Received: May 20, 2021. Revised: July 28, 2021.

International Journal of
Intelligent Engineering & Systems

http://www.inass.org/

SocioChain: A Distributed Ledger-based Social Framework for the Internet of Things

Ali Sohofi[1]* **Aliasghar Amidian[2]** **Yaghoub Farjami[1]**

[1] *Department of Computer Engineering and Information Technology, University of Qom, Qom, Iran*
[2] *Faculty of Post and Telecommunications (Ministry of ICT),*
University of Applied Science and Technology, Tehran, Iran
* Corresponding author's Email: a.sohofi@stu.qom.ac.ir

Abstract: The core vision of the Internet of Things (IoT) is to connect objects into a broad network and establish interactions between them to enhance society and human life through the society of objects. As this network expands and the number of connected objects increases continuously, more efficient, secure, and scalable architectures will be needed. Social IoT (SIoT) is a promising IoT architecture, which is based on establishing social relationships between objects. However, being centralized is one of its drawbacks, which leads to challenges like scalability, latency, and privacy issues. Distributed ledger technologies, especially blockchain, are distributed architectures that have received attention in recent years. This paper proposes a fully decentralized SIoT-based architecture for the IoT, exploiting a two-layer distributed ledger structure. This architecture is privacy-preserving and scalable; besides, it provides social relationships between objects. These social relationships are stored as transactions on the ledger. This study also calculates the required space to store ledger data locally. The number of established relationships between the objects is estimated by simulation. The results indicate that less than 10% of all possible relationships are based after 1000 days. Hence, it is possible to store ledger data locally. This approach leads to ten times less latency comparing to previous SIoT-based architectures for 10000 devices.

Keywords: Internet of things, Blockchain, Distributed ledger technology, Smart contract, Distributed architecture.

1. Introduction

Since Kevin Ashton introduced the Internet of Things concept in 1999, it has been developed dramatically. IoT is currently being used in various fields and is expected to play a more critical role in all aspects of human life [1]. It is predicted that more than 41 billion devices will be connected to and interact with each other through the IoT by 2025 [2]. Hence, an efficient IoT solution requires specific capabilities, including full interoperability among heterogeneous objects [3] and efficient service composition and discovery [4, 5]. An IoT architecture that has promised such capabilities is SIoT [6, 7]. In this architecture, a set of social relationships has been defined between objects. Each object interacts with other objects based on these relationships. Improving network navigability, enhancing scalability, establishing a level of trustworthiness, and the possibility of using social network analysis models in investigating IoT issues, are among the advantages of converging the concepts of social networks with IoT [8].

In contrast, however, the SIoT is a centralized architecture. Centralized architectures, which are based on the central server and cloud computing, utilizing effectively in some cases. Nevertheless, challenges such as flexibility, scalability, latency, privacy, presenting the central server as a single point of failure, and the possibility of information loss are some of the drawbacks of such architectures [9–11]. Distributed architectures have been introduced as the key to solving some fundamental challenges of IoT [12].

With the emergence of digital currencies, especially Bitcoin [13], the concepts of blockchain and distributed ledger technology (DLT) have attracted much attention in recent years. DLT is a secure, distributed, and tamper-resistant structure that can also be privacy-preserving [14, 15]. DLT is widely used in various domains, such as energy management, supply chain management, and healthcare [16]. Moreover, thanks to its superior characteristics, DLT can be considered an appropriate solution to dealing with IoT challenges, particularly the challenges of centralized architecture [17]. The smart contract is a concept that can be implemented using DLT. Smart contracts are scripts that reside on the distributed ledger and can apply rules and policies in a distributed, automatic, and inevitable manner [18]. This paper has employed DLT and smart contracts to implement the relationships between objects in SIoT to introduce a decentralized architecture for IoT.

The major contributions of this paper are as follows:

- We proposed a fully decentralized architecture based on social relationships by removing the central server (i.e., single point of failure) from the SIoT architecture and exploiting a two-level distributed ledger structure. This strategy improves the scalability, performance, reliability, and security of SocioChain. User privacy has also been taken into account in the new architecture.
- We implemented objects relationships and SIoT processes using DLT transactions and smart contracts. It allows heterogeneous objects to interact and interconnect autonomously based on user-defined regulations. Also, it makes transactions secure and tamper-proof. Furthermore, in the proposed architecture, DLT has been utilized simply as a module to store relationships between objects; thus, SocioChain is independent of the type of utilized DLT.
- We explored the possibility of storing ledger data locally by analyzing the SWIM mobility simulator's results. Accordingly, we compare the performance of the proposed architecture with previous SIoT-based implementations.

The rest of the paper is organized as follows. Section 2 reviews the related works. In section 3, the SocioChain architecture is introduced, and then objects' social relationships, storing transactions on the ledger and the resulting ledger's size are discussed. Section 4 is dedicated to the proposed architectural simulation and analysis of the number of relationships between objects. Finally, the concluding remarks are described in section 5.

2. Related work

The concept of the SIoT is thoroughly discussed in [7]. This architecture includes a network of smart objects with relationships similar to human societies. Objects in SIoT can exert automatic social relationships with each other based on the owner's rules. The main thrust areas of the SIoT include service discovery, network navigability, relationship management, and trustworthiness management [8]. Besides, scalability and reliability are improved using this architecture [19]. In addition to SIoT, other studies have been conducted to embed social network concepts into the IoT [20].

Several studies have been conducted in recent years to address the SIoT limitations by reducing the latency and increasing the scalability and flexibility. Cicirelli et al. [21] introduced a Java-based platform called iSapiens using the SIoT and edge-computing concepts for smart environments. SIoT was used in this platform to improve scalability and interoperability. An intra-network interface layer was introduced in the iSapiens platform using the edge computing paradigm to provide a smart environment to access the processing resources and local storage with the least latency and bandwidth usage. Although edge computing has been well applied in this architecture, the central server of SIoT is still a main part of the proposed scenario. Additionally, security and privacy are not taken into account in architectural design.

Lysis [22] is another SIoT-based architecture that proposed a four-layer model using a multi-layered structure from the iCore project [12]. In Lysis, objects in the real-world layer are converted to several social virtual objects in the virtual object layer using the virtualization layer. They can have social relationships with each other. This architecture uses the platform as a service (PaaS) model and deploys applications in the cloud environment. Even though the proposed framework is valuable, establishing new relationships based on existing ones is not supported in this system [23]. Also, the deployment of social virtual objects on the cloud can lead to centralized architecture challenges.

A user-centered access control mechanism to manage access control policies of the IoT is introduced using social relationships between owners and objects [24]. The architecture, called FOCUS, is constructed on a blockchain that used an identity management system to ensure security and privacy. In this architecture, the blockchain is used to record

Received: May 20, 2021. Revised: July 28, 2021.

