A block-chain based approach to Resource Sharing in Smart Neighbourhoods

No Author Given

No Institute Given

Abstract. Sharing resources is a growing trend in today's world, with many forms such as sharing for monetary compensation, bartering, or simply for social good. Sharing is a powerful way of enabling recycling and reducing wasteful practices that are the partly due to rigid ownerships. Sharing, in particular for social good, needs a level of trust and control on the terms and conditions, to provide sufficient assurance about the outcome. The goal of this paper is to design and provide a proof of concept implementation of a neighbourhood sharing platform that is based on a permissioned blockchain. The blockchain will be maintained by network providers who serve their subscribers in a neighbouring geographic area, that we refer to the neighbourhood. We use this geographic proximity as an initial basis of trust and convenience for sharing, and employ attribute-based access control to allow users to specify the terms of accessing their sharable items. We describe our proposed system that uses smart contracts to enforce the conditions of access, and analyse its security and information leakage that would lead to privacy loss.

Keywords: Access control·Permissioned blockchain·Resource sharing.

1 Introduction

As sharing economy grows [19,25,10] more users are willing to share their resources and being compensated by money or other rewards. Example services include sharing residences, vehicles, assets (e.g., lawn mover), or content (e.g., video or a file). Sharing requires some level of trust so that the owner have confidence about the outcome. This is commonly provided by a trusted intermediary service such as Uber [24] or Airbnb [1] that provides assurance for both sides of a sharing relation. There are also community sharing [13] that find ways of sharing things from clothing to hardware to digital goods. There are also groups emerging that consciously identify with the big-picture sharing movement. Sharing economy could responsibly mitigate over-consumption and move toward building community connections. Our goal is to empower neighbourhoods to engage in sharing economy and enjoy all the benefits of sharing goods and resources, as well as creating a richer and closer community.

Our starting point is a neighborhood that consists of a set of residential units, houses or units in a high-rise, that are in close proximity and are each equipped with a hub that provides connectivity to the Internet. We use geographic proximity as an initial basis for trust and convenience for sharing. As noted in [17], "In this view lies a, typically implicit, presumption that collocation facilitates trust formation, largely because

of greater opportunity of face-to-face[8,14]" and "In their seminal paper, Storper and Venables (2004) [21] argue that the type of interaction enabled by physical proximity requires that the actors invest time and effort in personal relationships and that this form of exchange renders greater understanding and human closeness."

The hub will also act as the first point of network connectivity for smart devices that are in the home. The goal of this paper is to provide an infrastructure for a neighborhood to allow home users to share their resources according to their personal preferences (policies). Although our focus is on sharing and do not consider exchange of money, our work can be extended to include exchange of goods and services for money.

The problem of sharing goods can be easily solved if one can assume a trusted authority (TA) that acts as an intermediary in all the interaction in the neighborhood:
- A user Alice with resource R can specify a policy p for accessing R (conditions of access), and send a digitally signed copy of (R,p) to the TA.
- The TA will verify that Alice is in the possession of R (this can be done by Alice providing sufficient evidence or making a deposit that will be taken away if her claim is false), and will publish (R,p).
-Users in the neighborhood can scan the published items. An interested user Bob will contact the TA with their credentials.
- TA verifies their credentials and if it satisfies the policy, grants access to the resource (e.g., by establishing a link between Alice and Bob and notifying Alice of Bobs interest).

This is a fully centralized system that in practice is unacceptable primarily because,
(i) TA will learn all the interactions and will be responsible for evaluating the policies, and must be trusted in every sense,
(ii) TA will have multiple roles & responsibilities that can be conflicting, and it requires significant process management, processing, and handling traffic, and
(iii) TA will be a single point of failure and can be subjected to various types of attack.

The goal of this paper is to design a blockchain based distributed ledger system that would simulate the TA in the above setting. Decentralized platforms such as Ethereum [6] and Hyperledger Fabric [3] can run arbitrary (Turing complete) programs (smart contracts) in a trusted way, and can be used to implement the functionality of TA, while avoiding a central point of trust and also single point of failure.

An architecture for smart home. Smart homes are one of the fastest growing areas of Internet of Things (IoT) with many providers such as Apple Home [4], Samsung SmartThings [20], Amazon AWS (Amazon Web Services) [2], IBM Watson [12], Microsoft Azure [16], etc. Existing smart homes are cloud-based: devices and actuators in the home communicate to the cloud through a hub whose main role is to provide the networking interface for the devices in the home, and Internet connectivity and communication with the cloud. In [5] and [7] this architecture was analyzed and a wide range of attacks on security and resiliency of the home were outlined. Authors also suggested an alternative architecture where the Hub provides a first level of intelligence to allow the home to function as a self-contained computing unit, and will use cloud as a powerful computing server to perform more complex tasks such as using advanced machine learning tasks that use multiple sources of data. We use this smart home architecture and refer to the hub as "EdgeHub", to emphasize its computing capability at the edge of the network. We

assume each EdgeHub is connected to the Internet through a network service provider, and for simplicity assume a smart home is associated with a single user. The EdgeHubs in a neighbouring geographical area, that we refer to a *smart neighborhood*, may share their content (e.g., movies), and computing resources (e.g., disk space), or other physical objects such as a book.

