Fuzzy Systems and Soft Computing
ISSN : 1819-4362

Novel trust consensus protocol and blockchain-based trust evaluation system for M2M application services

A.Pandi[1], Satya Ranjan Biswal[2]
[1,2]Department of Computer Science & Engineering, Gandhi Institute of Education & Technology, Bhubaneswar

Keywords:
Distributed Ledger
Blockchain
Consensus Protocol
Trust
Security
M2M
Service and Application

abstract

The increasing number of intelligent Machine-to-Machine Communication (M2M) devices in the end-user domain provide good resources for creating and sharing M2M applica-tion services. Therefore, transferring the role of a traditional centralized service provider to decentralized peers (end-users) acting as service providers is very promising. However, the future of decentralized M2M application services which are independently provided or consumed by several end-users in the M2M community depends on trust. Untrustworthy peers trying to deploy malfunctioning services for others mitigate the benefits of decen-tralized systems. Nowadays, the concept of distributed ledger and blockchain has an in-creased popularity regarding trustless computing among communities operating without centralized authorities. This research publication provides a comprehensive analysis and approach merging the concepts of M2M application services, trust and distributed ledger technologies. Moreover, this publication presents an optimized Trust Evaluation System which is used to ensure trustworthiness among peers in a M2M community. To improve the Trust Evaluation System the integration of blockchain for data storage is introduced. Additionally, blockchain technology is used to extend the existing trust model of the Trust Evaluation System to enable tamper-proof data and detection of untrustworthy peers. This publication reviews several existing approaches in the academic and industry sector to highlight the limitations of the blockchain regarding the consensus and proposes a novel Trust Consensus Protocol. This research publication also provides a practical evaluation of the proposed protocol.

1. Introduction

Devices, such as lights, cameras, and household appliances are becoming increasingly smarter and with complex con-trol/monitoring functionality. These intelligent devices can be used to realize specific M2M applications addressing several application fields such as smart home, building surveillance, energy or traffic management, and electro mobility. Most of these devices are present in the personal end-user environment and are therefore not accessible for external entities. The authors in [1] present an approach to integrate the end-user and the resources part of the end-user's personal environment in the M2M application provision process. Their approach introduces a fully decentralized and end-user friendly M2M ser-vice platform where internal resources are made accessible for external entities as a service in order to be integrated into external applications.

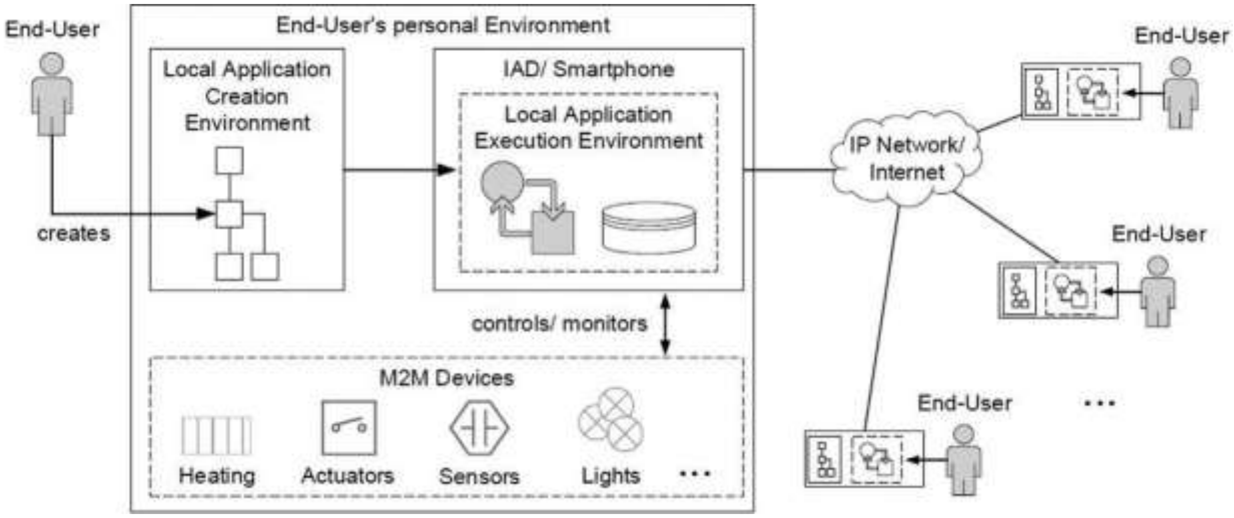


Fig. 1. Decentralized M2M application design and execution [4].

One disadvantage of decentralized M2M service platforms is that there is no centralized entity controlling the service creation process made by the end-user. Also, there is no authority that ensures that the created services meet the con-ditions for being deployed in the community. Moreover, the decentralized character of the approach could lead to several security issues performed by attackers [3]. Therefore, trust relationships between peers and services are necessary to miti-gate possible security attacks. These relationships are built by evaluating the trustworthiness of peers and services based on their behavior and the interactions with others.

Centralized trust evaluation systems are coming with single point of failure issues and decentralized ones rely on the reliability of the actors participating in the evaluation process. Blockchain, as a distributed ledger technology and tamper-proof data storage system can increase the integrity of the computed trust data and the whole trust process. However, the blockchain technology comes with some challenges regarding the way how consensus in the network is achieved specifically the required computational power which is a main drawback for using it in M2M environments.

This publication presents a comprehensive approach for optimizing and solving several identified limitations in decen-tralized M2M service platforms, trust evaluation systems, and distributed ledger technology. The following subsections (1.1 – 1.3) give a short overview of the state of arts for the three mentioned domains. Section 2 explains in detail an optimized trust evaluation system with blockchain integration for decentralized M2M service platforms. Moreover, it presents a trust evaluation example for a M2M application service. Section 3 reviews several existing consensus protocols and derives the strengths and weakness of them. This evaluation is used to introduce in Section 4 a new trust-based consensus protocol, which is practically evaluated in Section 5. Section 6 gives a conclusion of the presented analysis and approaches.

1.1. Decentralized M2M application services

To avoid limitations associated with single point of failure architectures, resource flexibility, and platform independency, the authors in [1] present a completely decentralized M2M system architecture where the M2M service platform itself is not provided by a platform operator but by end-users of the platform itself. End-users are able to design individual M2M application services and make them available for other end-users or central service providers. End-users have also the possibility to cooperate with each other in order to provide complex M2M application services. The author in [4] defines that "an M2M application service is a capability provided by a peer to other peers" and distinguish between technical peers (e.g. device) and non-technical peers (e.g. human). Thus, an end-user can provide services to others by being supported by technical devices (e.g. DSL router, personal computer, smartphone) where the M2M service platform is running.

To design and deploy M2M applications the resources which are available in the personal environment (permanent place of residence or current location) of the end-user are used. At permanent locations of the end-user, the Integrated Access Device (IAD) (e.g. DSL router) is used to execute the M2M service platform. The author in [4] state that "the IAD represents the M2M-Gateway in end-users' environment, which execute the M2M applications close to the end-user and processed the data where they are collected". At mobile locations the end-user's smartphone is used as execution system. The end-user's personal environment is illustrated in Fig. 1.

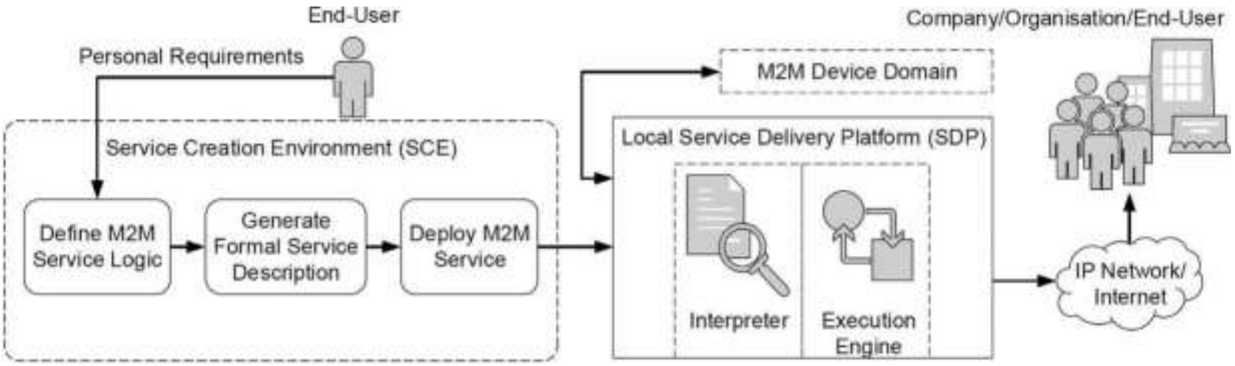

End-User
Personal Requirements
Service Creation Environment (SCE)
Define M2M Service Logic
Generate Formal Service Description
Deploy M2M Service
M2M Device Domain
Local Service Delivery Platform (SDP)
Interpreter
Execution Engine
Company/Organisation/End-User
IP Network/ Internet

Fig. 2. M2M Application service creation and provision process [4].

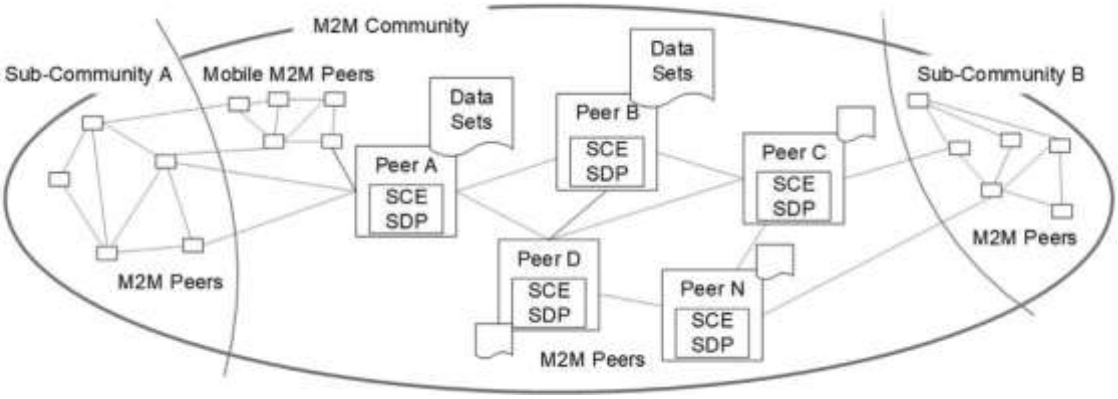


Fig. 3. P2P-based M2M community.

The M2M service platform for decentralized M2M application service provision presented in [1] includes a Service Man-agement Framework (SMF) which consists of a local Service Creation Environment (SCE) and a Service Delivery Platform (SDP) (illustrated in Fig. 2). Moreover, the SMF includes all available devices and services present in the personal envi-ronment of the end-user and integrates also remote services which are provided by other end-users. The Service Creation Environment (SCE) provides a Graphical User Interface (GUI) for designing graphically the behavior of a M2M application service (M2M application logic) and for combining building blocks representing the M2M service components, M2M devices and multimedia service components. Then, the SCE automatically transforms the M2M application logic into a formal ap-plication description which can be parsed, interpreted and executed by every underlying execution environment. Thus, the designed service can be provided as a service to other end-users.

The authors in [4] propose to use a Peer-to-Peer (P2P) network for communication and information storage between the peers which avoids centralized entities. The information exchange and communication between the peers for service utilization and application generation signaling is done directly (end-to-end) using M2M communication protocols. The in-formation storage is done through a distributed data storage using a P2P overlay network (e.g. Chord) which is formed out of all existing end-user nodes (peers).

An M2M community [2] serves for social networking between all the nodes and can be used to create interest groups by providing different sub-communities. It can also be used in order to address different application fields or geographical locations. End-users and the application services they provide can be part of several sub-communities at the same time. Fig. 3 shows the structure of P2P connected peers within a M2M community, where some of them are acting as service providers by providing services to other peers (service consumers).

Avoiding central instances and transferring all responsibilities for service provision to end-users can increase the risk of failures and malicious behaviors of the peers. Less technical knowledge or not manageable behavior of end-users could lead to serious problems in the M2M community such as wrong, malicious or not working services. In order to overcome such attacks trust among the peers should be ensured [3]. Therefore, an overall trust evaluation system which considers the decentralized nature of M2M application services is required and will be introduced in Section 2.

In the further course of this research, the term peer is associated to the end-device (e.g. IAD, smartphone, computer) used by the end-user as an M2M service platform and also part of its personal environment.

1.2. Trust evaluation for decentralized M2M application services

Several centralized and decentralized trust management systems for ensuring trust among peers within the M2M domain are reviewed in [5]. The review has shown that most of the existing approaches do not provide a secure and reliable way to store the trust data which are computed during the trust evaluation of the peers. Moreover, the existing trust management systems provide trust metrics which consider only one trust aspect of the community. For instance, they are focusing on recommendation or honesty between peers but are not analyzing the overall participation of the peers in the community, such as functionality or performance of services they provide, reliability of the information they share, or activity in several tasks of the community. Additionally, the review in [5] shows that none of the existing approaches in M2M provide a possibility for evaluating trust of new services entered to the community.

To overcome existing problems with new services joining the M2M community and their unknown trust score (trust level), the authors in [5] propose a system which tests the functional behavior and the performance characteristics of an incoming service based on the service description published from the service provider. For functional testing of the services a decentralized and autonomous framework was introduced in [5] which enables to generate automatically test cases from the system model of the service. To avoid centralized entities the authors in [5] propose to integrate all peers part of the M2M community in the testing process. Therefore, they will receive notifications about the arrival of new services and will independently from each other generate and execute test cases against the new services. By using the performance model of the new service, the node acting as a test agent will also perform performance testing in order to confirm the participation willingness of the new M2M service. The test results of both procedures, functional and performance testing, will be used by a trust evaluation function to evaluate the trust score about the initial behavior of the new service.

One problem of the computed data during the trust evaluation steps is the possibility to fake this information and to be-tray other nodes regarding the trust score of a node or a service it provides. Section 2 proposes the integration of blockchain technology to improve the secure storage limitations of existing trust management approaches. Moreover, it introduces an enhanced Trust Evaluation System covering several aspects of a peer, such as the quality of services it provides or the be-havior of a peer regarding several community tasks.

1.3. Distributed ledger technology – blockchain

The literature review has shown that the blockchain technology is a subbranch of the so-called distributed ledger. Ac-cording to [6], a distributed ledger is essentially an asset database that can be shared across a network of multiple sites, geographies or institutions. All participants within a network can have their own identical copy of the ledger. Any changes to the ledger are reflected in all copies in minutes, or in some cases, seconds. The security and accuracy of the assets stored in the ledger are maintained cryptographically through the use of “keys” and signatures to control what can be done by whom within the shared ledger [6].