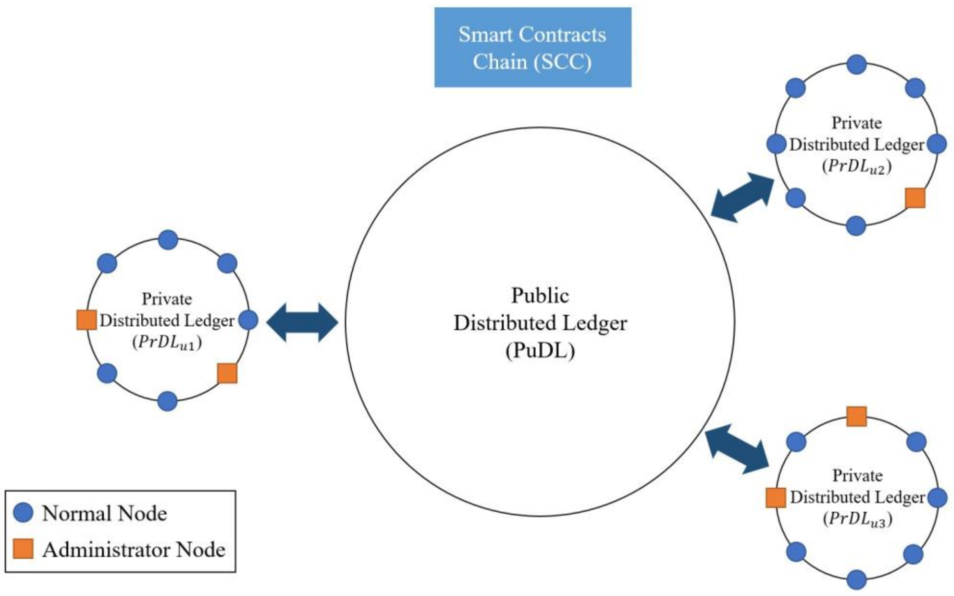

Smart Contracts Chain (SCC)
Private Distributed Ledger ($PrDL_{u2}$)
Private Distributed Ledger ($PrDL_{u1}$)
Public Distributed Ledger (PuDL)
Private Distributed Ledger ($PrDL_{u3}$)
Normal Node
Administrator Node

Figure. 1 An overview of sociochain

owner's identities and store them on peers in a trustless peer-to-peer network. The authors in [25] proposed a dynamic privacy-preserving and lightweight key agreement protocol for Vehicle-to-Grid in the SIoT to address security vulnerabilities. These studies focused on access control, security, and privacy in SIoT, yet they have not proposed a solution to the other challenges it faced.

Although several distributed architectures have been proposed in the literature for IoT [26], new architectures are introduced in recent years with the advent of DLT, especially blockchain. In [27], blockchain is integrated with the Internet of Vehicles (IoV) to provide big and secure data storage. The authors designed a multi-blockchain architecture in which each blockchain stores part of the data. Novo [28] provided a distributed access control system based on blockchain. In this architecture, encrypted data are stored on the blockchain. Accessing IoT data is managed by a token-based method and established policies in a smart contract. BeeKeeper [15] is an IoT system based on blockchain. In this system, the server processes user data and generates responses by performing homomorphic calculations on the information, but the server cannot get any data related to user privacy from the handled information. While these architectures have proposed specific solutions to solve IoT challenges by utilizing DLT, to the best of our knowledge, neither of them has highlighted the establishment of social relationships between objects using DLT.

3. Sociochain architecture

In this section, we discuss SocioChain in detail. The proposed architecture structure is first introduced, then the relationships and interactions between the objects are brought. The transactions' structure is then presented, and finally, the size of the ledger is analyzed.

3.1 An overview of the proposed architecture

Two distributed ledgers, including a private distributed ledger ($PrDL$) and a public distributed ledger ($PuDL$), were used in the SocioChain to decentralize the SIoT architecture. All objects are on the $PuDL$ and have access to its data. $PuDL$ is used to access public services and data. On the other hand, there is a $PrDL_u$ for each user, u, established and managed by the user. Only devices connected to $PrDL_u$ by the user have access to its data. In other words, each object O owned by user u is connected to two distributed ledgers; $O \in PuDL$ and $O \in PrDL_u$, and can receive and transmit data from/to both ledgers. It should be noted that DLT is only used to store transactions and execute smart contracts. Accordingly, the implementation of $PuDL$ and $PrDL_u$ is independent of DLT. The only criterion that needs to be met is to support the execution of smart contracts. Furthermore, the address and description of each service's smart contract are accessible in another distributed ledger called the smart contract chain (SCC). All objects have access to SCC to explore and discover services provided by other objects on the $PuDL$. Fig. 1 shows an overview of the proposed architecture.

In the IoT, it is assumed that heterogeneous objects with different processing and storage capabilities are connected to the network. In the

Received: May 20, 2021. Revised: July 28, 2021.

SocioChain, three categories are defined for the objects.

- Category 1 includes objects that are mobile and have proper processing and communication capabilities. Smartphones, tablets, and smart cars are some examples of this category.
- Category 2 includes fixed objects that have proper capabilities for processing, storage, and communication. Examples include router modems, network hard drives, smart TVs, and RSUs.
- Category 3 includes objects with poor capabilities of either storage or processing. Sensors, actuators, and RFID tags fall into this category.

To deal with the challenge of object heterogeneity, we introduced administrators in the SocioChain architecture. Administrator objects have proper processing and communication capabilities, and therefore, categories 1 and 2 can be considered as administrator objects. The primary function of the administrator is to perform activities that require processing or remote communication. The following subsection will elaborate on the functionalities of the administrator's objects.

3.2 Relationships and interactions in sociochain

Defined relationships in the SIoT [7] are used to describe the relationships in SocioChain.

Parental object relationship (POR) is established between objects from the same manufacturer and with similar models. The manufacturer provides the initial setting and configuration of the objects so they can connect to the $PuDL$.

Co-location object relationship (CLOR) is established between objects with similar geographical locations. This relationship does not necessarily mean a cooperative relationship.

Co-work object relationship (CWOR) is established between objects that work together to perform a task or provide a service.

Ownership object relationship (OOR) is established between objects having the same owner. The owner of an object is determined in the configuration procedure by the user.

Social object relationship (SOR) is defined between the objects that their owners have social relationships in the real world.

In addition to these relationships, a new relationship called the *administration object relationship (AOR)* is considered in this study. This relationship is established between objects and their administrators. An object administrator can manage the object. It is not a one-to-one relationship, and one object might have several administrators. Besides, an object that is defined as an administrator might have an administrator, too.

To decentralizing the proposed architecture, all the relationships and interactions of the SIoT should be implemented using the distributed ledgers.

Parental relationship: Manufacturers provide the capability of connecting the objects to the $PuDL$. To this end, each manufacturer issues a pair of public and private keys and a digital certificate for each manufactured object. Manufacturers can create a certificate authority (CA) hierarchy to make all the issued keys compatible. The objects use the issued certificate to interact with the $PuDL$.

Ownership relationship: In addition to the certificate issued by the manufacturer, another certificate is issued by the owner (or the administrator) for the object during the initial configuration. Objects can interact with the $PrDL_u$ of the owner (i.e., user u) using this certificate. Private information of the owner's objects is located on this ledger. The objects can use this private information in various ways, including calculating trust factors, establishing new relationships, collaborating with other objects, and discovering new services.