A Distributed ledger (DL) system for smart neighborhood. We consider a permissioned blockchain, called neighborhood blockchain or *N-chain*, that is developed and maintained by a set of nodes that we call consensus nodes (*C-nodes*), and have the role of (1) verifying the received transactions and if correct and agreeable, (2) publishing it on N-chain, and (3) maintaining a unified view of the chain at different nodes. Each block in the N-Chain includes the hash of the previous block and so provides an ordered sequence of events. N-chain enforces the user specified policies for accessing the sharable resources. Although our emphasis is on a neighborhood, the system can be extended to work with multiple N-Chains. A N-chain is well incentivized as it provides opportunity for small service providers to use the blockchain infrastructure to access and exchange goods and services. Establishing a blockchain for a neighborhood would be an attractive proposition for the Internet Service providers (ISP), each acting as a C-node. An ISP has a long-term relationship with a subscriber EdgeHub, and in addition to providing a point of entry to the Internet for the EdgeHub, it can also vouch for the EdgeHub's owner identity and attributes.

1.1 Our contribution

We propose an architecture for resource sharing and management in a smart neighborhood that facilitates efficient use of resources among the smart homes while ensuring secure operation and privacy awareness by which we mean, users are aware of the visibility of their data[1]. We consider a permissioned blockchain setting that is managed and maintained by a set of C-nodes that aim to provide computing and storage space for smart home communities. We also assume the following trusted services:

– **Enrollment Service (EnS)** that will generate a public and private key pair for an EdgeHub, and will provide the private key and a certificate for the public key to the EdgeHub. In practice this service will be provided by ISPs: each ISP will be a certificate authority for its subscribers.

– **Attribute Authority Service (AAS)** who will verify and share certificates for the attributes of the users and resources. Users will obtain certificates for their own attributes (e.g. $age > 18$) and their resources (e.g. a high definition video clip) from relevant authorities. The resource attributes can also be certified by its owner only. These certificates will form the basis of confidence that a requester will have. AAS can interact with a set of trusted authorities, each responsible for one or more types of attributes.

The N-chain will be used to store, (i) information about resources (name of the resource, type and attributes of the resource, attributes of the resource owner (e.g. public key)), (ii) the access control policies (e.g., resource α can be accessed by users with attribute β)

[1] In our future work we will consider privacy protection mechanisms that will guarantee privacy at the cost of higher system complexity.

that governs the access to the resources, (iii) authenticated supplementary information (e.g. state of the resource, such as "occupied") for evaluation of these policies as well as (iv) handling misbehaviour and penalties in case of a detected misbehaviour (e.g., when Bob tries to access Alice's resource more than once, his access is blocked).

Important properties of this system are: (1)*Expressive policies:* using ABAC, one can express complex policies. For example, Bob can only access the content if he has certain combination of attributes. (2) *Time-limiting rules:* of access can be implemented without the need for an external trusted global clock, by leveraging the updating (block creation) process of the DL. (3) *Financial incentives and penalties:* can be applied natively. For example, Alice can request that Bob makes a safety deposit, unlockable by a set of adjudicators. In the case that Bob damages the resource, he will lose his deposit. (4) *Support for reputation:* The ledger can also be used to maintain a reputation score for each EdgeHub (user) public key, and therefore a source of experience-based trust and learning from past experiences, when there is no direct interaction between users.

We design smart contracts to provide the required functionalities such as adding new users to the network, receive access requests, evaluate and provide access decisions. We describe the the system and its security goals in Section 3. We consider the following types of smart contracts: (i) *Registration Contract (RC)* holds ACC smart contract addresses and their functions each associated to a resource (identifier). Also, it provides functions to register a new resource and update or delete an existing resource. (ii) *Attribute Repository Contracts (ARC)* each representing attributes of resources or a user. (iii) *Access Control Contracts (ACC)* represent access control policies and their evaluation procedures for sharable resources. The ACCs also provide functions for adding, updating, and deleting access control policies. (iv) *Adjudicator Contract (ADJ)* handles all misbehaviour (for e.g., frequent access requests) reportings, and impose penalties on misbehaved users.
The flow of function calls and message transmissions is shown in Figure 2. For each access control request from a resource requester, the corresponding ACC function will be executed and verified by C-nodes in the system, ensuring the security (safety) of the access control. ACC also have a predefined instruction to find misbehaviours and provide these as a report to ADJ which judges the misbehavior and returns the corresponding penalty. In this paper, *Attribute-based access control (ABAC)* model is used to specify the access policies. ACCs retrieves the required attributes of resources, resource providers and resource requesters provided by the respective ARCs for the policy evaluation to make an access decision (e.g., Grant or Deny). We use smart contracts to achieve distributed and trustworthy access control evaluation to enforce the defined policies. In section 4.2, we analyze the security and privacy of the system. In Section 5, a case study is provided to demonstrate the application of the proposed architecture. For this case study, we use private Ethereum blockchain to develop and deploy all the above smart contracts, and provide the smart contracts and their explanation.

1.2 Related work

The idea of using smart contracts for access control was used in [15,18,26]. All these systems are based on permissionless blockchain (Ethereum) where proof-of-work is used

for achieving consensus. We use permissioned blockchain and use C-nodes to achieve consensus. This will significantly improve speed of transactions as well as not requiring the wasteful proof-of-work. The closest work among these compared to ours is [15] that uses a smart contract-based access control framework for IoT.

