DWH-DIM: A Blockchain Based Decentralized Integrity Verification Model for Data Warehouses

Jeroen Bergers*, Zeshun Shi*, Ken Korsmit†, and Zhiming Zhao*
*Informatics Institute, University of Amsterdam, Amsterdam, 1098 XH, the Netherlands
†Spatial Eye B.V., Culemborg, 4105 JH, the Netherlands
Email: jeroen.bergers@hetnet.nl, z.shi2@uva.nl, ken.korsmit@spatial-eye.com, z.zhao@uva.nl

***Abstract*—Data manipulation is often considered a serious problem in industrial applications as data tampering can lead to inaccurate financial reporting or even a corporate security crisis. A correct representation of company data is essential for the companies' core business processes and is requested by governments and investors. However, the current solution, third-party auditing, is expensive and cannot be fully trusted. In this paper, we present the Data Warehouse Decentralized Integrity Model (DWH-DIM) to validate the integrity of the data warehouse and replace the current process. To address the challenge that the existing distributed integrity verification models cannot handle GDPR and are limited by scalability, our model uses a distributed file system to store attributes that can be used for the integrity verification task. The blockchain further confirms the authenticity of the files. Based on the proposed model, we present a detailed implementation of the DWH-DIM tool. The implementation is tested with a use case and several benchmarks. Experimental results demonstrate that our proposed model is feasible and meets the requirement for certificate warehouse data.**

I. Introduction

Information and data are the most valuable resources of industrial companies and need to be handled properly to maximize their potential [1]. One frequently used data management solution is the data warehouse. The data warehouse is often a core technology within business intelligence; it is used for day-to-day decision making and can contain millions, or even billions of records [2]. The difference between a data warehouse and a conventional database is that the former is subject-oriented, non-volatile, integrated within the company, and time-variant [3]. One of the core components of the data warehouse is the extract, transform, load (ETL) system. It is responsible for extracting data from several sources, the customization and transposing of the data, and finally making the data available to users. Normally the ETL system cannot change the data, but the data stored in the source database can still be edited by internal or external actors directly in the source systems. This results in an existing likelihood of people tampering with the data to influence the outcome of the system. Current solutions to this problem are through internal or external audits. Limitations also exist in this approach [4]. For example, when using external auditing, there is always the probability of dishonesty.

Blockchain technology could be one of the promising solutions to this problem. In general, blockchain is a decentralized ledger technology with a "chained blocks" data structure; every block header includes the root of a Merkle tree, a timestamp, and a hash of the previous block [5]. When data inside the previous block is changed, the new hash is never referenced to another block, resulting in rejection by the network. Consequently, once data is committed to the chain, it is immutable, decentralized, traceable, and distributed. Combining these properties with a data warehouse can achieve the following features:

- Immutability of the data: Data stored in a blockchain is immutable; combining the blockchain with a data warehouse can create a tamper-proof data warehouse.
- Immutable access logging system: The immutability properties of the blockchain can be used to log all the read/write actions to create an immutable logging system.
- Automated access control: Various techniques have been designed to use the blockchain as a decentralized, scalable, and automated access control system [6]. In combination with the warehouse, this can lead to the right persons having the right access.

Correct implementation of blockchain within the data warehouse can verify the integrity of the data without the users having to worry about it, as the blockchain is tamper-proof [7]. This can replace the current process of expensive and not fully trusted third-party audit process by the blockchain, saving time and costs [8], [9]. The potentials of blockchain as a data management system has been investigated in recent studies [10], [11], [7], [4], [12]. This paper builds on top of the previous research and leads to the creation of the Data Warehouse Decentralized Integrity Model (DWH-DIM). Unlike previous models, DWH-DIM is designed to handle large amounts of data stored in a classical data warehouse while retaining integrity. DWH-DIM also prevents such malicious activities, i.e., not only third parties can behave maliciously, but also users within the organization.

A. Contributions

The contribution of the research is a novel decentralized model (DWH-DIM) to verify the integrity in an untrusted environment. Based on the proposed design, we introduce a second component called the Data Warehouse Decentralized Integrity Verification Tool (DWH-DIV). This tool is an addition to the traditional warehouse to make it decentralized. The main contributions of this paper can be summarized as follows:

- A decentralized model (DWH-DIM) using blockchain to verify the integrity of a data warehouse in an untrusted environment, where the degree of traceability can be customized based on the importance of the data.
- A methodology to verify data integrity via blockchain when privacy regulations require the data to be deleted.
- The DWH-DIV tool, a flexible addition on top of the traditional data warehouse that combines the model, algorithms, and protocols into a single tool with a user interface.
- The proposed approach is evaluated on a real data scenario. The results demonstrate that DWH-DIM is able to detect integrity-related issues and meet the use case's scalability criteria.