Co-location and co-work relationships: Smart contracts are used to implement these relationships. The administrator prepares the contract terms (e.g., the relationship type, trust measurement criterion, and the relationship duration) and deploys the smart contract on the ledger. After that, no more interactions are required between the object and its administrator. After calling the smart contract by any other objects, the CLOR and CWOR automatically established between the objects that meet the contract terms.

Social relationship: Social relationships can be established using smart contracts. The terms of these contracts can be adjusted based on the number of visiting of another object, the interval between consecutive visits, the visit duration, or other conditions determined by the administrator. Like CLOR and CWOR contracts, these contracts are adjusted by the administrator and then are deployed on the $PuDL$. If the conditions are met, the SOR is established after executing a smart contract without the administrator's intervention.

Besides the processes that establish relationships between objects, three types of processes, namely, new object entrance, service composition, and service discovery, are conducted in SocioChain architecture.

Any new object should be added to $PuDL$ and its owner's $PrDL_u$. During device manufacturing, the manufacturer issues a digital certificate and a pair of

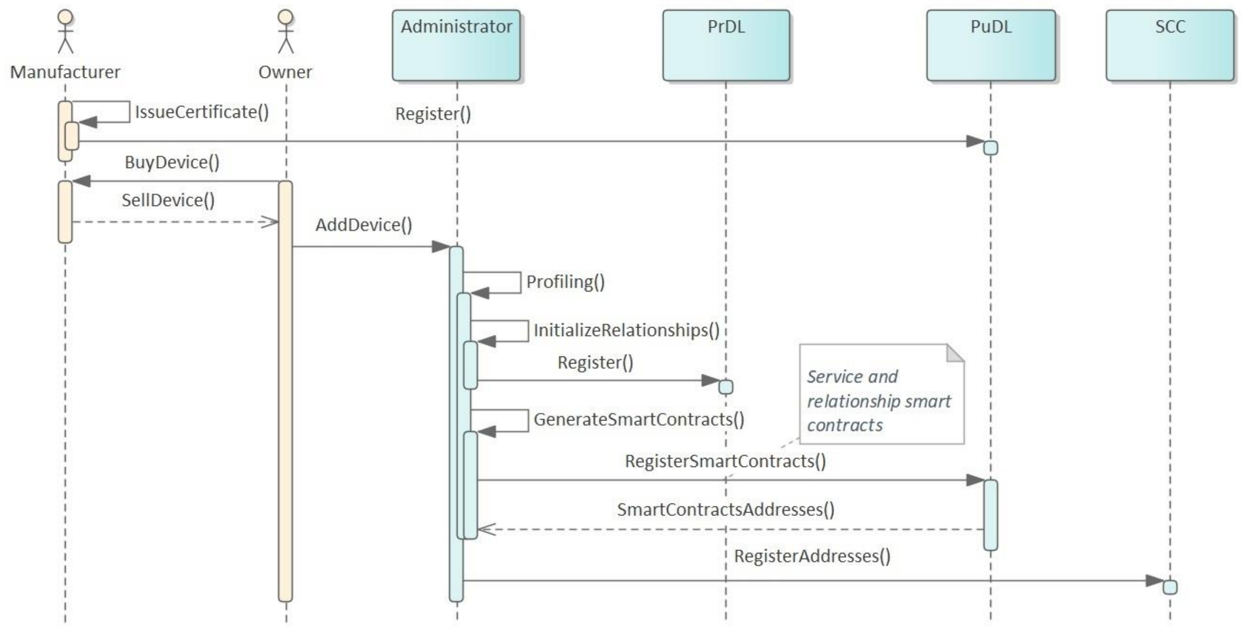


Figure. 2 Sequnce diagram of adding a new object

public and private keys for the object. So, the object can interact with the $PuDL$. To adding the object to $PrDL_u$ at the first step, the profiling will be conducted, where the type of the object, its capabilities (processing, storage, sensory, and functional), and the services offered by it are determined, and an ID along with a digital certificate is created for interacting by $PrDL_u$. The user does this process via an administrator. This process can be done automatically by the administrator based on the user's preferences without any intervention. The administrator can then inference and set the object's initial relationships using the data available on two ledgers and submit them on $PrDL_u$. Moreover, to reduce delay and speed up the service searching and ranking process, each object's relationships and friends are stored in a local database. The sequence diagram of adding a new object is depicted in Fig. 2.

To composite services, the administrator can deploy smart contracts on the $PuDL$. These smart contracts include services offered by the objects. The terms of these smart contracts are based on owner preferences. Also, the administrator submits smart contracts to SCC to make the services discoverable to other objects. The administrator may aggregate multiple services, extract a new service, and then publish it as a smart contract.

Services can be discovered in two distributed methods. The first method is searching on the SCC. After deploying each smart contract on the $PuDL$, the object's administrator publishes the description of the deployed smart contract on SCC. In this way, smart contracts will be accessible and explorable via SCC. Users or objects can search in SCC to find the service providers that can offer their desired service. The second method is searching in a P2P manner among friends. Search for the service is done by sending a direct request to all friends of the object. If any of its friends can fulfill the request, the service smart contract description is sent in the response; otherwise, the friend object will broadcast the request to its friends. This process continues until the desired service is founded. Both methods are fully distributed and independent of a central element. It effaces the single point of failure, and the service discovery process may continue if any nodes failed.

Service ranking is carried out after the service discovery process. It is a modular process in which each object can customize this process based on user preferences. Service ranking can be calculated based on various parameters considering different weights. These parameters can include the current relationships between objects and the score of previous interactions and utilized services. Furthermore, trust computation algorithms [29] can be employed to calculate the degree to which a service can be trusted. These data can be extracted from $PuDL$, $PrDL_u$, or the local database of the object. Each user can define an algorithm for extracting the services' scores to rank them for his own devices.

Finally, upon selecting the best service provider, the service requester calls the service's smart contract. Service provisioning is automatically conducted without a third party being involved based on the smart contract's conditions. If conditions are met, the

Received: May 20, 2021. Revised: July 28, 2021.

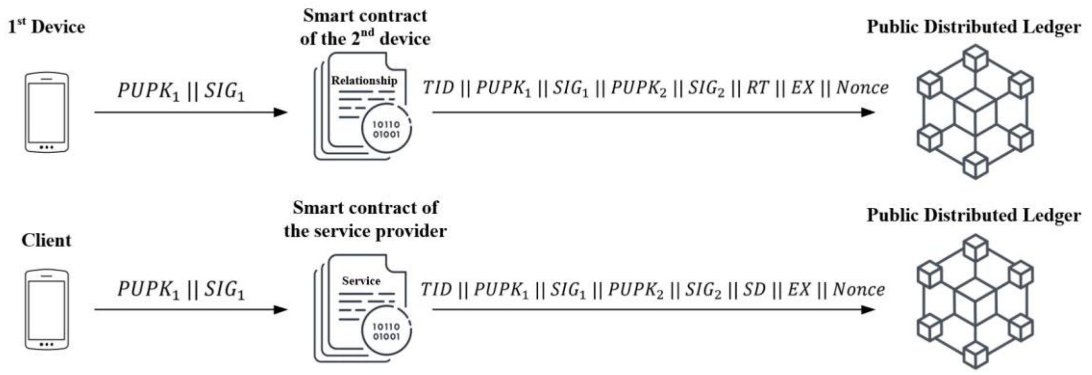

1st Device
Smart contract of the 2nd device
Relationship
10110 01001
Public Distributed Ledger
$PUPK_1 || SIG_1$
$TID || PUPK_1 || SIG_1 || PUPK_2 || SIG_2 || RT || EX || Nonce$
Client
Smart contract of the service provider
Service
$TID || PUPK_1 || SIG_1 || PUPK_2 || SIG_2 || SD || EX || Nonce$

Figure. 3 Flow of registering new relationship and requesting a service, on PuDL

smart contract is executed, and the service requester is provided with the service address and API access key. The permissions level of the access key is adjusted based on the conditions stipulated in the smart contract. After the service is received, the transaction's outcome is eventually stored on $PuDL$ and/or $PrDL_u$. Although it is possible to submit and share all the relationships between objects and service requests on $PuDL$, each of these transactions can be recorded only on $PrDL_u$ to protect privacy.

