Performance Analysis of Verifiable Data Registry Solutions for Decentralized Identifiers

Morteza Alizadeh
Department of Computer Science, Electrical and Space Engineering
Luleå University of Technology
Skellefteå, Sweden
morteza.alizadeh@ltu.se

Karl Andersson
Department of Computer Science, Electrical and Space Engineering
Luleå University of Technology
Skellefteå, Sweden
karl.andersson@ltu.se

Olov Schelén
Department of Computer Science, Electrical and Space Engineering
Luleå University of Technology
Luleå, Sweden
olov.schelen@ltu.se

2022 IEEE Asia-Pacific Conference on Computer Science and Data Engineering (CSDE) | 978-1-6654-5305-9/22/$31.00 ©2022 IEEE | DOI: 10.1109/CSDE56538.2022.10089278

***Abstract*—User identification in decentralized systems is a demanding task. Identification systems should work resiliently and have efficient performance. Moreover, identification systems should protect the data that they must store against hackers and saboteurs. Keeping a system with decentralized identification without any intervention in the middle has attracted attention to improve earlier centralized identification systems. Decentralized Identifiers (DIDs) constitute a solution for identification divided into different modules. The verifiable data registry is one of the main parts of this technology, which is distributed storage of identity properties. We analyze the decentralized identification data registry and compare the performance of verifiable data registry based on blockchain and the Distributed Hash Table (DHT) on different scales of systems. Our evaluation results show that DHT has better performance. Furthermore, a model based on DHT shows that in addition to immutable storage and faster query time, it makes systems handle or search in data storage with lower searching time compared to Ethereum Blockchain as another immutable secure technology. Finally, our results show that DHT is a better solution than other models in different scenarios. Although blockchain has promising results on a small scale, it still has problems with storage and query time in large-scale systems.**

***Index Terms*—Blockchain, Distributed Hash Table, Record keeping, Immutability, Data registry**

I. Introduction

Decentralized Identifiers (DIDs) enable verifiable and decentralized digital identity [1]. DIDs have been designed to remove the need for centralized registries, identity providers, and certificate authorities. DIDs constitute an identification system associated with a DID Document, which contains information, public keys, protocols, and available service endpoints. DIDs help to identify and track all malicious behavior. Usually, there are three actors in communication in a DIDs system: holder, issuer, and verifier. Furthermore, the data registry role constitutes the fourth main part of these systems and is the main time-consuming part of DIDs. Moreover, the "trust triangle" in a decentralized system represents the basic communication structure of the holder, issuer, and verifier, where a data registry is essential for collaborating with these angles. A data registry should have three primary features: fast, immutable, and traceable. A person, an organization, and a smart device can play any other role in the triangle [2].

A decentralized data registry for decentralized systems is challenging since data must be stored and accessed to avoid any single point of contact. Therefore, speed and immutability are two main challenges in smart identification-based systems that need to be improved. Furthermore, the number of stored records, execution speed, and the number of participants in a large-scale system are targeted in this paper.

Blockchain technologies, as decentralized data registry structures, provide an immutable and secure environment for identification applications. Therefore, blockchain is a primary module in DIDs, which helps them to be effective with a strong consensus and secure execution. Moreover, when the identification system reports any malicious behavior such as the wrong data type or incorrect data, the other applications can verify the declaration by communicating with the blockchain and punishing the malicious member or gateway if the claim is valid [3]. Although blockchains have several advantages, some, such as public blockchains, are not a quick solution when the system has to manage many peers with multiple records of information in a short time [4]. The main reason DIDs need blockchain for their verifiable credential is to resolve immutability and privacy issues in the authentication and record sharing part in DIDs. Privacy preservation is performed by proposing a user-centric control model for identity management without a central authority to manage it. DIDs leverage blockchain in the verifiable credentials part to show the trust between the parties and preserve part of personal data undisclosed for everyone on the blockchain.

A Distributed Hash Table (DHT), as another decentralized solution, can store data with a superfast look-up function for searching and querying data based on key-value pairs [5]. DHT can manage large-scale systems. It makes big systems fast. This DHT characteristic can be helpful for identification systems. Moreover, DHT technology has features similar to blockchain, such as immutability, resilience, and data sharing among network nodes. Additionally, DHT can be used as a data registry part in DIDs.

This paper's main contribution is finding the best solutions by considering different parameters (speed and scale) in a decentralized data registry subsystem that can be used to improve the performance of decentralized identification. Moreover, this paper aims to answer how decentralized solutions for data registries such as blockchain and DHT can positively affect an identification system.

In this paper, we measure and compare query times and fetching times to propose better solutions in three steps. First, we consider blockchain to be a resilient method to access data with low latency and low dependence on connectivity. In addition, blockchain provides immutable records in transactions and time stamping. Second, we compare the result with DHT. DHT can be another solution by considering Kademlia, which has quick lookup functions to load data [5]. Finally, this paper compares three models based on blockchain and DHT that can be used as verifiable data registries and measures query time with different nodes in resilient networks to achieve more efficient connectivity. The rest of this paper is organized as follows. In Section II, we perform a literature review. In Section III, we present the background and terminologies. Section IV describes different models. Section V evaluates different models and measurements. Finally, we discuss and conclude the paper in Sections VI and VII.