The rest of this paper is organized as follows: In section II, the requirements are established and DWH-DIM system components and workflow are described. Section III introduces the implementation details of DWH-DIM. In section IV, our model is tested to investigate whether it meets the requirements. Section V briefly reviews the related work. Finally, in section VI, the work is concluded and future work is presented.

II. DHW-DIM Overview

In this section, we will explain the system overview of DWH-DIM. The section is started with an analysis of the requirements. Next, the system components and workflow are described in detail.

A. Requirement Analysis

The requirements are made in compliance with *Spatial Eye*, a company that creates a data warehouse application. Despite this, the requirements are suitable to function with other warehouse systems. The input of DWH-DIM is the existing warehouse product. DWH-DIM must work with most of the existing data warehouse products without changing the core component of the data warehouse product, making it easy to implement the warehouse in existing warehouse products. The only requirement to the data warehouse is that data is uploaded in hourly, daily, or weekly batches. The output of DWH-DIM is a proof that the data stored in the warehouse is authentic. There will be three kinds of users interacting with the system, all needing a different user interface and different requirements.

1) Primary DWH User: The user in the presented use case is often a geographic information system (GIS) analyst working with a geographic information system. The GIS extracts data from the data warehouse. The primary user does not want their workflow to be changed. They expect the system to work fully automatic and do not want to perform different actions to validate the integrity.

2) Internal Auditor: The internal auditor makes the financial reports. The internal auditor expects that the asset data retrieved from the DWH is accurate. They do not have much IT knowledge, so the auditing process should be easy to set up. They expect DWH-DIM to be precise, so mistakes or wrongdoings can be identified early, accurately, and be recovered. He knows what assets are essential for good bookkeeping.

3) External Auditor: The external auditor can be an investor, external auditing company, or government agency. They expect a company to have perfect bookkeeping. They like to see a certificate to show that everything is correct. For the external auditor, it is essential that the system is explainable, as they need to trust the certificate.

B. System Components

DWH-DIM consists of four main components: The data warehouse (DWH), the Data Warehouse Integrity Verification tool (DWH-DIV), the distributed file system (DFS), and the blockchain. The primary connections are shown in Fig. 1.

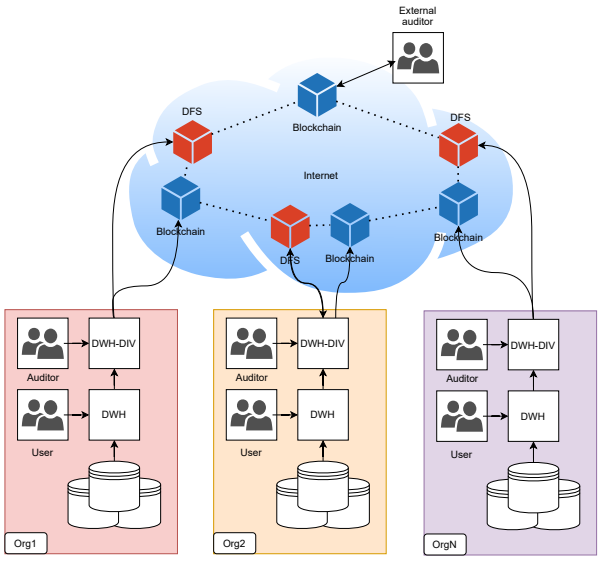


Fig. 1. Consortium overview of DWH-DIM.

1) Data warehouse: The data warehouse is the data source of the system. It takes care of the retrieval of data stored in the various connected databases. Data is added to the source databases via the warehouse in batches.

2) DWH-DIV: This tool is responsible for the communication between the system and users. It takes care of the encryption, authentication, and verification processes. It is the interface between the users, database managers, and the integrity verification system.

3) Distributed file system: The distributed file system takes care of the data storage because of the limited storage capabilities of blockchain. The file system should permanently store files and have a precise storage location, preferably based on file content. Options are the InterPlanetary File system (IPFS), Hadoop distributed file system, and blockchain-based file systems like StorJ or bigchanDB.