An essential capability of SocioChain is the use of smart contracts. Besides creating interoperability among objects, such contracts allow the automatic establishment of all relationships between the objects and service requests without user supervision or intervention. Consequently, every user can create predefined templates for smart contracts and deploy them automatically using an administrator.

3.3 Transaction structure of ledgers

As mentioned before, the architecture proposed in this study is independent of the utilized DLT. However, we use blockchain for more simplicity in describing the transaction structure in the SocioChain.

In SocioChain, three types of data are recorded on ledgers. The first type is entries that determine the parental relationship. Manufacturers add these entries on the $PuDL$. A block is added to the blockchain for a certain number of devices produced by the manufacturer. The manufacturer ID is included in the block's header, and the IDs of the manufactured objects in a specified time interval are inserted into the block's body. It is not needed to do this procedure at once by all manufacturers. They can adapt their devices to this architecture and add them to the $PuDL$ gradually.

The second type includes the entries that determine the relationships between objects. As mentioned in the previous subsection, after creating a CWOR, CLOR, or SOR between two objects, a transaction is recorded on $PuDL$ or $PrDL_u$. The transaction structure recorded in each ledger is different. The transactions on the $PuDL$ include the public key and the signature of both objects. In contrast, some relationships are only registered on the $PrDL_u$ for privacy. In this case, both objects' public keys are included in the transaction, but the object on the $PrDL_u$ only signs the transaction. Each of these processes is executed based on the smart contract terms, and the transactions are recorded on the corresponding ledger.

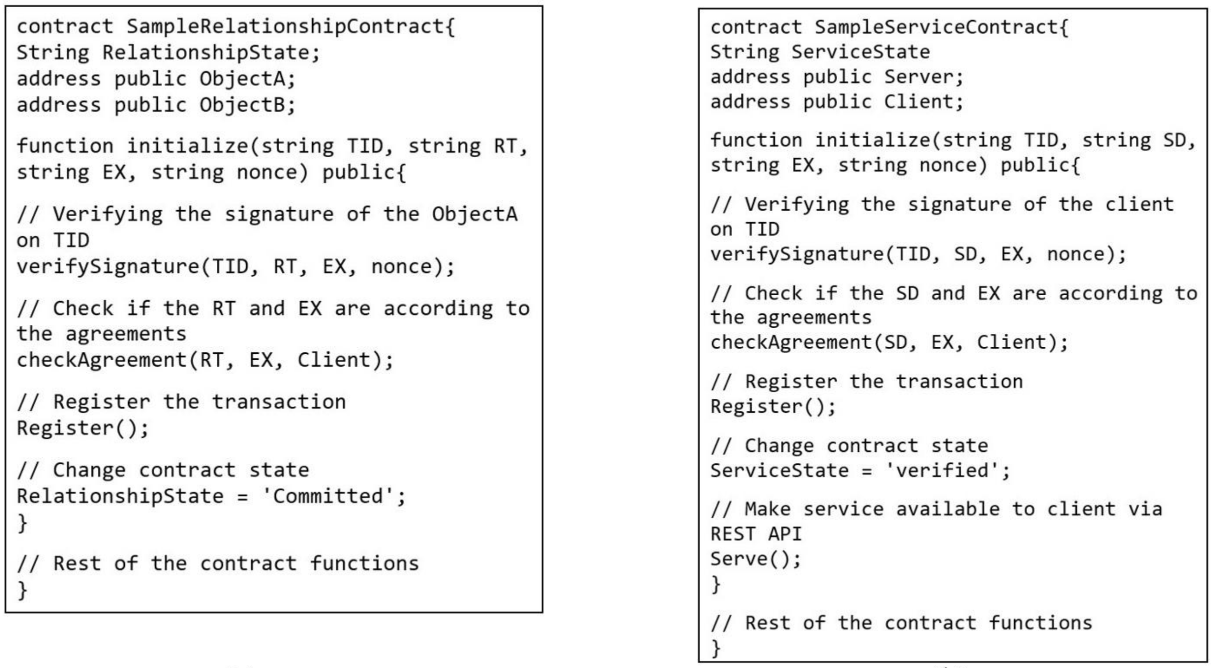
The structure of transactions on the $PuDL$ is as follows:

$$TID \,||\, PUPK_1 \,||\, SIG_1 \,||\, PUPK_2 \,||\, SIG_2 \\ ||\, RT \,||\, EX \,||\, Nonce$$

TID is the id of the transaction, which is the hash of the transaction content. $PUPK_1$ and $PUPK_2$ are the public keys of two objects for interacting on the $PuDL$. SIG_1 and SIG_2 represent the TID signed by first and second objects, respectively. RT indicates the relationship type (i.e., CWOR, CLOR, SOR). EX is used for extra information. It includes time boundaries or other constraints in a relationship. $Nonce$ follows this, which is a random number. An overview of the process is shown in Fig. 3.

In this case, either one of the two objects or their administrator calls the smart contract. The smart contract needs the public key and the signature of the first object as input. The first object calculates and signs the TID by hashing RT, EX, and $Nonce$ agreed by both parties. Therefore, the service provider cannot change these fields. The smart contract validates the signature and the signed fields, and then the transaction is submitted on the $PuDL$ with the specified structure. Fig. 4 (a) shows an example of such smart contracts.

On the other hand, the structure of the recorded transactions available on the $PrDL_u$ is as follows:



```
contract SampleRelationshipContract{
String RelationshipState;
address public ObjectA;
address public ObjectB;

function initialize(string TID, string RT,
string EX, string nonce) public{

// Verifying the signature of the ObjectA
on TID
verifySignature(TID, RT, EX, nonce);

// Check if the RT and EX are according to
the agreements
checkAgreement(RT, EX, Client);

// Register the transaction
Register();

// Change contract state
RelationshipState = 'Committed';
}

// Rest of the contract functions
}
```

(a)

```
contract SampleServiceContract{
String ServiceState
address public Server;
address public Client;

function initialize(string TID, string SD,
string EX, string nonce) public{

// Verifying the signature of the client
on TID
verifySignature(TID, SD, EX, nonce);

// Check if the SD and EX are according to
the agreements
checkAgreement(SD, EX, Client);

// Register the transaction
Register();

// Change contract state
ServiceState = 'verified';

// Make service available to client via
REST API
Serve();
}

// Rest of the contract functions
}
```

(b)

Figure.4 Sample smart contract for: (a) establishing relationship and (b) serve

$$TID \,||\, PRPK_1 \,||\, SIG \,||\, PUPK_2 \,||\, RT \,||\, EX \,||\, Nonce$$

This transaction is submitted on the $PrDL_u$ of each object by the objects themselves or their administrators. The TID field is the ID of the transaction. Each object should use public and private key pairs of the $PrDL_u$. The $PRPK_1$ is the public key of the object on the $PrDL_u$, while the SIG is the TID signed by this object's private key. $PUPK_2$ is the public key of the other object, which is on the $PuDL$. RT and EX indicate the relationship type and a field for extra information about the relationship, respectively. $Nonce$ is a random number.

