Journal of Parallel and Distributed Computing 144 (2020) 41–54

Contents lists available at ScienceDirect

J. Parallel Distrib. Comput.

journal homepage: www.elsevier.com/locate/jpdc

Blockchain-based access control management for Decentralized Online Social Networks

Mohsin Ur Rahman [a], Barbara Guidi [a,*], Fabrizio Baiardi [b]

[a] Department of Computer Science, University of Pisa, Italy
[b] Department of Philology, Literature, and Linguistics, University of Pisa, Italy

ARTICLE INFO

Article history:
Received 14 October 2019
Received in revised form 15 April 2020
Accepted 21 May 2020
Available online 4 June 2020

Keywords:
Decentralized Online Social Networks
Blockchain
Privacy issue
Access control list

ABSTRACT

Online Social Networks (OSNs) represent today a big communication channel where users spend a lot of time to share personal data. Unfortunately, the big popularity of OSNs can be compared with their big privacy issues. Indeed, several recent scandals have demonstrated their vulnerability. Decentralized Online Social Networks (DOSNs) have been proposed as an alternative solution to the current centralized OSNs. DOSNs do not have a service provider that acts as central authority and users have more control over their information. Several DOSNs have been proposed during the last years. However, the decentralization of the social services requires efficient distributed solutions for protecting the privacy of users. During the last years the blockchain technology has been applied to Social Networks in order to overcome the privacy issues and to offer a real solution to the privacy issues in a decentralized system. However, in these platforms the blockchain is usually used as a storage, and content is public. In this paper, we propose a manageable and auditable access control framework for DOSNs using blockchain technology for the definition of privacy policies. The resource owner uses the public key of the subject to define auditable access control policies using Access Control List (ACL), while the private key associated with the subject's Ethereum account is used to decrypt the private data once access permission is validated on the blockchain. We provide an evaluation of our approach by exploiting the Rinkeby Ethereum testnet to deploy the smart contracts. Experimental results clearly show that our proposed ACL-based access control outperforms the Attribute-based access control (ABAC) in terms of gas cost. Indeed, a simple ABAC evaluation function requires 280,000 gas, instead our scheme requires 61,648 gas to evaluate ACL rules.

1. Introduction

Today, millions of users are actively using Social Media, such as Facebook, Instagram, Twitter, etc. In particular, Online Social Networks (OSNs), such as Facebook, are used by 700 million users, and it attracts nearly 31% of global Internet users on a daily basis. People are attracted to social networks because they are free, fun, and offer useful social services, such as easy communication, marketing, games, etc. Usually, users trust the services of the service providers to manage their data and mine it only for specific purposes (i.e., targeted advertising). Furthermore, they also trust the providers for enforcing access control policies in order to protect their personal sensitive information such as profiles, pictures, videos, and other posted items. However, OSNs have not consistently achieved these objectives.

Due to the uncertain privacy guarantees of today's OSNs, online users are seeking for alternative data sharing techniques that offer them the possibility to gain control of their own data (i.e., to enable them to manage their data on their own), and to limit the support of the large service providers in controlling their own personal information. To meet these requirements, Decentralized Online Social Networks (DOSNs) have been proposed, ranging from Peer-to-Peer (P2P) decentralized solutions to hybrid solutions integrating private and external resources for storing users' data [20,21].

A DOSN is a decentralized social platform that facilitates social networking services in distributed environments [20]. It is worthy to note that the dynamic P2P nature of DOSNs necessitates distributed and lightweight access control techniques [13,14]. Due to the highly dynamic environment of DOSNs, guaranteeing data availability, defining lightweight and privacy-preserving access control algorithms, suitable algorithms for information diffusion in a distributed environment, are the current open problems in this area [20].

Several architectures for DOSNs have been proposed. Most of them use encryption in order to guarantee privacy [2,31,38], and only a minor part is based on the trust concept in order to guarantee privacy [10,19,28]. To guarantee privacy preserving in DOSNs,

[*] Corresponding author.
E-mail address: guidi@di.unipi.it (B. Guidi).

https://doi.org/10.1016/j.jpdc.2020.05.011
0743-7315/© 2020 Elsevier Inc. All rights reserved.

several techniques have been proposed. In particular, techniques, such as the Attribute-based Access Control (ABAC) [22], role-based access control (RBAC) [32], and rule-based access control, are proposed in order to increase the level of privacy and to guarantee more control over data [3,15,29,30]. Consider the current usage of social media, these techniques are not able to provide a scalable, manageable and efficient mechanism to meet the security requirements of DOSNs. The main issues in a DOSN are the distributed nature of the network, the need to manage private data of users which cannot be stored everywhere in the network, and the data availability. Access control techniques in current DOSNs require online nodes to evaluate the access. A new generation of DOSNs takes into account the blockchain technology, as explained in [18]. However, current Blockchain-based Online Social Networks are proposed principally to provide a way to manage the fake news issue, and to reward users for valuable content. Usually, personal information is stored in the blockchain and they are visible to everyone by excluding the privacy options. A critical question arises: how can we design an auditable and trustworthy access control system for DOSNs by exploiting the properties of the blockchain technology? Indeed, the emerging blockchain technology and the popular Ethereum platform (i.e., smart contracts) [9] can be used to address this challenging issue. In this paper we propose a blockchain-based decentralized social networks where blockchain is used as a tool for privacy preserving, and where context information is used to define privacy policies. The novelty of the proposed system is the application of the blockchain as a support for access control in a DOSN scenario, by exploiting an ACL model. For the sake of readiness, the proposed system takes into account the main characteristics of DOSNs by exploiting access control methods, in particular privacy policies with the support of the blockchain, which is used as a tool, instead of content storage, by providing a trustworthy access control system.

We test different implementations of our smart contracts and our experimental evaluation shows that the Resource Owner (RO) can create a single policy for a maximum of 14 subjects. Moreover, we tested different implementations of the reputation contract with different number of subjects and the results show that it can simultaneously evaluate the reputation for 66 subjects. Finally, the inspector contract simultaneously evaluates the punishment for 75 subjects in the DOSN network. These experiments aim to evaluate the effects of code complexity on gas consumption in the Ethereum network. Our approach requires 61,648 gas to evaluate ACL rules and it clearly outperforms a simple ABAC evaluation which requires 280,000 gas. We clearly observe that contract storage and contract operations have a direct impact on gas consumption in the Ethereum network.

The main contributions of this article are summarized as follows:

- the design of a new ACL-based access control model for DOSNs, by using the Ethereum blockchain which provides a unique address for each registered account, which is used as the identity for each user in this framework.
- the implementation of smart contracts over the Ethereum blockchain: the Access Control Contract able to control access to resources in the network, the Reputation Contract to provide a trustworthy system, the Inspector Contract able to inspect the behavior of users, and the Registrar Contract (RegC) able to find the identifying information of the access control contracts and the corresponding access control functions.
- the usage of the challenge–response authentication protocol to verify users' identities.
- the dynamic validation of the behavior of subjects in case of access requests.
- the design of a model which takes into account context information, such as time and location of access to define fine-grained access control policies. In particular, time and location are important information used to define a context in new generation of DOSNs, as described in [21]
- the evaluation of the proposed system on the Ethereum network in order to evaluate the model and the gas consumption.

The rest of the paper is structured as follows. Section 2 presents the related work concerning the major concepts considered in this paper. Section 3 discusses the access control requirements for DOSNs. Section 4 presents the architecture of our proposed framework. Section 5 introduces the set of smart contracts used in our proposed framework. Section 6 describes the Access Control Process. Section 7 reports the evaluation of the proposed approach. Finally, Section 8 concludes the paper with a summary of achievements and directions for future research.

2. Background and related work

In this section, we provide an overview of both the current DOSNs and blockchain-based solutions in order to explain the scenario in which our proposal is located. Furthermore, we provide the state of art concerning the access control, and a summary concerning the proposed scenario in order to explain the contribution of our paper.

2.1. Decentralized online social networks

The main difference among the current DOSN proposals concerns the technologies and techniques used to store and manage data. A possible classification considering this difference has been proposed in [20]. One of the first decentralized solutions, which has today more than 600,000 users, is Diaspora,[1] a federated DOSN, where users provide servers that are administered by themselves and that allow Diaspora users' profiles to be hosted on their servers. If Diaspora can be considered the distributed version of Facebook, Mastodon [34] represents the decentralized version of Twitter. Mastodon is a decentralized microblogging network based on open protocols and free, open-source software. During the last two years, Mastodon was increasing the number of users (about 2M of users), surpassing Diaspora. Mastodon is formed by a set of servers, known as instances, and each user is a member of a specific Mastodon instance, but can connect and communicate with users on other instances. Like Twitter, Mastodon supports direct, private messages between users, but unlike Twitter, Mastodon's messages can be either private to the user, private to the user's followers, public on a specific instance, or public across a network of instances. Mastodon is part of the Fediverse, an interconnected and decentralized network of independently operated servers, which includes platforms such as Diaspora, Friendica, GNU Social, PeerTube, etc. Fediverse is a common name for the union of various federated social networks which use a set of standard protocols: OStatus, ActivityPub, DFRN, Diaspora Network, and Zot. Diaspora, Mastodon, and all the other systems in the FeDiverse project are not completely decentralized. Indeed, they are federated and there are servers which communicate between them. As concerns fully-decentralized DOSNs, as the scenario proposed in this paper, several research applications have been proposed.