II. Literature Review

The process for authenticating DIDs includes three regulations from a high-level perspective. These three regulations are holders of a DID, the issuer of DID, and the verifier retrieves the DID Document from a data registry. DID Documents are stored in a registry that must perform the processes on documents, such as Create, Read, Update, Deactivate (CRUD) methods, and implement them securely [6]. Kang et al. [7] performed an empirical study on cloud and decentralized identifiers. They provided a framework for an integrated and efficient data access control policy that brought strong security, integrity, and availability to the cloud common data model. Although much research has been done and shows that the cloud is a good solution for data registry, there is still a centralized definition in cloud technology. Alizadeh et al. [8] clarified that the cloud is not a decentralized system from a management viewpoint. It is a distributed centralized system under organizational management. Thus, replacing this technology with a decentralized one is desirable. Alzahrani [6] and Farmer et al. [9] show that blockchain is a good solution for the decentralized identification data registry issue where the DID registry is replicated in all nodes. For example, Hyperledger Indy [10] is a type of public permissioned blockchain in which everyone can read data from the ledger without any permission but not in uploading data. The main problem of permissioned blockchains is that they are not fully decentralized but fast. Alizadeh et al. [11] showed that DHT in a combination of permissionless blockchain can increase the performance, but it is still slow. Augusto et al. [12] suggested a faster solution using socket and pipeline to blockchain and emitting data in real time.

This paper shows different solutions based on blockchain, pipeline to blockchain, and DHT for the data registry. The tuned Blockchain and DHT can be a replacement for the general blockchain. This paper illustrates the improved performance after analyzing the output of different proposed models as a fast, immutable, fully decentralized solution.

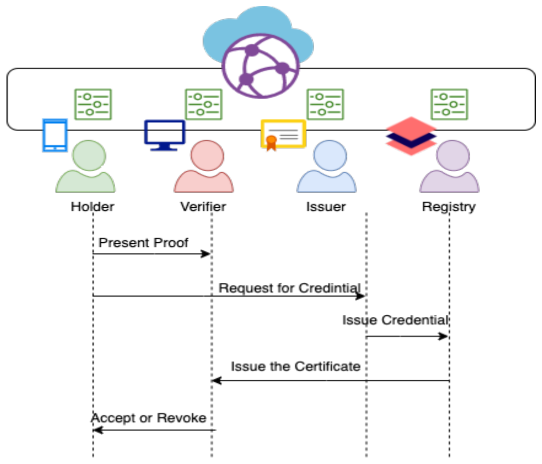


Fig. 1. DIDs internal actors

III. Terminologies

This section briefly explains different decentralized technologies, decentralized identifiers, blockchain, distributed hash table, the differences among decentralized technologies for the verifiable credential solution, and the publish/subscribe technique.

A. *Decentralized Identifiers (DIDs)*

Decentralized identifiers constitute a type of decentralized system that are managed with the help of all system participants. A decentralized system is different from a centralized system in that not all management tasks are handled by a single party. They are formed by nodes such as users, organizations, issuers, and validators. A peer-to-peer system is a popular protocol that is used for communication in decentralized systems, such as Distributed Ledger Technology (DLT) and DHT [11]. DIDs are decentralized, and no single organization manages the identification. They work in a peer-to-peer network where many nodes can be users, organizations, issuers, and validators. For example, self-sovereign identity as a kind of DID that represents a system that manages digital identities where users control their identity and privacy [13].

The issuer produces and manages unique IDs in a system after the holder registers his or her request to the system. Then, a verifier can approve that this user is a valid user with the help of information provided by the issuer by looking at the holder's credential. The holder can decide who can look at his or her information, which means the holder can control his or her privacy. All addresses and public keys can be stored on the data registry for future actions in an immutable way. Figure 1 shows the different modules in DIDs and how they communicate. For example, a holder can present his or her identity, like a driving licence to create a credential in the system, where the verifier can pick his or her case to check the validity. Then, the verifier can approve this holder by the summary of the earlier issuer's information in the data registry and credential. As the final step, the verifier can accept the login or reject it if the holder's documents do not satisfy the verifier. A DIDs node communicates through a DID Document that is a file representing different objects such as the associated repository and public key information. They may be used without a centralized registry, identity provider, or certificate authority. DIDs follow the Uniform Resource Identifiers (URIs) format and connect a DID subject to a DID Document. An example of a DID is `did:example:123456abcdef`, and this object is one of the objects in a DID Document [1].