In this case, even the second object publicly shares the information against the contract terms, the transaction will not be valid due to the lack of the first object's signature.

Finally, the third type of data recorded on the ledger is service transactions. Similar to the relationship entries, service entries can be recorded in either $PuDL$ or $PrDL_u$. The service transactions recorded on the $PuDL$ are public. However, when an object wants to use a service privately, for reasons such as privacy, the service transactions are only recorded on the $PrDL_u$. These transactions are only accessible by the objects on the $PrDL_u$.

The service request transactions recorded on the $PuDL$ have the following structure:

$$TID \,||\, PUPK_1 \,||\, SIG_1 \,||\, PUPK_2 \,||\, SIG_2 \\ ||\, SD \,||\, EX \,||\, Nonce$$

TID is the transaction identifier, which is calculated by hashing the transaction content. $PUPK_1$ is the public key, and SIG_1 is the signature of the service requester on the TID. $PUPK_2$ and SIG_2 are the public key and the service provider signature on the TID, respectively. SD is a description of the service, and EX is used to record additional service information. Fig. 3 shows an overview of the process.

When the ranking is performed and the service provider is determined, the service requester object or its administrator calls the smart contract. TID calculates by hashing SD, EX, and $Nonce$ fields and then signed by the service requester object. The smart contract receives the public key and the signed TID as inputs. In the smart contract, the signature and the signed fields' validity are first checked, then the transaction is submitted to the $PuDL$ with the specified structure. Finally, the service is provided to the service requester via a REST API. Fig. 4(b) shows an example of the structure of such smart contracts. Once the service is received, the service requester can record a transaction at any time to specify the result of the received service in $PuDL$.

Private service transactions are submitted to the $PrDL_u$ by the object or its administrator. The structure of these transactions includes:

$$TID \,||\, PRPK_1 \,||\, SIG \,||\, PUPK_2 \,||\, SD \,||\, EX \,||\, Nonce$$

TID is the transaction identifier. The service requester object should use public/private key pairs

Received: May 20, 2021. Revised: July 28, 2021.

Table 1. Parameters of PuDL size

Symbol	Definition
B	No. of transactions in a block
D	No. of all connected devices
H_P	Size of the header of POR blocks
H_R	Size of the header of REL blocks
H_V	Size of the header of SER blocks
P	Total no. of POR transactions
R	Total no. of REL transactions
V	Total no. of SER transactions
T_P	Size of a POR transaction
T_R	Size of a REL transaction
T_V	Size of a SER transaction
θ_R	Probability of sharing a relationship publicly
θ_V	Probability of sharing a service request publicly
C	Total no. of smart contracts

for the $PrDL_u$ to insert a transaction in it. $PRPK_1$ is the public key of the object on the $PrDL_u$, and the SIG is the signed TID by the private key of the object. $PUPK_2$ is the public key of the service requester object on the $PuDL$. SD is the service description, EX is a field for supplementary information of the service, and $Nonce$ is a random number. Since the transaction is recorded on the $PrDL_u$, the service provider signature is unnecessary because all the objects on the $PrDL_u$ have OOR, and their information is trustworthy. Later, the service requester object can add a transaction on the $PrDL_u$ to submit the service results.

3.4 Ledgers size analysis

As mentioned before, in SocioChain, users have the authority to declare their own objects' relationship and service transactions publicly on $PuDL$ or to store some information on their exclusive $PrDL_u$ to be accessible only by their own devices. Furthermore, there is no constraint imposed on the data storage location. Users can freely choose the storage location of $PrDL_u$ and $PuDL$. Under such circumstances, the user can store ledgers' data on local storage or any cloud service provider. In the former case, a remarkable limitation is the size of ledgers (i.e., $PrDL_u$ and $PuDL$). Since each user owns a limited number of devices, the number of OORs and private relationships are restricted. Consequently, the $PrDL_u$ of each user is small, and storing the $PrDL_u$ locally is not a challenge. Nevertheless, the size of the $PuDL$ can be significant. Data stored on the $PuDL$, includes public POR, SOR, C-WOR, C-LOR transactions, service transactions, and smart contracts. Table 1 shows the symbol notation used to calculate the size of the $PuDL$. It is assumed that any chosen DLT requires some space for each block's header.

As described in section 3, POR transactions are created and submitted by the manufacturer. If each block can store B transactions, the manufacturer submits one block on the $PuDL$ for each batch of B devices of the same model. The space required for the blocks is calculated as follows.

$$S_P = \frac{D}{B} \times (H_P + B \times T_p) \tag{1}$$

Users either share data with others or not for a variety of reasons [30]. θ_R and θ_V are added to the equations to consider this probability. The required space for SOR, CWOR, and CLOR transactions recorded on the $PuDL$ are as follows:

$$S_R = \theta_R \times (\frac{R}{B}) \times (H_R + B \times T_R) \tag{2}$$

The space required for the service transactions can be obtained as follows:

$$S_V = \theta_V \times (\frac{V}{B}) \times (H_V + B \times T_V) \tag{3}$$

Finally, the size of the required space to store smart contracts on the ledger is determined as follows:

$$S_C = C \times A \tag{4}$$

The summation of the above values calculates the total size of the ledger.

$$S_{DL} = S_P + S_R + S_V + S_C \tag{5}$$

4. Implementation and performance evaluation

The ledger size is an essential parameter in the possibility of implementation of the SocioChain. In this section, we first calculate the ledger size for different numbers of users using long-term simulation. Next, the performance of the proposed architecture is evaluated based on the results obtained in the first section.

4.1 Size of ledgers

The size of the $PuDL$ is one of the impactful parameters in implementing SocioChain in the real

Received: May 20, 2021. Revised: July 28, 2021.

world. It affects the possibility of the local storage of two ledgers and also the speed of data retrieval. The number of transactions in $PuDL$ determines its size. Due to the different sizes of the headers and blocks on the DLTs, the $PuDL$ size order can be estimated by assessing the number of transactions, i.e., the number of established relationships. To this end, we need to have real data of a large number of connected devices and their information, such as their capabilities, model, and brand, over time. Although some initial implementation of the SIoT is conducted [22], few real devices are now connected to it. A dataset is also provided for the SIoT using simulation [31], but it simulated the SIoT only for ten days. So, an alternative solution is required. To achieve this, We utilized SWIM [32], a mobility model, to simulate the devices' mobility and study their social behavior and relationships. This model considers social relationships among people through the simulation of their movements. In SWIM, the destination is determined based on its distance to the user's home or the place's popularity. The α parameter, varying from 0 and 1, specifies people's tendency to go to destinations closer to their home or more popular destinations.

