Verifiable Querying Framework for Multi-Blockchain Applications

1st Given Name Surname 2nd Given Name Surname 3rd Given Name Surname

***Abstract*—Effectively and securely retrieving data from various blockchain networks remains a critical challenge. We propose a novel framework that provides enhanced capabilities for authenticated data retrieval, empowering users and regulatory bodies. Our framework provides a scalable solution for information verification across diverse blockchain systems, enabling a variety of different types of actors to be integrated. Our framework allows metadata from various systems to be combined, also providing support for secure querying.**

***Index Terms*—Blockchain, Metadata, Data Query**

I. Introduction

Ensuring the security and authenticity of data has emerged as a critical issue in the domain of digital information. The widespread availability of digital data forms the basis for an extensive array of applications, spanning from simple data storage to sophisticated analytical tools. The key feature of these applications is not just the quantity of data, but also its accuracy. Ensuring the integrity of data is of the utmost importance, given that even minimal interference or manipulation can result in severe consequences for the insights drawn from this data.

Since data remains as the basis for many applications, it becomes a target for those who want to damage such services. Any manipulation of the data within the databases constitutes tampering, and if this is done by some malicious insiders or system administrators of an organisation, it would be hard to identify and would cause damage to its services. Traditional databases need additional security features to deal with such scenarios, and internal malicious agents can sometimes bypass these security features. Moreover, the traditional databases cannot ensure the state of the present data is the same as it was first inserted because it often does not store the historical changes that occur to the data [1].

Blockchain offers solutions for the problem of data tampering, and it is one of the reasons why it is used in different applications. Blockchain is a distributed ledger technology (DLT) that is immutable, tamper-resistant and stores data chronologically. According to [1], [2], blockchain as storage of data differs from traditional databases. Blockchain has decentralised storage, with heterogeneous data stored in an append-only structure on blocks with a pointer to the previous block and having inherent security mechanisms to provide immutability of data. The immutability of data enhances the data integrity and provides the complete history of a specific record. Unlike the traditional databases, blockchain does not support all the CRUD operations (Create, Read, Update, Delete) due to its immutability. Hence, the update will be another insertion, and the delete operation is unavailable.

Many companies are moving their applications to the blockchain [3] to improve trust and auditability. They use blockchain in applications that deal with IoT, asset management, health/medical, government audits and credit systems, to mention a few [2], [4]. The traditional business landscape has often faced issues related to transparency and the need for reliable information flow. Blockchain, with its unique characteristics, provides a solution to these challenges. The transparent and immutable nature of the blockchain ensures that every transaction or piece of information recorded is visible to all authorised participants in the network. Businesses adopt blockchain to leverage its transparency and traceability nature to enhance trust, streamline operations, and meet evolving customer expectations.

The present growth of these applications is limited within the organisations (more like private or consortium blockchain), and the development processes are often limited to the organisation's specific needs. Since blockchain, at its core, is a notarising agent/platform that brings trust, the applications may have to permit their stakeholders to access the blockchain to guarantee the trust. The increasing number of blockchain applications necessitates the need to query information between multiple blockchains.

Retrieving information from multiple blockchains can be quite challenging due to the diverse nature of blockchain platforms and the widespread use of private blockchains. Retrieving historical data involves traversing numerous blocks, leading to increased latency and reduced efficiency. Interoperability issues arise as different platforms often use distinct data structures and consensus mechanisms, making it difficult to retrieve data seamlessly. Private blockchains add an extra layer of complexity, limiting access to data and hampering comprehensive querying. Furthermore, the absence of standardized query interfaces and varying smart contract languages further complicates the process. The paper [1] identifies that the traditional blockchains only store data on a single topic, and there are no requirements for *join* operations. This is not the case while querying on multiple blockchains where there will be different data on the blockchains, and it will require at least a common field to perform the *join* operations. These challenges emphasize the need for a unified approach, such as a verifiable querying framework, to navigate the intricacies of querying across varying blockchain landscapes.

Our recent work [5] had proposed a taxonomy for data

management on the blockchain and this work uses concepts and ideas from it. The paper contributions are as follows.

- We propose a framework empowered by metadata to verify information from multiple blockchains based systems and it can easily scale to accommodate more entities.
- Our system, using the metadata processing empowers, the users or the regulatory bodies to gather raw data using authenticated data structures (ADS) and execute authenticated range queries.
- Our extensive experiments show that the system can efficiently verify data from multiple blockchains and perform authentication processes.

The rest of the paper is set as follows. Section II discusses a few related works. Section III provides the motivation, use case scenario and problem statement. Section IV details the design of the framework. Section V analyses the query process and integrity features. Section VI deal with the framework evaluation. Section VII considers a few use cases where the model can be implemented, and section VIII concludes the work.