There are however major differences between the two works, firstly because we use permissioned blockchain which allows us to implement an enrollment smart contract that is used to enroll EdgeHubs and provide a first level verification for access requests through C-nodes, and secondly, using ABAC as the access control model. Permissioned blockchains are significantly more efficient and with smaller processing delay, compared to non-permissioned blockchains. They also allow a level of access control on blockchain data that is essential for user privacy in our construction. The access control model in [15] in the basic access control matrix which is defined for pair subject-object. We use ABAC that provide a fine-grained access control and ability to express a wide range of access conditions. One of the important components of an ABAC system is assurance about the attributes. We introduce the required functions in the access control evaluation step (that is performed by the smart contract ACC) to provide verifiability for these attributes functions. We use an ADJ in our system that has the same role as the judge contract in [15], however the adjudication rules are completely different.

Organization of paper. Next section discusses the preliminaries and Section 3 introduces our system model, configuration, and architecture. Section 4 presents an access control model and discusses the privacy and outlines the use of attribute based encryption. Section 5 provides a proof of concept implementation and Section 6 concludes the paper.

2 Preliminaries

EdgeHub. It is the first point of contact to network connection for smart devices in the home, and home connectivity to the Internet and access to cloud service. It provides a range of functionalities including edge computing services to enable smart home to function on its own at a basic level (that can be defined by the service providers), to remove total reliance and constant interaction with the cloud. In this paper we focus on the functionality of EdgeHub as a registered node in a permissioned blockchain network, representing the smart home resources that are offered for sharing.

Blockchain and smart contracts. A blockchain is a decentralized, distributed, and oftentimes public, digital ledger that is used to record transactions across many computers. Transactions are stored in units of block which includes hash of the previous blocks. C-Nodes, who use a consensus protocol, are responsible to agree on the order of blocks and maintain a consistent view in all nodes, so that any involved record cannot be altered retroactively, without the alteration of all subsequent blocks. Blockchain has two types, permissionless and permissioned blockchain.

Permissioned blockchain. In a permissioned blockchain, there is a layer of access control which is used to manage the blockchain, and the access policies are written and modified by the blockchain administrators. The management policies determine who can write or read from the blockchain, or in a fine grained manner, they determine which part of a transaction is observable and which is not to other nodes.

In permissioned blockchain the task of executing and verifying transactions is performed by a set of nodes that are called endorsers. We assume that each EdgeHub subscribes to one ISP, which authenticate and provides service to that EdgeHub, and acts as endorser for all the transactions broadcasted to N-chain through that EdgeHub. Endorser also signs the transactions received, after verifying them, and then send them to a designated set of nodes that are called Consensus Nodes, or C-Nodes, which are responsible to validate the transactions and publish the transactions on the ledger while keeping their view consistent with each other. A membership service is usually available to add new nodes to the system and assign roles to them. In practice this service will be provided by ISPs.

Smart contract. A smart contract [6] is a public piece of code that resides on the blockchain, specifies a protocol that is run on the underlying consensus computer which guarantees trusted execution (assuming majority of consensus nodes are honest), and handles the required transfer of funds, if the blockchain is associated with coins. Smart contracts provide an attractive solution to achieve distributed and trustworthy operations for smart home networks. The smart contract is normally deployed in bytecode and uses Application Binary Interface (ABI) to specify the specifications of the functions in the contract. A smart contract includes data and functions that can be called by the EdgeHubs in the form of transactions or other smart contracts in the form of messages. A transaction is a package of data that is signed by an account and is aimed at another account or to execute the ABIs of a contract. A message is similar to a transaction, but is sent by a contract with the aim of running the ABIs of another contract.

Attribute based access control policy (ABAC). Controlling access to resources by authorized users will be provided by access control systems. Access control systems must provide expressibilty for the policy designer and security and efficiency for enforcement. There have been many access control proposals including capability-based and Role-Based Access Control (RBAC) [9] to provide the required efficiency and security. An access control system of growing significance and adoption, is ABAC [11] which assigns attributes to subjects, objects, and environment. Attributes are in the form of name-value pairs. Attributes are used in defining policies which are written as boolean expressions and specify the attributes of the subjects who can access an object with a given attribute set under the stated environmental conditions. In our setting, ABAC effectively decouples the owners (publishers) of the resources from the requesters (subscribers) of the resource, and let the owner to choose the rules of access, and by using environment ABAC will allow context-awareness. ABAC allows designing a fine-grained access control policies that well match the dynamicity and flexibility of the IoT environment.

3 System architecture and assumptions

We consider a smart home equipped with an EdgeHub that provides Internet connectivity and edge computing services to the home, through ISPs that are the C-nodes in the N-Chain (Figure 1). We consider neighborhood to be of a group of homes that are in the same geographic proximity, possibly a multi-story building with residential units. Communication of a user to the N-Chain is through EdgeHub and is in the form of a signed transaction that will be sent to the ISP that the EdgeHub is subscribed to, and so can verify the user's identity.