The authors in [7] differentiate between two architectures used in distributed ledger approaches. The first is blockchain, where transactions are stored in a block and all blocks are linked cryptographically with each other rendering the data tamper-proof. Second architecture is the Directed Acyclic Graph (DAG), where nodes in a graph are representing transactions and the edges of them indicate the direction of confirmation between the transactions.

Because of the advantages regarding decentralization, transparency and the immutability [34], this research publication is focusing on a blockchain as data structure. Nowadays, there is an increasing hype for using blockchain to secure systems in several application fields. Blockchain is a distributed database which records all transactions, agreements, contracts and/or other digital assets between peers participating in that community. A famous representative for using blockchain to store all transactions in the network is Bitcoin [8]. According to [9], trust is established due to the fact that everyone in blockchain has a direct access to a shared “single source of truth”. All transactions, which are public, comprise specific information, such as date, time, and number of participants. Every peer in the network has a copy of the blockchain and the transactions are validated by the so-called miners using cryptographic principles. These enable nodes to automatically recognize the current state of the ledger and every transaction in it [9]. As stated in [9], a corrupted transaction will be immediately refused by the nodes since they do not reach a consensus for validating that transaction. The authors in [10] state that “once a new block is formed, any changes to a previous block would result in different hashcode and would thus be immediately visible to all participants in the blockchain”.

Despite the numerous benefits of blockchain, there are still some limitations regarding the consensus between the nodes. A comprehensive review of existing consensus protocols is presented in Section 3. Finally, Section 4 introduces a novel Trust Consensus System, making it powerful to be used not only in blockchain processes but also for other tasks such as service provisioning between peers. Section 5 concludes with a theoretical and experimental evaluation of the new proposed consensus protocol.

2. Decentralized and optimized trust evaluation system for M2M

This section introduces a novel Trust Evaluation System which considers several aspects such as the initial trust level of peers, no centralized entity involved in the evaluation part, secure storage of sensitive data, and motivation for participation on community activities. An example is also provided in order to understand the workflow of this approach.

The trustworthiness of peers and services in decentralized M2M communities is one of the key goals of this research and should be ensured by an optimized and reliable trust evaluation system. Peers with bad intentions also providing malfunc-tioning services should be rated with a low trust score and disabled from the system. Malfunctioning services are classified in this research as services which: fail to provide the functionality they are initially claiming to provide; fail to provide the performance capability they are initially claiming to provide; perform on-off attacks in order to harm the system; do not provide the scalability they are claiming to provide. On the basis of the trust level of peers the security level in a decentral-ized environment can be increased.

2.1. Basic description of the trust evaluation system

As mentioned in Section 1, the Trust Evaluation System introduced in [5] provides an efficient way to calculate the trust score of peers by testing the functional and performance behavior of the new services they provide. This approach overcomes the limitation of several other trust system which do not consider the initial trust score of services or assign a default initial trust score without validating the services.

To include all activities done by a peer part of the M2M community in the trust evaluation process, this research proposes a Trust Evaluation System which analyzes the services provided by a peer, and also considers the participation of the peer in community tasks. As mentioned in Section 2, one of the community tasks is performing test activities against other services and peers. This role is called Test Agent. In order to enable a fully decentralized Trust Evaluation System, all peers part of the M2M community can do the tasks of the Test Agent, by testing the functionality and the behavior of new services. The motivation behind doing these tasks is the possibility to increase the own trust level which is used for several processes in the M2M community.

The Trust Evaluation System proposed in this research publication consists of three parts for evaluating the trust score of a peer: Service Trust Evaluation - evaluates the services the peer is providing; Behavior Trust Evaluation - evaluates the behavior of a peer based on the integrity of a service; Task Trust Evaluation - evaluates the activities as a Test Agent or other tasks done in the M2M community. The Service Trust Evaluation part consists of testing the service after the deployment to compute the initial trust score (Service Testing), monitor continuously the behavior and the performance of service based on several parameters, and consider the rating of a service done by all peers of the M2M community based on their own experience. In the Behavior Trust Evaluation part, the integrity of service information, such as trust data or service descriptions, are checked and in case of modifications the peer performing these changes is low rated. The Task Trust Evaluation part consist of monitoring the number of testing actions performed by a peer which is acting as a Test Agent. The more successful test actions are done by a peer the better will be the trust score of the peer. Besides testing activities there is also the possibility to consider other tasks for Task Trust Evaluation.

2.2. Blockchain integration for trust evaluation

The main benefits of using blockchain are ensuring data integrity and non-repudiation. Moreover, blockchain also pro-vides secure access and identity management possibilities. Thus, it provides great possibilities to make the M2M environ-ment more secure (an overview of possible security integration scenarios is given in [35] and in [36]). Regarding the integra-tion of blockchain in the application field of M2M/Internet of Things (IoT) the literature review provides several publications [11–14] dealing with secure data storage and data integrity in relation with blockchain. However, none of the publications [11–14] consider the blockchain technology for using in connection with trust management systems and the computed trust scores.

Managing trust scores in a decentralized M2M community is challenging because of the increasing number of peers joining and leaving the network and of their possible malicious behavior by removing or changing data which harm the system. The Trust Evaluation System is used to calculate the trust score of an observed service or peer. Based on this trust level, other end-users are able to check how much they can trust an M2M application service. As mentioned in the previous sections, one issue is that the evaluated trust data can be manipulated or removed from the network. In order to optimize the storage system of the trust management system and to ensure tamper-proof trust data, the authors in [15] propose to store all the evaluated trust data in the blockchain.

After a Test Agent has performed the trust evaluation steps and has computed the trust level (range from 1 to 5) of an M2M application service, it will send a blockchain transaction to the end-user providing the evaluated M2M application service. The blockchain transaction (see Fig. 4) consists of the trust level, Service ID, Service Instance (contact information about the service provider) and the Test Agent Username. This transaction will be broadcasted to all peers' part of the blockchain network and is going to be part of a block. Next, this block has to be validated from other peers in order to achieve a consensus for an identical version of the blockchain. An overview of the integration of blockchain for storing trust data is shown in Fig. 5.

To achieve a consensus for validating a transaction and creating a block the literature provides several consensus mech-anisms [16]. The first consensus mechanism introduced in blockchain is the Proof of Work (PoW) [6]. The PoW consists of nodes acting as "miners" by trying to solve a cryptographic puzzle that requires high computation power to solve. The node who solves first this puzzle will validate and add a new block of transactions to the blockchain. The performing activities are rewarded for the first successful miner for validation transactions and creating new blocks [6]. However, performing PoW

Fig. 4. Structure of the data part of a blockchain transaction [15].

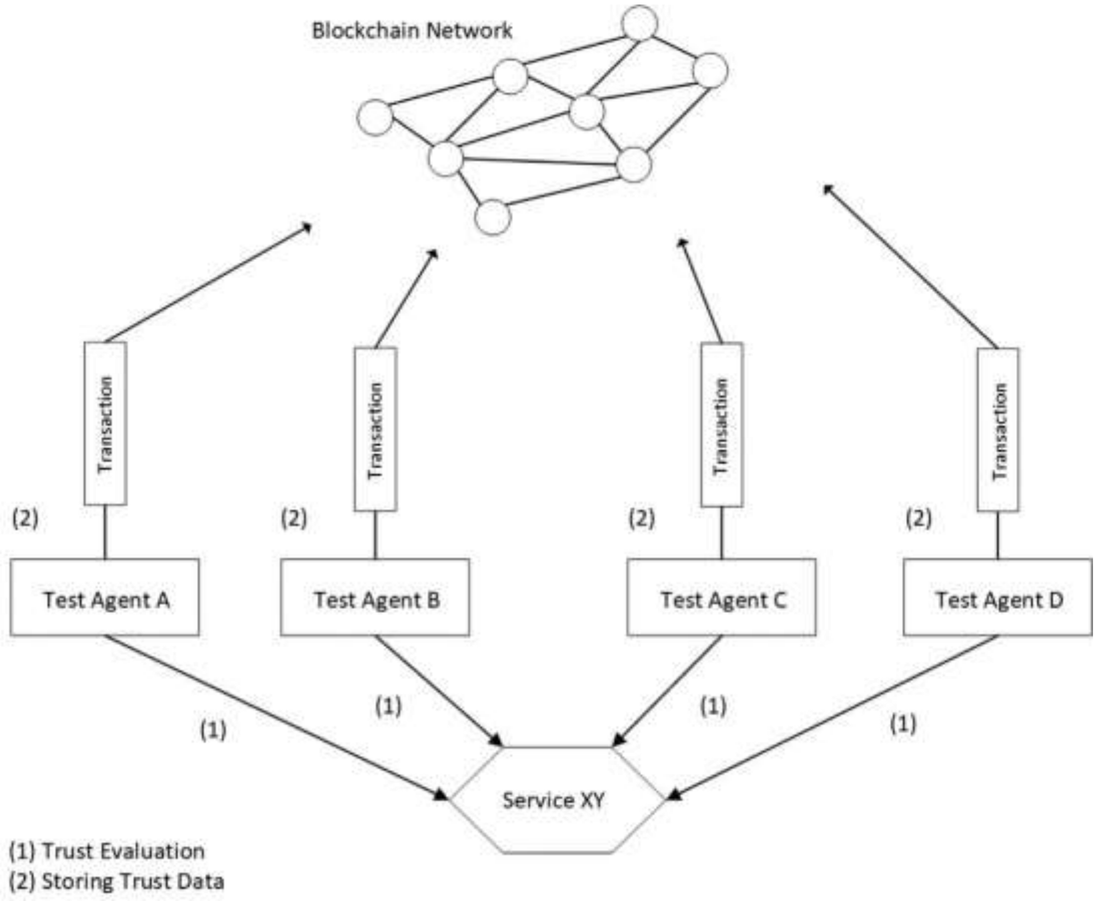


Fig. 5. Integration of blockchain for storing trust data [15].

requires hardware with high computational power and the mining process is an energy-intensive process. Another consen-sus mechanism is Proof of Stake (PoS) which tries to offer a more efficient way to validate transactions in the blockchain. This mechanism does not require high computing power and selects randomly nodes for mining based on several criteria (depends on the PoS version) [17]. In order to benefit from this energy saving character, this research proposes to initially consider the PoS consensus mechanism for validating new transactions and creating blocks within the presented blockchain approach. Later on, a review of several consensus protocols is going to be presented in this research paper.

2.3. Ensuring integrity using blockchain and P2P overlay

To increase the reliability in the M2M community the authors in [15] introduce the combination of the P2P overlay used in [1] and the blockchain network for trust data storage and for trust verification. Moreover, it is proposed in [15] to use this combination for several aspects such as end-user and service registration, and Interface Description verification for triggering the trust evaluation process. Fig. 6 shows an overview about the different aspects the combination of P2P overlay and blockchain network is proposed to use for in decentralized M2M application services and their trust computation.

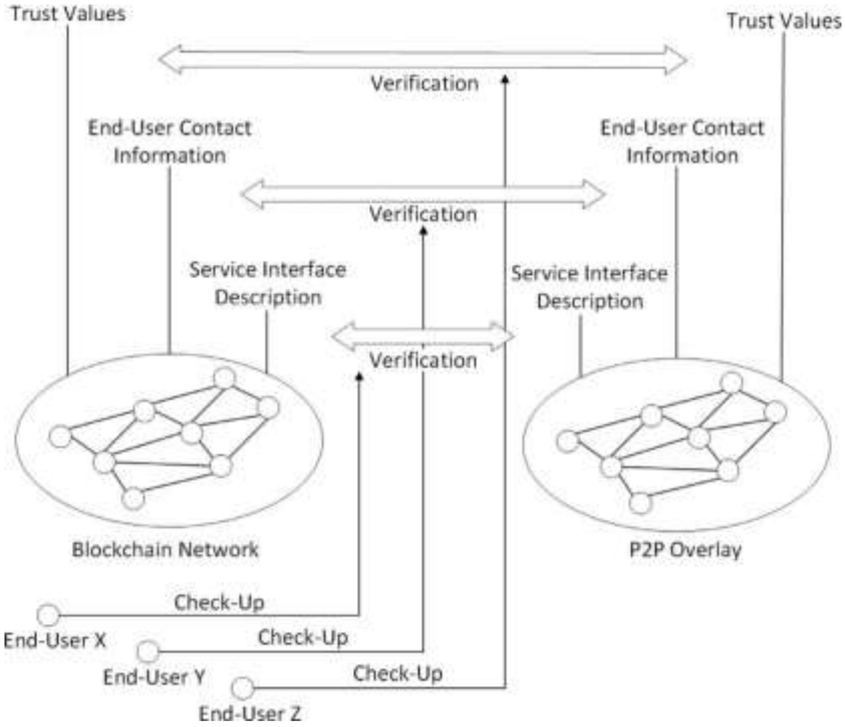


Fig. 6. Overview about the usage of P2P and blockchain combination [15].

2.3.1. Trust data verification

The data stored in the blockchain is tamper-proof and is used for integrity check-ups. However, to enable efficiency the P2P overlay introduced in [1], it is also proposed to use for storing the evaluated trust data of the M2M application services. The P2P overlay will only store the current status of the trust level and every end-user part of the M2M community has write permission for it. For simple and quick look-up about a trust score the end-user is able to do it first in the P2P overlay. Moreover, the end-user has the possibility to check the trust history about a M2M application service in the blockchain for past trust data entries in warily situations. For integrity check-ups, the end-user requires the data from the P2P overlay and compares/verifies it with the data from the blockchain. If the compared scores match, the end-user can rely on these entries and can continue selecting or not selecting the M2M application service [15].

2.3.2. Other storage inputs

The P2P overlay introduced in [1] is used for the management of M2M application services. The end-user providing a service will register the service using the P2P overlay network by storing the Interface Description (IDS) of the service and its associated personal temporary contact information. The storage in the P2P overlay could lead to several security attacks as evaluated in [3]. Besides, two main issues can appear. First, after storing the first version of an IDS in the P2P overlay, the service provider could change the IDS and then overwrite the first version without any notification. Thus, other end-users would not have the opportunity to check if there are many versions of the IDS, or what kind of changes happen. This information obtained by the end-user could help for a better overview about the behavior of the service provider. Another problem arises for Test Agents which are continuously performing tests for functional verification and trust eval-uation of M2M application services in the community. The Test Agents will test M2M application services when they are first deployed. However, they do not know when and after which time intervals to test an existing M2M application service. Another issue is that end-users providing a service are only identified by their temporary contact information which can be changed during their lifetime. An attacking peer is able to register several times with different identities and providing harmful services trying to break down the M2M community.