Safebook [11] is a three-tier architecture with the main focus on privacy, integrity and availability. Each user has a set of logical concentric structures called Matryoshkas. Matryoshkas are

[1] https://joindiaspora.com.

concentric rings of nodes built around each peer and provide a trusted data storage and communication obfuscation through indirection.

PeerSon [5] is a two-tier architecture in which one tier is implemented by a Distributed Hash Table (DHT) and it serves as a look-up service. The second tier consists of peers and contains the user data, such as user profiles.

My3 [28] is a privacy-friendly DOSN which exploits well-known interesting properties of Online Social Networks, for instance the locality of users and the trust among them. Users' profiles are hosted only on a set of self-chosen trusted nodes, called Trusted Proxy Set (TPS). Exploiting availability and performance goals, as its geographical location and the online time period of the user, populates the TPS of a user.

DiDuSoNet [19] is a two-tier system, where the lower level is implemented by a DHT, and the upper level is implemented by exploiting a social overlay [20]. In the social overlay, nodes are connected to other nodes with whom the tie strength computed on the interaction between them is higher. Social data are stored only on trusted nodes, and each node can choose two replicas to have a high level of availability.

Several DOSNs integrate the P2P layer with external resources, such as cloud storage services, to increase the quality of service. External resources are used to cope with the situations in which the users cannot deliver the service by themselves. Vis-à-Vis [36], Vegas [17], and SuperNova [37] are only three examples of approaches in which cloud services are used.

2.2. Blockchain-based online social networks

During the last years, several Blockchain-based Online Social Networks (BOSNs) [18] have been proposed. These platforms give more importance to the content by providing rewarding systems and they aim to address the problems of privacy and fake news using the blockchain technology.

Steemit[2] is a social media platform where everyone can receive a reward for creating and curating content, in the form of STEEM, which is the unit that is bought and sold for actual money on the open markets. It has more than one million users and it represents the most well-known BOSN. An important characteristic of Steemit is that it is fast, free, and scalable. Steemit operates based on one-STEEM one-vote, instead of one-user one-vote, as in other platforms. Within this model, individuals who have contributed the most to the platform have the most influence over how contributions are scored.

SocialX,[3] as all the previous platforms, is decentralized and allows users to give content feedback and reward tokens. All media files (photos and videos) and data (messages, posts etc.) are decentralized. The platform wants to face the problem of fake accounts, fake followers, and fake votes (likes, etc.). Indeed, the decision power is given to communities, which can decide the valuable content.

Sapien[4] is a decentralized social networking platform that is designed by using the Ethereum Blockchain. Interesting features of the platform include support for customization and rewarding contents creators without the need of central authorities. Furthermore, its cryptocurrency is called SPN, which is ERC20 complaint. The SPN is based on proof of value consensus mechanism in order to differentiate high-quality contents in the network.

2.3. Access control models

The Role-Based Access Control (RBAC) [33] model enables access to resources based on roles, and supports principles such as separation of duties, least privilege and partition of administrative functions. However, the basic RBAC model is vulnerable to the role-explosion problem, thereby making it unsuitable for the implementation of access control policies that involves complex DOSN scenarios. One of the main problems when implementing RBAC in distributed networks is the role-explosion problem. Solutions, such as self-management, can be used to solve the role-explosion problem in such an environment.

To address the limitations of the RBAC model in distributed network settings, a new Attribute-based Access Control (ABAC) model [22] was introduced to define fine-grained access control policies and to solve the role-explosion problem (i.e., to decrease the number of rules associated with the RBAC model). This model considers various attributes such as subject attributes and environmental attributes to define access control policies. Consequently, an AC authority can grant access rights by considering users' attribute certificates.

Authorization and access control to personal data are important properties in both OSNs and DOSNs systems. OSNs define access policies based on trust or the tie strength of users. In DOSNs, access control components need to be reactive to the dynamics of the network. Considering the proposed access control models, RBAC are not suitable for DOSNs due to the need of specific information that cannot be provided in this model [30]. The same problem with a rule-based access control, such as the one proposed in [8], where authors introduce a model for the users in OSNs, where the policies are based on social type information, such as relationship type and trust, stored in a server. This specific solution cannot be applied in a decentralized scenario due to the single point of failure of a server. On the contrary, Access control list (ACL) provides a popular technique to implement access control policies because it is suitable for situations in which the resource owners set permissions for their resources. ACL is suitable for DOSNs because it provides a fine-grained mechanism allowing users to restrict their sensitive content to a select subset of their friends [27]. The ACL includes a list of one or more subjects and a set of rules. Each of the subjects is associated with a set of operations that the subject can perform on the resource. The set of rules specify conditions at which a different set of operations is to be associated with one or more of the subjects in the list of subjects. With ACL, the resource owner matches subjects to resources on a one-to-one basis. A subject performs an operation on a resource if that subject or a group to which that subject belongs is specified in the ACL associated with the resource [35]. Moreover, ACL provides information about the mode of access that subjects are authorized to perform on resources. Access to a resource can be easily revoked by deleting the associated row containing an existing ACL-based policy.

2.4. Summary of the state of the art

The current status of DOSNs systems is evolving by exploiting new technologies, such as the blockchain. Current blockchain-based social media mainly exploit the blockchain to resolve the problem of fake news and to reward users for valuable contents, as described in [18]. Furthermore, these systems use the blockchain to store social data of users, which is not in line with the privacy issue. Privacy has been one of the most important point of DOSNs. Current P2P DOSNs provide methods for access control policies without the usage of the blockchain, such as [16, 30]. These methods are affected by the dynamics of peers and they require online peers to validate privacy policies. For this

[2] https://steemit.com/.

[3] https://socialx.network/.

[4] https://beta.sapien.network/.

reason, there is the need to manage the access control systems with a more stable solution which can address the problem of node churn. Our proposed system takes into account the main characteristics of DOSNs by exploiting privacy policies with the support of the blockchain, because the main characteristics of the blockchain technology can help to efficiently manage privacy issues.[5] The blockchain is a tool to guarantee a trustworthy access control system for DOSNs. The system is based on blockchain technology whereas the policies are enforced by it. As concerns the Access Control Requirements for DOSNs, in [30], authors propose a list of requirements. In particular the attention is posed to the resources without any central authority or trusted third parties, and to the definition of specific operation types on a resource.

The choice to combine ACL with the blockchain technology to implement our access control model has been done because in social relationships there is the need to provide access privileges which are mostly dependent on the social behavior of users, such as the same job. Indeed, the trust is a good way to specify the access privileges, and we want to provide a model to guarantee this by exploiting two important information concerning contexts: time and location.

The introduction of a blockchain-based access control for DOSNs can help to overcome the privacy preserving issues of these systems by exploiting the characteristics of the blockchain technology. In [23], a similar work has been proposed, however authors use the blockchain as a trusted server to provide central control services, such as user identity, news feed notification, and friend recommendation.

3. Requirements overview

Compared to OSNs which allow the service provider to manage the access control problem, DOSNs demand new requirements for access control due to the decentralization of the infrastructure for social networking services. In particular, special attention is needed for access control because DOSNs do not rely on a central authority. Furthermore, we need to think of a fine-grained access control model that fulfills the access control requirements. The following access control properties are essential to successfully tackle the access control problem in DOSNs:

- The provision of a user-driven and trustworthy access control for DOSNs in a distributed setting is still a challenging research problem [4,20].
- A fine-grained access control must address both users' resources (i.e., pictures, videos, files, etc.) and context information.
- Resources need to be protected in a user-driven manner without the support of a central authority or trusted third parties [20].
- The access control mechanism must be flexible that would allow users to define different permissions for different operations such as view, download, read and write etc. [30].
- The access control must be manageable that would allow users to easily modify the existing policies stored in the blockchain.
- The resource owner should be able to revoke a policy that he/she has already created. To perform this operation, he/she simply needs to make a new transaction specifying the policy that needs to be revoked [25].
- Users in the DOSN network should be able to create a policy for a single user or group of users as well as for a single resource or multiple resources. In other words, a user would be able to add a group of users to perform an operation on a single or multiple resources.
- The access control must be restrictive in the sense that a user would be able to perform policy-related actions (i.e., policy creation, update and revocation) and cannot perform these actions on behalf of other users in the DOSN network.

Our access control requirements for DOSNs are based on the needs and diversity of subscribers. We decide to use the access control lists (ACLs) [35] due to its flexibility to support distributed operation. Furthermore, ACL provides support for important elements such as groups/teams and context information. It is notable that, in our scheme, the access right validation is conducted by multiple entities (i.e., trusted nodes), thereby mitigating the single point of failure issue associated with a centralized entity. Moreover, context information, such as time and location of access, play an important role in designing a fine-grained access control solution [6]. Thus, our proposal allows users to create security policies based on these features while taking into account the features discussed in the requirements list. Context-aware access control is a security technique which considers different types of context information to control access to resources. In the last few years, several context-aware access control techniques have been proposed such as spatial information, temporal information, and both location and time [24]. However, collecting and sharing context information in DOSNs may lead to privacy problems because context data contains personal data. Consequently, a context-aware access control system requires sound privacy measures to ensure that the users' personal information are not linked to their real identities. It is worth noticing that our blockchain-based solution exploits pseudonymity to provide anonymity protection, which ensures that each registered user is associated with an anonymous ID that shows no information about the real identity of that user.