B. Blockchain

As a popular distributed ledger technology, blockchain is a decentralized system that delivers immutability for applications. A blockchain manages data records into blocks. Blockchains also help to protect transactions' data from being changed. One of the uses of the blockchain is storing a cryptographic signature of recorded data and events as a part of a data block on its chains [14]. Permissioned (private or consortium) and permissionless (public) blockchains are two forms of blockchain technologies. In a permissioned blockchain, a limited number of known trusted participants are allowed to record or even read a copy of the ledger. In some permissioned blockchains, everyone can read, but it is not open for access to all the stored data. On the other hand, a permissionless blockchain is an open system that allows everyone to join or cancel their account without relying on a central authority. Additionally, blockchain is a group of data packages (blocks) connected that represent the historical data of transactions. One of the main blockchain features is its unique timestamps and hash values that help to keep it traceable. Each block refers to the previous block by carrying the parent hash value. It is possible to return to the original ring, chain, or block by following the parents' hash values in a sequence. Each block can carry a group of transactions inside; therefore, the block creation process is complex. Most network participants must approve a new block by consensus, which is added to the list of approved blocks. However, to encourage other nodes to participate in the consensus, there is a reward that is cryptocurrency in the blockchain system. All nodes receive the copy after the consensus procedure is completed. This is known as replicate and is stored as transactional information. The transaction includes both sides' information such as Owner, Renter, Signatures, and Addresses. In addition, the timestamp is stored and shared through the network. Then, all network members keep a copy of the proof and know the time and who executed it. The cryptocurrency or credits are parts of the payment for those activities that lead to transactions. The process of approving, usually called consensus, has a different goal. Proof of work and proof of stake are two popular types of this process in which credits are the network's reward to participants who approved newly added information. The transactions are stored as an event with details in a Merkle tree that belongs to each block in the blockchain [15]. All blocks with their stored transactions are shared with other network participants as a ledger. This means that all nodes in a network receive a copy of the latest transactional event. This process makes the system resilient to recover itself after losing some information for any reason by copying other ledgers. Ethereum is a permissionless blockchain that uses smart contracts to operate, where Ether is its currency, and consensus is achieved under the proof of stake. A smart contract is a Solidity-based distributed computer program used for transaction mechanisms. Figure 3 (A) shows the interaction among the user, blockchain, and smart contract.

C. Relation between Verifiable Data Registry and Blockchain

Verifiable Data Registry is a lookup table in decentralized systems where anyone can request to read some part of the records to verify what organization a specific public DID belongs to. A verifiable data registry can consist of a group of information such as public DIDs, credential definitions, schemes, revocation registries, and proofs of consent for data sharing [16]. Blockchains, as verifiable data registries, store the public DIDs of the entities that issued the credential. Blockchains help to verify the authenticity or validity of the credential. For example, it is possible to check the blockchain ledger and see who issued the already registered transactions without contacting the issuing party. The blockchain is the structure that is used as a verifiable data registry. It provides control over the evolution of data between entities through a peer-to-peer network and ensures replication across the network nodes. In the data registry, blockchain allows everyone in the network to have a copy of the information about which credentials are valid and who is allowed to verify the data inside the credential without showing the actual data. The blockchain verifying parties do not need to check the actual existing data. Instead, the blockchain guarantees the correctness of each published data by the help of consensus. Verifiable credentials should provide trust between the parties and guarantee the authenticity of the data without actually storing any personal data on it. The Zero-Knowledge Proof (ZKP) is a common method in DIDs to allow others to prove that their record of information meets specific requirements without showing the exact details. A ZKP is a method to authenticate entities to prove to other entities that they see a piece of specific information or requirement without disclosing any details. This method is advantageous in DIDs where the proving entities do not trust the verifier and must prove itself to the verifier by disclosing specific information [17].

D. Distributed Hash Table (DHT)

DHT is a decentralized technology that uses hash as a solution and has a fast lookup time. The hash value is encoded by passing some data through a function that produces a result called a hash. There are different functions used for encoding, but the hash function is a one-way function. Returning to the encoded value is impossible without knowing a specific private key to decode. A hash table is known as a famous structure in databases because of its fast lookup time. It is an abstract data type that can map keys to values. In addition, the hash is a preliminary part used in decentralized systems. In the DHT, the user can obtain the appropriate file according to the unique hash address. The DHT can perform tracking of the data owner. It stores data packages by creating data chunks. The distributed component of DHT indicates that the table of hashes is shared over many network nodes. DHT can recognise which nodes have which data by using Kademlia technology designed in 2002 by Petar Maymounkov and David Mazieres. A peer-to-peer network is required to run DHT [5]. Kademlia is an algorithm that creates a distributed hash table for decentralized peer-to-peer computer networks that determine the format of the network, the communications, and exchanges of information through node lookups. Node IDs are specific unique ID numbers belonging to the participants that form an overlay network. All nodes can communicate through the User Datagram Protocol (UDP) protocol together. The Kademlia algorithm locates values with the help of node IDs that can be file hashes or keywords. The node ID delivers locations of file hashes, and that node stores information on where to obtain the file or resource. The search process includes several steps that work by finding the closest node in each step closer to the key until the desired node returns the value or no closer nodes are found. Kademlia uses a binary tree structure for the routing process and the XOR metric for calculating the distance between points in the key space. The routing process for looking up the specific value can be done in logarithmic time based on the number of nodes on the network O(log(n)) [5].

E. Blockchain versus DHT

Blockchain and DHT work based on sharing data among participants in a peer-to-peer network by dividing and creating replicas among nodes. Figure 2 shows the DHT and blockchain architectures. This figure shows an example of a DHT with a Chord [18] structure where nodes are visible in a circular fashion. Moreover, each node carries a piece of the distributed hash table. DHT enhanced an important load balancing method, which also allows the system to be transparently scalable since if peers become overloaded, new peers running the service can be added to the system [19].