To solve the above-mentioned problems, the authors in [15] propose to also use blockchain for storing the information about service providers and the services such as the Interface Descriptions. Due to the fact that data is overwritten and changed in the P2P overlay, Test Agents would have the possibility to identify this change in an Interface Description in order to perform functional verification and trust evaluation of M2M application services. Moreover, other end-users would also have the possibility to verify the information about a service or service provider by comparing the information in the P2P overlay and the blockchain network. The Interface Description of M2M application services is retrieved in the P2P over-lay at regular intervals. This description is compared to the description stored in the blockchain. If this information stays permanently stable, no further action must be taken from other nodes. If there are changes, the M2M community respec-

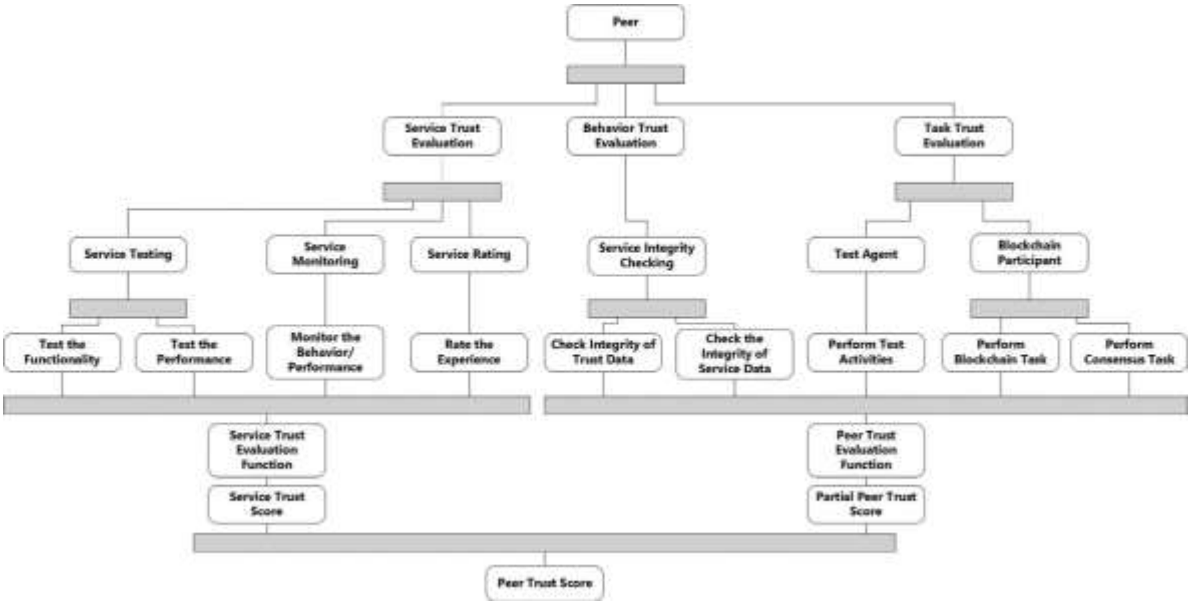


Fig. 7. Overview of trust evaluation system architecture.

tively the end-users acting as Test Agents will start testing the M2M application service in order to verify its functionality. This verification could increase or decrease the trust scores of the M2M application service [15].

The blockchain within the M2M community can also be used in order to check which end-user has provided what services and when in the past. In order to mitigate the problem with end-users and their different temporary contact infor-mation and the many services they can provide, this research proposes to define a permanent username for every end-user and to store these information both in P2P and blockchain for further integrity analysis and to comprehend what services they offer [15].

2.4. Trust evaluation system architecture and trust evaluation steps

To evaluate the trustworthiness of peers and the services they are providing it is required to have a trust evaluation system which covers not only experiences between peers but also other parameters such as functionality and performance of the services. Moreover, the participation of the service provider or peer in several community tasks should also be motivated and honored with a good trust score. Thus, an overall trust score of peers and services can be computed and used by other peers to decide for a provider-consumer relationship or not.

The following Fig. 7 presents an overview about the architecture of the introduced Trust Evaluation System. The new proposed Trust Evaluation System aims to derive the trust score of a peer based on its services it provides, its behavior, and its participation in the community tasks. These trust evaluation parts are defined as Service Trust Evaluation, Behavior Trust Evaluation, and Task Trust Evaluation.

The Service Trust Evaluation part consists of Service Testing, Service Monitoring, Service Rating. In Service Testing the functional behavior and the performance of a service based on the service description is done [18]. The functional testing verifies the functionality of a service and concludes with the result if the service is behaving like it is mentioned in its sys-tem model (SCXML service description and service interface description). In parallel, performance testing is done in order to confirm the participation willingness of the service regarding several requests by service consumers. Functional and perfor-mance testing are used to compute the initial trust score of the new service and to include the service in the service list of the M2M community. Literature review [5] has shown that there are no known approaches in the context of M2M to deal with services which have no initial trust score. Most of the approaches assume a specific trust level for the initial service and deal only with the future trust level of that service. After the service is part of the M2M community it is not anymore a new service but it is considered as an existing service. To evaluate the trust score of an existing service, this research paper proposes to monitor the behavior and the performance of the service by considering parameters such as number of online/offline actions or ratio of positive/negative responses. Additionally, the service is rated by other users based on their experience with the used service. The results of Service Testing, Service Monitoring, and Service Rating are computed using a Service Trust Evaluation Function which concludes with a Partial Trust Score of the Service.

Another part of the Trust Evaluation System is the Behavior Trust Evaluation which considers the integrity of a service to compute the trust score of a peer. Therefore, this research publication introduces the integration of blockchain to benefit from its tamper-proof feature to check the integrity of data which is stored in the blockchain (on-chain) with the data which is stored outside the chain (called off-chain e.g. P2P overlay). The data stored in the blockchain as mentioned in the previous section are service data about the services and trust data about the computed service trust scores. If the data in the off-chain is different with that in the on-chain, then the information in the off-chain has been changed by a user. The “last edited information” in the on-chain will be used to punish the user who makes false information about a trust score or a service data by decreasing his trust score.

To evaluate the trustworthiness of a peer this publication also considers another part called Task Trust Evaluation. Based on the participation on several tasks (e.g. Test Agent, Blockchain node) in the M2M community the peer can increase or decrease his own trust score. In order to ensure a decentralized environment without centralized entities this research propose to distribute all community tasks among the participating nodes. One of the tasks is to perform the test, mon-itoring and rating tasks which are used in the other parts of the Trust Evaluation System. Another task is to maintain the blockchain which is proposed for sensitive data storage. The maintenance of the blockchain consists on all functions such as block deployment, block validation and voting (are going to be explained in Sections 3 and 4). The results of the Behavior Trust Evaluation and Task Trust Evaluation are used to compute the Partial Trust Score of the peer using the Peer Trust Evaluation Function. To compute the overall trust score of the peer the Service Trust Scores of all services provided by the peer are combined with the Partial Peer Trust Score of the peer gained by behavior evaluation and task participation.

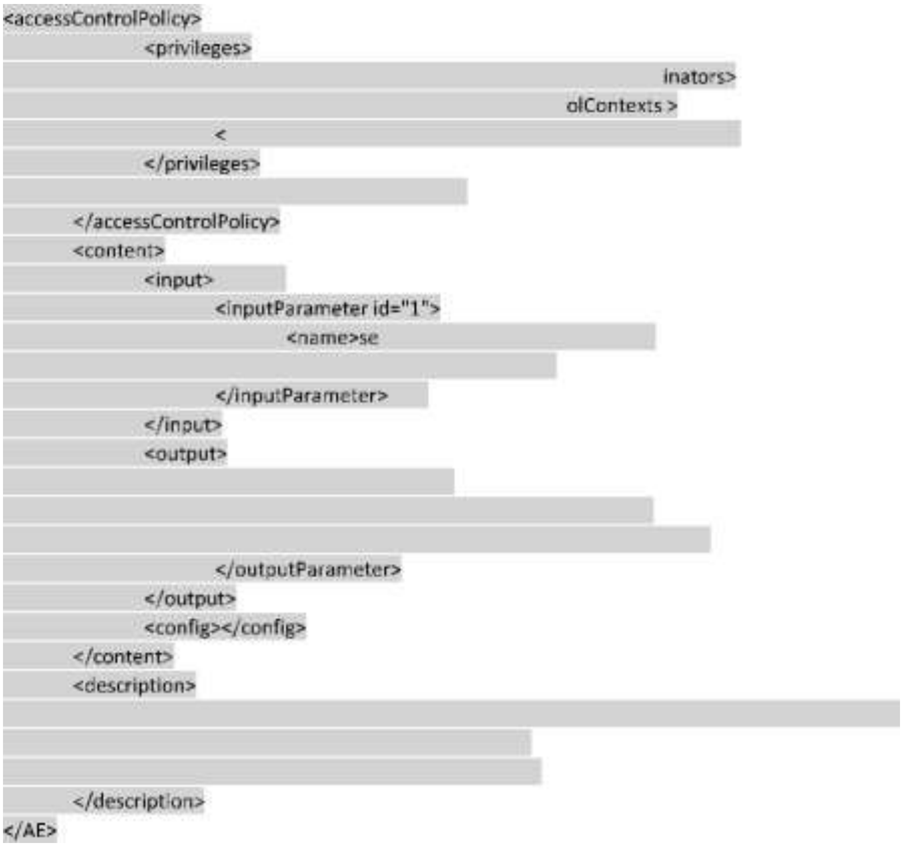
2.5. More details and example of proposed trust evaluation system

In the following an example for the trust evaluation of an M2M application service part of the M2M community is presented. This research assumes a trust score (trust level) range from 1 to 5 where 5 is the best and 1 the worst score.

Information about the example service: *PeerXYZ* is offering a service with ID-Nr. 1331, which provides the functionality to get and evaluate weather data information. The weather data information is provided by a different service (is not relevant). Based on the evaluated weather data, *Service1331* determines which user has to be informed. Prerequisite is that there are users which have registered that service in order to get be alarmed for a specific input. *Service1331* is able to receive two kinds of inputs: user contact information input and rain or snow weather data information. For instance, there are three users registered for *Service1331*. User A and User C are registered to get an alert for rain weather data, User B wants to get informed for snow.

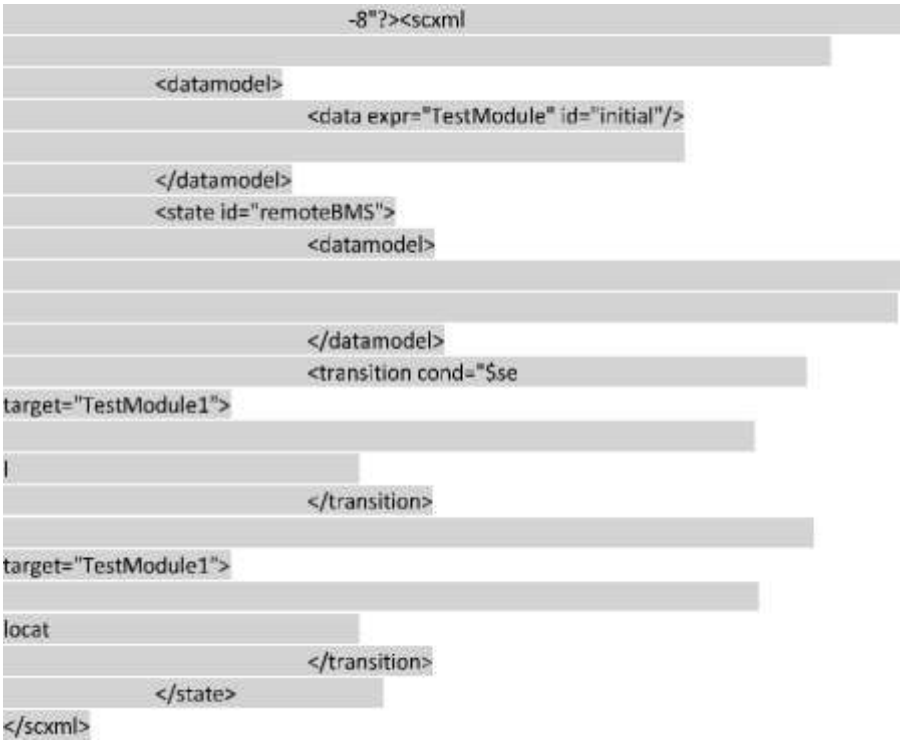
General workflow of the service and its trust evaluation: an end-user (peer) is creating or deploying *Service1331* for the first time. After the end-user has defined the service logic of that service (using the Service Management Framework), he generates a Service Interface Description (see Fig. 8) and an SCXML Service Description (see Fig. 9) of that service [1]. These two descriptions are stored in the P2P overlay (Distributed Hash Tables such as Chord). Afterwards *Service1331* is deployed in the M2M community. Other peers’ part of the M2M community will receive a notification that there is a new service in the community without trust information. Thus, peers do not have the possibility to determine if they can trust or not the new service at the beginning. Therefore, the initial trust level should be evaluated. It is proposed to test the initial functional behavior and performance of a new service. The testing should detect possible malfunctioning behavior of the service which intentionally or unintentionally is provided by the peer (service provider). These tests can be done by other peers’ part of the M2M community. Every peer has the possibility to test and evaluate the behavior of that service by requesting the service information (interface description and SCXML service description). After getting the service information, the peer acting as a test-agent will use the service-information in order to generate test cases which can be used to test the functional behavior and the performance of the service. For instance, after the test cases are executed against the service the test report gives a result of 80% of pass test cases.

The second step of the trust evaluation of the new M2M service is the initial performance testing. Therefore, the perfor-mance capabilities defined in the service specifications (not illustrated here) are analyzed and from these model’s perfor-mance test cases are generated. In the service specifications the peer providing Service1331 defines the maximum response time and the maximum number of requests with which can deal the service. From these definitions test cases for per-formance tests are generated and against the system executed. This example assumes that Service1331 passes 60% of the performance test cases. The outcome of the functional and performance behavior of Service1331 will be used to generate an initial trust level for that service, as there is no pre-information about that service. Thus, the results of functional and performance testing are combined in order to calculate the average of successful test cases (70%). This result is sent to the Service Trust Evaluation Function which calculates the initial service trust level of Service1331. For simplicity this example will provide only simple mathematical average calculation for computing the trust level. If there is a trust score range from 1 to 5 then 70% of pass test cases would result in the trust score 3,5. Thus, the initial trust level of Service1331 is evaluated as “average trustworthiness”. For data integrity reasons this result is stored in the blockchain and in the P2P overlay (e.g. Chord). Fig. 10 shows an overview of the steps for evaluating the initial trust level of a new service provided in the M2M community. All the M2M community members will have the possibility to know the initially trust behavior of the new service. They have the possibility to use or not to use that new service based on that trust score.



```
<accessControlPolicy>
                <privileges>
                                                                                    inators>
                                                                        olContexts >
                        <
                <privileges>

        </accessControlPolicy>
        <content>
                <input>
                        <inputParameter id="1">
                                <name>se

                        </inputParameter>
                </input>
                <output>



                        </outputParameter>
                </output>
                <config></config>
        </content>
        <description>



        </description>
</AE>
```

Fig. 8. Service interface description of *service1331*.



```
                                        -8"?><scxml

                <datamodel>
                                <data expr="TestModule" id="initial"/>

                </datamodel>
                <state id="remoteBMS">
                                <datamodel>



                                </datamodel>
                                <transition cond="$se
target="TestModule1">

l
                                </transition>

target="TestModule1">

locat
                                </transition>
                </state>
</scxml>
```

Fig. 9. SCXML service description of service1331.