4. Access control framework

In this paper, we address how blockchain can support DOSNs in order to provide a distributed, auditable, user-driven, and scalable access control system using smart contracts. This paper proposes a novel context-aware access control system using ACLs implemented using smart contracts. The emergence of blockchain has eliminated the need for a central agency, by allowing applications and services to operate in a decentralized fashion, without the support of a central authority.

Our framework proposes a set of smart contracts: the Access Control Contract (ACC) able to control accesses to resources in the network, the Reputation Contract (RC) to provide a trustworthy system, and finally the Inspector Contract (IC) able to inspect the behavior of users.

The main actors of our framework include the Resource Owner (RO), who is the owner of a specific resource, nodes trusted from the RO, and subjects who are interested to perform different operations on the resources of the RO. Our proposal allows the RO to define access rights for his/her resource using ACLs, thereby preventing trusted nodes from fraudulently denying the access rights granted by ACLs. The RO uses the IDs of subjects to create fine-grained and auditable access control policies using its personal ACC. Furthermore, the approach allows users to check the blockchain at any time to determine the status of access rights. It is worth noticing that a subject interested in performing an operation on a resource of the RO must assert that it possesses

[5] https://helios-social.com/project/.

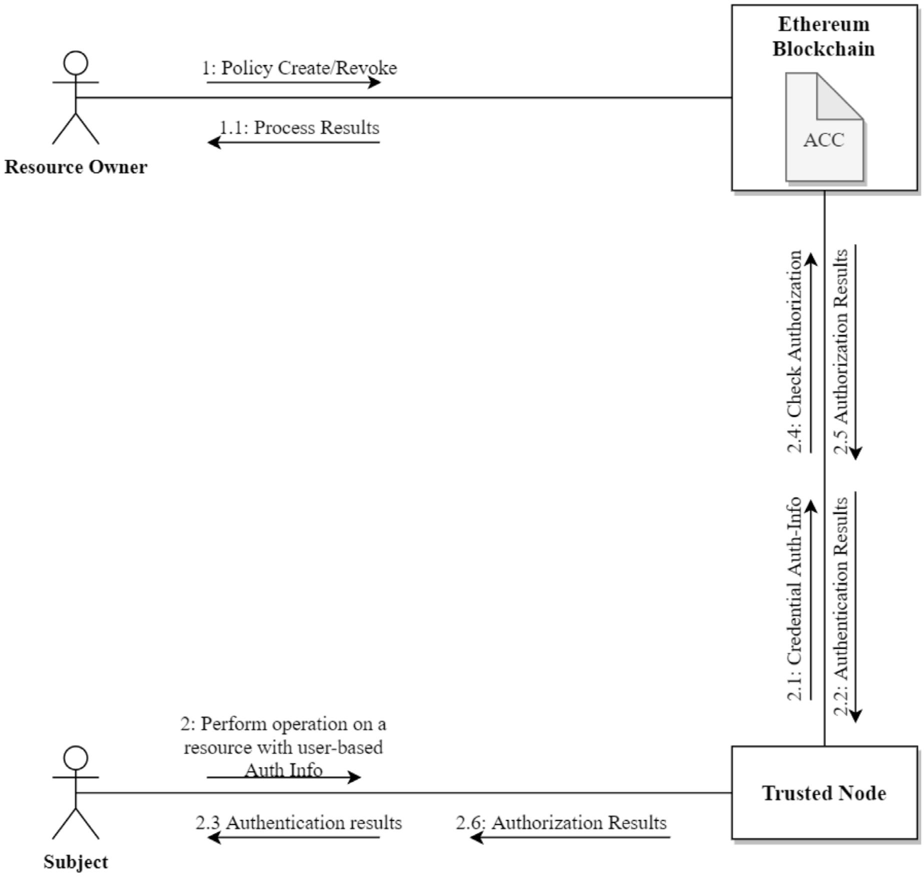
Ethereum Blockchain
ACC
Resource Owner
1: Policy Create/Revoke
1.1: Process Results
2.4: Check Authorization
2.5 Authorization Results
2.1: Credential Auth-Info
2.2: Authentication Results
2: Perform operation on a resource with user-based Auth Info
2.3 Authentication results
2.6: Authorization Results
Trusted Node
Subject

Fig. 1. Illustration of the policy creation, authentication and authorization process.

a unique ID that was used by the RO to create a policy in the blockchain.

As illustrated in Fig. 1, a RO in a DOSN can send transactions to its personal Access Control Contract (ACC) in order to define access rights for a resource. Furthermore, the ACC consists of necessary functions to perform the task of access right validation based on the policy list published on the blockchain.

The RO has a set of trusted nodes where data are stored, as described in [15], in order to guarantee the data availability even when the user is offline. The choice of trusted replica nodes is outside the scope of this article. A generic request is sent to the RO when it is online, otherwise it is sent to one of the trusted nodes chosen at random. Each RO has a unique personal ACC deployed on the blockchain to control access to resources. Trusted nodes get information about the RO's ACC from the registrar (RegC) contract. These nodes then use the RO's ACC to check access permission for the requested operation. Based on the evaluation of access control policies, the subject is either allowed or denied to perform the requested operation on the resource of the RO. It is worth noticing that trusted nodes only check the status of access permission in the blockchain and all the policy ownership operations such as policy creation, update and revocation are performed by the RO. Fig. 2 shows the architecture of our access control framework.

Fig. 3 describes the smart contract system included in our proposal. As shown, the smart contract system consists of three different smart contracts: the Access Control Contract, the Inspect Contract, and finally the Reputation Contract. Each of these has a specific role in our proposal. Indeed, the RO can use its personal ACC to control access to resources in the network; the Inspect Contract inspects the access requests sent to the ACC and punishes the subjects if they do not follow the rules of the ACC; the Reputation Contract (RepC) computes the reputation score of users in the network; and the Registrar Contract (RegC) which enables trusted nodes to find the identifying information of the access control contracts and the corresponding access control functions. In OSNs, the access to resources are managed by server in a centralized way. In a DOSNs, there is the need to prevent privacy issues, and in our specific scenario we need to check the behavior of users in a decentralized way. This important service is a challenge in a decentralized system due to the dynamics of the network and the lack of a central server which can check all the users. For this reason, we decide to introduce in our framework, a reputation score which highlights the behavior of a user, and thanks to the Reputation Contract, each reputation score is stored in the blockchain and public available. This provides a trustworthy access control system.

It is worthy to note that the reputation record is visible on the blockchain, thereby enabling the resource owner (RO) to exclude users from the policy list if they have negative reputation in the network.

A subject will be classified as *dishonest* for different reasons which are defined and given as an input of our system. For example, a user can be considered dishonest if it sends too frequent access requests to access the resources of the RO. In that specific case, the access frequency is a parameter, and it should take into account real social information, such as the behavior of bots, which are able to access to resource several times per seconds.

5. The system of smart contracts

In this section, we provide a detailed overview of our proposed smart contract system shown in Fig. 3.

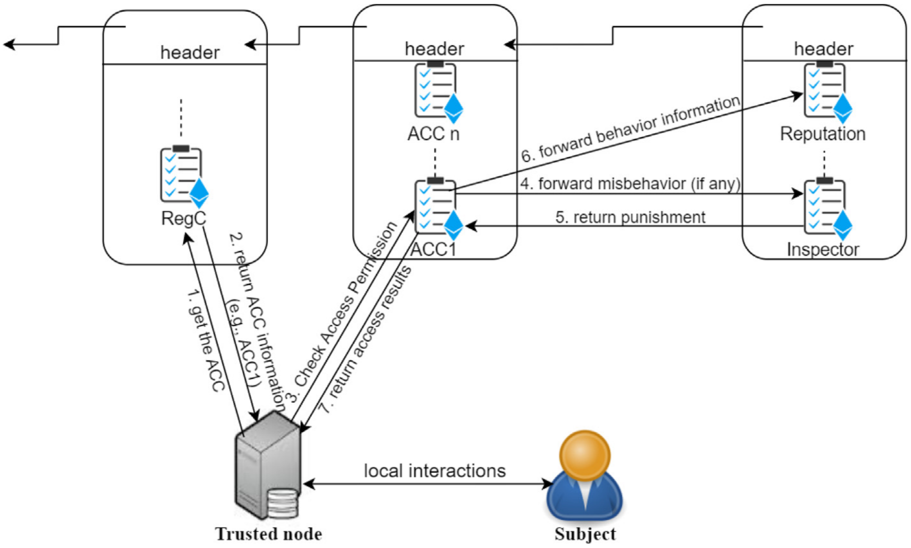

Fig. 2. Blockchain-based architecture for Decentralized Online Social Networks.

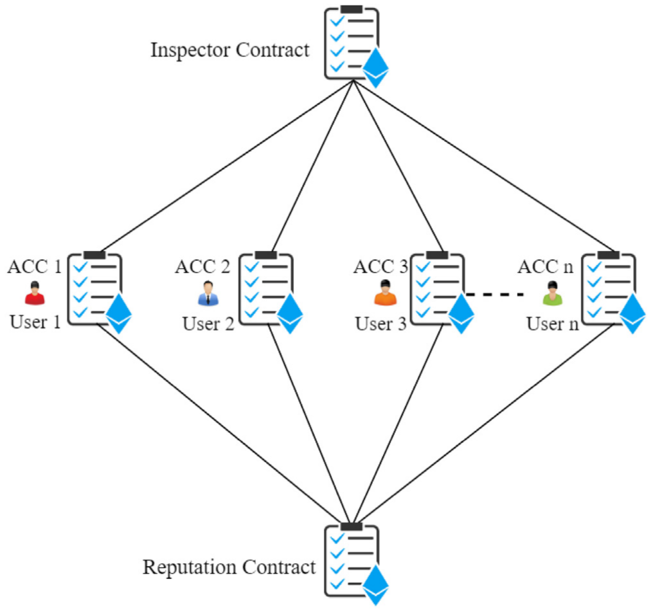

Fig. 3. Illustration of the smart contract system.