II. Related Works

The paper [1] surveys query processing in blockchain-based systems and provides a comprehensive overview of the existing methods. The work identifies the increasing significance and importance of efficient data retrieval processes in blockchain while shedding light on the challenges in this process. It explains how blockchain is different from the traditional database in the way it stores and retrieves information. While elaborating on the query processes on the blockchain, the paper identifies and compares the query capabilities of the traditional databases and maps the same to see whether those features are available on the blockchain. The paper surveys existing research and systems developed for data retrieval on blockchain and categorises them based on their capabilities. The paper [6] examines different mechanisms for querying blockchains in both literature and practical applications. It analyses these mechanisms from three critical perspectives: efficiency, verifiability, and security. For each of these perspectives, the paper further categorizes and explains the strengths and weaknesses of each approach. Additionally, it provides several use cases to illustrate the challenges that blockchain queries must overcome.

The paper [7] analyses the possibility of searching or querying data in the blockchain, and it identifies that the native querying on the blockchain is very limited. The existing platforms only allowed queries either with block hash or transaction hash. Since it was no possible to query anything other than these limited options, they proposed an architecture that decouples the data available on each transaction, stores them on a secondary database (MongoDB), and queries them using traditional SQL. The framework (EtherQL) used the data from the Ethereum mainnet and, with the help of API, provides the possibility to query the data from the database. Due to the possible forks, the framework has at least ten blocks less than the mainnet. EtherQL claims to be the first to make this querying possible and performs well compared to the native methods. The paper [4] proposes a framework (FalconDB) based on the blockchain and uses ADS as a way to store and query data with multiple parties or nodes. It aimed to provide a collaborative platform where the nodes could work and share data without worrying about the presence of any malicious nodes. It uses ADS to store the data, and the digest generated from it is stored on the blockchain. The features of ADS coupled with blockchain give a new look to this model by moving the storage load to the server, and the lightweight clients need to have only hashes. FalconDB could execute various queries and can provide provenance. Though FalconDB requires more storage on the server-side (to capture the historical data), the overall performance was quite good. They identified that the query generation is fast, but the authentication is slow, so they separated both to achieve higher performance. The honest (full) nodes are incentivised, and false information would lead the nodes to lose their stakes.

The paper [8] emphasises the role of ADS as an efficient way to store data but points out that the gas expenses for such operations are high. They propose a framework called Gas Efficient Merkle Merge Tree (GEM^2 Tree) that optimises the gas expenses by making bulk transactions rather than individual insertions at the cost of more computations required. The paper [9] identifies that blockchain has weak semantics and insufficient operations to support authentic queries. Hence they propose a model that combines on-chain and off-chain data for queries. They add relational semantic features to the data and provide an interface for executing the queries. The paper explains the framework with a use-case of the donation system where the general information is on-chain while the private/sensitive information is off-chain. The on-chain data, based on the content, are made into some indices for providing faster query results. The framework provides SQL-like commands to create new tables, insert transactions and query the data using select commands. It supports the thin clients to query data with the API with ease. The paper [10] demonstrates the inefficiency of the native query features available and proposes a cloud-based architecture (VQL). The framework uses the data from the Ethereum maninet, decouples the possible information and stores them on a cloud DB (MongoDB). During regular intervals, the fingerprint of the cloud DB is generated. The miners with the available data generate the fingerprints and compare the same with that available on the cloud database. The miners will store the fingerprint in the blockchain if the information is matched. This process ensures the integrity of data present on the cloud DB. The framework uses Merkle Patricia Tree (MPT) to store the database digests generated, and thin clients can use them to verify the integrity of the query results. Comparing the framework with the Ethereum client (*Geth*) shows that the proposed framework performs better than the client on the data set considered.

III. Motivation and Problem Statement

A. Motivation

Contemporary data management is witnessing a transformative force in the form of the blockchain, which promises unparalleled transparency, security, and decentralisation. Businesses from various industries are increasingly embracing blockchain technology to establish trust, enhance traceability, and streamline operations. However, as more entities utilize blockchain, a critical challenge arises – how to efficiently retrieve information from multiple blockchain networks.

To explain further, let us consider a pharmaceutical use case. There are genuine and fake medicines available in the markets, though their presence may vary from developed to developing countries. The covid-19 pandemic has brought the urgency to track fake medicines [11]. Drug companies use various chemicals in their products and may have their own supply chains to monitor the process. Some of these chemicals need to be regulated and controlled by government agencies. If the regulatory bodies want to get details of the use of these chemicals in the medicines produced, they may have to query various drug supply chains. Pharmaceutical products undergo various stages in the production cycle, and different entities may participate in the process. Consider the example of blockchain-enabled pharmaceutical production [12], [13] and a logistic company that uses a blockchain platform [14]. Tracing and tracking drugs to avoid counterfeit products require querying on multiple blockchains. Figure 1 gives this scenario. If the consumers want to know the drug they use is authentic or not, they need to find out whether the drug is produced by the original company and the logistics company had delivered it to the right pharmacy/agency. Gathering the data from the these entities and analysing them would guarantee that the drug is authentic. If the query does not return any data, then it would indicate that the drug is fake/counterfeit. While individuals can get information of a single product, the auditors or the regulatory bodies can perform range queries over multiple blockchains.