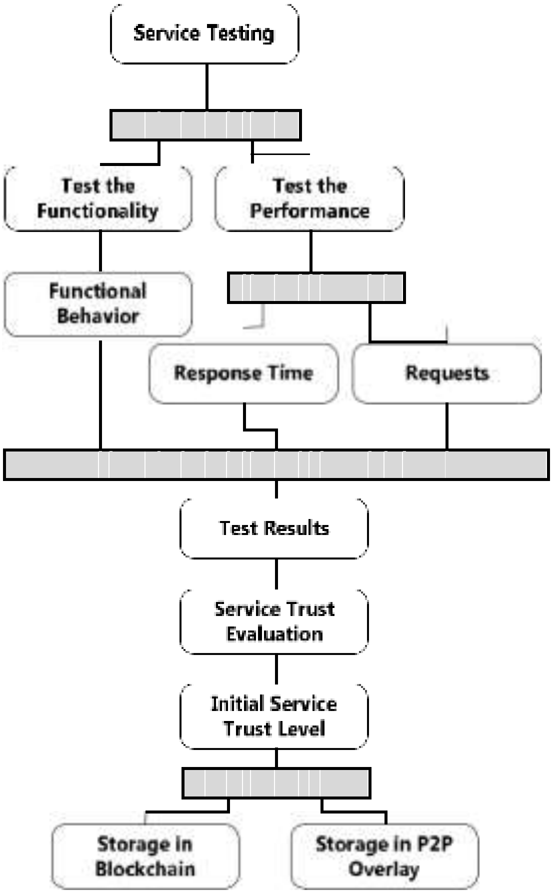


Fig. 10. Service trust evaluation for new service.

After the service is initially tested and evaluated it is part of the M2M community, if the trust level is above a prede-fined trust level. After a while a new service will change its status to an existing service which is used by several other peers. The trust evaluation of an existing service is done through two processes: service monitoring and service rating (see Fig. 11). Service monitoring is done by the Service Management Framework introduced in [1] (part of the peer pro-viding that service) which monitors independently different parameters such as the number of online/offline actions of a service. The service monitoring is also done by the other peers' part of the M2M community. Other parameters are the number of attendance in various applications, number of execution nodes, number of positive/negative responses, service online/offline time. For each parameter a threshold has to be defined which are used for calculation of the evaluation re-sults. For instance, *Service1331* in this case performs an average monitoring result of 90%. Next to service monitoring, other peers will have the possibility to rate *Service1331* based on their own experience. This example assumes that the rating score for *Service1331* derived from all other peers using that service is 95%. The monitor and rate results are combined and the average percentage of 92,5% is calculated. This result is used from the Service Trust Evaluation Function to calculate the partial current trust level of Service1331. In this case, 92,5% in the trust score range from 1 to 5 means the trust score 4625.

In order to calculate the total trust level, the initial trust level of 3,5 (explained at the beginning of this section) and the partial current trust level are combined with each other. Calculating the average of the sum of 4625 and 3,5 results with a current total trust level of 4,0625. This is the current total trust level of *Service1331*. This trust level is stored for data integrity reasons in the blockchain and in the P2P overlay (such as Chord). Moreover, this result is con-sidered for the evaluation of the trust level of the peer which is providing the service (this will be explained in the following).

Besides the Service Trust Level, there is also a Peer Trust Level. A Peer Trust Level is determined for all peers acting as service providers. The trust evaluation of a peer is done through four processes: Service Trust Level Average, Service Integrity Checking, acting as a Test Agent, and acting as a Blockchain Participant. The Service Trust Level Average is a trust metric which considers all trust levels of all services provided by the peer. In this case, *PeerXYZ* is providing only *Service1331*, so the Service Trust level Average (or partial peer trust level) would be 4,0625 out of 5. The second process, Service Integrity

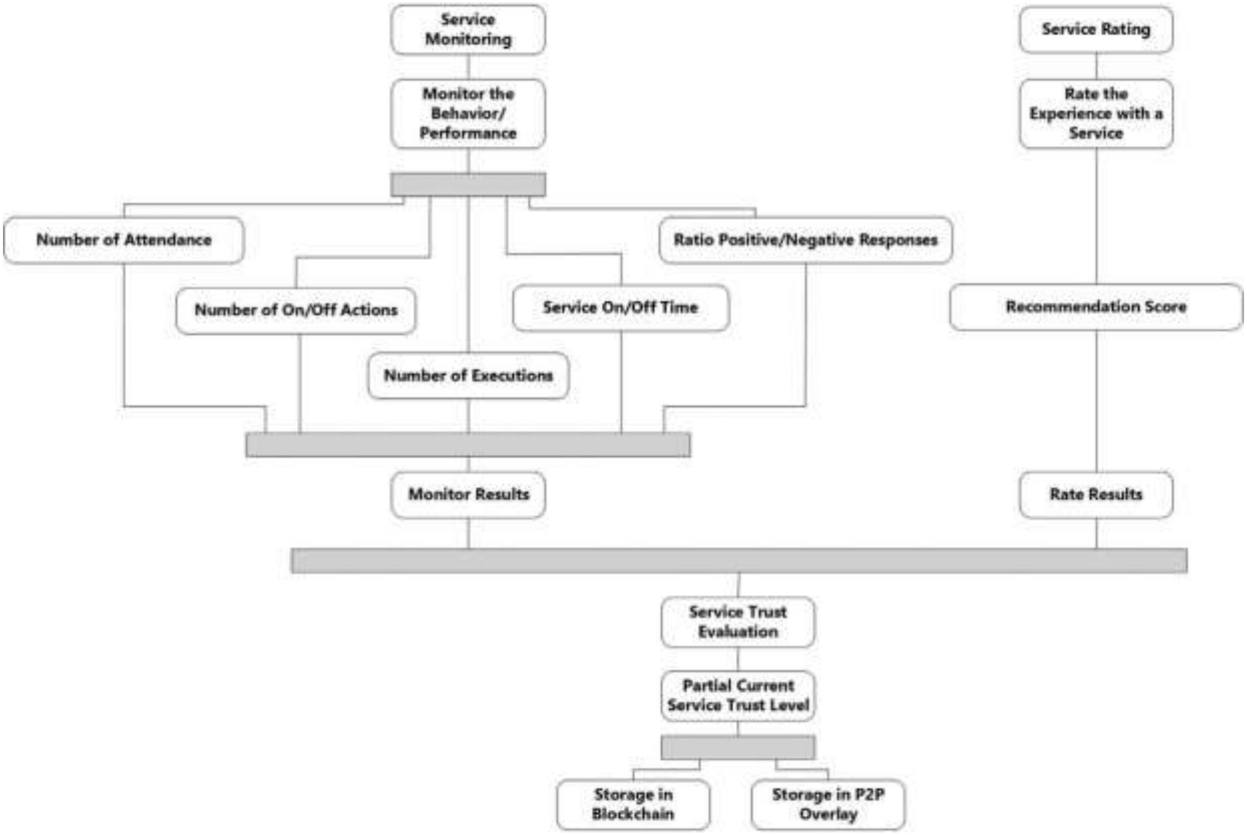


Fig. 11. Service trust evaluation for existing service.

Checking (see Fig. 12) consists of checking periodically the entries in the blockchain and in the P2P. This process is also done by other peers' part of the M2M community. Possible changes should be fixed and the peer who has changed the entries should get a low trust percentage for the Service Integrity Check process. In this case, the integrity check-up is successful, the partial peer trust level for this process is 5.

The third trust evaluation process for peer trust evaluation is the participation of the peer acting as a Test Agent. As mentioned at the beginning of Section 2, all peers part of the M2M community can act as Test Agents in order to test and evaluate other services and peers. The motivation to act as a Test Agent is getting a higher trust level. A high trust level in the M2M community is necessary in order to be able to make the own services more attractive for others and to be also able to get access to other services provided by others. In our case, *PeerXYZ* is acting very rare as a Test Agent. The number of testing actions is very low and based on a predefined threshold, the partial trust level for participating as a Test Agent is 2.

The fourth trust evaluation process for peer trust evaluation is the participation of the peer as a Blockchain Node. The Blockchain is used to store relevant peer and service information and the maintenance of the blockchain is done by the peers' part of the M2M community. The motivation to participate in the blockchain processes is the same as for the role of the Test Agent: getting a higher trust level. Every peer acting as a Blockchain Node will get a good trust score for a successful created block or validation of other blocks and will get also a bad trust score for a failed block (block which does not fulfil the blockchain rules, block which is created to harm the system) or block validation. Fig. 13 shows an overview about Peer Trust Evaluation for Community Task Participation. In this example, *PeerXYZ* is not participating in the blockchain community, receiving the partial trust level 1 out of 5 for that.

Finally, Table 1 shows the trust evaluation results of *PeerXYZ* considering several trust aspects introduced in this chapter.

The average calculation of all the above listed partial trust levels of *PeerXYZ* results with the total trust level 3,85 out of 5. This result is also stored in the blockchain and in the P2P overlay (such as Chord). Every other peer will have the possibility to check the trust level of the peer and to check the trust level of each of the services provided by the peer. Other peers will have the possibility to decide if they want to trust the service provider, or the new/existing service provided by them.

The Trust Evaluation Procedures for peers and services are connected with each other, disabling the possibility for bad peers to harm the system by providing malfunctioning services. Moreover, the trust evaluation procedures enable a fully

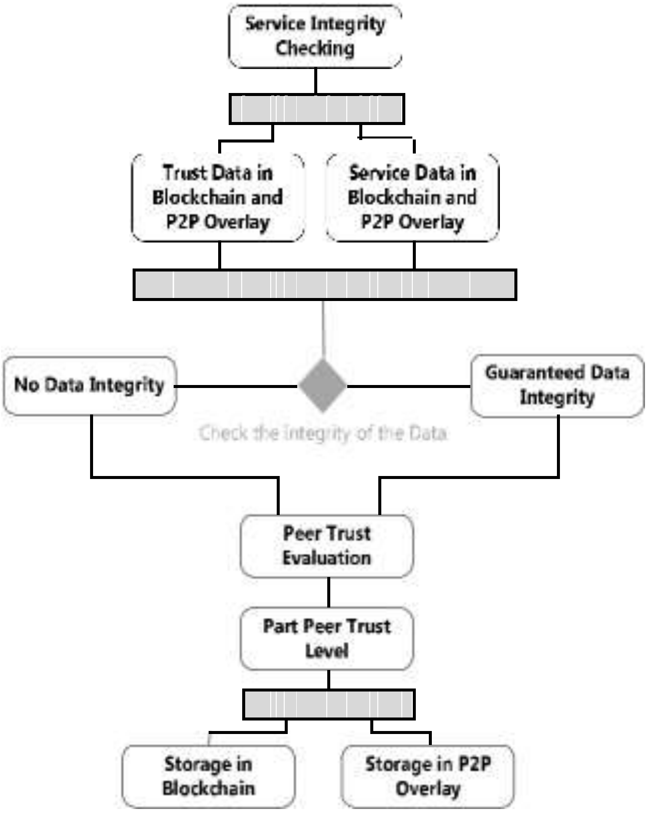

Service Integrity Checking
Trust Data in Blockchain and P2P Overlay
Service Data in Blockchain and P2P Overlay
No Data Integrity
Guaranteed Data Integrity
Check the Integrity of the Data
Peer Trust Evaluation
Part Peer Trust Level
Storage in Blockchain
Storage in P2P Overlay

Fig. 12. Service integrity checking.

Table 1
Example - Trust evaluation results of *PeerXYZ*.

Trust Evaluation *PeerXYZ*	
Trust Metric	Result
Service Testing	3,5
Service Monitoring	4,5
Service Rating	4,75
Service Integrity Checking	5
Test Agent and Blockchain Activities	1,5
Total Trust Level	3,85

decentralized environment without the need of a centralized entity for testing services, for trust evaluation and for data storage, making the Trust Management System compliant to a fully decentralized M2M service provisioning environment with an M2M community where decentralized M2M services are provided.

3. Consensus protocols used in distributed ledger approaches

This section is reviewing several existing consensus protocols used in distributed ledger approaches. Moreover, it intro-duces requirements which has to be fulfilled by a consensus protocol in order to deal with the decentralized nature of the M2M community. Finally, the evaluation results with the strengths and limitations of the protocols against these require-ments are illustrated.

In previous sections the integration of blockchain technology has been introduced in order to benefit from several ad-vantages of this technology. Moreover, blockchain is introduced for storing information such as trust evaluation data and service data in M2M environments. The storage of these information in the blockchain network increases the reliability and the availability of the data.

In order to agree for the same copy of the ledger, there is a need of a consensus between the nodes. In the distributed ledger terminology, there exist several definitions [19–22] for the terms of consensus, consensus protocol, and consensus mechanism. One of them is defining the consensus as a process to reach a global view of all transactions between the

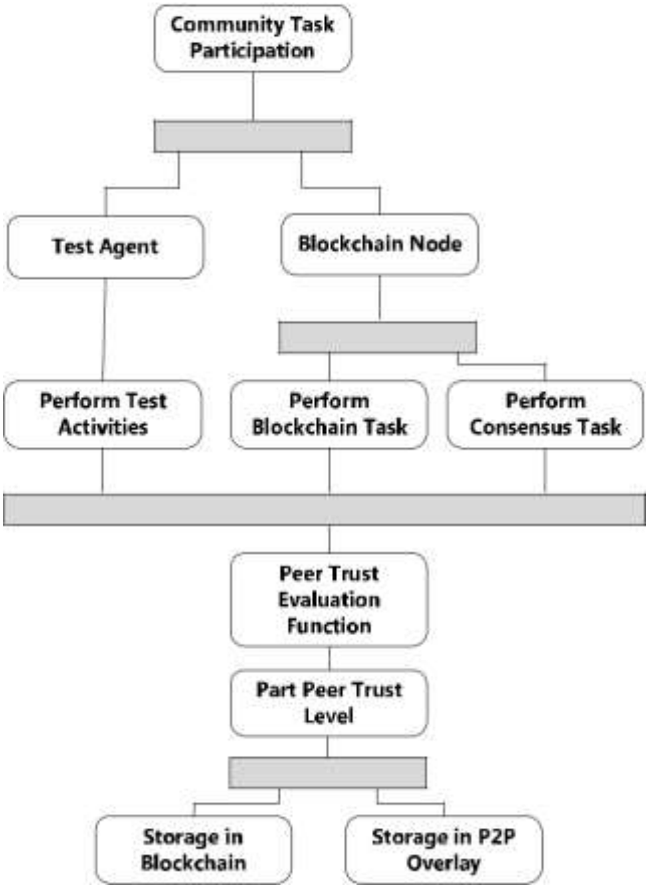


Fig. 13. Community task participation.

nodes in the blockchain [22]. Another definition [20] states that consensus is "a series of procedures from approving a transaction as an official one and mutually confirming said results by using" consensus mechanisms or protocols. A consen-sus mechanism is defined in [21] as "a set of steps that are taken by all, or most, nodes in order to agree on a proposed state or value". Main properties of a consensus are "agreement indicating that if two non-faulty participants decide they decide on the same block, validity indicating that the decided block should be one of the blocks that were proposed and termination indicating that eventually a correct participant decides" [19].