A DHT system stores public data through a distributed database and is basically just a large key/value store. Each node holds a small shard of the DHT. However, this process is similar to the sharing ledger in blockchain but with a different sharing mechanism [20]. Additionally, the DHT stores multiple copies of each piece of data, so the information is available even when a failure occurred. This property shows the high resilience and availability of DHT. This means that nodes can still interact with each other when some failures occur. In the recovery process of DHT, each partition in the hash table is replicated on more than one cluster node. Any replica can be used to recover, but all replicas must be updated during adding or removing processes with the help of the surviving members of the partitions' replica group. Additionally, this replica group membership is dynamic, meaning that after a failure, all of the group's replicas are removed from the current groups.

In the blockchain architecture, all nodes carry a distributed ledger with its Merkle tree. Additionally, all blockchain nodes receive an updated copy of the ledger after the consensus has been reached.

The amount of data that all nodes hold increases constantly in a blockchain. Therefore, keeping the system scalable is difficult. Furthermore, this weakness leads to a decrease in the number of nodes that can store all data. For these reasons, designing a lookup method seems complex specifically in permissionless blockchain to support anyone joining [21]. On the other hand, DHT does not cover cryptography at this stage and is not a transactional environment. However, in DIDs systems, specifically, an identification proposed by `W3C` [1] uses blockchain to keep some parts of information distributed unrelated to payments and transactions. Hence, the need to have such a transactional and payment environment system is weak, where DHT can be a more scalable and faster solution for this issue.

F. Publish/Subscribe

The publish/subscribe is a messaging pattern. Usually, message senders, called publishers, do not directly send the messages to specific receivers, called subscribers. Instead, publishers publish messages into classes without knowledge of who the subscribers are. Additionally, subscribers show interest in one or more classes and receive messages after applying for their interests without knowing which publishers have already published them. Publish/subscribe is generally one part of a larger message-oriented middleware system, and many systems support both the traditional messaging methods based on the request–response model and the publish/subscribe messaging model. This method provides a better more scaled network, real-time data processing, and a more dynamic network topology [22].

G. Performance

A performance study systematically explains the action or process of performing a task or function. The performance relies on different parameters. Most similar works calculated performance based on user accessibility and response time. This paper defines high performance as the models' output providing immutability and being fast. In addition, these systems usually manage larger network sizes and tasks.

IV. Models

There are three data registry structures that we analyze in this section: two models use blockchain for the data registry, while the third model uses DHT.

A. Model A: Blockchain as a Data Registry

In this model, the blockchain acts as a data registry that can store information as a transaction on the ledger. All this information is accessible for all the network parties. Security and immutability are the purposes of using blockchain as a

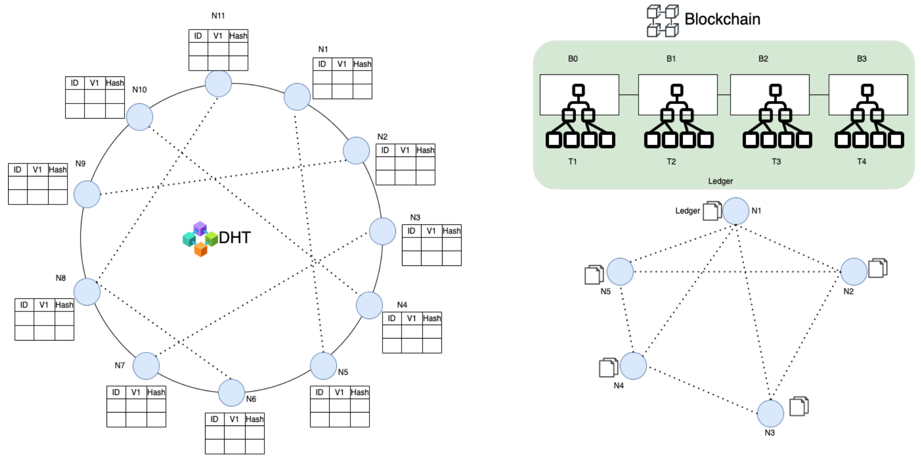
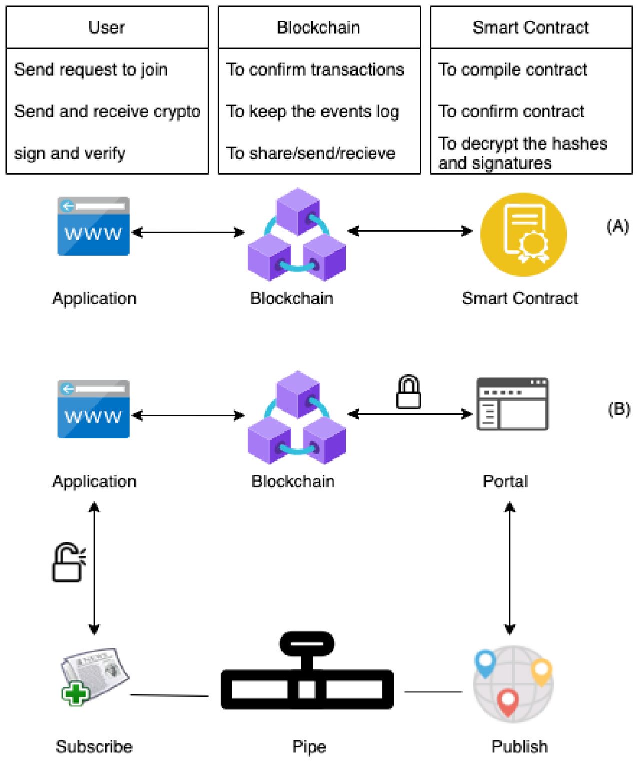