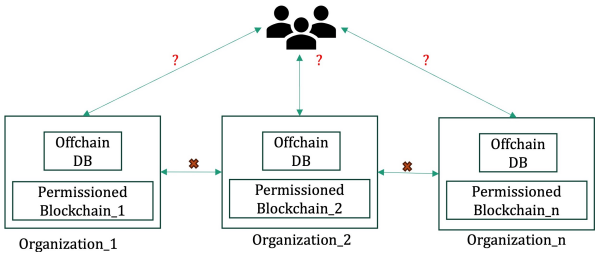

Fig. 1. Current Scenario

B. Problem Statement

Blockchain technology promises transparency and security, but accessing and querying distributed data is complex. Each blockchain operates as an independent entity with unique data structures, making it challenging to execute coherent queries across multiple platforms. Private blockchains restrict access to a defined set of participants, which complicates the querying process. These limitations hinder the free flow of information and demand innovative solutions for secure and authenticated data retrieval.

In order to tackle the challenges related to querying information from distributed blockchain networks, there is a strong need for a verifiable querying framework. Such a framework should allow regulatory bodies and users to execute authenticated queries. This will help in building trust and scalability in blockchain systems. This paper proposes and elaborates on such a framework to provide a comprehensive solution to the complexities involved in querying systems based on blockchain.

IV. Verifiable Query Framework

In this section, we shall detail our proposed framework, designed to cater for the specific needs of the use case. There can be several organisations in the use case, and each organisation has several entities. They are data owners, validators (blockchain is part of it), cloud ADS (CADS), query verifiers, query miners and query layer. Figure 2 shows the framework diagram.

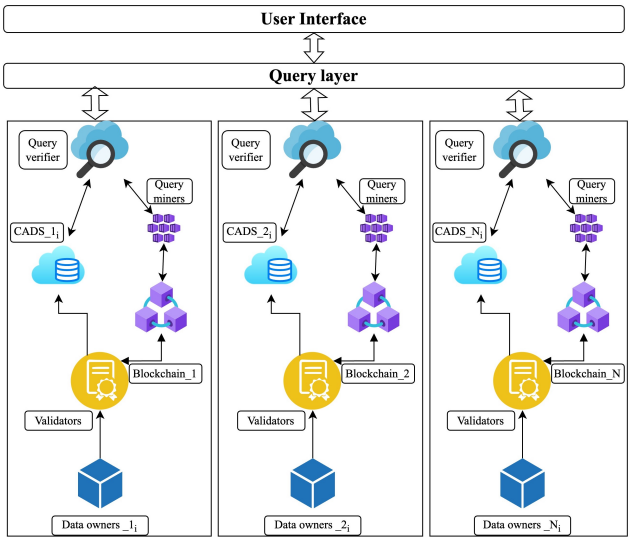

Fig. 2. System Architecture

A. Data Owners

They are the owners of data and decide what data should be made available through blockchain. They communicate directly and only to validators. It has a one-way communication with validators as they only send the data and not read it from the validators. In our scenario, the data owners are the pharmaceutical, logistics and retail pharmacy companies who want to make some of their data available for public access.

B. Validators

They are the entities that receive data from data owners and then push it to the blockchain. Before explaining further, let us discuss the consensus algorithm used to understand

the validators. The consensus algorithm considered is proof of authority (PoA). It was proposed by Gavin Wood, the former CTO of Ethereum, on 2017, and it came as a part of the permissioned environment of Ethereum. PoA requires fewer message exchanges which increases the performance. It was proposed to solve one of blockchain's biggest problems: scalability. PoA is a reputation-based consensus protocol which is efficient and can tolerate up to 50% malicious nodes. Proof of Work (PoW) consensus stakes its computational power, and Proof of Stake (PoS) consensus stakes coins in the consensus process. In contrast, PoA stakes its reputation, i.e., PoA consensus accepts node identity as a means to secure its blockchain. In PoA, a few validators are arbitrarily selected as trustworthy entities to perform the transactions. Since PoA relies on a few validators for consensus, the process is highly scalable. PoA permits organisations to maintain their privacy while providing a private blockchain's features. It is suitable for permissioned or private blockchains and does not require native coins to run the systems. PoA does not provide complete decentralisation but improves the performance. PoA is an effective and acceptable solution to various applications, including supply chain [15], [16]. Only the validators push the data to the blockchain and CADS. The blockchain access is limited to validators and to query miners. Validators are the full nodes that has the complete access to data (actual data and blockchain data). The process of selecting the validators in PoA is not under this work's scope.