The literature review in scientific libraries and in the industry are providing a considerable amount of consensus proto-cols. The review done in this publication is focusing on two categories of consensus protocols: the first category consists of the most relevant consensus protocols used in the distributed ledger community. The second are consensus protocols which are adding trust elements to their consensus reaching steps.

3.1. Requirements for an ideal consensus protocol for M2M

Integrating distributed ledger technology for data storage in the M2M community comes with several challenges. One of them is that most of the peers' part of the M2M community are light-weight devices which do not have the compu-tational capability to perform high computing actions to maintain the distributed ledger and to achieve consensus among other peers. Therefore, one of the main requirements for a consensus protocol is a protocol which does require less *com-putational effort (1)* for performing the rules to achieve consensus. Another issue is the way how a peer is selected to add a new transaction or block to the ledger. It could be that a peer with bad intention is selected to deploy new transaction into the ledger and instead of adding real transactions, the node is favorizing fake transactions for confirmation. Therefore, the consensus protocol should ensure a *trusted selection of the block creator (2)* is required. Distributed ledger technologies provide a secure way to store data, however the trustworthiness of the data and the data originator are not aim of it. The consensus protocol should consider the differentiation between the transactions and the transaction originator trustworthiness. Besides the selection of the block creator, the blocks that are created should also be trusted which means they should not include fake transactions *(trusted block creation (3))*. Therefore, *trusted transactions (4)* should be consid-ered for inclusion in the consensus process. Another very important requirement is the *decentralized architecture (5)* of the distributed ledger network and the participating peers. There should be no static single entity or master peer which main-tains a part or the whole network. This ensures the defense of single point of failures or monopoly of peers which tries to harm the system. In order to motivate peers for correct participation in the consensus process, there should be a *trust reward/punishment (6)* system which honors or declass peers for their consensus contributions. A *dynamic and reliable trust model (7)* which is used to evaluate peers should be also part of the consensus process. Moreover, the trust model should generate reliable trust scores and should enable dynamic allocation of trust scores. Additionally, the consensus protocol should ensure that the community and the participating peers are *robustness/resilient against fake transaction attacks (8)* be-cause it could be that there are several malicious peers trying to harm the blockchain system by flooding fake transactions or peers trying to add fake new blocks to the blockchain. Finally, the decisions done by the peers for accepting new blocks to the blockchain should be reliable *(Reliable Validation Decision Process (9)*).

3.2. Review of traditional consensus approaches

The Proof of Work (PoW) consensus protocol is introduced in Bitcoin and is one of the first mechanism for achieving consensus on the same ledger between the nodes in a blockchain. The PoW consists of nodes acting as “miners” by trying to solve a computationally puzzle called hash. The node who solves first this puzzle will validate and add a new block of transactions to the blockchain. The performing activities are rewarded for the first successful miner for validation. [8] How-ever, performing PoW requires hardware with high computational power and the mining process is an energy-intensive process.

Another consensus mechanism is Proof of Stake (PoS) which tries to offer a more efficient way to validate transactions in the blockchain. This mechanism does not require high computing power and selects randomly nodes for mining based on several criteria (depends on the PoS version). The first version of PoS defines proof of ownership of a currency and the coin age consumes by a transaction as criteria to select nodes for being able to add new blocks. In case of different concurrent chains in the network the blockchain with the highest score is selected as the main chain. This score is computed based on the consumed coin age of every transaction which is part of the block. [17] The PoS protocol minimizes the computing effort and provides more energy efficiency. However, selecting leaders based on the stake ownership percentage could lead to the problem of centralization and monopolization.

One of the extended versions of PoS is the Delegated Proof-of-Stake (DPoS) where the block producers are elected by other nodes. The votes are weighted based on the network stake of each voter. In DPoS the number of block producers is fixed, and every block producer is allowed to produce one block per round. If a block producer does not perform the right actions he can be voted out by the community. The DPoS provides a pretense democracy by providing voting. [23] However, similar to PoS one of the disadvantages of this protocol is that as the votes are weighted based on the stake which is owned by each node, rich nodes could increase their impact on the network and building a monopoly. Another problem is the fixed small number of block producers that can be elected. This increases the problem of centralization in the system.

Nano is a consensus protocol which uses a block-lattice structure where each block contains only one transaction and every node (account holder) has its own blockchain (not the whole but only a single view about it). A sending transaction from the sender of the funds and a receiving transaction from the receiver are required to send funds from one account to another. If there is a conflict on a transaction the system can start a voting mechanism where representatives chosen on behalf of account holders vote for transactions and their voting power is calculated based on the sum of all balances of account that have chosen them. In order to mitigate double-spending attack Nano uses a light-weight version of the PoW. [24] The Nano protocol is similar to the Proof-of-Stake protocol and lead to the problem of supernodes in form of representatives. Moreover, the problem of monopoly is also here present.

A different consensus approach is provided in Ripple where each node has a Unique Node List (UNL) and a ledger. The UNL contains a list of nodes which are chosen by a node with the assumption that they will not behave maliciously. The consensus works in that way that a node votes by comparing the transaction received by the UNL with other transactions from previous rounds or other nodes if they are matching. Transactions that receive a negative vote will be considered for the next round of the consensus or they are going to be discarded from the network. The voting mechanism runs until the transaction receives 80% of the votes which qualifies them to be included in the ledger. [25] Ripple is an energy-saving protocol which relies on voting based on transaction similarity and thus does not require PoW computing. However, a drawback in Ripple is that the consensus is done by a fixed number of peer leading to a less decentralized system.

Another consensus protocol is introduced in IOTA which uses Tangle as an underlying distributed ledger technology, and which is based on a directed acyclic graph (DAG). In a Tangle all transactions are linked with each other where unconfirmed transactions are called tips and confirmed transactions sites. In order to deploy a new transaction to the community, a node first needs to validate two previous transactions by performing a light-weight PoW (used to avoid double-spending) and other validating steps (for instance, if the transaction is in conflict with the history of the tangle). After validating two other transactions, the current transaction is linked to them and waits until it is validated by others. Additionally, IOTA adds weight to the sites (transactions). The weight of each site is calculated based on the time a node spent to do PoW for confirming that transaction. Each transaction also has a cumulative weight which is the sum of its own weight plus the sum of own weights of all transactions that approve this transaction. [26] Tangle is a light-weight consensus protocol which uses a coordinator in order to protect the system from double-spending attacks. Therefore, the coordinator has periodically to issues transactions called milestones. Other transactions are confirmed if they are referenced by a milestone [27]. The integration of a coordinator is a centralized element in Tangle and can be considered as a drawback.

3.3. Review of trust-based consensus approaches

The authors in [28] propose to use a blockchain to log service transactions part of a crowdsourcing site in it. To avoid the limitations of PoW, they introduce a consensus mechanism where the blockchain is maintained by a ledger management and the leader of it is selected using an election algorithm which is based on votes. Moreover, the selected leader selects a service transaction validation group to validate transactions which has to be included in the blockchain. The list of service transaction validation group is sent to other members part of the blockchain which vote based on their own trust database for or against that list. Based on the received votes the leader forms the validation group and broadcasts the final list to the network. The validation nodes select transactions for including them in the next block and broadcasts it to other nodes. These other nodes called consortium nodes, also vote on the proposed transactions by the validation nodes. Votes of both the validation nodes and consortium nodes are summarized by the leader in order to select the winning transactions to be included in the block. The consensus protocol introduced by [28] avoids the use of Proof of Work and adds trust to the whole process. However, deciding for a leader for blockchain management leads to the problem of centrality and single point of failure. Moreover, the authors do not describe the motivation of the participating nodes in the overall process. There are also missing information regarding the local trust databases whether or not they differ from node to node. Another point is that the authors consider all transactions to be part of the blockchain without checking the trustworthiness of the transactions.

Another trust-based consensus protocol is proposed in [29] where the authors combine a so-called trust graph encoded in the blockchain and PoW. Based on the trust graph the trust score of every node can be derived and this score is used in order to “assign the amount of energy that has to be sent for traditional PoW”. First, a trust graph is created where every node expresses its own trust towards other nodes. Based on that graph trust metrics for all consensus nodes are computed and in order to avoid centralization all nodes with the highest trust score are discarded from the consensus part. The consensus protocol introduced in [29] selected every round randomly and in proportion to its total trust, derived from the trust graph a node which can decide for transactions to be deployed in the blockchain. The authors in [29] also claim that the difficulty of the cryptographic puzzle depends on the trust score of the selecting node proposing a new block. Nodes with a higher trust level are required to mine using a lower difficulty level of the puzzle. The approach presented in

[29] represents a light-weighted version of the PoW minimizing the computing power for mining and also integrating trust elements in the overall process. However, they do not provide a concept how transactions are validated after a node with a high trust score has proposed a block. Moreover, the process of generating a trusted candidate set by the previous miner adds an element of centralization in the concept. The authors also miss to add more trust in the process of creating a block or selecting transactions. Moreover, the participation of the nodes in the consensus process is not rewarded or punished.

The authors in [30] propose a consensus protocol which is operating in a DAG (Directed Acyclic Graph) and where trans-actions receive trust scores based on the trust score of the sending node. In order to deploy a transaction to the DAG, the node has to validate two prior unconfirmed transactions. Moreover, the node has to perform light-weight form of PoW to disable double spending attacks. These transactions are selected based on the current trust score of the current transaction (trust scores which are close to each other). “This results in the formation of trustchains in the cluster. The cumulative trust score of such a chain is the sum of the trust score of all the transactions making up the chain”. To check if a transaction is confirmed, the cumulative trust score of its trustchain has to be compared against the pre-set global confirmation thresh-old. The advantages of the approach provided in [30] is that it also considers in comparison with other approaches the trust score of the transaction originator and the transaction in the consensus process. Moreover, the confirmation of a transaction depends on the trust score of the trustchain and that transaction. The authors in [31] describe COTIs trustchain as a network where the trust score of nodes is considered to evaluate the level of proof of work necessary to confirm a transaction. How-ever, the authors in [31] state that that the consensus is realized by a small number of highly trusted nodes. Disadvantages of this is the risk of centralization and monopoly during the consensus process.

In order to enable a decentralized smart contract settlement and validation the authors in [32] introduce a Proof-of-Trust protocol. The idea of this protocol differs from the others reviewed above and is to evaluate and pre-validate information entered in the smart-contract which are contested by other participant nodes. “Whenever data input is contested by one of the smart contract parties, a decentralized network of qualified participants validates the data before the execution of the contract”. The Proof-of-Trust protocol is performed by so called delegates which are initially considered as trustworthy and legitimate throw a known-your-costumer process. For each smart contract a delegate is assigned, and the contract is “executed with a valid input”. The delegates will have at the beginning the same high trust score (100%) which remains or is changed based on the inputs provided by them. If there is a “contestation to an input, a full network call is incited and all delegates in the network vote in an auditing process of the original query to provide majority consensus”. Based on this voting the trust score of the delegate is going to be changed or not. The authors in [32] introduce a reward/punishment system for the inputs provided by the delegates. Another advantage of this approach is the fact that the inputs added to a blockchain are prechecked and a kind of trust is attached to a transaction. However, the selection of the delegates where they are considered initially as trustworthy without a trust evaluation process is a disadvantage and also provides elements of centralization.

3.4. Evaluation of existing consensus protocols

The research and development activities, described in Sections 3.2 and 3.3, provide an overview for blockchain-based consensus protocols. All these approaches include their relative benefits and limitations which partially and briefly are de-scribed in the previous sections and which are going to be summarized in this section.

Most of the reviewed approaches [8,17,23–25,29,30] require computational effort for achieving consensus and validating new transactions. However, the protocols described in [25,28] minimize this effort by removing the need to perform “Proof of Work” and to solve a computational puzzle. Traditional-based consensus protocols [8,17,23–26] also do not fully satisfy the requirement to select fairly a node to produce a new block. They use computational effort or amount of stake to decide for a block creator. In contrast to them, trust-based protocols [28–30,32] add some trust elements in the selection process by considering the trust score of peers. Another point is the way how new blocks are created and if they consist of cor-rect transactions or not. Except the protocol in [30] which consider the trustworthiness of the inputs in the blockchain, all other reviewed protocols [8,17,23–26,28,29,32] do not or partially satisfy the requirement for trusted block creation. To avoid monopoly and single point of failures, a decentralized architecture with decentralized responsibilities is required. Only the PoW fully satisfies this requirement by including all participating nodes in the consensus process to find the required nonce value to solve the puzzle and validating other blocks. Other reviewed approaches [17,23–26,28,30,32] partially satisfy the requirement of a decentralized architecture. Mostly the semi-centralized representatives or limited number of nodes taking part in the consensus hinder the system to be fully decentralized. Another important element of a consensus is how the network motivates the nodes to participate in the consensus process and how to deal with misbehaving nodes. Specifically, a trust reward/punishment system is required. The authors in [33] mention a ban score which is assigned to misbehaving nodes which are broadcasting false information. However, this only partially fulfill the requirement as it does not include incentives for nodes to participate in all consensus steps. Another approach which partially fulfill the defined requirement is [32] which introduce a reward/punishment system, but only for nodes which send an input to the blockchain without considering the validating nodes or block creators. None of the reviewed approaches except the protocol introduced in [29] use or describe any trust model which can be used to evaluate the trustworthiness of the participating nodes of the blockchain. The trust level of the nodes ensures a reliable and secure network mitigating several misbehaving attacks. In this context, participating nodes could flood the network with fake transactions trying to bring false information in the blockchain. Against this scenario, protocols like [8,25,26,28,30,32] partially satisfy the requirement of being resilient against these attacks. The required high computing power to add new blocks to the blockchain in [8] mitigates the integration of blocks with fake transactions in the blockchain because not every malicious node could solve the cryptographic puzzle. Be-sides them, the integration of fake transactions within a blockchain in [8] is not considered. The approach in [25] requires transaction similarity to add a transaction to the blockchain, therefore, a malicious node is required to maintain a high per-centage of nodes in order to proceed its fake transactions in the blockchain. The Tangle protocol also partially satisfies this requirement by considering the cumulative weight of transactions to decide which transactions are valid. Most of the trust-based approaches [28,30,32] consider trust weight or scores of transactions in their solution. However, they cover with their trust approach not all considered attacks of fake transactions. Therefore, they partially fulfill the requirement being resilient against fake transaction attacks. Finally, only the protocols introduced in [25,28,32] partially support a reliable validation decision process where in Ripple [25] the validation is done throw transaction similarity check-ups and in [28,32] throw voting of all community members.