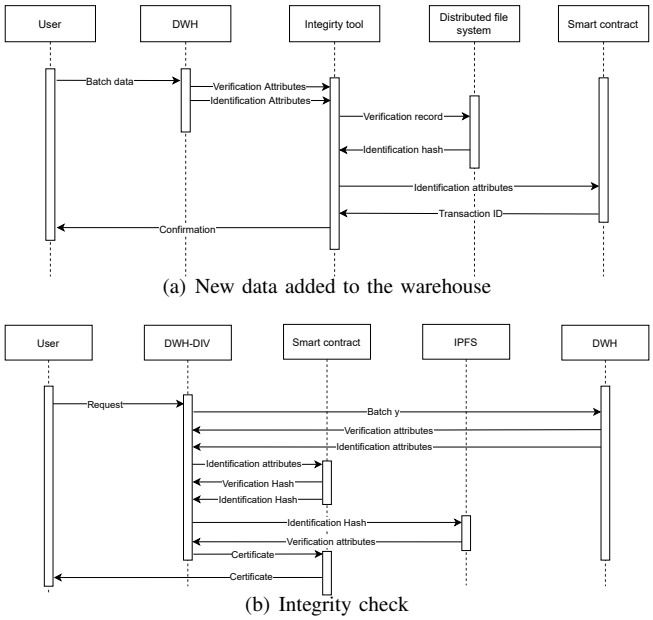
4) Blockchain: The final component is the blockchain. The blockchain is the connection between different companies. The immutability and decentralized properties are being exploited to store and share integrity verification hashes and

identification attributes. If more parties join the blockchain network, the degree of protection increases as the system is more decentralized.

C. DWH-DIM Workflow

A motivating scenario based on a utility company is used to explain the system for easy understanding of the system. However, the possibility of use is not limited to utility and telecom companies. The scenario has been split into three steps: 1) adding new data to the data warehouse, 2) the internal integrity check, and 3) the external integrity check.

1) Adding Data: A new internet cable has been put into the ground. This new cable is sent to the network architect that adds this together with 300 other changes in the electricity network to the geographic information system. The geographic information system uploads the network changes as batch number 1200 to the data warehouse at the end of the day. The DWH-DIV tool builds on top of the warehouse extracts identification and verification attributes from batch 1200. The identification attributes s_i are keywords to identify the batch, in this case this can be: $s_i = \{TableId : LowVoltage,\ BatchId : 1200,\ Timestamp : 2017-12-09, ... : ...\}$. The verification attributes s_v are to verify the integrity of the warehouse batch and can include the vertical column hashes. The verification attributes can vary between data warehouse tables based on the importance of the data. Still, they should at least consist of an SHA-256 hash H_r of the content of the batch table and a description of what other attributes are included in s_v. It is recommended to have at least one SHA-256 hash for each column in s_v. Both s_i and s_v are added together into a verification record r ($r = s_v + s_i$). The verification record r is encrypted using a generated key P_s, the returned cyphertext CT is stored on the distributed file system. A location hash is returned from the distributed file system; this location hash H_L is stored together with integrity verification hash H_r and the identification attributes s_i by broadcasting a transaction to the blockchain network. The private key P_s is stored on a private ledger database with the identification attributes. An overview of the actions are performed can be seen in Fig. 2(a).



(a) New data added to the warehouse

(b) Integrity check

Fig. 2. The process diagram of the DWH-DIM workflow.

2) Internal Integrity Check: When an internal auditor requests an audit, the DWH-DIV interface is used. An overview of these steps can be seen in Fig. 2(b). First, the auditor selects what database to validate. For this example, the low voltage power cables are being audited. The first step in the audit process is to acquire the identification attributes s_i, and verification attributes s_v for batch N from the low voltage power cables. Next, the identification attributes s_i are used to search the blockchain network via the smart contract for h_v and h_l. Now h_v" retrieved from the blockchain can be compared to h_v received from the batch data with s_v. If both are the same value, we can confirm the integrity of this specific batch. However, if this number is not the same, h_v can extract r from the distributed file system and P_s from the private ledger database. Now r can be decrypted and compared to the voltage table extracted s_v for differences. If the integrity verification is successful for all the batches, a transaction containing the date, auditor, and the result is broadcasted to the blockchain.

3) External Integrity Check: The external party can confirm that the company's ledgers are valid by checking the blockchain for the certification report.

III. DWH-DIM Implementation

This section describes the implementation choices and technical considerations of DWH-DIM and DWH-DIV. The creation of DWH-DIM is possible using multiple exciting technologies. Our code implementation can be found in the Github repository[1].