The user perception radius is another influential parameter in the SWIM model. It determines the communication radius for each user, i.e., how far each user can communicate with another user. Since the available space is in the range of [0, 1] in the model, horizontal and vertical axes are divided based on this parameter. The number of simulation cells is determined based on it. Therefore, to increase the number of cells in the model, we need to decrease this parameter.

The output of the SWIM model is a trace of individuals' movements. In the simulation of the devices' movement, it is assumed that each user owns a set of devices. Some of the devices are in the user's house, while the rest are mobile and move with the user. To assign the devices to users, we used the data from the Global Web Index report in 2019 [33]. It is provided by a survey of more than 270,000 Internet users. According to the report (Table 2), a set of devices were assigned to each user in the SWIM model. Based on the resulting network, it is possible to create traces of the devices' movements and their interactions.

The purpose of this simulation is to show how the number of relationships changes over time. To this end, we conducted the simulation using a virtual server with 20 vCPUs and 16 GB RAM by adjusting the α equal to 0.9 [31]. We simulated two scenarios. In the first scenario, the simulation runs for a varied number of users for 100 days. The second scenario simulates 1000 users for 1000 days. Table 3 shows the information and configurations used in the simulations.

Table 2. Device ownership

Device	Ownership (%)	Type
Smartphone	95	Mobile
Pc/Laptop	70	Static
Smart TV	39	Static
Tablet	37	Mobile
Game Console	22	Static
TV Streaming Device	15	Static
Smartwatch	13	Mobile
e-Reader	12	Mobile
Smart Wristband	11	Mobile
Smart Home Devices	12	Static

Table 3. Simulation configuration parameters

No. of Users	No. of Mobile Devices	No. of Static Devices	Total No. of Devices	User Perception Radius
500	841	803	1644	0.0141
1000	1674	1608	3282	0.0100
2000	3416	3180	6596	0.0070
4000	6773	6378	13151	0.0050

Since the number of objects is fixed, the number of PORs does not change. The objects' ownership is assumed to be permanent, and thus, the number of OORs is fixed too. CLOR is established between the static objects that are in the communication range of each other. Each user has a specific number of static devices in two scenarios, and therefore, the number of CLOR does not change over time.

CWOR and SOR are two types of relationships whose numbers increase over time in these scenarios. CWOR is created between a mobile object and a static object if the meeting duration is longer than T_C. It was assumed that T_C is 1 hour. SOR is established between two mobile objects. It is set if the two objects meet each other N_M times, the meeting duration is more than T_M, and the time interval between two consecutive meetings is longer than T_I. In this scenario, N_M, T_M, and T_I were considered equal to 3, 1h, and 6h, respectively. Additionally, we introduced T_L to closing the long-term simulations' results to real-world data. If the time interval between two consecutive meetings is longer than T_L, the N_M will be reset to zero. In other words, the N_M will hit zero if the two objects have not established a relationship yet and didn't meet in the interval of T_L.

Fig. 5 and 6 show the number of established CWOR and SOR in the first scenario for 100 days, respectively. We assumed $T_L = \infty$. The results show that the number of established relationships is a small

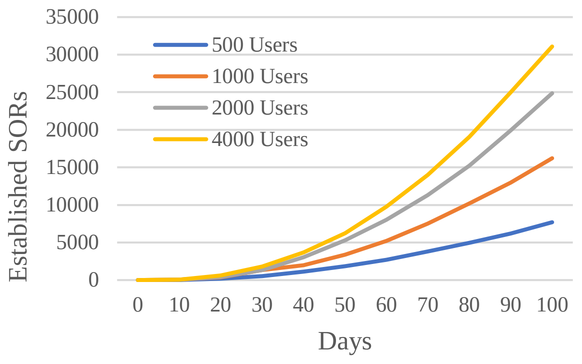


Figure.5 No. of established SOR over time

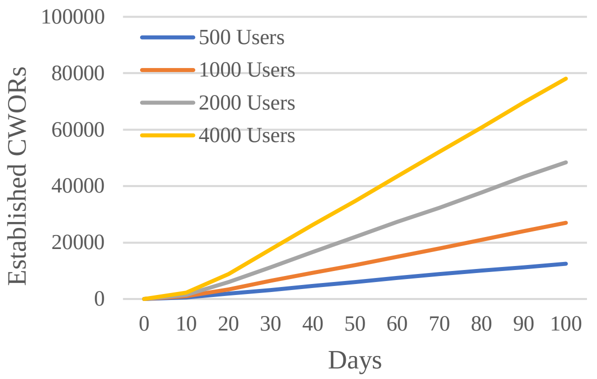


Figure.6 No. of established CWOR over time

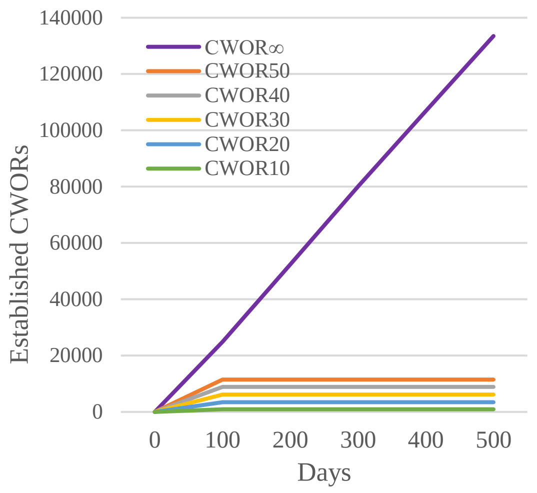


Figure.7 No. of established CWOR over the time for 1000 users

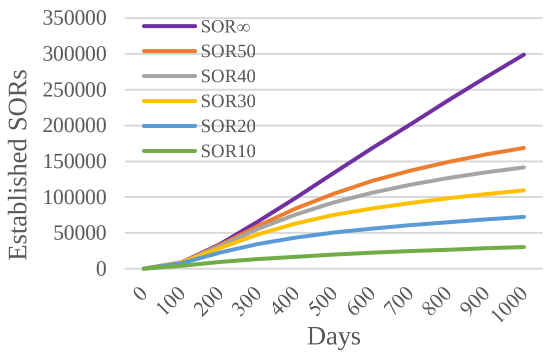


Figure.8 No. of established SOR over the time for 1000 users

percentage of all possible relationships between the objects. For example, 0.36% of all CWORs are set for 4000 users after 100 days. This value is only about 0.14% for the complete graph of SOR. The trend of establishing new relationships in Fig. 5 shows a nearly linear function. In Fig. 6, the function initially has an ascending slope due to the constraintsconsidered to establish the SORs. Some objects might not meet each other enough times in the initial days. SOR is established between them as the meeting number increases over time. Therefore, the number of SORs has a higher slope at the first 60 to 70 days, but the function has become linear later.