Table 2 summarizes the analyzed approaches and a list of the requirements resulting from the strengths and weaknesses of the protocols.

4. Proposed trust consensus protocol for blockchain technology

The review of several existing consensus mechanisms shows several advantages and disadvantages in the existing tech-niques. This section proposes a novel consensus mechanism by reusing some positive ideas of the existing techniques and avoiding the disadvantages they provide. The presented consensus mechanism is a mix of the benefits of several existing techniques and trust. Thus, the proposed consensus is considered as a Trust Consensus Protocol (Trust-CP).

4.1. General description and key components

The Trust-CP aims to solve several trust issues in a decentralized community where the nodes do not need to spend to much energy and waste computational power for performing blockchain activities. Main component of the proposed consensus protocol is the *dynamic trust model* (Trust Evaluation System) which is introduced in Section 2 and is used to assign trust scores to peers from whom blocks and transactions inherit the scores. This novel trust model considers all relevant trust aspects of a node including the initial trust level, the participation in the community and the experience rating between the nodes. Another important component of the Trust-CP is the *trust selection* of block creator nodes (nodes which can collect transactions and add a new block to the blockchain). To enable reliable verification of new blocks a *trust-based voting system* is integrated to the consensus protocol. For encouraging the nodes to participate positively in whole blockchain process a *trust reward/punishment mechanism* is included in the Trust-CP.

Table 2
Evaluation of consensus protocols.

Protocol	Computational Effort (1)	Trusted Block Creator Selection (2)	Trusted Block Creation (3)	Trusted Transaction (4)	Decentralized Architecture (5)	Trust Reward/ Punishment (6)	Dynamic and Reliable Trust Model (7)	Resilient Against Fake Transaction Attacks (8)	Reliable Validation Decision Process (9)
Traditional-based									
PoW [8]	-	o	-	-	+	o	-	o	-
PoS [17]	o	o	-	-	o	-	-	-	-
DPoS [23]	o	o	-	-	o	-	-	-	-
Nano [24]	o	o	-	-	o	-	-	-	-
Ripple [25]	+	n/a	o	-	o	-	-	o	o
Tangle [26]	o	-	o	o	o	-	-	o	-
Trust-based									
Zou [28]	+	+	o	-	o	-	-	o	o
Bahri [29]	o	+	-	-	-	-	o	-	-
COTI [30]	o	+	+	+	o	-	-	o	-
i-CASH [32]	n/a	n/a	n/a	o	o	o	-	o	o

Assessment criteria: + satisfied, o partial satisfied, - not satisfied, n/a not defined.

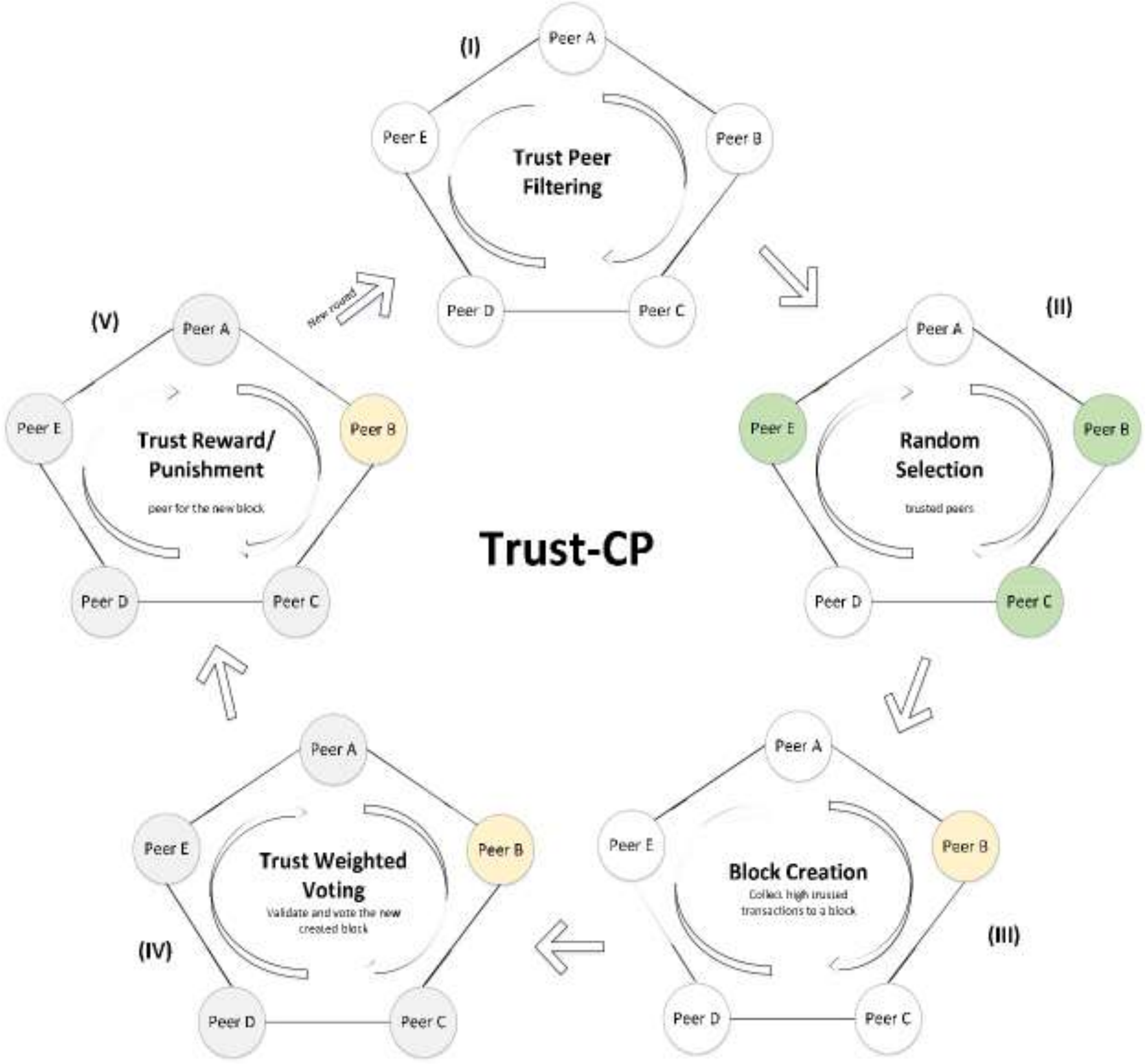


Fig. 14. Lifecyle of trust consensus protocol.

4.2. Workflow in the blockchain network

It is considered that all nodes which are part of an M2M community also participate in the blockchain network. As the M2M community is a decentralized environment without centralized authority, the nodes part of it are continuously evaluated in order to derive their current trust score. Part of the trust evaluation is also the active participation in the blockchain network. As mentioned, all nodes in the blockchain network will have their own trust score. During the lifetime, transactions are sent from one node to other nodes. All transactions are assigned with the trust score of the transaction initiator (trust score of the sender node). In this research the blockchain is used to store information about the trust level of nodes and service information. For instance, node A is evaluating the trust score of node B using the trust model described in Section 2. After the trust score is computed, node A will send a transaction to node B containing the trust information. Transaction like that are also sent by other nodes. These transactions are unconfirmed and are waiting to be approved by the blockchain network. Before the approval process starts, the transactions have to be included in a block. This is done by so-called block creators which are nodes selected from the blockchain network to perform these tasks.

The proposed Trust Consensus Protocol consists of five main phases (see Fig. 14): Trust Peer Filtering, Random Selection; Block Creation; Trust Weighted Voting; Trust Reward/Punishment. These phases are explained in the following paragraphs.

Most of the existing approaches are selecting randomly or by a leader the block creator. Other approaches are using computational challenges or amount of stake to select a node acting as a block creator. As mentioned in Section 3, the reviewed approaches have several limitations. To overcome these problems, this research introduced a trust-based selection mechanism where the block creators are selected based on the combination of trust score, chance, and criteria. For every round of block generation, an algorithm is going through the nodes to select randomly on of them as block creator based on its trust score. To ensure that only nodes with a high trust score are going to be selected, the selection algorithm considers only nodes with a trust score of 80% and higher of the highest trust score (see Fig. 14, Phase I and II). The threshold of 80% is set exemplary and is changed through an adaptative adjustment system of thresholds. All other nodes are not considered in the random selection process. In case that there are no nodes with a trust score above a dynamic threshold a different block creator selection mechanism is going to be used. In this case the different selection mechanism is based on voting of all nodes part of the blockchain. Every node will vote for another node to be selected as a block deployer. The nodes are voting for nodes part of their own neighbor list. The voting is weighted with the trust score of the voting node and the node which is voted.

After a node is selected to be the block creator, it will collect pending transactions to a block (see Fig. 14, Phase III). Another information added to the block is the trust score of the block creator and the voting information about its selection process. Additionally, the created block contains Block ID, previous block header hash, timestamp (no difficulty target and no nonce value in comparison with the Proof of Work), Merkle root and transactions. Ideally, only transactions with a high trust score are considered to be part of the block. Transactions below the dynamic threshold are not considered to be part of the block. If there is no transaction pending at the block creator node which is fulfilling the criteria, then the block creator node drops his task and a new block creator has to be selected with the above-mentioned algorithm.

After the block is generated it will be broadcasted to other nodes for validation and confirmation (see Fig. 14, Phase IV). Other nodes will receive the block and will verify it by checking the trust score of the block creator node, the trust score of the transactions part of the block, and the hash values of the block. If the block contains the right information and also fulfills the criteria of the system, then it will be positively voted by the validating node and the block is forwarded to other nodes. The criteria are fulfilled if the block is created by a trusted block creator, the block contains the right hashes, and the transactions part of the block also are trusted. If the block does not meet the conditions, it will receive a no-vote. The votes are weighted based on the trust score of the validators. If the block receives majority (is a dynamic percentage e.g. 80% of the high trusted nodes should vote positively) of the trust score of all nodes, then it is part of the blockchain.

In order to motivate the nodes in the community to participate in the blockchain community this research publication proposes to reward, or to punish respectively, nodes for the actions they are taking or not taking (see Fig. 14, Phase V). If, for instance, a block creator provides a fake block, he will get a lower trust level by the validating nodes and is going to be downgraded by loosing the possibility to create blocks for further rounds. In general, the block creator node will receive positive or negative trust scores based on the approval or refusal of the proposing block.

4.3. Example and summary of storing trust information and achieving consensus in the M2M community

PeerA evaluates the trust score of *Service1331* which is provided by *PeerB*. The trust evaluation is done through previous defined trust evaluation techniques. The evaluated trust score is stored in the P2P overlay network (Distributed Hash Ta-bles such as Chord) and for integrity check-ups in the blockchain. To store the trust score in the blockchain, *PeerA* sends a transaction to the service provider, in this case *PeerB*. This transaction contains information about the trust evaluation process, the service provider and the service. All the peers' part of the blockchain will receive this unconfirmed transaction which afterwards is stored in the local storage of each Peer. Then, a block creator is selected randomly based on the trust score of it (should be above a predefined threshold e.g. 80% of the highest trust score). In this case, *PeerD* is selected as a Block-Creator which will create a Block with unconfirmed transactions and broadcast them to other nodes. These nodes will verify the collected transactions and the Block if the conditions for creating a Block are fulfilled. Then the validating node is sending a reply-message with a confirmation if the block is accepted or rejected. This confirmation is weighted with the trust score of the validating node and if the Block is positively confirmed, the Block-Creator gets trust points which are used to increase its trust score. If no, the opposite will happen by decreasing the trust score of the Block-Creator. The response is forwarded to the next nodes which will repeat the same step. In order to be considered in the blockchain, the block should be positively validated by the major of the nodes (e.g. 80% of high trusted peers) participating in the blockchain network.

5. Practical evaluation of proposed trust consensus protocol

This section describes an attack model which is used to evaluate the attack robustness of the introduced Trust Consensus Protocol in relation to existing ones. Moreover, it argues theoretically the behavior of the protocol against attacks. Finally, this section concludes with a practical experiment using several scenarios.

5.1. Attack model

The behavior of the proposed Trust Consensus Protocol is assessed exemplary under several attack types. The attack types include attacks performed by malicious nodes acting as sybil nodes and sending several fake transactions.

Attack Type 1: One or several nodes try to spam the local memory pools of other by broadcasting fake transactions. These fake transactions part of the local memory pool of every node can be added by the block creation nodes to a block. Moreover, this block is distributed to other nodes for validation purposes and inclusion to the blockchain.

Attack Type 2: The block creator node tries to harm the system by adding fake transactions to a block and broadcasting it to other nodes for validation. Other nodes will validate the transactions and include the fake transaction in the blockchain.

5.2. Experiment settings

To demonstrate the essential functionalities and the security of the proposed consensus protocol a prototype has been developed. The prototype is developed using the Java programming language and several virtual machines running on a PC. The proposed Trust Consensus Protocol is implemented and evaluated besides the existing traditional consensus protocols (PoW, PoS, DPoS, Nano, Ripple, and Tangle).

To simulate the described attack model this experiment assumes several scenarios with the following settings/conditions on the blockchain and consensus protocols:

- Every participating blockchain node has 1000 transactions in its local memory pool.
- The percentage of malicious nodes is changed through the scenarios.
- The number of fake transactions within the 1000 transactions is increased through the scenarios from 100 to 900 fake transactions.

- Every block includes one transaction.

5.3. Results

The attacks are run individually against the Trust Consensus Protocol, PoW, PoS, DPoS, Nano, Ripple, and Tangle. The aim of the experiment outcomes was to identify the ratio between correct and fake blocks for the different consensus protocols. Therefore, the number of fake transactions and the percentage of malicious nodes are increased during the experiment to analyze the behavior of the protocols.