5.1. Access Control Contract (ACC)

The ACC is the contract used to control access to resources. Each RO has a unique ACC deployed on the Ethereum blockchain to control access to resources in the DOSN. The RO can create or update an access control policy by sending transactions to its personal ACC that contains information about the policy. Moreover, each ACC is associated with a policy list that contains information about access control policies. An example of the policy list is provided in Table 1. Each field of the policy list is discussed below:

- Subject: This field indicates the unique id (i.e., Ethereum address) of the subject in the DOSN network
- Resource: This field indicates the name of the resource that needs to be included in the access control process.
- Action: It specifies the action that can be performed on the given resource such as view, download, read, write, etc.
- Location: This field holds information about the access location of the subject.
- Time Range: This field can be used by the RO to specify a time range in order to restrict the access time to access resources.
- Permission: This field is used to describe the permission on a certain (resource, subject) pair such as allow or deny.
- Last Request Time (LRT): This field records the time of the last access request from the subject.

For the sake of simplicity, we suppose that each row of the policy list contains a single subject ID only. However, our proposed solution is flexible because the ACL rules can be configured to include multiple subjects as well. Indeed, the RO can create an ACL-based policy for multiple subjects in a single transaction. Moreover, the RO can easily revoke an existing policy by sending a policy revocation transaction specifying the identification information of the policy that needs to be revoked. It is worth noticing that only the RO is eligible to invoke the main functions of the ACC. Indeed, this mechanism enhances the security of our access control system because we want the system to be restrictive as already discussed in the requirements section.

The *Action* field indicates the type of operation that the listed subject is allowed to perform on a certain resource. The RO specifies different operations such as read, write, download, view, etc. The download action rises a question about the content download policy. For instance, if subject A downloads content X, modifies it slightly and it becomes its owner. It is worth noticing that the RO is primarily responsible to manage content download policy. Indeed, the RO will specify a download action in the policy if he/she trusts a subject to perform this operation. As it is common in real-life environments, only trusted users share contents with each other. Our solution also assumes that the RO trusts the subject who downloads the content when downloadable policies are created in the blockchain. Moreover, we assume that owners who offer downloadable content to subjects draft and publish simple privacy policies. Further details on how to enforce such privacy policies are outside the scope of this article.

The *permission* field is used to ensure the validation of access control policies, whereas the *Last Request Time (LRT)* field is used to ensure the dynamic validation of the subjects. The LRT field is essential to inspect the behavior of the subject. For instance, if a subject sends too frequent requests then the ACC is programmed to report such information to the inspector contract, which inspects the behavior and decides an appropriate punishment. It

Table 1
Illustration of the RO's policy list.

Resource	Subject	Location	Time Range	Action	Permission	LRT
File A	user 2	Location A	10:00–15:00	download	deny	2019-06-05 12:20
File B	user 9	Location B	20:00–22:00	write	deny	2019-06-06 20:15
obj 2	user 6	Location C	14:00–15:00	view	allow	2019-06-07 14:11
obj 1	user 6	Location D	10:00–12:00	view	allow	2019-05-20 11:11
File D	user 7	Location E	08:00–11:00	write	allow	2019-05-20 11:11

is worth noticing that only the subjects specified by the RO are allowed to send access requests, and it is expected that they will behave honestly by following the rules of the ACC. Therefore, we configured the ACC to keep track of misbehaving subjects who send too frequent access requests to perform operations on resources. Each ACC maintains a table to record the id of the subject, the type of misbehavior, the time of occurrence, and the type of punishment.

Table 2 shows an example of record to store subjects' misbehavior. To explain a possible scenario, we consider the case listed in Table 2 concerning User 2 who is sending recurrent access requests to the ACC of the RO, and for this reason it has been included in the Table. A possible suitable punishment to handle this situation is to timeout the involved subject. Another example concerns user 6, which is not trusted because the reputation contract has calculated a negative reputation for it during a previous access request. A possible punishment to tackle this situation is to block such as subject. Furthermore, the access requests from user 5 and user 1 are detected as incorrect because they are not included in the policy list of the RO. As a result, an access denied punishment is necessary to tackle such a situation. In addition, user 3 is sending access request from a location that is not specified in the RO's policy list and an access denied decision is returned. Finally, user 7 is sending access request at a time that does not lie within the time range field of the policy and an access denied decision is used to handle this situation.

Ethereum smart contracts provide Application Binary Interfaces (ABIs) or functions, which can be executed by sending transactions to perform policy-related operations [41]. Therefore, our proposed ACC supports various transactions that are necessary to execute these ABIs. A brief discussion of the different functions of our proposed ACC is given below:

- *policyAdd():* This function is used by the ACC to create a new policy on a certain resource, subject pair. When this function is executed, the resulting policy is added to the existing policy list stored in the blockchain.
- *policyUpdate():* This function is used to update an existing policy defined on a specific resource, subject pair. Please note that a policy can be updated at anytime by changing the subject ID, context information, action, the access permission or a combination of these elements.
- *locationUpdate():* This function can be used by the RO to update the access location specified in the ACL-based policy. It requires the identification information of the policy that requires changes in location. Furthermore, the RO needs to specify the new access location and make a transaction to execute the *locationUpdate()* function of the ACC smart contract.
- *timerangeUpdate():* This function can be used by the RO to update the existing time range field specified in the policy. It also requires the identification information of the policy and a new value of time range to update the existing value in the policy.
- *policyDelete():* This function of the ACC can be used to delete an existing policy stored in the blockchain. To accomplish this goal, the RO needs to provide the identification information of the policy that needs to be deleted.
- *accessControl():* The RO or trusted nodes use this function to validate access control policies on the blockchain. Please note that only trusted nodes specified by the RO in the ACC are capable to send transactions in order to validate access permission in the blockchain. Furthermore, this function requires identification information of the policy (i.e., subject ID and name of the resource), context information and the current access time in order to determine the permission associated with the policy. Moreover, this function not only performs static validation of access control policies, but it also checks the behavior of the subjects by considering context information and the number of recurrent requests in a short span of time. Finally, an access request is authorized if it passes both the static and dynamic validation criteria.
- *deactivate():* This function can be used to disable the ACC deployed on the blockchain. We use the *selfdestruct* function [40] to completely free the storage occupied by the code of this smart contract.

5.2. Inspector Contract (IC)

The Inspector contract inspects the behavior of subjects and evaluates their interactions when access requests are sent to the ACCs. We assume that honest subjects will follow the rules of the ACCs when sending requests to perform operations on resources. On the other hand, dishonest subjects behave abnormally by violating the rules of the ACCs. The Inspector contract implements a misbehavior detection mechanism, which inspects the subject's misbehavior and evaluates a punishment according to the type of misbehavior. Once a subject is detected as suspicious by the ACC, a report is forwarded to the IC contract, which evaluates the type of misbehavior and determines a suitable punishment, and returns this information back to the ACC contract. Finally, the ACC enforces the punishment on the involved subject.

The IC contains a special ABI or function that is used to evaluate the behavior of subjects. This function obtains data from the ACC, evaluates it, and decides an appropriate punishment. A discussion of the different functions of this contract is given below:

- *inspectBehavior():* This function is used to inspect the behavior information received from the ACC in order to decide a suitable penalty. It evaluates the behavior and returns information about the punishment to the ACC. Finally, the ACC enforces the decision on the involved subject in the DOSN network.
- *deactivate():* This ABI can be used to disable the inspector contract deployed on the blockchain.

5.3. Reputation Contract (RepC)

Trust management systems allow nodes to decide whether a subject is trustworthy or not, and consequently enables the ROs to reward or punish the subject. Trust and reputation techniques are mainly exploited for large open system. Reputation is usually a global value which indicates the character (such as honest, dishonest, reliable) of an entity, which can be an agent, or a person. On the contrary, trust indicates a personalized reflecting

Table 2
Illustration of the misbehavior list for the smart contract.

Subject	Misbehavior	Time	Punishment
user 2	Recurrent access requests	2019-06-05 12:20	Timeout
user 6	Not from a trusted address	2019-06-07 14:11	Timeout
user 5	incorrect access request	2019-07-05 22:20	Access denied
user 1	incorrect access request	2019-07-05 21:30	Access denied
user 3	Location does not match	2019-04-04 11:30	Access denied
user 7	Time does not match	2019-05-20 11:11	Access denied
....	....	....	....
....	....	....	....

Table 3
Illustration of the reputation record maintained by the RepC contract.

Subject	Reputation score	Time
User 2	-1	2019-06-05 12:20
User 7	$+1$	2019-06-02 22:20
User 4	$+1$	2019-07-03 20:50
User 6	-1	2019-02-08 13:50
....	....	....
....	....	....

of a user's judgment. For sake of the readiness, a trust value is referred to a specific couple of users. Indeed, trust can be obtained from a user's own experience with another user. Literature review reveals that existing reputation management systems can be categorized into two types, i.e., centralized and decentralized [26]. Centralized systems usually consider a central server to store and process the reputation score, e.g., cloud server. On the other hand, decentralized trust management systems consider nodes in the network to calculate and store the reputation record [39].