In our model, the blockchain does not store the actual data but only hash of the data. Upon receiving the data *d*, the validators generate hash *h(d)* and store it on the blockchain. After storing the data it would retrieve the following information from the blockchain: transaction data (*TD*), transaction hash (*TH*), block height (*BN*) and block hash (*BH*). The actual data *d*, and *h(d)* are stored in the local database of the validator. For a given database D with $d \in D$, validators send *d*, *h(d)*, *TD*, *TH*, *BN* and *BH* to CADS, where they will be stored. Algorithm 1 shows the process of storing data on blockchain and CADS.

Algorithm 1 Validator process

Input: data *d*
Output: Storing information on Blockchain and CADS
1: **for** every data $d \in D$ **do**
2: Generate *h(d)* and push to Blockchain
3: Retrieve *BD* ← *TD*, *TH*, *BN* & *BH*
4: Store *d* and *h(d)* on internal database
5: Send *d*, *h(d)*, and *BD* to CADS
6: **end for**

C. Cloud ADS (CADS)

ADS have been used to improve the efficiency of the applications by reducing the data retrieval process on the source of data [17]. A few researches [8]–[10] have made use of ADS on blockchain and [4] provide a detailed description of the construction of ADS for a collaborative database platform. By its design, ADS supports clients to query and verify the data on the database and provides proof of the data's correctness. In case of any doubt about the data from such databases, the clients can check the soundness and correctness by retrieving its proof from the source (in our case, the blockchain). The ADS is designed in such a way that there are only insert and read functions. The update or delete functions are unavailable in the database. If there are any corrections, it would be another insertion operation. In this way, the data immutability is achieved on ADS. The purpose of placing the ADS on the Cloud is that they can always be available and can leverage the potential of the Cloud to have better performance.

Apart from the function of storing the data from the validators, CADS has the following function as well. It can return results for a collection of queries Q performed on its database D. The function to query $q \in Q$ searches for records $r \in D$. If there are results that satisfy q, then it will return result $R = q(D)$ and a proof π (in our case, the proof is the hash of the specific data, *h(r)*). Due to the inherent properties of hashing mechanism, no two hashes will be the same, i.e., $h(r_i)) \neq h(r_j))$ where $r_i \neq r_j$. The data returned has different format for simple verification and blockchain verification which will be discussed in section V-A2.

CADS stores the information sent only by the validators, and only verifiers can retrieve that information from the CADS. The authentication function and the security features of the CADS will be discussed in section V.

D. Query Verifiers

Query verifiers do not store data but can query data from CADS and retrieve its proof from the blockchain. They communicate with the query layer and provide the information requested by them. Upon receiving the query request from the query layer, the verifiers send the request to the CADS, which returns the results along with transaction details. Depending on the kind of request, it will further send the request for blockchain proofs. The results and their proofs are then communicated with the query layer.

E. Query Miners

Validators have the responsibility to write data to the blockchain and not retrieve it during the query process. Query miners are the entities that has the role of reading the transaction details from blockchain during the query process. In this way, validators are isolated from other entities. In the context of blockchain verification, the query verifiers send the transaction details from CADS to query miners who would retrieve the verification objects from the blockchain and return them to query verifiers. Query miners would communicate only with the query verifiers.

F. Query Layer

It is responsible for collecting query requests and processing them. The query layer gathers information from multiple organisations and returns query result to users. It uses the metadata information from the organisation for the query

process. Every organisation must send its metadata and query structure to the query layer, which has a metadata engine instructing where to search for data. The metadata engine helps the query layer to execute parallel searches across multiple systems, optimises the query process and aggregates the results. In some situations, organisations can change their schema; they can update the metadata with the query layer, ensuring that the query process returns the requested details. The metadata engine plays a crucial role in making the query layer easily scalable to add additional organisations dynamically and manages the changes in schema from these organisations. Before updating the metadata, it validates the data to avoid errors. Figure 3 shows the query layer process.

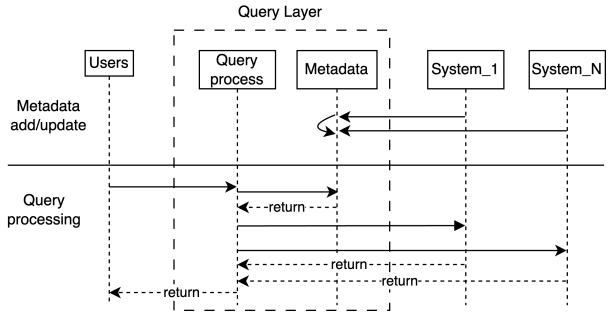

Fig. 3. Query Layer Process

V. Query Process and Integrity Analysis

In dealing with queries, one of the fundamental requirements is ensuring the soundness and correctness of the data; that is, the query returns all the data that satisfies the condition, and no data has been modified after it had been inserted into the database. Its authenticity is assured if the data is consistent with the blockchain. The section elaborates on the query process and integrity analysis.

A. Query Process

In the proposed framework, data retrieval is through the query layer, which is empowered by the metadata engine. This section explains the role of metadata in the query process and the query process itself.