Fig. 7 and 8 showing the number of established SORi and CWORi in the second scenario in which index "i" is the value of T_L in terms of days. For example, in CWOR10, T_L is set to 10 days. We have done the simulation for $T_L = 10, 20, 30, 40, 50$ days and also $T_L = \infty$. In the two figures, if $T_L = \infty$, the function increases linearly over time. If the valid interval for establishing of relationship (i.e., T_L) is limited, the number of established relationships decreased. In the case of CWOR, because users often visit constant locations over time, all of CWORs are set in the first 100 days. Besides, by limiting T_L, the number of established SORs shapes a concave function whose slope has decreased over time. The results show that in a more realistic scenario, by limiting T_L, the number of established relationships doesn't jump up very high. A small percentage of all possible relationships between the objects are set after 1000 days. For example, with $T_L = 50$, only about 0.85% of all CWORs and 12.04% of all SORs are established for 1000 users after 1000 days.

The simulation results demonstrated that storing $PuDL$ in local storage is feasible; however, solutions like Light Node [34] and Pruning [35] can be applied when its size has become increasingly large. The scalability of SocioChain depends on the degree to which the utilized DLT is scalable. Currently, solutions such as Sidechain Technology [36] and Sharding [37] are being used to improve the scalability of DLTs. Utilizing such solutions can help us reach more than 1,000,000 transactions per second, which has highly exceeded the current maximum transactional throughput of centralized solutions [38].

4.2 Comparison and performance analysis

Scalability and latency are two important aspects that should be considered in SocioChain. We implemented an experiment to demonstrate the efficiency of this architecture and compare it with the

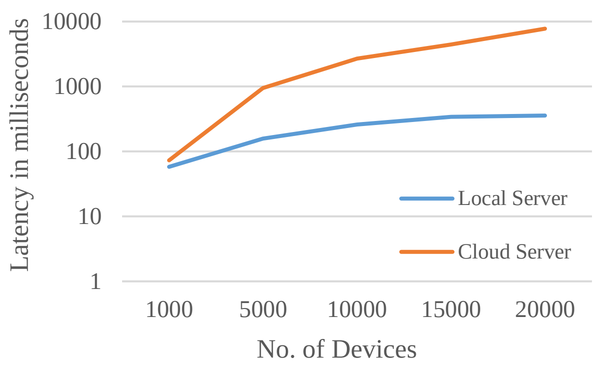


Figure.9 Comparison of response latency

previous cloud-based/centralized SIoT architectures. One of the main operations in SIoT is accessing the details of the relationship between objects. According to the last section results, it is possible to keep ledgers data locally. Ledgers data must be indexed to be explorable. Using relational database structure is a common approach for indexing ledgers data. An Example of such indexers is Eth-indexer (https://github.com/getamis/eth-indexer). We developed a web application using Python, which is connected to a MySQL database, containing information about the relationships between objects.

We conducted our experiment in two scenarios. In the first scenario, the database and web application are deployed on a server featuring eight vCPUs cores of Intel(R) Xeon(R) E-2176G. In the second scenario, a Raspberry Pi 3 Model B was employed as an IoT device on which the database and web application were deployed. We assumed that 10,000 new entries are added to the relationships database for every 1000 devices added to the network in both scenarios. In the first scenario, 10% of the devices send requests to the server simultaneously. In our second scenario, only the database size increases as more devices are added to the network, and there will be only one request from the device itself at any moment.

To dimension our experiments, we use a benchmark tool called ApacheBench, which is a tool for measuring the performance of web servers. Fig. 9 presented the results of implementing the experiments in both scenarios on a logarithmic scale. In the cloud server scenario, the latency has grown dramatically by increasing the number of devices and simultaneous requests. In the local server scenario, although response time has increased with the growing number of relationships in the database, the search operation is still conducted in an admissible time. This demonstrates that removing the central SIoT server in SocioChain leads to more efficient and scalable architecture comparing to other SIoT implementations.

5. Conclusion

This paper proposed a distributed architecture for social relationships between IoT objects, using the promising SIoT architecture. The new architecture is more scalable, secure, and privacy-preserving; besides, it exploits establishing social relationships between objects. In this architecture, objects relationships and service transactions are stored as entries in a two-layer distributed ledger. Public transactions are added to $PuDL$, and private transactions are inserted in $PrDL_u$.

A challenge in implementing SocioChain is storing ledgers' data locally. We initially present an equation to calculate the size of $PuDL$. Then, we simulate the social movements and relationships of users and their devices using the SWIM simulator. The results show that less than 10% of all possible relationships are established after 1000 days. So, ledgers' data may be kept in the local storage. Based on these results, we conducted another experiment to compare latency in Sociochian and previous SIoT implementation. It revealed that the proposed architecture has a significant advantage in latency and is more scalable. For instance, response latency in SocioChain is ten times less than previous SIoT-based architectures for 10000 devices.

Our future work plan is to develop an implemented prototype of SocioChain to analyze its efficiency in the real world and various domains, such as smart cities and vehicular networks. Also, we will study and analyze diverse DLTs to determine the best one for SocioChain with the highest efficiency and scalability.

Conflicts of interest

The authors declare no conflict of interest.

Author contributions

Conceptualization, A.Sohofi and A.Amidian; methodology, A.Sohofi and Y.Farjami; software, A.Sohofi; validation, A.Sohofi; formal analysis, A.Sohofi; investigation, A.Sohofi; resources, A.Sohofi; data curation, A.Sohofi; writing—original draft preparation, A.Sohofi; writing—review and editing, A.Sohofi, A.Amidian, and Y.Farjami; visualization, A.Sohofi; supervision, A.Amidian, and Y.Farjami.

References

[1] S. Nižetić, P. Šolić, D. L. D. I. G. D. Artaza, and L. Patrono, "Internet of Things (IoT): Opportunities, issues and challenges towards a smart and sustainable future", *J. Clean. Prod.*,

Vol. 274, p. 122877, 2020.

[2] M. Shirer, *The Growth in Connected IoT Devices Is Expected to Generate 79.4ZB of Data in 2025, According to a New IDC Forecast*, 2019. https://www.idc.com/getdoc.jsp?containerId=prUS45213219

[3] M. Noura, M. Atiquzzaman, and M. Gaedke, "Interoperability in Internet of Things: Taxonomies and Open Challenges", *Mob. Networks Appl.*, Vol. 24, No. 3, pp. 796–809, 2019.

[4] M. Aziez, S. Benharzallah, and H. Bennoui, "A full comparison study of service discovery approaches for internet of things", *Int. J. Pervasive Comput. Commun.*, Vol. 15, No. 1, pp. 30–56, 2019.

[5] P. Asghari, A. M. Rahmani, and H. H. S. Javadi, "Service composition approaches in IoT: A systematic review", *J. Netw. Comput. Appl.*, Vol. 120, pp. 61–77, 2018.

[6] L. Atzori, A. Iera, and G. Morabito, "SIoT: Giving a Social Structure to the Internet of Things", *IEEE Commun. Lett.*, Vol. 15, No. 11, pp. 1193–1195, 2011.

[7] L. Atzori, A. Iera, G. Morabito, and M. Nitti, "The social internet of things (SIoT) - When social networks meet the internet of things: Concept, architecture and network characterization", *Comput. Networks*, Vol. 56, No. 16, pp. 3594–3608, 2012.