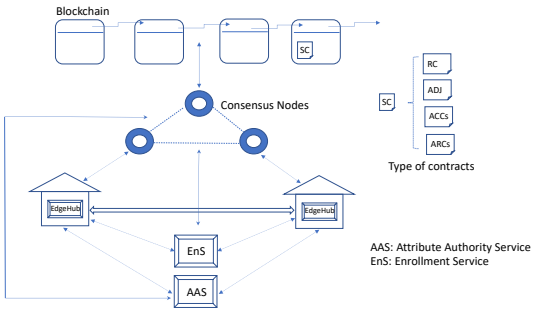

Fig. 1. Different entities in the proposed infrastructure

There are also (i) an Enrollment service (EnS) that enrolls the users in the system, and provides a certified public key and the corresponding secret key to their EdgeHub, and (ii) an Attribute Authority Service (AAS) which issues certificates for the attributes of resources and users after performing verification.

An user attribute information is published as a smart contract (called Attribute Repository Contract *ARC*) on the blockchain under their public key. Each EdgeHub also publishes a second smart contract called Access Control Contract (ACC) that includes conditions of access for each resource. There are also two smart contracts which are deployed by the N-chain administrators and updated by the EdgeHubs by sending message to N-Chain: Register Contract (RC)and Adjudicator Contract (ADJ). RC includes all sharable items, the address of the corresponding ACCs, the public key of the resource owner, and the ABIs that are related to each item in ACC. ADJ records the public key of the malicious requester (subject), the misbehavior information, and the penalty which is considered. The format of each contract will be describe in details in Section 4.

EdgeHubs use the N-Chain to advertise their sharable content and their associated access conditions. An EdgeHub A sends a transaction to the blockchain on behalf of the home-owner (user), requesting a a resource from a home B. If the request satisfies the access policy of the resource, as specified by B, the access is granted, and the EdgeHubs A and B establish a direct communication channel to complete the request. A request will be signed by the EdgeHub and will include the claimed attributes of the owner that will be used in the evaluation of the request by the access control system that is implemented by the blockchain.

3.1 Security Goals and Trust Assumptions

There are four types of entities. (i) EdgeHubs, (ii) C-Nodes, (iii) Trusted authorities and services, and (iv) outsiders.

Goals: The security and privacy goals of the system at a high level are (i) access to resources will be provided only to requests that satisfy the access policy of the resource, and (ii) transactions do not leak more information compared to what is publicly available on the blockchain.

We note that one of the important properties of the blockchain is the ability to provide a trusted platform for sharing, and so by nature the information that are on the blockchain

will be voluntarily shared. A successful request should enable the requester and the resource owner to establish direct communication link. We require that the information state of entities do not change after a transaction. In particular,
- C-Nodes will not learn the identity of EdgeHub owners that are not their subscribers;
- The two sides of a transaction only learn the necessary information to complete the transaction. In particular if the transaction fails, none of the parties learn any new information, and if the transaction is successful, they only learn the required link or access information for the requested resource.

As our analysis in Section 4 shows, our design provides security, but in its basic form and without using cryptographic mechanisms cannot provide privacy. In Section 4.3 we outline the use of attribute based encryption (ABE) to provide privacy.

Trust assumptions. We assume EdgeHubs are implemented as tamper-evident boxes that will ensure that the installed software will be run as specified. However the owner will not be trusted in their claims: they may offer resources, or claim quality or attributes for their resources, or themselves that they do not have. They are also interested in learning details of other users in the system through their interaction with the N-Chain and the users when the transaction is not successful. The communication between each Edge-hub and the C-nodes is over TLS.

The C-Nodes and other authorities are honest but curious: while they participate in the protocols correctly and function as specified, they will attempt to learn extra information about participants and transactions including their real identities.

4 Securing access using N-Chain

Let E_s and E_r denote two EdgeHubs: E_s (sending request) requests a resource from E_r (receiving request). In the following we use $S(C) \rightarrow R(C) : m$ to show S sends message m to R, and use C if the sender/receiver is a smart contract. We outline the flow of EdgeHub's transactions as follows (Figure 2 shows the flow of the protocol in detail):

1. Registering with EnS: Each Edge-hub subscribe to one of the ISPs who is also one of the C-Nodes, and provides its identity. Then, it receives a public key and an authentication certificate.
2. Registering with AAS: Each edge-hub contacts the attribute authorities and receives certificate for attributes that it holds. For example for the object, movie, the attributes can be the title of the movie, the description (type of movie, appropriate age-range, ..), duration of the movie. For subjects, the location and the age-range of the user can be considered as related attributes. These attributes can be validated through different authorities, and hence different certificates will be assigned to each of them.

Each certificate has an expiration time which is verified by the N-chain when an access request is executed. If the certificate is not valid anymore, the access will be denied. We consider four types of smart contracts and all of them have a deleteContract() ABI that performs selfdestruct operation which can only be called by the owner of the contract.