The Reputation Contract receives information about the behavior of subjects from the ACC and uses it to compute reputation score. Depending upon the access control behavior (i.e., whenever a subject/peer sends access requests to the ACC of the RO), dishonest peers will get negative reputation (i.e., -1) while honest peers will obtain positive reputation (i.e., +1). Moreover, due to the auditability property of blockchain, the reputation record of peers is publicly visible on the blockchain, consequently any user can know at any time the reputation value assigned to another user. This solution achieves transparency, thereby enforcing users to behave honestly in the network. Furthermore, the classification of honest and dishonest peers is necessary to make the system robust and to allow fair access to resources. The main ABIs of this smart contract are briefly discussed below;

- *computeReputation():* This ABI receives information about the subjects from the ACC, and use it to calculate the reputation score. After that, the reputation contract then publishes the result of the evaluation on the blockchain.
- *getLatestReputation():* This ABI can be called offline without sending transaction to obtain the reputation score of a subject. This function requires the unique ID of the subject to retrieve reputation information from the blockchain.
- *deactivate():* This ABI can be used to disable the reputation contract deployed on the blockchain. We use the selfdestruct function to completely free the storage occupied by the code of this smart contract.

The reputation contract maintains a record that contains a list of peers who are identified by their reputation values. For instance, peers who behave honestly in the network have a positive reputation score and vice versa. In this case, the reputation contract maintains a table to record the type of subjects/peers according to their behavior, the corresponding reputation score and the time of computation as shown in Table 3.

5.4. Registrar Contract (RegC)

The Registrar Contract is used to enable trusted nodes to find the identifying information of the ACCs and the corresponding access control functions. As illustrated in Table 4, this contract maintains a record, which is used to achieve the distributed access control functionality in the network, thereby offering a platform that can be used by trusted nodes to look and obtain the necessary information for access control. Indeed, a trusted node can send transaction to check access permission in the blockchain once it has obtained the necessary information such as ID of the desired ACC and the corresponding access control function as illustrated in Table 4.

When the RO is offline or down in the network, a subject can send access request to an online trusted node chosen at random. When sending an access request to a trusted node, a subject S must specify that it is willing to access a resource of the RO, where S and RO indicate the identifiers of the subject and RO, respectively. Once this information is declared in the access request, the trusted node uses the identifier of the RO to obtain the ID of the ACC and the associated access control function defined by the RO as illustrated in Table 4.

With the help of the subject ID, ACC ID and the corresponding access control ABI, the trusted node sends transactions to the required ACC to check access rights in the blockchain. The policy list of the RO will either contain an allow or deny decision for the requesting subject ID. A brief overview of the main fields of this record is given below.

- RO ID: This field shows information about the ID of the owner of the ACC.
- ACC ID: This field reveals information about the id of the ACC smart contract deployed on the blockchain.
- Access Control ABI: This field shows information about the access control ABI used by the ACC contract.

The RO can register the necessary information such as ID of the ACC and the corresponding functions by sending transaction to the RegC contract deployed on the blockchain. Indeed, this mechanism is useful to keep the information in the lookup table consistent. This contract provides the following essential functions to keep the necessary information up to date:

- *register():* This function can be used by the RO to register information about the methods of the ACC into the lookup table. It requires the name of the method, and ID of the ACC as parameters.
- *update():* This function can be used by the RO to update the ID of the ACC or to update information about an existing access control method in the ACC.
- *remove():* This function can be used by the RO to delete information from the lookup table maintained by the RegC contract. This is helpful in situations when the RO is not interested in the distributed access control mechanism, and wants to allow direct access to resources without relying on trusted nodes. Furthermore, it is also necessary when the data of the RO is not replicated at a set of trusted nodes in the network.

Table 4
Illustration of the record maintained by the registrar contract.

RO ID	Access Control Contract (ACC) ID	Access Control ABI
User 3	0 × 109CabceC02C6C3C08b8F06ad72ca47240c0aB23	accessControl ( )
User 2	0xEd8863e30c22C0A03dD34617daE413aE92D089f4	accessControl ( )
User 6	0 × 109CabceC02C6C3C08b8F06ad72ca47240c0aB22	accessControl ( )
User 8	0 × 109CabceC02C6C3C08b8F06ad72ca47240c0aA33	accessControl ( )
....	....	....
....	....	....

- *getACCID():* This function can be called offline to retrieve information about the ID of an ACC and it requires the ID of the RO

Please note that only the RO can send transactions to invoke the register, update and remove functions of the RegC contract. This essentially means that the RegC contract verifies information on the blockchain before making any changes to the lookup table. Once it is determined that the RO has a unique ACC deployed on the blockchain, the RegC contract will make changes to the lookup table. Indeed, the verification process is intended to prevent the insertion of false information into the lookup table.

6. The access control process

The access control policies defined by ACLs are published on the blockchain, thereby allowing the RO as well as trusted nodes to check access rights when an access request is sent by the subject. An overview of the main functions of our proposed system is given below.

- **Policy Creation**: The RO can create a policy for any subject, which defines the access rights on a certain resource. Consequently, the RO needs to provide the ID of the subject, name of the resource that needs to included in the policy, context information, and an access permission in order to create a fine-grained smart policy that considers a wide range of elements.
- **Policy Update**: The RO can update any policy at any time by sending a policy update transaction to the ACC smart contract. Furthermore, the RO needs to provide the identification information of a policy that needs to be updated.
- **Policy Revocation**: The RO can easily revoke a policy from the policy list by sending a policy revocation transaction to the ACC smart contract. This transaction requires the identification information of the policy that needs to be revoked.
- **Access Control**: The ACC contract also provides the functionality to check access permission in the blockchain when an access request is sent by the subject. This function allows the RO or trusted nodes to check access permission in the blockchain.
- **ACC Deletion**: The resource owner can also disable the ACC smart contract by making a contract deactivation transaction in order to free the storage occupied by the code of the ACC.

Our proposed framework is privacy preserving because the real identities of DOSN users are not disclosed to other users in the network. The main rationale behind our approach is motivated from the fact that Ethereum uses a single ID (i.e., Ethereum address) for each account holder. It is worth noticing that the auditability property of blockchains may lead to privacy issues. For instance, the access control policies and the access requests are transparently stored on the blockchain. Moreover, a public blockchain is designed in such a way that all the connected users and even miners can read the data when executing the code of a smart contract. However, current blockchain platforms exploit pseudonymity to provide privacy protection, which ensures that each registered user is associated with an anonymous ID that does not disclose information about his/her real identity. Indeed, our framework exploits the pseudonymity feature to ensure that subjects' anonymous IDs are not directly linked with their real identities.

A trusted node can obtain the ID of the ACC smart contract either from the RO or from the lookup table associated with the RegC smart contract. Our proposed system requires the following essential steps to successfully complete the access control process:

- The RO sends a policy validation transaction to the blockchain which contains the necessary information for access control. Furthermore, the RO will get the results of execution (i.e., allow or deny) once the block containing this transaction is mined by some miner in the Ethereum blockchain network.
- If the RO is offline, a trusted node acts as agent to check access rights on its behalf. Thus, a trusted node sends a transaction to the blockchain containing the required information for access control such as subject ID and name of the requested resource. Consequently, a trusted node will get the results of execution once the block containing this transaction is mined in the network.
- The ACC contract forwards behavior information to the Inspector contract and to the Reputation contract deployed on the blockchain. Furthermore, the reputation smart contract examines the report to calculate a positive or negative score indicating honest and dishonest subject, respectively.
- Finally, the ACC returns the access decision (i.e., to allow the operation or to deny it).

Our proposed solution is *user-centric* because each user can use its own personal ACC smart contract to control access to resources in the network.

The access control algorithm of our proposed system is shown in Algorithm 1. A *smart policy* is uniquely identified by a resource, subject pair. Subjects who do not follow the rules of the ACC and behave dishonestly are blocked for a specific amount of time. To achieve this goal, we use a variable called *BlockUntil* that indicates the time instance until which the requesting subject is blocked. By default, this field contains a value of 0. Furthermore, a two-dimensional mapping is used to uniquely identity a policy when an access request is sent by the subject. The resource field is used as a primary key, while the subject field is used a secondary key to generate the policy list. The following fields are added to the previous privacy policies:

- *minInterval:* It stores the minimum acceptable time interval between two recurrent requests. Consequently, a request will be treated as recurrent if the difference between the current request time and the last request time is less than or equal to minInterval. It is worth noticing that we add a

Algorithm 1 Access Control Algorithm

```
    Input: Resource, Action, Subject, Location, Time
    Output: result, punishment
    Require: BlockUntil, policyCheck ← false, RepC instance reputation, IC instance inspector, policy list policies
 1: $p \leftarrow policies[resource][subject]$
 2: if $BlockUntil \leq time$ then
 3:     if $BlockUntil > 0$ then
 4:         $p.NoRR \leftarrow 0$, $p.LRT \leftarrow 0$, $BlockUntil \leftarrow 0$
 5:     end if
 6:     if $p.policy =$ "$allow$" then
 7:         $policyCheck \leftarrow true$
 8:     else
 9:         $policyCheck \leftarrow false$
10:     end if
11:     if $p.Location \neq Location$ then
12:         $punishment \leftarrow inspector.inspectBehavior(subject, lmsb)$
13:         $reputation \leftarrow reputation.computeReputation(subject, lmsb)$
14:         Push lmsb into the misbehavior list of subject
15:     else if $Time \notin [t_i, t_j]$ then
16:         $punishment \leftarrow inspector.inspectBehavior(subject, tmsb)$
17:         $reputation \leftarrow reputation.computeReputation(subject, tmsb)$
18:         Push tmsb into the misbehavior list of subject
19:     else if $time - p.LRT \leq p.minInterval$ then
20:         $p.NoRR \leftarrow p.NoRR + 1$
21:         if $p.NoRR \geq p.threshold$ then
22:             $punishment \leftarrow inspector.inspectBehavior(subject, famsb)$
23:             $reputation \leftarrow reputation.computeReputation(subject, famsb)$
24:             $BlockUntil \leftarrow time + punishment$
25:             Push famsb into the misbehavior list of subject
26:         end if
27:     else
28:         $p.NoRR \leftarrow 0$
29:         $reputation \leftarrow reputation.computeReputation(subject)$
30:     end if
31: end if
32: $p.LRT \leftarrow time$
33: $result \leftarrow policyCheck$
34: Trigger event returnResult(result, punishment)
```

policy to the ACC with *minInterval* set to 60 s by default. However, the parameter can be decided during the setting phase.