Fig. 2. DHT and Blockchain structure

data registry. However, the main problem of this model is the query time for querying and searching specific data in the ledger and historical data when it has a large number of stored records. Nevertheless, the process of storing is not time-consuming, and it can be done in a constant amount of time. However, the searching and query process takes time, and this period contains communication with a smart contract on the blockchain network, validation process by the network, and registering the transaction on the ledger, as shown in Figure 3 (A). There are three modules illustrated in Figure 3 (A): the application, smart contract, and blockchain network. The name of the application in communication with blockchain is Decentralized Applications (DApp). The DApp is on the user side, and users can install it free and run it when they are one registered node in the blockchain network. Usually, developers fix and publish DApp on the blockchain network. Additionally, DApp is connected to a smart contract. This means that users who want to use this application must accept the policies defined by the developer (default terms) in the smart contract. A transaction can be executed when users press the button, which means the user accepts the contract. This process appears on the users' electronic wallets when they execute the DApp and pay using the cryptocurrency on their wallets. This process is an agreement that the user agrees to policies. Finally, a transaction with smart contract fields is stored as a record in the ledger, and the ledger is distributed through the blockchain network.



User	Blockchain	Smart Contract
Send request to join	To confirm transactions	To compile contract
Send and receive crypto	To keep the events log	To confirm contract
sign and verify	To share/send/recieve	To decrypt the hashes and signatures

Fig. 3. Query time challenge in blockchain

B. Model B: Publish/Subscribe to Blockchain Historical Records

In this model, the blockchain acts as a data registry similar to Model A that can control storing information in the form of transactions on the ledger. Additionally, all network parties have the right to access this information. The permissionless blockchain is used as a data registry that is secure and immutable, similar to the previous model. We propose to communicate with the portal of blockchain and fetch data by opening a pipe to this web application with the help of socket programming. This helps to reduce the long query time for transactions and fetches and decodes data from the network. However, the main problem of this model is the query time for searching specific data in the history, which takes up to one minute. The performance of this model is high when the system has a large number of transactions and historical data on the ledger. In this model, all transactions are decoded later on the user side with the help of the Application Binary Interface (ABI) file. The ABI format is like a map with all variables and functions that the decoder functions need to know for the decoding process. Nevertheless, the process of storing is not time-consuming, which is the case for Model A. The model is illustrated in Figure 3 (B). `Infura.io` [23] is a web portal that provides a link to open a pipeline to the Ethereum blockchain by knowing the network ID and the hash address of the network.

C. Model C: DHT as an Immutable Data Storage

In this model, the DHT method is used. The DHT method is derived from the Kademlia algorithm. As previously explained, this method's data retrieval and reading speed are very high, and the look-up time is very low. In addition, this method is wholly distributed and is an immutable solution that can be used to store decentralized information. As illustrated in Figure 2, it is clear that in a peer-to-peer network, all users are linked to others to insert and query information in a distributed manner. Some attributes such as open access and being free to execute commands in DHT cause this technology to be replaced with blockchain in an extensive system that does not need payment subsystems. Moreover, the information is stored in the form of many slices of data in a distributed manner. Therefore, DHT can be one of the best solutions for verifiable data registries.

The main difference between Model C and the other blockchain-based models is that DHT splits the data block into many small pieces and shares them in a distributed fashion (data partitioning). Instead, the blockchain shares the same copy with all the network nodes. Model C can protect data from malicious attackers deleting it because it is difficult to delete all pieces at the same time. Furthermore, although DHT splits the data into small pieces, it makes replicas in a cluster layer to prevent losing data in failures and when nodes leave the network.

V. Evaluation and Measurement

In this section, we discuss how to implement the three models described above. The main process is divided into starting to send a read request from the user into internal layers and then trying to gain query time in the data registry part for the three models.

The holder, verifier, and issuer parts are not related to the main contribution of this article. Therefore, we relaxed our study to only cover the data registry so that it does not stress all parameters. In this paper, the different sizes or numbers of nodes and records (10, 20, 100, 1,000) are considered as test cases. This means that 10 is a small system with just ten records inside, and 1,000 is the largest.

Table I shows our analysis and our implementation, indicating that Model C has logarithmic computational time complexity in the query process. $T_{\text{Model A}}$ is the time complexity of Model A, where N is the number of records stored on the ledger, $T_{\text{Model B}}$ is the time complexity of Model B, N is the number of records in the blockchain, and $T_{\text{Model C}}$ is the time complexity of running Kademlia (time to look up). T_{BC} is the network connection time independent of algorithms, depending on the blockchain and the complexity of running a smart contract on it. T_{SoK} shows the socket time to connect by the subscriber to the `Infura` gateway. T_{Fetch} is the extraction time of the output of the pipeline. T_{Decode} is the time for decoding the transaction records with the help of the ABI file.