1) Metadata Processing: Metadata is a fundamental element in the field of data management, providing crucial insights into the nature of data itself. It plays a pivotal role in improving the query process by illuminating the underlying structure, relationships, and attributes of the data, which facilitates precise retrieval and manipulation. Metadata guides and orchestrates the query process by mapping attributes to their corresponding locations, enabling swift and accurate data retrieval. It streamlines data management and enhances the overall effectiveness of the query process. The Metadata engine has the following functions: it manages the metadata, constantly waits for the new or updated metadata from the entities, and directs the query process to send the requests to the correct locations. Upon receiving the metadata, it checks whether the received metadata has the correct structure. If it has the proper structure, it will be added or updated, else it will be discarded. In every query, the metadata engine is called upon, which guides where to search for the data.

2) Query Verification: A query is constructed from the metadata engine and sent to the respective location(s). A query can either be a simple verification or a blockchain verification. In FalconDB [4] a query and its verification is separated, as the query can be faster while the verification process from the blockchain tends to be slower. Another reason is the amount of gas fees required for executing transaction. Though the data retrieval process does not change the blockchain state, it still requires a gas fee to incentivize the miners. Hence, separating the blockchain verification from simple verification (which returns results completely from CADS) allows the users to get the information from the systems without paying the gas fee. Figure 4 shows the sequence diagram illustrating how the query gets executed and retrieves information in a single system/ organisation.

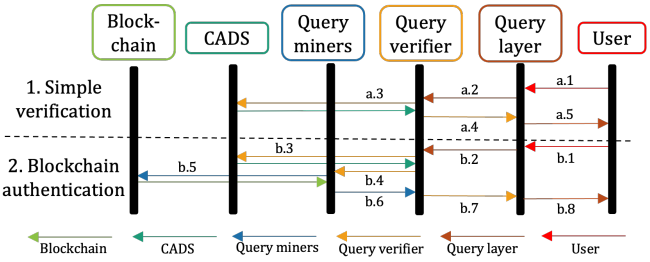

Fig. 4. Verification Process

The simple verification process tries to leverage the features of CADS in providing fast data access and its verifiability. Depending on the metadata information, the query layer can perform various queries. In simple verification process, the request from the user (a.1) reaches the query layer, which, using the metadata engine, fires the query to the respective systems (a.2). In the system, the first entry point is to the query verifier, which diverts the query to CADS and gets its results (a.3). If any results that satisfies the query, it will return them. The query verifier then returns the results to the query layer (a.4) and then to the user (a.5).

In the blockchain verification process, the request from the user (b.1) reaches the query layer, which, using the metadata engine, fires the query to the respective systems (b.2), which diverts the query to CADS and gets its results (b.3). If any results satisfy the query, it will return the results and the verification object. The query verifier sends the verification object to the query miner (b.4), which retrieves the corresponding verification object from the blockchain (b.5) and returns it to the query verifier (b.6). The query verifier then compares and aggregates the results and sends them to the query layer (b.7). Finally, the query layer provides the results to the user (b.8). The initial processes in both simple and blockchain verification are the same. To reduce the complexity of the overall process, a flag is set to distinguish between the two processes. Depending on the condition, the system process query differently.

The query process can be further elaborated as follows. The user wants to get some information; It may constitute a simple verification (SimVer) or blockchain verification (BlockVer). The request may return normal result *R* or verified result *VR*. As the query layer (*QLayer*) receives the request, it checks whether the user can access the information. If the user has permission, then the *QLayer*, using the metadata engine, will check where to search for the data. Let us consider that there are a total of *n* possible locations (L_n) in which the required information may be present in *i* locations (L_i). Metadata engine then would formulate the query and the *QLayer* would send the request to L_i in parallel. Each location will have a query verifier ($QVer_i$), $CADS_i$, and query miner ($QMiner_i$). Upon receiving the request from the *QLayer*, $QVer_i$ will send it to $CADS_i$, which retrieves the information if it exists. In the case of SimVer, $CADS_i$ retrieve the records R_i, and its proof π_i (hash of the record r_i) and return to the ($QVer_i$). If the query is for *VR*, then $CADS_i$ retrieves R_i, π_i and $CADS_i$ verification object VO_{CADS_i} (which is transaction hash TH_{CADS_i} and transaction data TD_{CADS_i}). Upon receiving R_i from $CADS_i$, $QVer_i$ send them directly to *QLayer*, which verifies the R_i and π_i and returns them to the user. If $QVer_i$ received *VR* from $CADS_i$, then $QVer_i$ extracts TH_{CADS_i} from VO_{CADS_i} and send to the $QMiner_i$, which retrieve the corresponding data from the blockchain and constitutes the blockchain verification object VO_{B_i} which is the data stored on blockchain (TD_{B_i}) for the respective transaction hashes. Having the VO_{B_i} , $QMiner_i$ send them to the $QVer_i$ which compares VO_{CADS_i} and VO_{B_i} and produces *VR*. $QVer_i$ will send the *VR* to the *QLayer* which publish the same to the requested user. The algorithm 2 shows the query process.