[8] M. S. Roopa, S. Pattar, R. Buyya, K. R. Venugopal, S. S. Iyengar, and L. M. Patnaik, "Social Internet of Things (SIoT): Foundations, thrust areas, systematic review and future directions", *Comput. Commun.*, Vol. 139, pp. 32–57, 2019.

[9] J. A. Muhtadi, M. Qiang, K. Saleem, M. AlMusallam, and J. J. P. C. Rodrigues, "Misty clouds—A layered cloud platform for online user anonymity in Social Internet of Things", *Futur. Gener. Comput. Syst.*, Vol. 92, pp. 812–820, 2019.

[10] Y. Sahni, J. Cao, S. Zhang, and L. Yang, "Edge Mesh: A New Paradigm to Enable Distributed Intelligence in Internet of Things", *IEEE Access*, Vol. 5, pp. 16441–16458, 2017.

[11] Z. K. Zhang, M. C. Y. Cho, C. W. Wang, C. W. Hsu, C. K. Chen, and S. Shieh, "IoT security: Ongoing challenges and research opportunities", In: *Proc. IEEE 7th International Conference on Service-Oriented Computing and Applications, SOCA 2014*, pp. 230–234, 2014.

[12] C. Sarkar, A. U. Akshay, R. V. Prasad, A. Rahim, R. Neisse, and G. Baldini, "DIAT: A scalable distributed architecture for IoT", *IEEE Internet Things J.*, Vol. 2, No. 3, pp. 230–239, 2015.

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

[14] H. N. Dai, Z. Zheng, and Y. Zhang, "Blockchain for Internet of Things: A Survey", *IEEE Internet Things J.*, Vol. 6, No. 5, pp. 8076–8094, 2019.

[15] L. Zhou, L. Wang, Y. Sun, and P. Lv, "BeeKeeper: A Blockchain-Based IoT System with Secure Storage and Homomorphic Computation", *IEEE Access*, Vol. 6, pp. 43472–43488, 2018.

[16] J. A. Jaoude and R. G. Saade, "Blockchain applications - Usage in different domains", *IEEE Access*, Vol. 7, pp. 45360–45381, 2019.

[17] M. S. Ali, M. Vecchio, M. Pincheira, K. Dolui, F. Antonelli, and M. H. Rehmani, "Applications of Blockchains in the Internet of Things: A Comprehensive Survey", *IEEE Commun. Surv. Tutorials*, pp. 1–1, 2018.

[18] K. Christidis and M. Devetsikiotis, "Blockchains and Smart Contracts for the Internet of Things", *IEEE Access*, Vol. 4, pp. 2292–2303, 2016.

[19] I. Mashal, O. Alsaryrah, T. Y. Chung, C. Z. Yang, W. H. Kuo, and D. P. Agrawal, "Choices for interaction with things on Internet and underlying issues", *Ad Hoc Networks*, Vol. 28. pp. 68–90, 2015.

[20] A. M. Ortiz, D. Hussein, S. Park, S. N. Han, and N. Crespi, "The cluster between internet of things and social networks: Review and research challenges", *IEEE Internet Things J.*, Vol. 1, No. 3, pp. 206–215, 2014.

[21] F. Cicirelli, A. Guerrieri, G. Spezzano, and A. Vinci, "An edge-based platform for dynamic Smart City applications", *Futur. Gener. Comput. Syst.*, Vol. 76, pp. 106–118, 2017.

[22] R. Girau, S. Martis, and L. Atzori, "Lysis: A platform for iot distributed applications over socially connected objects", *IEEE Internet Things J.*, Vol. 4, No. 1, pp. 1–1, 2017.

[23] S. Ali, M. G. Kibria, M. A. Jarwar, H. K. Lee, and I. Chong, "A Model of Socially Connected Web Objects for IoT Applications", *Wirel. Commun. Mob. Comput.*, Vol. 2018, pp. 1–20, 2018.

[24] X. Zhu and Y. Badr, "Fog Computing Security Architecture for the Internet of Things Using Blockchain-Based Social Networks", In: *Proc. of 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. 1361–1366, 2018.

[25] K. Park, Y. Park, A. K. Das, S. Yu, J. Lee, and Y. Park, "A Dynamic Privacy-Preserving Key Management Protocol for V2G in Social Internet of Things", *IEEE Access*, Vol. 7, pp. 76812–76832, 2019.

[26] J. Lin, W. Yu, N. Zhang, X. Yang, H. Zhang, and W. Zhao, "A Survey on Internet of Things: Architecture, Enabling Technologies, Security and Privacy, and Applications", *IEEE Internet Things J.*, Vol. 4, No. 5, pp. 1125–1142, 2017.

[27] T. Jiang, H. Fang, and H. Wang, "Blockchain-based internet of vehicles: Distributed network architecture and performance analysis", *IEEE Internet Things J.*, Vol. 6, No. 3, pp. 4640–4649, 2019.

[28] O. Novo, "Blockchain Meets IoT: An Architecture for Scalable Access Management in IoT", *IEEE Internet Things J.*, Vol. 5, No. 2, pp. 1184–1195, 2018.

[29] M. Nitti, R. Girau, and L. Atzori, "Trustworthiness management in the social internet of things", *IEEE Trans. Knowl. Data Eng.*, Vol. 26, No. 5, pp. 1253–1266, 2014.

[30] M. M. L. Wasko and S. Faraj, "Why should I share? Examining social capital and knowledge contribution in electronic networks of practice", *MIS Q. Manag. Inf. Syst.*, Vol. 29, No. 1, pp. 35–57, 2005.

[31] C. Marche, L. Atzori, and M. Nitti, "A Dataset for Performance Analysis of the Social Internet of Things", In: *Proc. of IEEE International Symposium on Personal, Indoor and Mobile Radio Communications, PIMRC*, Vol. 2018-Septe, pp. 1–5, 2018.

[32] S. Kosta, A. Mei, and J. Stefa, "Large-Scale Synthetic Social Mobile Networks with SWIM", *IEEE Trans. Mob. Comput.*, Vol. 13, No. 1, pp. 116–129, 2014.

[33] "The device trends to watch in 2019", 2019. https://www.globalwebindex.com/reports/device

[34] "Light client protocol". https://eth.wiki/en/concepts/light-client-protocol

[35] "Block file pruning". https://github.com/bitcoin/bitcoin/blob/v0.11.0/doc/release-notes.md#block-file-pruning

[36] A. Singh, K. Click, R. M. Parizi, Q. Zhang, A. Dehghantanha, and K. K. R. Choo, "Sidechain technologies in blockchain networks: An examination and state-of-the-art review", *J. Netw. Comput. Appl.*, Vol. 149, p. 102471, 2020.

[37] H. Dang, T. T. A. Dinh, D. Loghin, E. C. Chang, Q. Lin, and B. C. Ooi, "Towards scaling blockchain systems via sharding", In: *Proc. of the ACM SIGMOD International Conference on Management of Data*, pp. 123–140, 2019.

[38] "Scaling DLT to over 1M TPS on Google Cloud", https://www.radixdlt.com/post/scaling-dlt-to-over-1m-tps-on-google-cloud/.