TABLE I
Time Complexity

Time	Description	Complexity
$T_{\text{Model A}}$	Complexity of Model A (n is the number of records)	Linearithmic time
$T_{\text{Model B}}$	Complexity of Model B (n is the number of records)	Linearithmic time
$T_{\text{Model C}}$	Complexity of Model C (n is the number of nodes)	Logarithmic time
T_{BC}	Time to execute and decode by smart contract	Independent constant time
T_{Sok}	Pub-Sub connection time	Independent constant time (up to 60s)

$T_{\text{Model A}}$ is calculated by the sum of the time to connect to the blockchain network and the time to call a smart contract for each record, where N is the number of records (Equation 1). T_{SC} is the time of decoding a transaction by the smart contract. $T_{\text{Model B}}$ is calculated based on subscribed time plus time to decode each record on a user side (Equation 2). $T_{\text{Model C}}$ is the same log (number of nodes) as the lookup time in Kademlia (Equation 3). T_{BC} is much greater than T_{Sok} when the system has a large number of data records (more than 100 records).

$$T_{\text{BC}} >> T_{\text{Sok}} \quad | \quad where\ number\ of\ records\ are\ big$$

$$T_{\text{Model A}} = \sum_{i=1}^{N}(T_{\text{BC}_\text{i}} + T_{\text{SC}_\text{i}}) \quad (1)$$

$$T_{\text{Model B}} = T_{\text{Sok}} + \sum_{i=1}^{N}(T_{\text{Fetch}_\text{i}} + T_{\text{Decode}_\text{i}}) \quad (2)$$

$$T_{\text{Model C}} = T_{\text{DHT}} = \log(M) \quad (3)$$

`M: number of nodes`

`N: number of records`

In Model A, we use the `Ethereum` blockchain. Then, `Ganache` helps to bring up the local blockchain test environment. Additionally, we test to query data from the Kovan test network that is an Ethereum real test network for developers with the help of `Metamask` as a wallet and `web3` and `Truffle` as communication languages. Table II shows a comparison between query time for Model A in the local machine (Ganache) and real blockchain. The differences between these two measurements indicate that communicating with the real test network with the several included nodes is a time-consuming process.

In Model B, we use the Ethereum blockchain web portal `infura.io`, which helps to bring up the blockchain transactions with the help of pub-sub and socket programming to the `infura.io` portal. The test environment is local on the computer with the help of the `web3-providers-ws` library on `web3` and `infura.io` as communication languages deployed under the Visual Studio Code editor.

In Model C, we implement `DHT-Kademlia` with the help of the Go language. Then, we store a text file to the network locally. We use `Docker containers` to bring up the network nodes on the local machine.

We repeated the Experiment 10 times. As a test, there are ten nodes in the Ganache for blockchain and the Docker for the DHT test. The results are listed in Table III, where the average for the abovementioned functions is indicated in each caption. We used a MacBook Pro (16-inch, 2019) laptop with the following configuration: macOS Monterey Version 12.3.1, 2.6 GHz 6-Core Intel Core i7 processor, 32 GB 2667 MHz DDR4 memory, 780/850 Mbps average download/upload data rates, Google Chrome Version 102.0.5005.61 (Official Build), go V1.18 darwin/amd64, node V16.14.2, npm V8.5.0, truffle V5.0.5, online Remix IDE, ganache V2.5.4, Docker desktop V4.7.1, and web3 v1.7.3.

VI. Results and Discussion

We compared different models' query times in tables and diagrams. Fetching time is the time to find the exact file by knowing the file's address on the network, while query time is the time to find all content that is shared among the nodes. Model C has better result than the two other models during our tests. The large difference between them covers the lookup time in DHT, which is logarithmic. Model A has better performance in a small environment, and Model B has scalable performance. However, Model A could not manage a large number of records, as shown in Figure 5. Referring to a smart contract for each record to decode it and finding a similar record in the search algorithm is the weakness of Model A. The publish/subscribe method can complete the query process approximately in one minute in Model B independent of the number of stored records (depends on `infura.io` portal internet speed). An Application Binary Interface (ABI) is an interface that obtains information from two binary program modules. An ABI file is needed to decode the transaction. ABI is a `JSON` file that describes all parameters and functions in the smart contract and their data types. This file helps to decode transactions in the local machine.

Figure 4 shows four different query times by the different models. The Kovan was not mentioned earlier in the model's section and is a product of model A in a real test network with multiple nodes. The measured times are in milliseconds and range from 0 to 100 records (200 records in Figure 5). As shown in Figure 4, Model B is worse than the other models in query time when there is a limited number of records (100). Moreover, the estimation on Model B indicates that Kovan communicates with several nodes. Moreover, while Model A converges to infinity soon, which means that this model cannot answer large-scale problems in a shorter time than Model B, as shown in Figure 5. Therefore, DHT is the best solution and is very fast in fetching data.