- *NoRR:* the number of recurrent requests in a short time period.
- *threshold:* the threshold on the NoRR. The smart contract judges a misbehavior if the NoRR field is equal to or larger than this threshold. We add a policy to the ACC with threshold set to 3. However, the threshold can be changed depending on system requirements.

Two important information which permit us to implement a context-aware access control system are location and time. Considering the location information, if the location of the subject does not match with the location specified in the policy list, the inspector contract is called to deal with this misbehavior as shown from line 11 to line 14. Furthermore, the reputation contract is called to calculate the reputation of the subject because he/she is behaving dishonestly. Indeed, the reputation contract computes and assigns a negative reputation due to location misbehavior. It is worth noticing that we define location misbehavior as a situation in which the subject sends access request from a location that does not match with the location specified by the RO in the policy list.

Furthermore, the access can be performed during a time specified in the time-range field of the policy. If the subject is sending access requests at a time that does not lie within the time-range field in the policy, the inspector contract is called to deal with this problematic. Here, the valid current time means that the condition, $Time \in [t_i, t_j]$, must be satisfied at the access request. Please note that t_i, respectively t_j, indicate the initial, respectively, the final time in the time-range field. Furthermore, the reputation of the subject is also affected due to time misbehavior. Line 15 to line 18 show details about this misbehavior.

Line 19 to line 25 show a misbehavior when the subject is sending recurrent access requests. Line 19 checks if the difference between the current request time and the last request time (LRT) is less than or equal to *minInterval*. Consequently, the NoRR field is incremented as shown in line 20. Furthermore, line 21 checks if the NoRR is greater than or equal to the predefined threshold. Consequently, the functions of the inspector and reputation contracts are called to deal with this misbehavior. The punishment variable can hold a value in minutes or seconds to block the requesting subject. Furthermore, the value of this variable is added with the current time to calculate the blocking time and the result is stored in the BlockUntil variable as shown in line 24. For instance, if a subject is sending recurrent access requests at 10:00 PM and the punishment is set to 30 min then the subject will be blocked until 10:30 PM. Finally, line 28 to line 29 show that the subject is behaving honestly. Consequently, the number of recurrent request (NoRR) field is set to 0. Moreover, the reputation contract is called to compute a positive reputation for the subject due to honest behavior in the network.

The output of our proposed algorithm is in the form of result and punishment. The former variable holds the access decision (i.e., true or false indicating allow or deny, respectively). The latter variable stores information about punishment (i.e., blocking time or an access denied decision) which is evaluated on the type of misbehavior.

6.1. System configuration

We need to make the following basic configurations to apply the Ethereum platform [7] in our proposed access control system.

- Each user must be associated with an Ethereum account to uniquely represent itself in the network. Consequently, this account allows each peer to claim the deployment of a smart contract and to send transactions to create, update or revoke policies. Furthermore, it can also be used by a subject to identify itself during the access control process.
- All users can configure and run the Ethereum client on their devices. DOSN users can use their Ethereum clients to directly interact with the blockchain. Indeed, they can also send transactions to execute the functions of their personal smart contracts.
- An oracle [1] operated by a location server or mobile network operator provides network based positing information to the ACC. We rely on a reliable infrastructure to provide location information instead of the users in the DOSN network.
- Trusted nodes act as agents to store the data of the resource owner and conduct access control for the resources when the resource owner is offline or down in the network. To achieve this goal, the subject sends access request to an online trusted node, which uses the accounts of the subject and resource owner to check access permission in the blockchain. We assume that trusted nodes are benign and selected by the resource owner in the network

6.2. Challenge–response authentication

The Challenge–Response protocol is executed when a subject sends an access request to the RO or to a trusted node when the former is offline. The basic steps of this protocol are discussed as follows;

- **Declaration:** When sending an access request, the subject must declare its unique ID (i.e., Ethereum account), type of operation (i.e., read, write, download, etc.) that the subject is willing to perform on a resource of the RO.

- **Information Verification:** Given the assertion of the subject, the RO sends a transaction to the *accessControl()* function of its personal ACC in order to determine whether the subject is eligible to perform the requested operation on the resource R. Hence, the access control algorithm checks the data on the blockchain to validate access permission. Furthermore, if the RO has specified location information in the policy, the algorithm will interact with an oracle to verify the access location of the subject. Similarly, the algorithm determines the validity of the current access request time. Consequently, the subject is allowed to perform the requested action on the resource R if the permission in the policy is allow; otherwise the operation is denied. Please note that an online trusted node performs the verification process if the RO is offline or down in the network. Finally, the node receiving the request challenges the subject in order to verify the authenticity of the Ethereum account.
- **Challenge:** The RO or a trusted node selects a random message m, and asks the subject to sign it.
- **Response:** Response from the subject determines the real owner of the Ethereum account because the private key of the subject is required to sign the message. Therefore, the creation of a correct signature is only possible if the subject possesses a valid private key. To accomplish this goal, we use the following function:

  $$S = Sign(pk; ID; m) \tag{1}$$

  where pk represents the private key of the subject, ID represents the address of the Ethereum account and m represents the random message sent to sign. Thus, a correct signature is only possible if the subject possesses the corresponding private key which is uniquely defined for each Ethereum account, and it is also unique for each DOSN user. The subject then sends S back to the RO or to the trusted node, depending upon the access request scenario.
- **Response Confirmation:** The RO or trusted nodes will allow the subject to perform the requested operation on a resource R if a correct signature is generated by the subject. To accomplish this goal, the RO or trusted node uses the following verification function:

  $$confirmResponse(ID; m; S). \tag{2}$$

Finally, the RO or trusted node will allow access to the resource if and only if the verification process is successful. Please note that the authentication protocol can be executed offline, and the Solidity sha3 [12] function can be used to securely generate the signature. This technique can be used to generate a message signature without disclosing the private key.

7. Performance evaluation

Every transaction that is used to invoke the function of a smart contract requires the payment of a fee in order to compensate the mining node for the execution of transaction and saving it on the blockchain. Ethereum uses gas to express this fee. Users can purchase gas from the mining nodes by paying Ether. Please note that Gas and Ether are two distinct terms because gas indicates a constant cost of performing an operation on a Blockchain network, whereas Ether is a volatile virtual currency, which is used to pay for the network resources.

7.1. Experiment setting

We use the solidity [12] programming language to develop prototypes of our proposed smart contracts, and deployed them on the Rinkeby Ethereum testnet. The experiments were conducted in the month of September 2019 during which we observed that 1 Ether ≈ 189 USD and an average gas value of ≈ 0.000000021 ETH. Experimental results show that the ACC contract requires 3,180,418 gas for deployment. Costs of the remaining contracts are discussed in the following section.

Table 5
Costs of the different functions of the ACC smart contract.

ACC Function	Gas used	Cost (ether)	USD ($)
policyAdd()	145,625	0.000146	0.0275
policyUpdate()	84,336	0.000084	0.0158
policyDelete()	84,462	0.000084	0.0158
thresholdUpdate()	32,382	0.000032	0.0060
locationUpdate()	32,500	0.000032	0.0060
timerangeUpdate()	32,500	0.000032	0.0060
accessControl()	61,648	0.000062	0.0117
deactivate()	13,455	0.000027	0.0051

Table 6
Costs of the different functions of the Inspector smart contract.

Inspector function	Gas used	Cost (ether)	USD ($)
Contract Deployment	1,310,013	0.00131	0.2475
inspectBehavior()	220,350	0.00022	0.0415
deactivate()	13,455	0.000027	0.0051

Table 7
Costs of the different functions of the Reputation smart contract.

Reputation function	Gas used	Cost (ether)	USD ($)
Contract Deployment	741,141	0.000741	0.1400
computeReputation()	112,407	0.000112	0.0211
deactivate()	13,455	0.000027	0.0051

Table 8
Costs of the different functions of the Registrar contract.

Registrar function	Gas used	Cost (ether)	USD ($)
Contract Deployment	1,310,013	0.0085	1.6065
register()	29,848	0.00003	0.0056
update()	44,640	0.000045	0.0085
remove()	54,613	0.000055	0.0103

7.2. Results

Table 5 shows the one-time constant costs of the different functions of the ACC smart contract.