Fig. 15 shows the scenario where all participating nodes are good. This scenario assumes different number of fake trans-actions in the local memory pool of every node. The outcome shows that the proposed Trust Consensus Protocol will include only correct transactions in the ledger no matter how many transactions in the local memory pool are faked. This is argued with the fact that the proposed Trust Consensus Protocol assigns a trust score for every transaction based on the transaction originator. Moreover, it includes only transactions with high trust scores (in this case 80% or above). Thus, the Trust Consen-sus Protocol filters out the fake transactions and includes the good ones in the ledger. Fig. 15 also shows that the outcomes of the other protocols are similar to each other since they do not have a mechanism to determine if a transaction is good

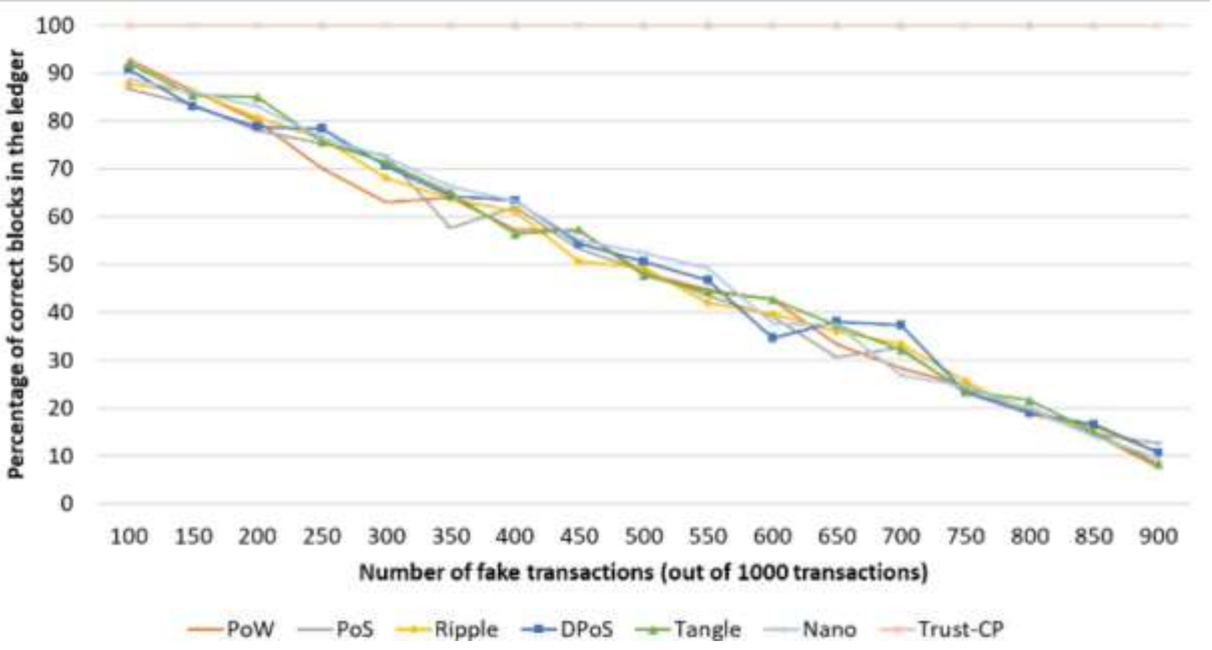


Fig. 15. Fake transaction in various protocols (0% Malicious nodes).

or fake. Only the Ripple consensus protocol relies on transaction similarity, that means that with an increasing number of fake transactions the probability to pass the similarity check is high, thus also the number of fake blocks in the ledger.

Figs. 16–20 consider the existence of malicious nodes which control the block creation and validation part. Except the new introduced Trust Consensus Protocol, which also considers trustworthiness for the block creator selection and block creation process, these figures show that with the increasing number of malicious nodes the resilience of the traditional protocols decreases. This can be argued with the fact that existing consensus protocols rely on computing power or simple selection algorithms of the block creator node without considering trust. Thus, if a malicious node is being selected as a block creator, it would create a block, solve the cryptographical puzzle, and broadcast the block to others for validation. The

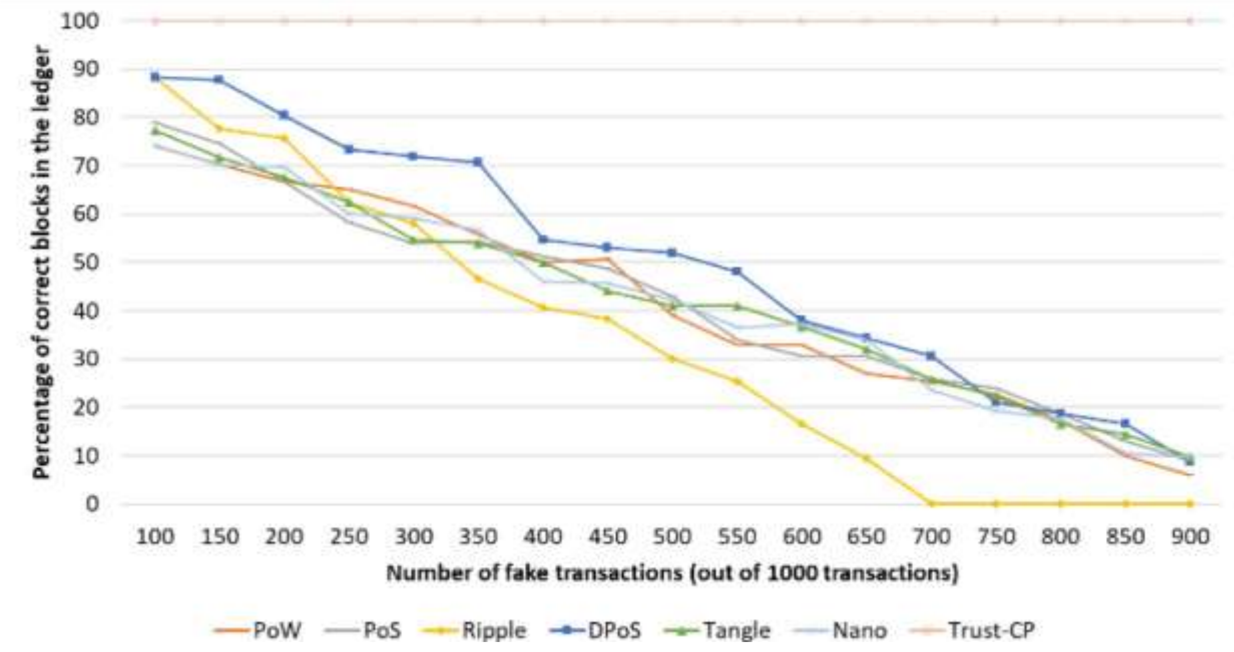


Fig. 16. Fake transaction in various protocols (16% Malicious nodes).

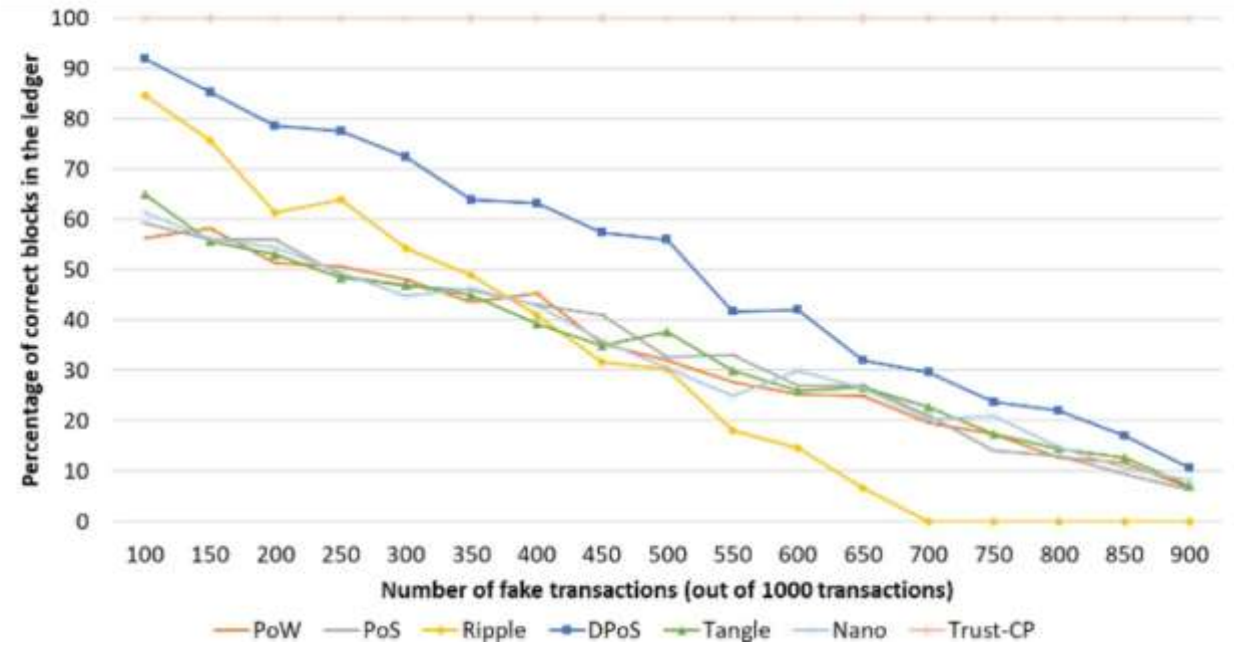


Fig. 17. Fake transaction in various protocols (32% Malicious nodes).

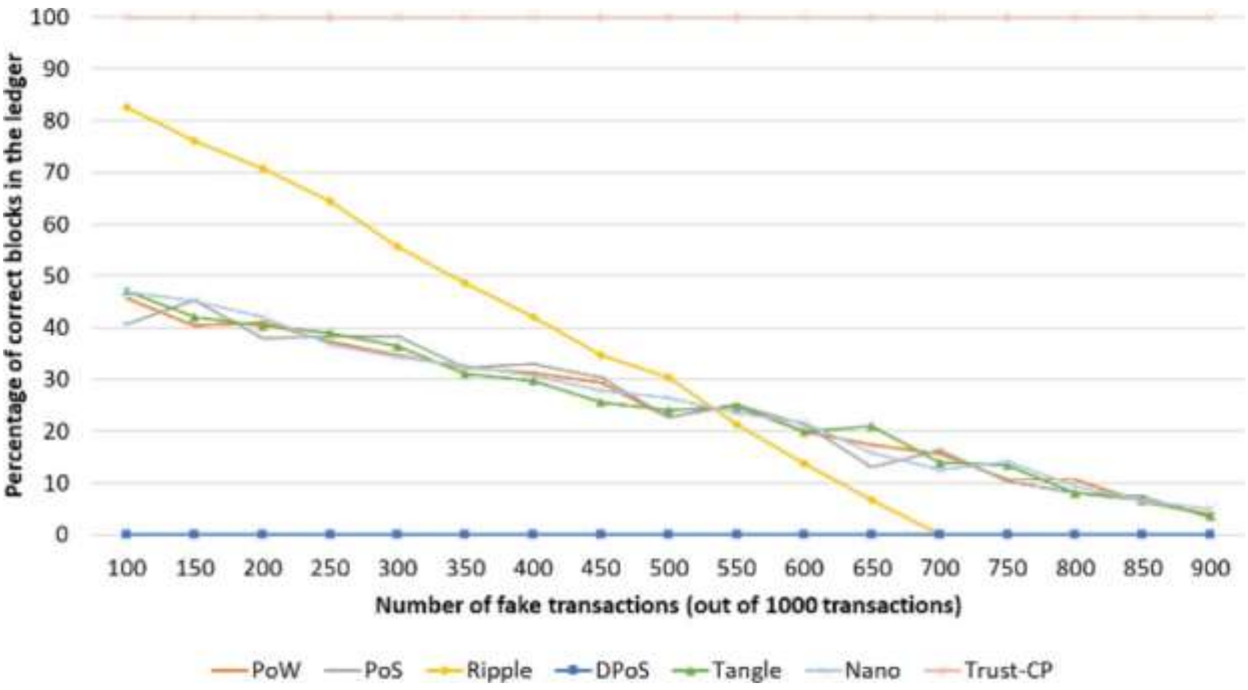


Fig. 18. Fake transaction in various protocols (50% Malicious nodes).

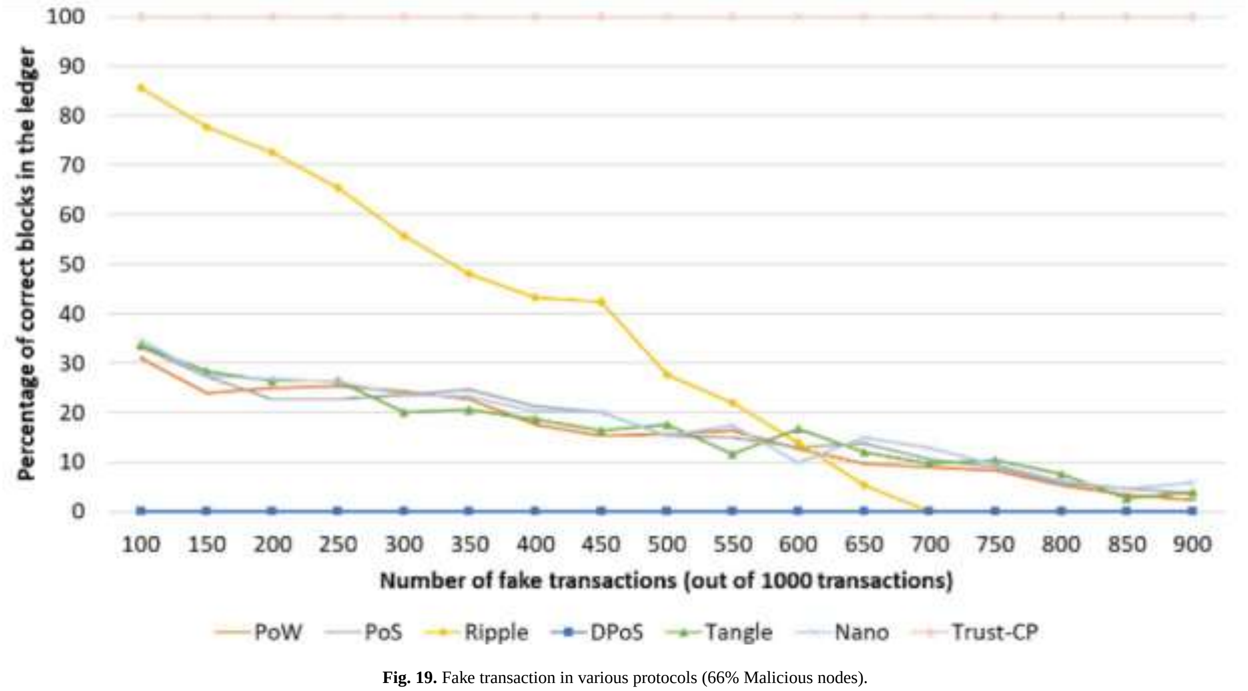


Fig. 19. Fake transaction in various protocols (66% Malicious nodes).