Algorithm 2 Algorithm for Query Process

Input: Query q
Output: R or VR

1: User send q to *QLayer*
2: **if** permission == True **then**
3: *QLayer* checks the metadata engine and returns L_i
4: **for** each L_i **do**
5: $QVer_i$ send q to $CADS_i$
6: $CADS_i$ execute q and get R_i
7: **if** $R_i > 0$ **then**
8: **if** SimVer == True **then**
9: R_i,$\pi_i \leftarrow r_i$, $h(r_i) \in D_i$
10: **return** R_i,π_i to $QVer_i$
11: **else if** BlockVer == True **then**
12: $VO_{CADS_i} \leftarrow TH_{CADS_i}$,$TD_{CADS_i}$
13: R_i,$\pi_i \leftarrow r_i$, $h(r_i) \in D_i$
14: **return** R_i,π_i, VO_{CADS_i} to $QVer_i$
15: **end if**
16: **else**
17: **return** $\emptyset$
18: **end if**
19: $QVer_i$ checks whether it is a SimVer or BlockVer
20: **if** SimVer == True **then**
21: $QVer_i$ send R_i,π_i to *QLayer*
22: **else if** BlockVer == True **then**
23: send VO_{CADS_i} to $QMiner_i$
24: **for** each $TH_{CADS_i} \in VO_{CADS_i}$ **do**
25: $VO_{B_i} \leftarrow TD_{B_i}$
26: **end for**
27: **return** VO_{B_i} to $QVer_i$
28: **end if**
29: $QVer_i$ compare R_i,π_i, VO_{CADS_i}, VO_{B_i} and send VR_i to *QLayer*
30: **end for**
31: Repeat in parallel steps 4 to 30 in L_i
32: $R \leftarrow (R_1,....,R_i)$ or $VR \leftarrow (VR_1,....,VR_i)$
33: *QLayer* send *R* or *VR* to User
34: **else**
35: **return** no access
36: **end if**

B. Integrity Analysis

1) Integrity of the validator: In the proposed model, the validators have the role of writing data to the blockchain and the CADS. After pushing the data to the blockchain, they retrieve on-chain information and, along with the original data, write them to the CADS to support the data retrieval process. The validators cannot read, update or delete data on CADS but only write the information to it. That means data flow between validators and CADS is one-directional. Though only the hash of the data is stored on the blockchain, the validators would maintain an internal database to store the actual data for verification. The validators will not need to access the internal database until a situation arises when the data on CADS does not match with the data on the full node. The consensus protocol considered here is the proof of authority (PoA), and its basic assumption is that the validators are trustworthy. The model uses permissioned blockchains, and the participating entities can decide the incentives for the validators depending on their agreement.

2) Integrity of the CADS: During the query process, the verifiers get the information from the CADS and it is important to ensure its integrity. By its design, only read and write operations are permitted on CADS. The write permissions are only with validators and since the validators are trustworthy, the data they write on CADS will be reliable. The verifier nodes can only read the data from CADS and no other operations are possible for them. In practical deployment, there can be multiple CADS that are distributed to reduce latency and improve load balancing. The aspect of immutability is brought into CADS by removing the update or delete operations.

VI. Framework Evaluation

We have created a prototype to evaluate the practicality and effectiveness of our framework. This section will detail its prototype implementation and evaluations.

A. Experimental Setup

To demonstrate our framework, we have used a pharmaceutical supply chain scenario with 4 organizations: a manufactur-

ing company, a logistic company and two pharmaceutical retail chains. Each entity manages its ADS, blockchains, validators and verifiers. The users, using API, can connect to the query layer and request various information that suits their needs or retrieve the complete product history. One of the assumptions is that the organizations have standardised naming conventions for their databases and provide the correct schema to the metadata engine. Every entity of the proposed framework implements asymmetric key cryptography and assumes that the keys are shared between the entities in a safe manner. The prototype uses 4 servers symbolizing 4 different organizations. Each server is empowered with Intel Xeon E3-1220, having 16GB of RAM running Ubuntu 18.04. Each organization manages their blockchain built on Ethereum, and the ADS is built on MongoDB. The prototype uses a pub-sub mechanism to communicate between the entity, and communications between each entity are secured using asymmetric key cryptography.

B. Performance Evaluation

To evaluate the system, we have considered requesting several records from 4 organisations varying between 10,000 to 40,000 and assessing their latency. The evaluation is performed for both simple verification and blockchain verification. The readings were taken multiple times, and the averaged values were considered while discussing the results. In the case of simple verification, the requests return the desired results within a time frame between 0 to 3 seconds. Here, the records are returned only from ADS, not involving the blockchain. As the number of organisations increases from 1 to 4, the latency increases. This may be due to the processing time taken at the user end to receive and process the information from multiple entities. Though the increase in the number of records does not affect latency significantly, the increase in the number of organisations increases the latency visibly, as seen in figure 5.