There is a comparison of query time between the blockchain system compiled on the local machine and the real test network. In the real network, no information shows how many nodes are on the network at different times, which is the reason this form does not have a good result compared with

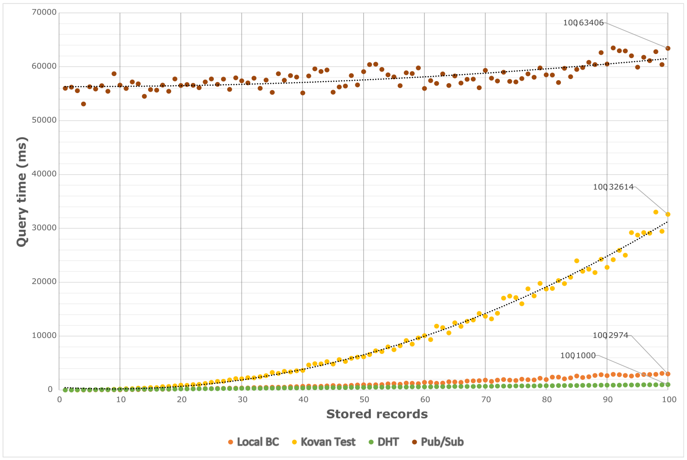


Fig. 4. Query time comparison

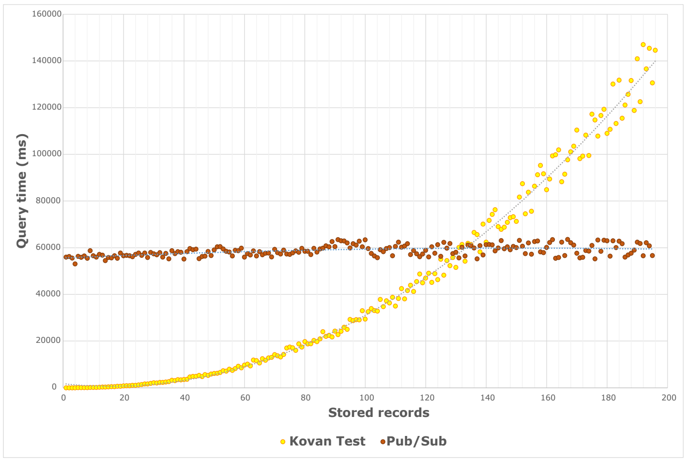


Fig. 5. Query time comparison between Models A and B

a local version of blockchain that has just 100 nodes. Table II lists the average of ten repeats for this comparison. The main comparison among the three models is summarized as query time in Table III, in which the data are calculated as the average of ten repetitions.

Table IV displays a comparison among a local version of

TABLE II
Query time for Model A in local machine and Kovan test network

Records	**10**	**20**	**100**	**1,000**
Local	140 ms	160 ms	3,500 ms	10,000 ms
Kovan	2,800 ms	5,300 ms	32,000 ms	Not Responding

TABLE III
Query time with different record number comparison between models

Records	**10**	**20**	**100**	**1,000**
Model A	120 ms	130 ms	1,300 ms	10,000 ms
Model B	58,000 ms	60,000 ms	63,000 ms	65,000 ms
Model C	50 ms	50 ms	400 ms	700 ms

TABLE IV
QUERY TIME BY 10 AND 100 NODES IN BLOCKCHAIN AND DHT

	Records	10	20	100	1,000
10 Nodes	Blockchain	120 ms	130 ms	1,300 ms	10,200 ms
	DHT	50 ms	50 ms	400 ms	700 ms
100 Nodes	Blockchain	170 ms	183 ms	4,000 ms	51,100 ms
	DHT	40 ms	50 ms	700 ms	9,000 ms

TABLE V
FETCHING TIME COMPARISON

	Records	10	20	100	1,000
10 Nodes	Blockchain	80 ms	120 ms	810 ms	9,051 ms
	DHT	5 ms	7 ms	50 ms	540 ms

blockchain and DHT with different numbers of nodes and records. Again, DHT is superior to the blockchain method.

Therefore, the difference between fetching time and query time relies on the number of records and nodes. For example, Table V shows the fetching time in both blockchain and DHT networks with ten nodes. DHT has better response time.

Our study's overall achievement is comparing some principal solutions experimentally for a decentralized identifier data registry and performing some quantitative comparative performance evaluation. However, performance evaluations of larger-scale studies are left as future work.

VII. CONCLUSION

Current research has achieved promising results with regard to decentralized identification systems that capture users' trust in such an open environment for all to be inside. First, this paper presented challenges such as registry and storage problems of decentralized identification. Then, it highlighted how the DLT and DHT's immutability could be useful as a fast solution. Then, this paper addressed how a DHT-based architecture is an excellent way to keep all systems decentralized while avoiding memory constraints. As one of the new approaches, we suggested replacing DHT with blockchain in DIDs for better performance. We compared three models and measured the query time in the data registry part. The output showed that DHT performs better than permissionless blockchain. Although the results indicated that blockchain with socket connectivity is better for large numbers of records, DHT is still the best solution. Therefore, we plan to find a solution for the query time of large-scale systems to have a strong and reliable decentralized identification system.

ACKNOWLEDGEMENT

This work was supported by the Kolarctic CBC project DIT4BEARS under grant KO4096.