A. Design choices

The design choices are made in compliance with the utility company *Spatial Eye*. However, for other use cases, there could be more suitable technologies available.

1) Spatial Data Warehouse: The data warehouse product used for this implementation is the *Spatial Warehouse* from *Spatial Eye*[2]. The *Spatial Warehouse* focuses on the advanced data structure: geographical locations. Nevertheless, it is also capable of working with numerical and alphabetical data. The advantage of this tool is that it can track changes in the data over time, making the data a time series.

2) DWH-DIV: The DWH-DIV tool has been made in Python using Jupyter notebook. Jupyter notebook is an open-source, web-based interactive computational environment[3]. It is often used by data scientists working in Python, or R. The decision for Python has been made because it is one of the most popular programming languages and contains the necessary external packages, minimizing the complexity of the tool.

[1]https://github.com/jeroenbergers/DWH-DIM

[2]https://www.spatial-eye.com/product-spatial-warehouse.html

[3]https://jupyter.org/

3) IPFS: For the decentralized storage system, IPFS is chosen[4]. IFPS meets all our requirements set in section 3 as it is decentralized and uses a hash over the file's content as a file location. This gives the advantage that data cannot be stored twice as this will result in the same identification hash, saving redundant storage space [13].

4) Hyperledger Fabric: Hyperledger Fabric is an open-source permissioned blockchain infrastructure. The permissioned blockchain was chosen because, in the proposed model, the participation companies work together in a consortium. Individually they cannot be trusted, but together they can. More trust in the network means that a less heavy consensus mechanism can be used, increasing scalability and decreasing resource waste. The choice for Hyperledger has been made based on research [14] showing that Hyperledger has the best performance in terms of latency, throughput, and privacy compared to other permissioned blockchains.

DWH-DIM is built with Hyperledger Fabric v2.2. The recommended consensus protocol in this version is Raft. The Raft consensus protocol works with one leader node, determining the order of the transactions and all the other nodes as followers, replicating the leaders' decisions [15]. Raft is a Crash fault tolerance algorithm, and the focus is on performance when nodes go offline. It can tolerate f-number of faulty nodes in $n = 2f + 1$ where n is the total number of nodes [15]. The smart contract in Hyperledger Fabric is called chaincode (CC) and can be written in Go language, Node.js, or java. For this implementation, the Go-language is used to program the chaincode. For communication with the Hyperledger Fabric network, the node.js SDK is used, as shown in Fig. 3.

B. Communication

The communication between the components happens via REST protocol. This allows using HTTP operations as GET and POST for interaction. Using the Node.js SDK it is possible to run a Node.js server next to Hyperledger Fabric to make alterations to the network. The Python package *Request* can be used to send the actual operations to the network. Both CouchDB and IPFS support the possibility of sending HTTP requests via their API, which means that operations can be sent using the same logic. An overview of the communication within the local network can be found in Fig. 3.

C. Chaincode

To implement DWH-DIM, three smart contracts are introduced: 1) the main contract to store the identification attributes on the chain and retrieve them; 2) a contract to store the private key on a private peer; and 3) a contract to store the validation report. All smart contracts are deployed using the four-step Fabric chaincode lifecycle [15].

1) CC - Integrity Verification : The integrity verification CC is responsible for the data structure, and it consists of the functions: $CreateRecord$, $QueryById$, and $QueryAll$. The

[4]https://ipfs.io/

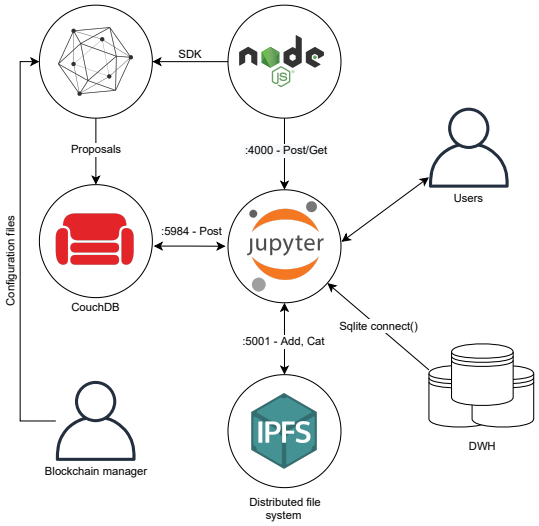


Fig. 3. Overview of the components and the internal connection.

invoking of the functions works via the Node.JS SDK. The defined structure of the records is the identification attributes. The used index key is the identification hash. The structure can be seen in Fig. 4.

```
type VerificationAttributes struct {
    Organisation    string `json:"Organisation"`
    Table_name  string `json:"Table_name"`
    Batch_ID string `json:"Batch_ID"`
    Verification_Hash  string `json:"Verification_Hash"`
}
```

Fig. 4. Structure of the verification attributes inside the create record chaincode.

