disadvantages of the Blockchain Technology". In: *IEEE 6th Workshop on Advances in Information, Electronic and Electrical Engineering (AIEEE)* (2018), pp. 1–6. DOI: 10.1109/AIEEE.2018.8592253.

[16] Chung-Wei H, Anup K, and Munindar P. "Behind the Curtain: Service Selection via Trust in Composite Services". In: *IEEE 19th International Conference on Web Services* (2012).

[17] Fang H et al. "Reputation mechanism for e-commerce in virtual reality environments". In: *Electronic Commerce Research and Applications* 13.6 (2014), pp. 409–422. DOI: 10.1016/j.elerap.2014.08.002.

[18] Yahyaoui H et al. "Trust-Based Management in Iot Federations". In: *SSRN Electronic Journal* (2022). DOI: 10.2139/ssrn.4047084.

[19] Jo HJ and Choi W. "BPRF: Blockchain-based privacy-preserving reputation framework for participatory sensing systems". In: *PLOS ONE* 14.12 (2016). DOI: 10.1371/journal.pone.0225688.

[20] L Huaiqing, W Xuezhi, and LHaitao. "Hierarchical fuzzy trust management for peer-to-peer network". In: *n Proceedings of the 2009 Second ISECS International Colloquium on Computing, Communication, Control, and Management, CCCM* (2009), pp. 377–380.

[21] A Josang, R Hayward, and S Pope. "Trust network analysis with subjective logic". In: *Proceedings of the 29th Australasian Computer Science Conference* (2006).

[22] S D Kamvar, M T Schlosser, and H Garcia-Molina. "he EigenTrust algorithm for reputation management in P2P networks". In: *in Proceedings of the 12th International Conference on WorldWideWeb* (2003), pp. 640–651.

[23] D Kaur and J S Gupta. "Proposed P2P reputation-basedmodel to secure grid". In: *Proceedings of the International Conference on Recent Advances Trends in Information Technology* (2012).

[24] H Kurdi, S Alnasser, and M Alhelal. "AuthenticPeer: a reputation management system for peer-to-peer wireless sensor networks". In: *International Journal of Distributed Sensor Networks* 11.11 (2015).

[25] H A Kurdi. "HonestPeer: An enhanced EigenTrust algorithm for reputation management in P2P systems". In: *King Saud University—Computer and Information Sciences* 27.3 (2015), pp. 315–322.

[26] A Lahbib et al. "Blockchain based trust management mechanism for IoT". In: *AIEEE Wireless Com* 11.11 (2019), pp. 1–8. DOI: 10.1109/WCNC.2019.

[27] S Y Lee et al. “A reputation management system in structured peer-to-peer networks”. In: *in Proceedings of the 14th IEEE International Workshops on Enabling Technologies: Infrastructure for Collaborative Enterprises* (2005), pp. 362–367.

[28] Resnick P et al. “Reputation systems”. In: *Communications of the ACM* 43.12 (2000), pp. 45–48. DOI: 10.1145/355112.355122.

[29] Thanasis G Papaioannou and George D Stamoulis. “Effective Use of Reputation in Peer-to-Peer Environments”. In: *Computer Networks* 50.4 (2006), pp. 563–578. DOI: 10.1016/j.comnet.2005.07.024.

[30] Thanasis G Papaioannou and George D Stamoulis. “Reputation-based estimation of individual performance in collaborative and competitive grids”. In: *Future Generation Computer Systems* 26.8 (2010), pp. 1327–1335. DOI: 10.1016/j.future.2009.05.007.

[31] I Petri et al. “Trustmodelling and analysis in peer-to-peer clouds”. In: *International Journal of Cloud Computing* 1.2/3 (2012), pp. 221–380.

[32] R Di Pietro et al. “A blockchain-based trust system for the internet of things”. In: *Access Control Models and Technologies* (2018), pp. 77–83.

[33] R Di Pietro et al. “A blockchainbased trust system for the Internet of Things”. In: *Access Control Models Technol* (2018), pp. 77–83. DOI: 10.1145/3205977.3205993.

[34] Chen I R, Bao F, and Guo J. “Trust-Based Service Management for Social Internet of Things Systems”. In: *IEEE Transactions on Dependable and Secure Computing* 13.6 (2016), pp. 684–696. DOI: 10.1109/tdsc.2015.2420552.

[35] Dennis R and Owenson G. “Rep on the Roll: A Peer to Peer Reputation System Based on a Rolling Blockchain”. In: *International Journal for Digital Society* 7.1 (2016), pp. 137–150. DOI: 10.20533/ijds.2040.2570.2016.0137.

[36] Mehdi R et al. “Utility-driven data acquisition in participatory sensing”. In: (2013).

[37] Hien T et al. “A Trust and Reputation Model Based on Bayesian Network for Web Services. IEEE International Conference on Web Services”. In: (2010).

[38] Liu W et al. “Optimization of PBFT Algorithm Based on QoS-Aware Trust Service Evaluation”. In: *Sensors* 22.12 (2015). DOI: 10.3390/s22124590.

[39] Adewuyi A Cheng H Shi Q Cao J Wang X and Zhou B. “SC-TRUST: A Dynamic Model for Trustworthy Service Composition in the Internet of Things”. In: *IEEE Internet of Things Journal* 9.5 (2022), pp. 3298–3312. DOI: 10.1109/jiot.2021.3097980.

[40] L Xiong and L Liu. “PeerTrust: supporting reputation-based trust for peer-to-peer electronic communities”. In: *IEEE Transactions on Knowledge and Data Engineering* 16.7 (2004), pp. 843–857.

[41] Zhou Z et al. “Blockchain-Based Decentralized Reputation Management System for Internet of Everything in 6G-Enabled Cybertwin Architecture”. In: *Journal of New Media* 3.4 (2021), pp. 137–150. DOI: 10.32604/jnm.2021.024543.

[42] Zhou Z et al. “Blockchain-based decentralized reputation system in E-commerce environment”. In: *Future Generation Computer Systems* 126 (2021), pp. 155–167. DOI: `10.1016/j.future.2021.05.035`.

[43] H Zhao and X Li. “VectorTrust: Trust vector aggregation scheme for trust management in peer-to-peer networks”. In: *in Proceedings of the 18th International Conference on Computer Communications and Networks* (2009).

[44] R Zhou and K Hwang. “Gossip-based reputation aggregation for unstructured peer-to-peer networks”. In: *in Proceedings of the 21st International Parallel and Distributed Processing Symposium* (2007).

[45] R Zhou and K Hwang. “PowerTrust: a robust and scalable reputation system for trusted peer-to-peer computing”. In: *IEEE Transactions on Parallel and Distributed Systems* 18.4 (2007), pp. 460–473.