REFERENCES

[1] World Wide Web Consortium (W3C), "Decentralized Identifiers (DIDs) v1.0, core architecture, data model, and representations," Accessed on October 22, 2022. [Online]. Available: https://www.w3.org/TR/did-core/

[2] G. Laatikainen, T. Kolehmainen, and P. Abrahamsson, "Self-sovereign identity ecosystems: benefits and challenges," in *Scandinavian Conference on Information Systems*, 2021, pp. 1–16.

[3] P. Singh, M. Masud, M. S. Hossain, and A. Kaur, "Cross-domain secure data sharing using blockchain for industrial IoT," *Journal of Parallel and Distributed Computing*, vol. 156, pp. 176–184, 2021.

[4] V. K. Tan and T. Nguyen, "The Real Estate Transaction Trace System Model Based on Ethereum Blockchain Platform," in *2022 14th International Conference on Computer and Automation Engineering (ICCAE)*. IEEE, 2022, pp. 173–177.

[5] P. Maymounkov and D. Mazieres, "Kademlia: A peer-to-peer information system based on the XOR metric," in *International Workshop on Peer-to-Peer Systems*. Springer, 2002, pp. 53–65.

[6] B. Alzahrani, "An information-centric networking based registry for decentralized identifiers and verifiable credentials," *IEEE Access*, vol. 8, pp. 137 198–137 208, 2020.

[7] Y. Kang, J. Cho, and Y. B. Park, "An empirical study of a trustworthy cloud common data model using decentralized identifiers," *Applied Sciences*, vol. 11, no. 19, 2021.

[8] M. Alizadeh, K. Andersson, and O. Schelen, "A survey of secure Internet of Things in relation to blockchain," *Journal of Internet Services and Information Security (JISIS)*, vol. 10, no. 3, pp. 47–75, 2020.

[9] C. Farmer, S. Pick, and A. Hill, "Decentralized identifiers for peer-to-peer service discovery," in *2021 IFIP Networking Conference (IFIP Networking)*. IEEE, 2021, pp. 1–6.

[10] The Linux Foundation, "Hyperledger Foundation," Accessed on October 22, 2022. [Online]. Available: https://www.hyperledger.org/use/hyperledger-indy

[11] M. Alizadeh, K. Andersson, and O. Schelén, "DHT- and blockchain-based smart identification for video conferencing," *Blockchain: Research and Applications*, vol. 3, no. 2, pp. 1–12, 2022.

[12] L. Augusto, R. Costa, J. Ferreira, and R. Jardim-Gonçalves, "An application of ethereum smart contracts and iot to logistics," in *2019 International Young Engineers Forum (YEF-ECE)*. IEEE, 2019, pp. 1–7.

[13] J. Yin, Y. Xiao, Q. Pei, Y. Ju, L. Liu, M. Xiao, and C. Wu, "Smartdid: a novel privacy-preserving identity based on blockchain for IoT," *IEEE Internet of Things Journal*, 2022. [Online]. Available: https://doi.org/10.1109/JIOT.2022.3145089

[14] M. Alizadeh, K. Andersson, and O. Schelén, "Efficient Decentralized Data Storage Based on Public Blockchain and IPFS," in *2020 IEEE Asia-Pacific Conference on Computer Science and Data Engineering (CSDE)*. IEEE, 2020, pp. 1–8.

[15] X. Qi, "S-store: A scalable data store towards permissioned blockchain sharding," in *IEEE INFOCOM 2022-IEEE Conference on Computer Communications*. IEEE, 2022, pp. 1978–1987.

[16] B. Kim, W. Shin, D.-Y. Hwang, and K.-H. Kim, "Attribute-based access control (ABAC) with decentralized identifier in the Blockchain-based energy transaction platform," in *2021 International Conference on Information Networking (ICOIN)*. IEEE, 2021, pp. 845–848.

[17] X. Sun, F. R. Yu, P. Zhang, Z. Sun, W. Xie, and X. Peng, "A survey on zero-knowledge proof in blockchain," *IEEE Network*, vol. 35, no. 4, pp. 198–205, 2021.

[18] S. Ratnasamy, I. Stoica, and S. Shenker, "Routing algorithms for DHTs: Some open questions," in *International Workshop on Peer-to-Peer Systems*. Springer, 2002, pp. 45–52.

[19] V. Iancu and N. Ţăpuş, "Towards a Highly Available Model for Processing Service Requests Based on Distributed Hash Tables," *Mathematics*, vol. 10, no. 5, 2022.

[20] S. D. Gribble, E. A. Brewer, J. M. Hellerstein, and D. E. Culler, "Scalable, Distributed Data Structures for Internet Service Construction," The University of California at Berkeley, Tech. Rep., 2000.

[21] K. Matsuoka and T. Suzuki, "Blockchain and DHT Based Lookup System Aiming for Alternative DNS," in *2020 2nd International Conference on Computer Communication and the Internet (ICCCI)*. IEEE, 2020, pp. 98–105.

[22] A. Lazidis, K. Tsakos, and E. G. Petrakis, "Publish–subscribe approaches for the IoT and the cloud: Functional and performance evaluation of open-source systems," *Internet of Things*, vol. 19, pp. 1–17, 2022.

[23] Infura Inc, "INFURA," accessed on October 22, 2022. [Online]. Available: https://infura.io/