(1) ARC, attribute repository is a contract which holds the attribute of objects and subjects and it is deployed by the Edge-hub. Attributes are in the form of key-value pairs

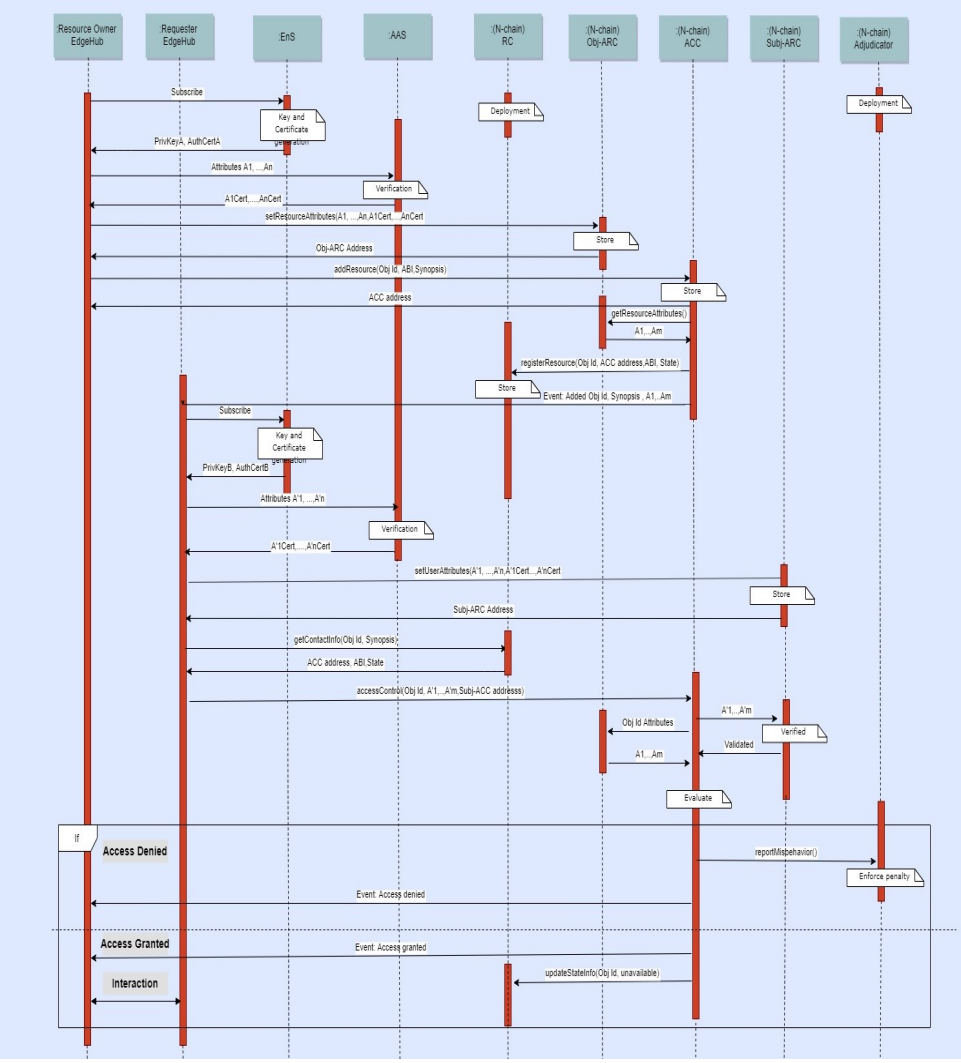

Fig. 2. Access control flow- (1) Registering the resource owner, (2) certifying the resource attributes, (3) deploying the access control contract, (4) registering a resource, (5) registering the requester, (6) certifying requester attributes, (7) access request (8), access evaluation.

and they are certified by relevant trusted attribute authorities (AAS) who validate the attributes. Each Edge-hub is responsible to get the required certificates for its resource attribute (or user attributes) and store the set of attributes and certificates in $Obj-ARC$ (or $Subj-ARC$) (please see Table 4) . A collection of attributes may have only one certificate. We require that each requester only sees the information that is needed for issuing an access, therefore we store the attributes of the subjects in separate contracts. The attributes of a resource will be stored in the Obj-ARC, while the Edge-hub can acts as both subject and resource owner in the system.

Fig. 3. $Obj-ARC$ look up table

Obj Id	$Attr_1$	...	$Attr_n$	$Cert_1$	...	$Cert_n$
M_i	*title*:Tom and Jerry	...	*Description*: Cartoon	0xabc....	...	0xa89....

ARC has two methods, depending it is an Obj-ARC or Subj-ARC. The methods can have same functionalities;
-*setUserAttributes()*: is used in Subj-ARC, and setResourceAttributes() is used in Obj-ARC to recored the attributes and certificates of users and resources. -*getUserAttributes() and getResourceAttributes()*: are used to retrieve the attributes and certificates of an user or a resource when ACC is evaluated.
-*deleteUserAttributes() (or deleteResourceAttributes()):* is used when the resource is not sharable, or the user is not in the possession of an attribute anymore.

(2) ACC stores the attribute based access control policies, and required function to evaluate the policies for a request. This contract is deployed by resource owner E_r, and multiple of them can exist at the same time. All the contracts related to one E_r have the same owner. An example of a access policy can be like this: grant access to anyone who pays p coins and has the attribute of being adult, having age greater than 18 (which is retrieved from resource attribute repository).