Rahman et al. [32] proposed the Role-based access control for social network using Ethereum smart contract. Their experimental results show that creating a simple RBAC policy that compares the subject's asserted role with the role specified in the RBAC policy in the Ethereum blockchain requires 27,864 gas. Moreover, they also implemented a policy evaluation (i.e., access control) function that compares the subject's role with the policy in the blockchain and returns allow or deny. This evaluation function requires 22,808 gas consumption on the Rinkeby testnet.

Table 6 shows the one-time constant costs of the different functions of the Inspector smart contract. The Inspector contract receives misbehavior information about the subject together with the time of misbehavior to determine an appropriate punishment. Table 7 shows the one-time constant costs of the different functions of the reputation smart contract. This contract requires a one-time deployment cost of 0.000741 ether. The *computeReputation()* is the main function that is used to evaluate the reputation score of users in the DOSN network. Finally, Table 8 shows the one-time constant costs of the different functions of the Registrar smart contract deployed on the Ethereum blockchain.

We performed an experiment to evaluate the effects of the number of subjects in the ACL-based policy on Gas consumption

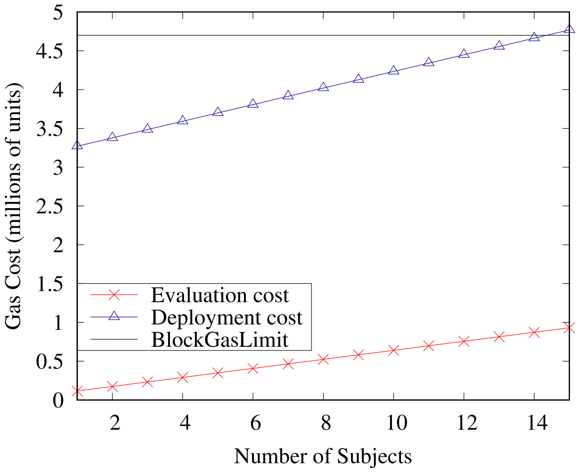

Fig. 4. Number of Subjects in the *policyAdd()* function vs. Gas cost.

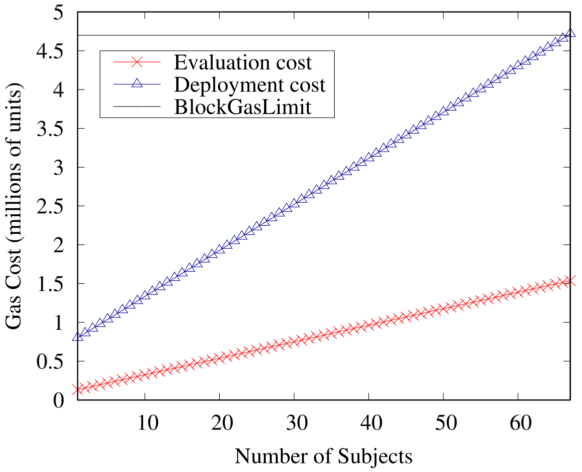

Fig. 5. Number of Subjects in the *computeReputation()* function vs. Gas cost.

in the Rinkeby testnet as shown in Fig. 4. Results of our experiment reveal that both the function evaluation and ACC contract deployment cost increases linearly as we increase the number of subjects in the *policyAdd()* function. However, as the number of subjects exceeds 14, the BlockGasLimit is reached. The gas limit is the maximum amount of gas that a transaction is allowed to consume, if a transaction exceeds this amount, the gas is spent but the execution effects on the contract's state are discarded. Indeed, this mechanism is intended to prevent long computations that would stall the EVM. Furthermore, each block has an associated block gas limit that limits the number of computations that can be executed by all transactions in that block. The value of this limit on the Rinkeby testnet is 4,700,000. In short, the given results reveal that the RO can create a single policy for a maximum of 14 subjects in the network. It is worth noticing that the transaction fee depends on the complexity of transaction sender wants to make. Consequently, the more complex a transaction, the more Gas we need to pay for its execution on the blockchain. Moreover, operations performed by the smart contract code and storing information in the smart contract are the two complexity factors that determine the amount of gas that a transaction will consume.

The *computeReputation()* function of the reputation contract is flexible because it can be easily configured to evaluate the reputation score for multiple subjects in the DOSN network. Indeed, multiple subjects can be provided as arguments together with the type of misbehavior in order to evaluate and simultaneously assign the same value to each subject. We evaluated the gas consumption costs by varying the number of subjects in this function as shown in Fig. 5. Results of the experiment show that both the contract deployment and function evaluation costs increase linearly as we increase the number of subjects. However, when the number of subjects exceeds 66, the BlockGasLimit is encountered. This essentially means that the proposed reputation contract can simultaneously evaluate the reputation for 66 subjects in the network. Although, the current solution evaluates the reputation for a single subject, more complex scenarios may require the consideration of multiple subjects.

The *inspectBehavior()* function of the Inspector contract currently requires the ID of a single subject and the type of misbehavior to evaluate a possible penalty. However, we also evaluated this function by passing multiple subjects and the type of misbehavior as arguments. Results of the evaluation show that this function can simultaneously evaluate the punishment for up to 75 subjects in the network. However, as the number of subjects exceeds this limit, the Ethereum BlockGasLimit is encountered, thereby discarding transactions containing more than 75 subjects (see Fig. 6).

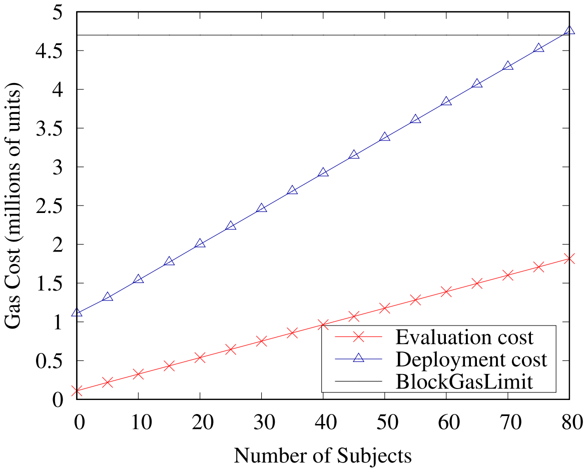

Fig. 6. Number of Subjects in the *inspectBehavior()* function vs. Gas cost.

8. Conclusions

In this research, we proposed an auditable and trustworthy access control system for DOSNs using access control list (ACL). The resource owner has a unique personal ACC deployed on the blockchain, which is used to control access to resources in the network. The proposed solution consists of different smart contracts, which are designed to meet the trust and security requirements of DOSN. The proposed solution achieves the auditability property successfully because all functions that are invoked on the smart contracts are reflected on the Ethereum blockchain. Consequently, all the actions performed by a user are publicly visible on the blockchain. Indeed, a user will not be able to perform a secret action without the knowledge of other users. Furthermore, resources are available to authorized users even when the resource owner (RO) is not available or disconnected. We proposed a smart contract system which achieves the verification property because the challenge–response protocol is used to perform a secure verification of the user's identity and to verify the status of access permission in the blockchain.

Finally, we implemented and evaluated the approach by exploiting the Rinkeby Ethereum testnet to deploy the smart contracts, and the experimental results show the feasibility of our proposed scheme. Indeed, our scheme requires 61,648 gas to evaluate ACL rules. As future work, we plan to evaluate our framework in a real DOSN scenario, and we plan to investigate more in depth the problem of Access Control by comparing our approach with other different models in order to highlight the benefits in a DOSN. In particular, we plan to investigate the cross-context misbehavior of subjects, as well as the changing of behavioral patterns within a specific context or in a cross-context scenario.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgment

This work was partially funded by the European Commission under contract number H2020-825585 HELIOS.

References

[1] J. Adler, R. Berryhill, A. Veneris, Z. Poulos, N. Veira, A. Kastania, Astraea: A decentralized blockchain oracle, in: 2018 IEEE International Conference on Internet of Things (iThings) and IEEE Green Computing and Communications (GreenCom) and IEEE Cyber, Physical and Social Computing (CPSCom) and IEEE Smart Data (SmartData), IEEE, 2018, pp. 1145–1152.

[2] R. Baden, A. Bender, N. Spring, B. Bhattacharjee, D. Starin, Persona: an online social network with user-defined privacy, in: ACM SIGCOMM Computer Communication Review, Vol. 39, No. 4, ACM, 2009, pp. 135–146.

[3] O. Bodriagov, G. Kreitz, S. Buchegger, Access control in decentralized online social networks: Applying a policy-hiding cryptographic scheme and evaluating its performance, in: 2014 IEEE International Conference on Pervasive Computing and Communication Workshops, PERCOM WORKSHOPS, IEEE, 2014, pp. 622–628.

[4] S. Buchegger, A. Datta, A case for P2P infrastructure for social networks-opportunities & challenges, in: 2009 Sixth International Conference on Wireless on-Demand Network Systems and Services, IEEE, 2009, pp. 161–168.

[5] S. Buchegger, D. Schioberg, L. Vu, A. Datta, Implementing a P2P Social Network - Early Experiences and Insights from PeerSoN, in: Second ACM Workshop on Social Network Systems, Co-Located with EuroSYs 2009, 2009.