2) CC - Private Storage: Hyperledger Fabric offers the ability to create private data collections. Private data collections are used to store only an individual's company their private data. Other participants on the network cannot view the private data when they are not given access. In DWH-DIM, private data collection is used to store the keys necessary to decrypt the integrity verification file. The advantage of using this private data collection is that keys can be easily shared with external auditors when needed. The CC contains a $CreateKeyStorage$ and a $QueryKeyStorage$ function. The difference between this CC and the integrity verification CC is that the data needs to be transient when working with private data. The index key used for querying is the identification hash.

3) CC: Integrity Certificates: The integrity certificate CC is responsible for the storage of the certificate showing that data is authentic. It has the $CreateCertificate$ function invoked when all the data has been identified as honest.

D. Fabric SDK - Node.JS

The fabric SDK has three functions: it checks for authorization, including the possibility of adding new users, invoking

transactions, and query transactions.

1) Authorization: In Hyperledger, users are assigned through a certificate authority (CA). The CA gives a certificate to a user that can be used to sign and authenticate a transaction. For this implementation, we have given every company one CA and used the default elliptic curve digital signature algorithm (ECDSA). The secret key is stored offline on the server side; as a result, the certificate does not have to be sent via the REST API or stored on the user's computer. However, we still need to identify and authenticate the user. For this, the express-jwt module is used, which allows us to work with JSON web tokens (JWT) to sign the REST-API requests and invoke transactions. A new user can be created using a Post request on the registered URL. The system returns a JWT token that can be used as a bearer token for the organization's transactions.

2) Invoke: In order to invoke a transaction to the blockchain, a Post request needs to be made to the SDK. The Post request must mention the chaincode and name of the chaincode function to invoke a transaction. The Node JS server uses the JWT token in the header to determine if the user has the proper authorization. The payload of the transaction is a JSON string; for private data, this is transient.

3) Query: Querying happens via the same logic as invoking but is used when a Get request is received. Instead of invoking a transaction, the querying happens via the $EvaluateTransaction$ function.

E. DWH-DIV - Python - Jupyter Notebook

DWH-DIV has been created in Jupyter Notebook and is responsible for the communication and transformation of the data. It is the interface between the data warehouse, the blockchain, the distributed file system, and the user. In addition, it is responsible for the creation and comparison of verification attributes and sending the certificate of authentication to the blockchain. Every user has their own Jupyter notebook.

1) Interface: In the requirements, three types of users are identified. All three users use a different part of the system and require a different interface. This is why three notebooks have been created, one to upload the data to the warehouse, one to check the integrity, and one to check the integrity certificate.

2) Add Database: For this implementation, the data is extracted from the warehouse as an SQ-Lite database. Using the $sqlite3$ package in Python, we can directly make queries in the database and only extract the table and batch needed for the upload or verification tasks.

3) Verification Attributes: If the database is loaded, individual verification files can be created based on the number of batches and tables in the database. The verification attributes are JSON files containing a different amount of fields. The fields are dependent on the importance of the data. Storing more verification attributes increases the traceability and recovery of the warehouse. However, it is important to write in the JSON file what the properties are and how they are designed, because the attributes must be recreated for the verification task. In addition, it is necessary to know which columns are allowed to be changed for the purpose of GDPR comparison. With this in mind, we propose three different verification files in Fig. 5, the first one minimum recovery, the second one for minimum recovery but maximum traceability, and the final one with maximum recovery and maximum traceability but a more significant computational overhead and limiting scalability. For more complex data types as images, other configurations could be useful. For this implementation, only option one has been implemented. The first verification hash H_V is used in the integrity document and sent to the blockchain together with the identification attributes.