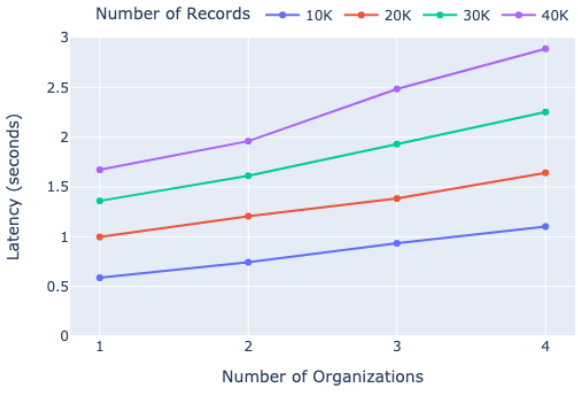

Fig. 5. Simple Verification Latency

When similar evaluations for blockchain verification are performed, the latency increases significantly. As discussed earlier, this process involves gathering information from ADS and that of the underlying blockchain. Though the total time looks similar for the same number of records, there is a slight difference of 0 to 1 second. The processes in blockchain consume the major part of the data retrieval process. As seen in the figure 6, it is clear that the majority of the total time required to retrieve information comes from the retrieval processes in the blockchain. It can also be noticed that the time required for retrieving information from the blockchain remains constant and is not affected by the increasing number of organisations.

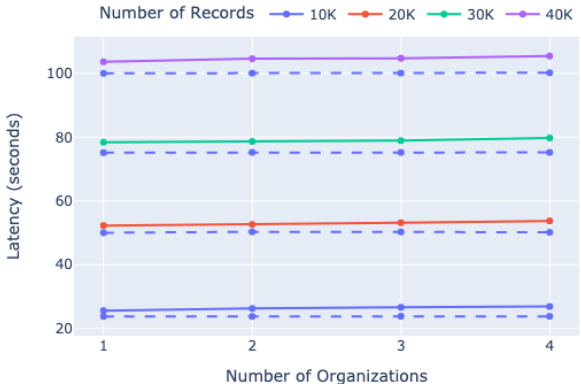

Fig. 6. Blockchain Verification Latency

VII. Further Use cases

Our framework has a versatile design which enables it to be applied beyond its primary use case, allowing for the expansion of its functionality and potential usability. This section highlights additional scenarios where our framework is useful, demonstrating its adaptability to a diverse set of applications.

A. Food Supply chain

In a food supply chain, various products (e.g. wine, meat, fruits, etc) can use blockchain to track the origin and provenance of a product. This is essential to enable better market positioning of a product and also to comply with regulatory requirements. There may be instances that require a product to be made from two or more other products, which may have separate supply chains. Suppose the end-user (maybe a customer, shopkeeper, or regulatory body) would like to know the quality of the raw materials used in the production. In that case, they may have to query the origin of the product being considered from multiple blockchains.

B. Healthcare

Research [18] demonstrates the use of blockchain in the health sector that combines hospitals and pharmacies. A patient's medication information may be shared with pharmacies, even though hospitals and pharmacies may have different blockchains – being part of different organisations, e.g. a private hospital and pharmacy chain. The on-chain data can be medical details, while personal details of a patient must be kept on the off-chain databases (to comply with privacy legislation, e.g. GDPR). Suppose a regulatory body or private insurance company wants to get the details of a patient and their medication history to investigate an allegation or process a

claim. Situations of this sort necessitate gathering information from multiple blockchains.

C. Financial Management

Blockchain technology has strong ties with the finance sector since it traces its origins to cryptocurrency. Blockchain provides the features of immutability and transparency, which are very important to financial systems. There may arise situations where companies use blockchain for managing their finances and trading. If a conflict arises between the companies that use blockchain in their business, the regulatory bodies or arbitrators may have to query data from the blockchain to gather the details.

D. Registries

Registries maintain various information like land, buildings, vehicles, and other information regarding individuals. Research [19] proposes using blockchain to store the details of these registers, which will act as a trusted entity. Consider a situation when a bank would like to find information on the land and vehicle details of an individual or retrieve information for a mortgage or loan application. In such a situation, a need arises to query different registry blockchains that manage the related data.

VIII. Conclusion

In this paper, we propose a novel framework capable of retrieving information from multiple blockchains. Our system is a versatile solution that combines a robust metadata processing framework with secure querying methods. Robust data retrieval mechanisms are crucial as blockchain technology advances. Our work not only empowers users and regulatory bodies but also lays the foundation for a more resilient and adaptable blockchain ecosystem. Our metadata-driven framework enables scalability, accommodating a broader range of entities and ensuring relevance in the evolving blockchain landscape. In future, we plan to extend and deploy our framework to support applications in multilayer blockchains.