[6] S. Bugiel, S. Heuser, A.-R. Sadeghi, Flexible and fine-grained mandatory access control on android for diverse security and privacy policies, in: Presented As Part of the 22nd {USENIX} Security Symposium , {USENIX} Security 13, 2013, pp. 131–146.

[7] V. Buterin, et al., Ethereum: A Next-Generation Smart Contract and Decentralized Application Platform, Vol. 7, 2014, URL https://github.com/ethereum/wiki/wiki/%5BEnglish%5D-White-Paper.

[8] B. Carminati, E. Ferrari, A. Perego, Rule-based access control for social networks, in: OTM Confederated International Conferences" on the Move to Meaningful Internet Systems", Springer, 2006, pp. 1734–1744.

[9] L.W. Cong, Z. He, Blockchain disruption and smart contracts, Rev. Financ. Stud. 32 (5) (2019) 1754–1797.

[10] M. Conti, A. De Salve, B. Guidi, F. Pitto, L. Ricci, Trusted dynamic storage for dunbar-based p2p online social networks, in: OTM Confederated International Conferences" on the Move to Meaningful Internet Systems", Springer, 2014, pp. 400–417.

[11] L.A. Cutillo, R. Molva, T. Strufe, Safebook: A privacy-preserving online social network leveraging on real-life trust, Commun. Mag. 47 (12) (2009) 94–101.

[12] C. Dannen, Solidity programming, in: Introducing Ethereum and Solidity, 2017, pp. 69–88.

[13] A. Datta, S. Buchegger, L.-H. Vu, T. Strufe, K. Rzadca, Decentralized online social networks, in: Handbook of Social Network Technologies and Applications, Springer, 2010, pp. 349–378.

[14] A. De Salve, R. Di Pietro, P. Mori, L. Ricci, A logical key hierarchy based approach to preserve content privacy in decentralized online social networks, IEEE Trans. Dependable Secure Comput. (2017).

[15] A. De Salve, B. Guidi, P. Mori, L. Ricci, V. Ambriola, Privacy and temporal aware allocation of data in decentralized online social networks, in: International Conference on Green, Pervasive, and Cloud Computing, 2017, pp. 237–251.

[16] A. De Salve, P. Mori, L. Ricci, A privacy-aware framework for decentralized online social networks, in: Database and Expert Systems Applications, Springer, 2015, pp. 479–490.

[17] M. Durr, M. Maier, F. Dorfmeister, Vegas–A Secure and Privacy-Preserving Peer-to-Peer Online Social Network, in: Privacy, Security, Risk and Trust, PASSAT, 2012 International Conference on and 2012 International Confernece on Social Computing, SocialCom, 2012, pp. 868–874.

[18] B. Guidi, When blockchain meets online social networks, Pervasive and Mobile Computing 62 (2020) 101131.

[19] B. Guidi, T. Amft, A. De Salve, K. Graffi, L. Ricci, Didusonet: A P2P architecture for distributed dunbar-based social networks, Peer-to-Peer Netw. Appl. 9 (6) (2016) 1177–1194.

[20] B. Guidi, M. Conti, A. Passarella, L. Ricci, Managing social contents in decentralized online social networks: A survey, Online Soc. Netw. Media 7 (2018) 12–29.

[21] B. Guidi, A. Michienzi, K. Koidl, K. Kapanova, A multilayer social overlay for new generation DOSNs, in: Proceedings of the 5th EAI International Conference on Smart Objects and Technologies for Social Good, 2019, pp. 114–119.

[22] V.C. Hu, D.R. Kuhn, D.F. Ferraiolo, J. Voas, Attribute-based access control, Computer 48 (2) (2015) 85–88.

[23] L. Jiang, X. Zhang, BCOSN: A blockchain-based decentralized online social network, IEEE Trans. Comput. Soc. Syst. 6 (6) (2019) 1454–1466.

[24] A. Kayes, J. Han, W. Rahayu, T. Dillon, M.S. Islam, A. Colman, A policy model and framework for context-aware access control to information resources, Comput. J. 62 (5) (2019) 670–705.

[25] D.D.F. Maesa, P. Mori, L. Ricci, Blockchain based access control, in: IFIP International Conference on Distributed Applications and Interoperable Systems, Springer, 2017, pp. 206–220.

[26] S. Marti, H. Garcia-Molina, Taxonomy of trust: Categorizing p2p reputation systems, Comput. Netw. 50 (4) (2006) 472–484.

[27] M. Mondal, Y. Liu, B. Viswanath, K.P. Gummadi, A. Mislove, Understanding and specifying social access control lists, in: 10th Symposium on Usable Privacy and Security, {SOUPS} 2014, 2014, pp. 271–283.

[28] R. Narendula, A. Papaioannou, K. Aberer, A Decentralized Online Social Network with Efficient User-Driven Replication, in: IEEE SocialCom, 2012.

[29] R. Nasim, S. Buchegger, XACML-Based access control for decentralized online social networks, in: 2014 IEEE/ACM 7th International Conference on Utility and Cloud Computing, 2014, pp. 671–676, http://dx.doi.org/10.1109/UCC.2014.108.

[30] R. Nasim, S. Buchegger, Xacml-based access control for decentralized online social networks, in: Proceedings of the 2014 IEEE/ACM 7th International Conference on Utility and Cloud Computing, IEEE Computer Society, 2014, pp. 671–676.

[31] K.P. Puttaswamy, B.Y. Zhao, Preserving privacy in location-based mobile social applications, in: Proceedings of the Eleventh Workshop on Mobile Computing Systems & Applications, ACM, 2010, pp. 1–6.

[32] M.U. Rahman, F. Baiardi, B. Guidi, L. Ricci, Protecting personal data using smart contracts, in: International Conference on Internet and Distributed Computing Systems, Springer, 2019, pp. 21–32.

[33] M.U. Rahman, B. Guidi, F. Baiardi, L. Ricci, Context-aware and dynamic role-based access control using blockchain, in: International Conference on Advanced Information Networking and Applications, Springer, 2020, pp. 1449–1460.

[34] A. Raman, S. Joglekar, E. De Cristofaro, N. Sastry, G. Tyson, Challenges in the decentralised web: The mastodon Case, 2019, arXiv preprint arXiv:1909.05801.

[35] R.S. Sandhu, P. Samarati, Access control: principle and practice, IEEE Commun. Mag. 32 (9) (1994) 40–48.

[36] A. Shakimov, H. Lim, R. Cáceres, L.P. Cox, K. Li, D. Liu, A. Varshavsky, Vis-a-vis: Privacy-preserving online social networking via virtual individual servers, in: Communication Systems and Networks, COMSNETS, 2011 Third International Conference on, 2011, pp. 1–10.

[37] R. Sharma, A. Datta, SuperNova: Super-peers based architecture for decentralized online social networks, in: COMSNETS, 2012, pp. 1–10.

[38] A. Tootoonchian, S. Saroiu, Y. Ganjali, A. Wolman, Lockr: better privacy for social networks, in: Proceedings of the 5th International Conference on Emerging Networking Experiments and Technologies, ACM, 2009, pp. 169–180.

[39] Y. Wang, J. Vassileva, A review on trust and reputation for web service selection, in: 27th International Conference on Distributed Computing Systems Workshops, ICDCSW'07, IEEE, 2007, p. 25.

[40] M. Wöhrer, U. Zdun, Design patterns for smart contracts in the ethereum ecosystem, 2018.

[41] R. Xu, X. Lin, Q. Dong, Y. Chen, Constructing trustworthy and safe communities on a blockchain-enabled social credits system, in: Proceedings of the 15th EAI International Conference on Mobile and Ubiquitous Systems: Computing, Networking and Services, ACM, 2018, pp. 449–453.

Mohsin Ur Rahman received his Master of Philosophy from the Department of Computer Science, University of Malakand with distinction in the class. He served as a research assistant on a malware detection projected, namely, an artificial immune system based general-purpose intrusion detection system which was a collaborative research project between National University of Science and Technology (NUST), Islamabad and the University of Malakand. He worked on several topics on mobility issue in Mobile Ad hoc networks as well as the analysis and detection of security attacks. He is currently pursuing his Ph.D. with fellowship from the department of computer science, University of Pisa. His research interests include IoT security, distributed ledger technology, access control systems and malware detection.

Barbara Guidi is currently an Assistant Professor at the Department of Computer Science of the University of Pisa. She received her B.Sc. and M.Sc. in Computer Science from the University of Pisa, Italy, in 2007 and 2011, respectively. She received her Ph.D. degree in Computer Science from the University of Pisa, in 2015. In 2014, during her Ph.D., she was a visitor at the Heinrich Heine University of Dusseldorf. She was a Co-Chair for the conference EAI GoodTechs 2017, and Co-Chair of several workshops. She has been involved in the TPC of several international conferences and workshops, and has been a reviewer for relevant scientific journals. She received three Best Paper Awards: at the International Conference DCNET 2013, at the workshop LSDVE 2017, and at LSDVE 2018. She is part of the UNIPI Team in the H2020 HELIOS project. Her current research interests include distributed systems, P2P networks, complex networks, Social Network Analysis, Decentralized Online Social Networks, dynamic community detection, and the Blockchain technology.

Fabrizio Baiardi graduated at Università di Pisa where he is currently a full professor in computer science. His main research interest is cyber risk assessment and management. He is a cofounder of Haruspex, a startup that develops risk assessment and management tools based upon adversary emulation. He holds some patents on intrusion detection.