ACC can include multiple resources and their access policies. This smart contract is deployed on the blockchain and its address is stored by E_r. ACC has the following ABIs:
- *setResourceARC():* is used to register the address of the $Obj-ARC$ in the contract which is used to retrieve the attributes of a resource.
-*setAdj():* is used to set the address of the ADJ.
- *addResource():* is used to add a resource identified by a reference, $ObjId$, to ACC (each access control can support multiple resources) and also register the resource in RC. ACC first retrieves the information about $ObjId$ from $Obj-ARC$ and sends a message to the RC to register the resource in RC (by using *registerResource()* method). The state of the resource is set to available at the beginning, and can be changed by E_r or after a successful granted access. Additionally, this method emits an event to inform all listening EdgeHubs about the sharable resources that exist in this contract. All the EdgeHubs who listen to the blockchain network will store these information in their local storage. - *accessControl_i():* is called by requester E_s to evaluate the resource access policy, and takes the resource identifier $Obj\,Id$ as input. The output of this ABI is either access granted or access denied. When access is granted, the state of the resource in RC is changed by ACC (through calling

(3) RC acts as directory for all other smart contracts and will be deployed by the N-Chain providers when initiating the system. The RC contract keeps record of the resource identifiers $Obj\,Id$, the corresponding ACC address, the required ABI, the public key of the resource owner who can modify, or delete that resource from the table table, and state information (See Table 4 for details). State field reflects the state of the resource, for example state="unavailable" means that resource is temporarily unavailable maybe because other users are using it. State information can be updated by ACC when it grants access, or by the resource owner. RC has the following ABIs:

Fig. 4. RC look up table

Obj Id	ACC address	Pk_r	ABI	state
M_i	0xf2453jddkkd..	0x456788433...	accessControl_1(M_i,..), Inputs:[]	unavailable

- *registerResource():* is used to register the resource information in RC table.
- *updateStateInfo():* is used by ACC or resource owner E_r to set resource state.
- *getContractInfo():* is used by E_s to retrieve the information (ACC address, ABI, and state) of an object identified by $Obj\,Id$.

(4) ADJ handles the complaints and provides dispute resolution rules. For example it details and executes the necessary steps if the quality of a resource does not match the advertised content. It stores the information about the resource, the misbehaving party, and the description about the misbehavior, the time, and the penalty in a table (please see Table 4). ADJ has ABI, *reportMisbehaviour()*, that is called by ACC or victims to set the information about a misbehavior.

Fig. 5. Adjudicator look up table

Obj Id	Pk_s	Misbehavior	Time	Penalty
M_i	0xf2453jddkkd..	Frequent access	7:43 12-10-2019	slash P coins

4.1 Requesting an access

To get access to a specific resource, Edge-hub E_s, first searches in its local database and finds the synopsis and the identifier of the desired resource. Then, E_s sends a request to RC and gets either an empty result which shows the object has been deleted (or not registered), or the address of the ACC contract which specifies the access policies plus the ABI of that contract. In the latter case, the E_s sends a request to the ACC together with its attributes, also provides the address of $Subj-AR$ for validating the attributes it is claiming, and a temporary public key Pk_{temp} generated by itself. ACC evaluates the attributes against policies and issues an event which shows the result of access rule evaluation.
$E_s \rightarrow RC : Obj\ Id, Synopsis$
$RC \rightarrow E_s : ACC \quad address, ABI, state$
$E_s \rightarrow ACC : accessControl_i(obj\ Id, \cdot), A'_1, .., A'_m, ARC\ address$
$Event\{\text{“}AccessResult\text{”} : ...\}$

Connecting to resource owner: If access is granted, the E_r encrypts its IP address (or any link to the resource with the public key of E_s and sends it to the $N-Chain$. E_s decrypts the message and retrieves the resource. $E_r \rightarrow E_s : Enc_{Pk_{temp}}(IP)$

4.2 Security analysis

Correctness. We want to ensure that access to resources will be provided only to requests that satisfy the access policy of the resource written in ACC. This implies that: (i) outsiders should not be able to send a request to access the resources, (ii) The requesters who are cheating should be detected, and (ii) the requester who has the required attributes end up with access granted.

For (i), we argue that each access request is executed by the C-Node who has registered the requester at the enrollment stage. As C-Nodes are trusted entities they will reject any request which has been originated from outsiders. Additionally, outsiders cannot deploy ACC contracts and share resources with EdgeHubs in a neighbourhood. Such attempts

will be immediately hindered by C-Nodes. For (ii), recall that the real identity of each EdgeHub is known by the C-Nodes that EdgeHub has subscribed to. So, if a cheating is detected, the corresponding EdgeHub can be tracked and sued by judge. ADJ records the misbehaviors upon access request, and can impose penalty on misbehaving entities, which these will discourage misbehaviors. To detect the misbehaviors, ACC keeps track of requests, and checks the request history for each new request it receives. Based on the work flow of protocol, all the attributes of the resource and their certificates are verified before they are added to $Obj-ARC$ contracts. However, it is possible that a certificate is expired when the access is requested. At that time, the transaction fails and the resource owner will be reported as misbehaving party to ADJ. For (iii) we rely on trustworthiness of the C-Nodes, in executing the access control policies in ACC.