other nodes in existing approaches will just verify the hash values and the keys of the created block and will accept them, again without considering the trustworthiness of the block, block creator or the transactions part of it. By increasing the number of malicious nodes in the network, the probability of being selected as a malicious node is increased. Thus, the performance of the reviewed traditional consensus protocol is continually decreasing. The Ripple consensus protocol had in the first scenario, without malicious nodes, a quite similar result like the others. However, by adding malicious nodes to the network, the security breaks down. This happens because of the feature of similarity checks. By increasing the number of malicious nodes, the probability that the malicious nodes will have the same fake transactions for the similarity check is

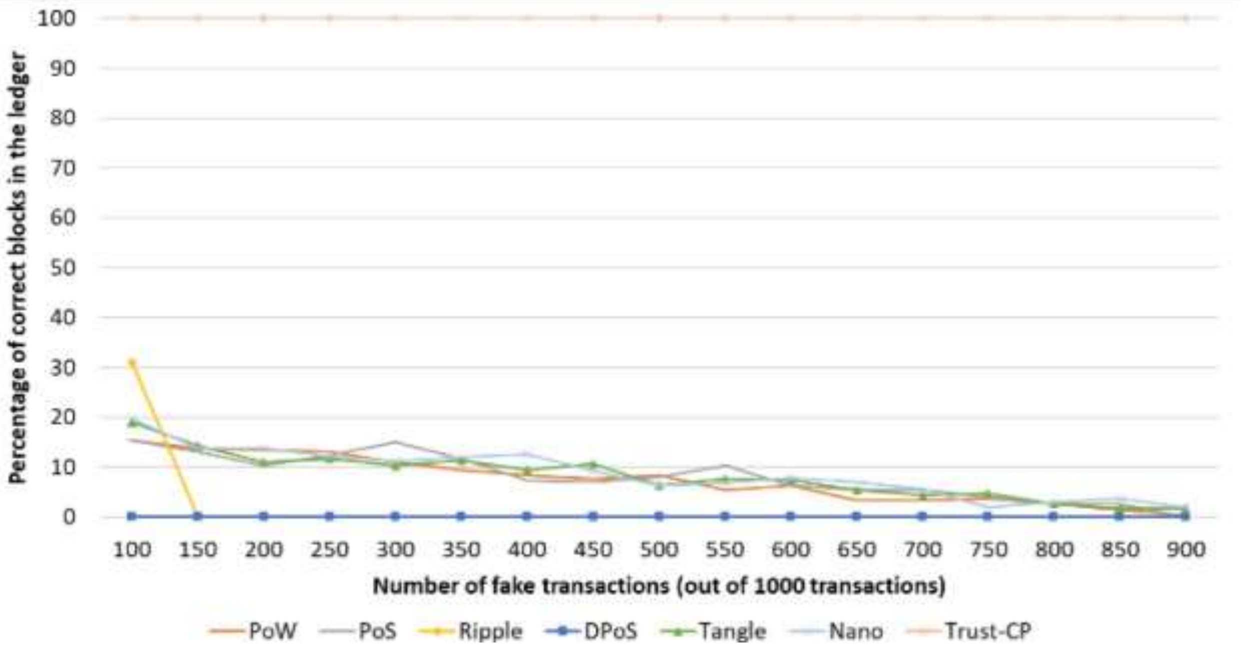


Fig. 20. Fake transaction in various protocols (84% Malicious nodes).

high and therefore, the number of fake transactions in the ledger is also high. The DPoS consensus protocol shows a better security performance at the beginning (see Figs. 16 and 17) than the other traditional protocols. This can be argued by the fact that DPoS is using a voting mechanism to select the block creation leader. Until the number of malicious nodes in the network is not equal or higher than 50%, the probability that a good node is voted by the other nodes as the leader is higher and thus, results in a higher number of correct blocks in the ledger.

6. Conclusion

Decentralized peers acting as service providers within a M2M community come with a risk of failures and malfunc-tioning behaviors. Therefore, this publication presents an optimized and enhanced trust evaluation system which is used to determine the trustworthiness of participating peers in the M2M and the decentralized M2M application services they are providing. The introduced trust evaluation system is an overall framework considering several trust aspects, such as the service functionality, service quality, as well service acceptance. Moreover, it includes the behavior of a peer providing a service and includes the willingness of the peer to participate in several community tasks. To secure data generated through the trust evaluation, this publication proposes to use distributed ledger technology. Blockchain, as a part and data structure of distributed ledger technologies, is used to store trust and service data. Moreover, the combination of P2P overlay and blockchain is introduced for verification and comparison of data such as trust scores, end-user contact information, and service information.

Distributed ledger technologies provide several advantages such as tamper-proof storage and no centralized entities. However, one major problem of DLTs is the way how consensus between the nodes for the same ledger is achieved. This publication presents a comprehensive review of several existing consensus protocols, deriving the different benefits and lim-itations the protocols provide. Based on the review outcomes, a novel Trust Consensus Protocol is introduced which covers the trustworthiness of transactions, the block creator selection, and the block validation step. Finally, the resilience against fake transaction attacks of the Trust Consensus Protocol and existing traditional consensus protocols is shown through a practical evaluation with comparison results.

References

[1] M. Steinheimer, U. Trick, W. Fuhrmann, B. Ghita, Autonomous decentralised M2M application service provision, in: 7th IEEE International Conference on Internet Technologies & Applications (ITA 17), Wrexham, UK, 2017, pp. 18–23. https://doi.org/10.1109/ITECHA.2017.8101904.

[2] M. Steinheimer, U. Trick, W. Fuhrmann, B. Ghita, P2P-based community concept for M2M Applications, in: 2nd IEEE International Conference on Future Generation Communication Technologies (FGCT 2013), London, UK, 2013, pp. 114–119. https://doi.org/10.1109/FGCT.2013.6767198.

[3] B. Shala, P. Wacht, U. Trick, A. Lehmann, B. Ghita, S. Shiaeles, Trust integration for security optimisation in P2P-based M2M applications, in: IEEE International Conference on Trust, Security and Privacy in Computing and Communications (TrustCom 17), Sydney, Australia, 2017, pp. 949–954. https://doi.org/10.1109/Trustcom/BigDataSE/ICESS.2017.335.

[4] M. Steinheimer, Autonomous Decentralized M2M Application Service Provision PhD Thesis, School of Computing and Mathematics, University of Ply-mouth, UK, 2017 http://hdl.handle.net/10026.1/11957.

[5] B. Shala, U. Trick, A. Lehmann, B. Ghita, S. Shiaeles, Trust-based composition of M2M application Services, in: 10th IEEE International Conference on Ubiquitous and Future Networks (ICUFN 2018), Prague, Czech Republic, 2018, pp. 250–255. https://doi.org/10.1109/ICUFN.2018.8436992.

[6] Distributed ledger technology: beyond blockchain, Government Office for Science (2016) available from: https://www.gov.uk/government/uploads/system/uploads/attachment_data/file/492972/gs-16-1-distributed-ledger-technology.pdf (accessed 19 March 2019).

[7] D. Burkhardt, M. Werling, H. Lasi, Distributed Ledger, in: IEEE International Conference on Engineering, Technology and Innovation (ICE/ITMC), Stuttgart, Germany, 2018, pp. 1–9. https://doi.org/10.1109/ICE.2018.8436299.

[8] S. Nakamota, Bitcoin: a Peer-to-Peer Electronic Cash System. (2008) available from: https://bitcoin.org/bitcoin.pdf (accessed 12 January 2017).

[9] S. Hashemi, F. Faghri, P. Rausch, World of empowered IoT users, in: 1st IEEE International Conference on Internet-of-Things Design and Implementation, Berlin, Germany, 2016, pp. 13–24. https://doi.org/10.1109/IoTDI.2015.39.

[10] M. Samaniego, R. Deters, Using blockchain to push software-defined IoT components onto edge hosts, in: International Conference on Big Data and Advanced Wireless technologies (BDAW'16), Blagoevgrad, Bulgaria, 2016. https://doi.org/10.1145/3010089.3016027.

[11] Z. Huang, X. Su, C. Shi, Y. Zhang, L. Xie, A Decentralized solution for ioT data trusted exchange based-on blockchain, in: 3rd IEEE International Confer-ence on Computer and Communications, Chengdu, China, 2017, pp. 1180–1184. https://doi.org/10.1109/CompComm.2017.8322729.

[12] M.Y. Jung, J.W. Jang, Data management and searching system and method to provide increased security for IoT platform, in: IEEE International Confer-ence on Information and Communication Technology (ICTC), Jeju, South Korea, 2017, pp. 873–878. https://doi.org/10.1109/ICTC.2017.8190803.

[13] X. Liang, J. Zhao, S. Shetty, D. Li, Towards data assurance and resilience in IoT using blockchain, in: IEEE Military Communications Conference (Milcom 2017), Baltimore, MD, USA, 2017, pp. 261–266. https://doi.org/10.1109/MILCOM.2017.8170858.

[14] B. Liu, X.L. Yu, S. Chen, X. Xu, L. Zhu, Blockchain based data integrity service framework for IoT data, in: 24th IEEE International Conference on Web Services, Honolulu, HI, USA, 2017, pp. 468–475. https://doi.org/10.1109/ICWS.2017.54.

[15] B. Shala, U. Trick, A. Lehmann, B. Ghita, S. Shiaeles, Blockchain-based trust communities for decentralized M2M application services, in: F. Xhafa, FY. Leu, M. Ficco, CT. Yang (Eds.), Advances on P2P, Parallel, Grid, Cloud and Internet Computing, 3PGCIC 2018, 24 (eds), Springer, Cham, 2019, pp. 62– 73. Lecture Notes on Data Engineering and Communications Technologies https://doi.org/10.1007/978-3-030-02607-3_6 .

[16] N. Chalaemwongwan, W. Kurutach, State of the Art and challenges facing consensus protocols on blockchain, in: IEEE International Conference on Information Networking (ICOIN), Chiang Mai, Thailand, 2018, pp. 957–962. https://doi.org/10.1109/ICOIN.2018.8343266.

[17] S. King, S. Nadal, Ppcoin: peer-to-Peer Crypto-Currency with Proof-of-Stake, (2012) available from: https://peercoin.net/assets/paper/peercoin-paper.pdf (accessed 12 April 2018).

[18] B. Shala, P. Wacht, U. Trick, A. Lehmann, B. Ghita, S. Shiaeles, Framework for automated functional testing of P2P-based M2M applications, in: 9th IEEE International Conference on Ubiquitous and Future Networks (ICUFN 2017), Milan, Italy, 2017, pp. 916–921. https://doi.org/10.1109/ICUFN.2017.7993932.

[19] C. Natoli, V. Gramoli, The Blockchain Anomaly, in: 15th IEEE International Symposium on Network Computing and Applications (NCA), Cambridge, USA, 2016, pp. 310–317. https://doi.org/10.1109/NCA.2016.7778635.

[20] Survey on Blockchain Technologies and Related Services FY 2015 Report, Japan, 2016 available from: https://www.meti.go.jp/english/press/2016/pdf/ 0531_01f.pdf (accessed 22 March 2019).

[21] I. Bashir, Mastering Blockchain, Packt, Birmingham, UK, (2017), ISBN 978-1-78712-544-5.

[22] K. Christidis, M. Devetsikiotis, Blockchains and smart contracts for the internet of things, IEEE J. Mag. (2016) 2292–2303. https://doi.org/10.1109/ACCESS.2016.2566339.

[23] M. Snider, K. Samani, T. Jain, Delegated Proof-of-Stake: features & Tradeoffs, (2018) available from: https://multicoin.capital/wp-content/uploads/2018/ 03/DPoS_-Features-and-Tradeoffs.pdf (accessed 18 March 2019).

[24] C. LeMahieu, Nano: a Feeless Distributed Cryptocurrency Network, available from: https://nano.org/en/whitepaper (accessed 18 March 2019).

[25] D. Schwartz, N. Youngs, A. Britto, The Ripple Protocol Consensus Algorithm, (2014), available from: https://ripple.com/files/ripple_consensus_ whitepaper.pdf (accessed 20 March 2019).

[26] S. Popov, The Tangle, Whitepaper (2018), available from: https://docs.iota.org/ (accessed 20 March 2019).

[27] IOTA is Centralized, (2017) available from: https://medium.com/@ercwl/iota-is-centralized-6289246e7b4d (accessed on 14 March 2019).

[28] J. Zou, B. Ye, L. Qu, Y. Wang, M. Orgun, L. Li, A proof-of-trust consensus protocol for enhancing accountability in crowdsourcing services, IEEE Trans. Serv. Comput. (2018). https://doi.org/10.1109/TSC.2018.2823705.

[29] L. Bahri, S. Girdzijauskas, When trust saves energy: a reference framework for proof of trust (PoT) blockchains, in: Web Conference (WWW'18), 2018, pp. 1165–1169. https://doi.org/10.1145/3184558.3191553.

[30] The Trust Chain Consensus, COTI: a Decentralized, High Performance Cryptocurrency Ecosystem Optimized for Creating Digital Payment Networks and Stable Coins, White Paper, (2018), available from: https://coti.io/files/COTI-technical-whitepaper.pdf (accessed on 22 March 2019).

[31] Coti Network, available from: www.medium.com/cotinetwork (accessed on 12 March 2019).

[32] I-CASH, A Smart Contract Origination and Settlement Platform Leveraging the Proof of Trust Protocol, White Paper, available from: https://cdn2.hubspot.net/hubfs/4276960/ICASH%20whitepaper.pdf?t=1524150610703 (accessed on 13 March 2019).

[33] Bitcoin Developer Guide, available from: https://bitcoin.org/en/developer-guide (accessed on 7 March 2019).

[34] Blockchain vs. DAG Technology, available from: https://medium.com/nakamo-to/blockchain-vs-dag-technology-1a406e6c6242 (accessed on 25 March 2019).

[35] N. Kolokotronis, K. Limniotis, S. Shiaeles, R. Griffiths, Secured by blockchain: safeguarding internet of things devices, IEEE Consum. Electron. Mag. 8 (3) (May 2019) 28–34. https://doi.org/10.1109/MCE.2019.2892221.

[36] D. Minoli, B. Occhiogrosso, Blockchain mechanisms for IoT security, Elsev. J. Internet Things 1–2 (September 2018) 1–13. https://doi.org/10.1016/j.iot. 2018.05.002.