IX. Acknowledgements

Supported in part by the Engineering and Physical Sciences Research Council "Digital Economy" programme: EP/V042521/1 and EP/V042017/1.

References

[1] D. Przytarski, C. Stach, C. Gritti, and B. Mitschang, "Query processing in blockchain systems: Current state and future challenges," *Future Internet*, vol. 14, no. 1, 2021.

[2] H.-Y. Paik, X. Xu, H. M. N. D. Bandara, S. U. Lee, and S. K. Lo, "Analysis of data management in blockchain-based systems: From architecture to governance," *IEEE Access*, vol. 7, pp. 186 091–186 107, 2019.

[3] M. LIM, "81 of top 100 companies use blockchain technology, blockdata research shows," 24/09/2021. [Online]. Available: https://forkast.news/81-of-top-100-companies-use-blockchain-technology-blockdata/

[4] Y. Peng, M. Du, F. Li, R. Cheng, and D. Song, "Falcondb: Blockchain-based collaborative database," in *In Proceedings of the 2020 ACM SIGMOD International Conference on Management of Data*. ACM, 2020, Conference Proceedings, pp. 637–652.

[5] S. Wilson, K. Adu-Duodu, Y. Li, E. Solaiman, O. Rana, S. Dustdar, and R. Ranjan, "Data management challenges in blockchain based applications," *IEEE Internet Computing*, 2024 (Accepted for Publication).

[6] Q. Zhang, Y. He, R. Lai, Z. Hou, and G. Zhao, "A survey on the efficiency, reliability, and security of data query in blockchain systems," *Future Generation Computer Systems*, vol. 145, pp. 303–320, 2023.

[7] Y. Li, K. Zheng, Y. Yan, Q. Liu, and X. Zhou, *EtherQL: A Query Layer for Blockchain System*, ser. Lecture Notes in Computer Science. Cham: Springer, 2017, vol. 10178, pp. 556–567.

[8] C. Zhang, C. Xu, J. Xu, Y. Tang, and B. Choi, "Gem$\hat{2}$-tree: A gas-efficient structure for authenticated range queries in blockchain," in *IEEE 35th International Conference on Data Engineering (ICDE)*, 2019, Conference Proceedings, pp. 842–853.

[9] Y. Zhu, Z. Zhang, C. Jin, A. Zhou, and Y. Yan, "Sebdb: Semantics empowered blockchain database," in *IEEE 35th International Conference on Data Engineering (ICDE)*. IEEE, 2019, Conference Proceedings, pp. 1820–1831.

[10] H. Wu, Z. Peng, S. Guo, Y. Yang, and B. Xiao, "Vql: Efficient and verifiable cloud query services for blockchain systems," *IEEE Transactions on Parallel and Distributed Systems*, vol. 33, no. 6, pp. 1393–1406, 2022.

[11] "Unodc develops a holistic strategy to combat crime related to falsified medical products in west and central africa," 28/05/2021. [Online]. Available: https://www.unodc.org/westandcentralafrica/en/2021-05-28-falsified-meds-west-africa-ccpcj.html

[12] M. Kumari, M. Gupta, and C. Ved, "Blockchain in pharmaceutical sector," *Applications of blockchain in healthcare*, pp. 199–220, 2021.

[13] R. Raj, N. Rai, and S. Agarwal, "Anticounterfeiting in pharmaceutical supply chain by establishing proof of ownership," in *2019 IEEE Region 10 Conference (TENCON)*. IEEE, 2019, Conference Proceedings, pp. 1572–1577.

[14] "Dhl and accenture unlock the power of blockchain in logistics," 03/12/2018. [Online]. Available: https://www.dhl.com/global-en/home/press/press-archive/2018/dhl-and-accenture-unlock-the-power-of-blockchain-in-logistics.html

[15] P. B. Honnavalli, A. S. Cholin, A. Pai, A. D. Anekal, and A. D. Anekal, "A study on recent trends of consensus algorithms for private blockchain network," in *Blockchain and Applications: 2nd International Congress*. Springer, 2020, pp. 31–41.

[16] "Proof of authority explained," 08/12/2018. [Online]. Available: https://academy.binance.com/en/articles/proof-of-authority-explained

[17] R. Tamassia, *Authenticated Data Structures*. Berlin, Heidelberg: Springer, 2003, vol. 2832, pp. 2–5.

[18] A. P. Singh, N. R. Pradhan, A. K. Luhach, S. Agnihotri, N. Z. Jhanjhi, S. Verma, U. Ghosh, D. S. Roy *et al.*, "A novel patient-centric architectural framework for blockchain-enabled healthcare applications," *IEEE Transactions on Industrial Informatics*, vol. 17, no. 8, pp. 5779–5789, 2020.

[19] S. K. Panda, G. B. Mohammad, S. Nandan Mohanty, and S. Sahoo, "Smart contract-based land registry system to reduce frauds and time delay," *Security and Privacy*, vol. 4, no. 5, p. e172, 2021.