Privacy. We want to ensure that transactions do not leak more information compared to what is publicly available on the blockchain. It worths to first mention that outsiders do not get any information about the shared resources and the requests. The reason is that we assume the communication from each EdgeHub to N-Chain is over TLS. What's more, any function call from outsider to contracts inside a neighborhood will be hindered by C-nodes. Thus, our concern is to show that:

- C-Nodes will not learn the identity of EdgeHub owners that are not their subscribers; In our system, all interactions to N-Chain is signed by a public key which has been assigned to EdgeHubs by EnS. This public key can only be linked to real identity of the EdgeHubs by the C-Nodes which the user has subscribed to. All other C-Nodes can only see the linkage between contracts and the public keys that have no information regarding the real identity of EdgeHubs or owners of them. This happens because, each resource owner deploys three contracts ACC, $Subj-ARC$, $Obj-ARC$, using its registered public key as owner. Moreover, the information of the owner of resources is also registered in the RC. If a resource owner misbehaves, its public key will be recorded in ADJ. Thus, all the contracts related to an EdgeHub are linked with each other through the public key of the owner of the EdgeHub. Furthermore, when an access is granted, the IP of the owner is encrypted using a temporary key which has been determined by the requester which prevents the counter-party C-Nodes to observe it.

- The two sides of a transaction only learn the necessary information to complete the transaction: It is not satisfied in our system. Note that N-Chain is permissioned which is run by C-Nodes who are not members of the neighborhood. Therefore, the view of EdgeHubs are limited to the output of the functions that they are allowed to call.

When a requester sends access request to ACC, its access history which includes its public key will be recorded in ACC, and it is observable by resource owner (by using $getRequestHistory()$). In $Subj-ARC$, there is a method $getUserAttributes()$ which can be used by any ACC contract to retrieve the attributes and the certificates for further evaluation. Thus, $getUserAttributes()$ is observable by all entities (it is defined as a public function). This reveals all the attribute of the requester to the resource owner who knows the public key of the requester. Note that the reverse is not true; the requester cannot see the public key of the resource owner (when $RC.getContractInfo()$ is called by the requester, the public key of the owner will not be returned). Note that ensuring

privacy of a resource is not a priority for a (volunteered) sharing system. However, it is essential to prevent the resource owner from learning all the attributes of requester recorded in $Subj-ARC$, even if they are not necessary for granting an access.

4.3 Towards a privacy preserving N-CHain

The above analysis shows that the resource holder can learn the attributes of a requester. To prevent the attribute values to be leaked to resource holder one can use ciphertext policy based attribute based encryption (CP-ABE). In CP-ABE, a ciphertext is attached with a policy that determines those who can decrypt it. The access rule of a resource will encrypt an access token using a CP-ABE, and with a policy that matches the resource holder access policy for the resource. A requester with the matching access can decrypt the token and present it to the resource holder for access. The system will require an attribute key authority (AKA) that will issue attribute keys for the attributes, after verifying them.

5 Proof of concept implementation

In this section we consider the usecase of sharing a movie in a smart neighbourhood to show the feasibility of the proposed infrastructure.

Truffle & Ganache. Truffle provides a development environment for building, testing, and deploying Ethereum smart contract and decentralized applications (dApp) [22]. Ganache is a personal blockchain that allows developers to create smart contracts, dApps, and test software [23]. The following smart contracts: RC, ACC, ARCs and ADJ are developed and tested on Truffle and Ganache.

Experiment. We simulate a neighbourhood of 5 smart homes each equipped with an EdgeHub. Each EdgeHub obtains its private key and secret key from the enrollment service and uses them for authentication while transacting with the blockchain. Let us consider that one of the EdgeHubs E_R would like to share a movie with the other EdgeHubs in the neighbourhood. For this, E_R provides the movie attributes (as $\{key,value\}$ pairs) such as movie name, type of the movie (for e.g., adult or kids), size of the movie, etc., to attribute authority AAS that verifies and provides a certificate that certifies these attributes. E_R then stores these attributes including their certificate on the blockchain using the *ResourceARC* smart contract and gets its address. We assume that each EdgeHub has already stored their attributes using their ARC.

Then the EdgeHub E_R provides the movie (resource) information using *addResource*() function to the ACC. The function takes input parameters: resourceType (e.g., Movies), movieName (e.g., Cars) and E_RARC address. The ACC also provides another important *accessControl*() function that contains the access request evaluation procedure. It uses the information provided by *addResource*() and retrieve the attribute values of both E_R and E_S from E_RARC and E_SARC for access request evaluation purpose. This function is explained in much details later in the section. After publishing the ACC on the blockchain, E_R assigns an unique identifier to its movie and calls the function *registerResource*() to register the movie to RC passing the identifier, the ACC address, ACC ABI and E_R's public key. Fig. 6 shows successful registration of a movie in RC.

Fig. 6. Registering the resource and there corresponding ACC

Fig. 7. Request for access control

After successful execution of *registerResource*(), the ACC generates an event including the movie index, ACC address, ABI, E_R public key and synopsis of the movie (more information about the movie that E_R wants to advertise) to inform all the EdgeHubs in the neighbourhood. The event information is stored in a local database at each EdgeHub in the neighbourhood.

EdgeHub E_S that is interested in the movie, sends a request to RC to get more information like the ABI that it requires to request for the movie access. Using this information, E_S sends its access request calling accessControl() with the parameters values movieName and the E_S ARC address. The function is executed to make a decision based on the attributes of the movie $ResourceARC$, E_RARC and E_SARC. Fig. 7 illustrates the successful execution of the accessControl() for the E_S access request.

The ACC also provides a function that identifies any misbehaviour by the E_S. For instance, in this usecase, we identify it as a misbehaviour if an E_R sends more than 1 access request in 100 ms time similar to the one discussed in [26]. When the number of requests are exceeded the limit, a report including the time of the last request and time of the current access request from the E_R is provided to ADJ. On verifying this information, if the misbehaviour is true, ADJ denies the access request and block further requests form the E_R. If misbehaviour is not detected, the ACC grants the access and informs both the E_R and E_S. Then the interaction between the EdgeHubs is off-chain and share the movie using a secure communication link.

6 Conclusion

We proposed a neighborhood sharing platform based on a premissioned blockchain, which provides levels of trust and convenience that are expected from a sharing service. Users employ attribute-based access control to specify the conditions and terms of access for their items, and the access control is enforced by the blockchain. We gave details of the system, assumptions, and discussed the security of the system. We also showed the feasibility of our scheme by proof of concept implementation. According to our analysis, ensuring the privacy of requesters needs further study; we suggested a possible solution using attribute-based encryption which can initiate an interesting future direction.

References

1. Airbnb: Book homes, hotels, and more on Airbnb (2019), https://www.airbnb.ca/
2. Amazon: AWS IoT Framework (2018), https://aws.amazon.com/iot
3. Androulaki, E., Barger, A., Bortnikov, V., Cachin, C., Christidis, K., De Caro, A., Enyeart, D., Ferris, C., Laventman, G., Manevich, Y., et al.: Hyperledger fabric: a distributed operating system for permissioned blockchains. In: Proceedings of the Thirteenth EuroSys Conference. p. 30. ACM (2018)
4. Apple: The smart home just got smarter (2018), http://www.apple.com/ios/home/
5. Avizheh, S., Doan, T.T., Liu, X., Safavi-Naini, R.: A secure event logging system for smart homes. In: Proceedings of the 2017 Workshop on Internet of Things Security and Privacy. pp. 37–42. ACM (2017)
6. Buterin, V., et al.: A next-generation smart contract and decentralized application platform. white paper **3**, 37 (2014)
7. Doan, T.T., Safavi-Naini, R., Li, S., Avizheh, S., Fong, P.W., et al.: Towards a resilient smart home. In: Proceedings of the 2018 Workshop on IoT Security and Privacy. pp. 15–21. ACM (2018)
8. Dupuy, J.C., Torre, A.: Cooperation and trust in spatially clustered firms. Trust and Economic Learning, Edward Elgar, Cheltenham pp. 141–61 (1998)
9. Ferraiolo, D., Kuhn, D.R., Chandramouli, R.: Role-based access control. Artech House (2003)
10. Higginbottom, J.: Governments finally embrace the sharing economy (2018), https://www.ozy.com/fast-forward/governments-finally-embrace-the-sharing-economy/89688/
11. Hu, V.C., Ferraiolo, D., Kuhn, R., Friedman, A.R., Lang, A.J., Cogdell, M.M., Schnitzer, A., Sandlin, K., Miller, R., Scarfone, K., et al.: Guide to attribute based access control (abac) definition and considerations (draft). NIST special publication **800**(162) (2013)
12. IBM: IBM Watson IoT Platform (2018), https://internetofthings.ibmcloud.com/#/
13. Johnson, C.: The Rise of the Sharing Communities (2012), https://www.shareable.net/the-rise-of-the-sharing-communities/
14. Lazaric, N., Lorenz, E.: Trust and economic learning. Edward Elgar Publishing (1998)
15. Maesa, D.D.F., Mori, P., Ricci, L.: Blockchain based access control services. 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). pp. 1379–1386. IEEE (2018)
16. Microsoft: Microsoft azure suite (2019), https://azure.microsoft.com/
17. Nilsson, M.: Proximity and the trust formation process. European Planning Studies **27**(5), 841–861 (2019)
18. Ouaddah, A., Abou Elkalam, A., Ait Ouahman, A.: Fairaccess: a new blockchain-based access control framework for the internet of things. Security and Communication Networks **9**(18), 5943–5964 (2016)
19. Rinne, A.: 4 big trends for the sharing economy in 2019 (2019), https://www.weforum.org/agenda/2019/01/sharing-economy/
20. SmartThings, S.: How it works-smartThings (2018), https://www.smartthings.com/getting-started
21. Storper, M., Venables, A.J.: Buzz: face-to-face contact and the urban economy. Journal of economic geography **4**(4), 351–370 (2004)
22. truffle: truffle suite (2019), https://www.trufflesuite.com/truffle
23. truffle: truffle suite (2019), https://www.trufflesuite.com/ganache
24. Uber: Drive with Uber, Earn on your schedule. (2019), https://www.uber.com/ca/en/
25. Williams, R.: Forrester: Millennials boost growth of sharing economy (2018), https://www.mobilemarketer.com/news/forrester-millennials-boost-growth-of-sharing-economy/515851/
26. Zhang, Y., Kasahara, S., Shen, Y., Jiang, X., Wan, J.: Smart contract-based access control for the internet of things. IEEE Internet of Things Journal **6**(2), 1594–1605 (2018)