JSON

```json
{
  "h_v": "SHA-256 over subset",
  "traceability": "1",
  "col": "AllColNames",
  "rows": "Number of rows",
  "gdpr": "ColNames containing GDPR data",
  "gdpr_hash": "Sha256 over the subset without the GDPR cols",
  "ColHash": "Vertical calculated SHA256 for each column"
}
```

JSON

```json
{
  "h_v": "SHA-256 over subset",
  "traceability": "2",
  "col": "AllColNames",
  "rows": "Number of rows",
  "gdpr": "ColNames containing GDPR data",
  "gdpr_hash": "Sha256 over the subset without the GDPR cols",
  "ColHash": "Vertical calculated SHA256 for each column",
  "RowHash": "Horizontal calculated SHA256 for each row"
}
```

JSON

```json
{
  "h_v": "SHA-256 over subset",
  "traceability": "3",
  "col": "AllColNames",
  "rows": "Number of rows",
  "gdpr": "ColNames containing GDPR data",
  "gdpr_hash": "Sha256 over the subset without the GDPR cols",
  "Data": "AllTableData"
}
```

Fig. 5. Three options of the proposed verification files.

4) Encryption and Decryption: For encryption and decryption of the JSON integrity files, the Advanced Encryption Standard (AES) is used. Before using the encryption function, the file needs to be encoded into bytes. Then, the file is encrypted using a secret key and a randomly created nonce. Both the 16 bytes secret key as the nonce is stored on the private blockchain node. Then, the encrypted message is sent to the IPFS. For decryption, the nonce and secret key can be extracted from the private peer if the user is from the proper organization and decrypts the IPFS file. Finally, the decrypted file is decoded and changed back to JSON format to be used in the verification task.

5) IPFS Connection: For the IPFS connection, the IPFS desktop daemon is installed. Then, files are uploaded using the IPFS HTTP package, using simply the add function. This returns the identification hash stored on the blockchain and can be used to retrieve the file.

6) Blockchain Connection: Blockchain transactions from the notebook are sent via the request package. The JSON payloads, besides the access token, function name, and chaincode

name, are the identification attributes and verification hash shown in Fig. 4. A search request is directly sent to the peer for querying an individual batch, containing the verification attributes. To query all, a request is sent to the integrity CC.

7) Verify I and Verify II: We have developed two identification functions. Verify I is supposed to be the faster identification method. It uses the verification hash received from the blockchain and compares it with the new verification hash created from the database. If this hash is equal, the data has not been changed, and a deeper investigation is unnecessary. Verify II is a deeper comparison, as it compares the two integrity files. This takes more time as the file needs to be retrieved from IPFS and be compared to the current state of the database. Therefore, verify II is only used when verify I results that there is something wrong.

IV. DWH-DIM Evaluation

In this section, extensive experiments have been performed to determine whether our model can tackle the gaps and requirements. Especially, the analysis has been split into computational benchmarks and throughput experiments to test whether DWH-DIM has the required scalability.

A. Computational Benchmarks

To verify if DWH-DIM can handle large and complex data warehouses, the implementation has been tested to multiple testing use cases. The goal of the benchmarks is to find out whether our implementation of DWH-DIV is fast enough to handle the large amount of data stored in a data warehouse. Further, we want to know if our implementation is able to handle complex data types as geometry data and what the limiting factors of our implementation are.

To test the system's performance, we repeatedly run the implementation of DWH-DIM with different database settings to find out how long it takes to perform the functions. The database will vary in the number of records per batch and the number of batches per table. It will be evaluated in seconds since the activation of the function. For the database, an SQLite database is used containing information about properties in the Netherlands to simulate a realistic scenario. A subset table is used to perform the benchmarks containing 28 columns, including two columns containing complex geometries. The subsets are made by randomly selecting 1.000, 10.000, 100.000, 1.000.000, or 10.000.000 records. Then, each subset is duplicated, resulting in 10 subsets. Half of the subsets contain one batch. The other subset contains one batch for every 1000 records. An overview of the subsets can be seen in Table I. The subset containing 10.000.000 records has only been tested once. All other benchmarks are performed in triplicate.

For the benchmarks, there are four points of interest: 1) the time in seconds for uploading the subset to the network, 2) the time in seconds for performing integrity check Verify I, 3) the time in seconds for performing integrity check Verify II, and 4) the time in seconds of the individual functions in the system. In Fig. 6 (1) the first representation has been shown with variation in the number of batches.

TABLE I
Overview of Variation in Benchmarks and Experiments

Experiment:		1	2	3	4	5
Benchmark I	Records:	10^3	10^4	10^5	10^6	10^7
	Batches:	1	1	1	1	1
Benchmark II	Records:	10^3	10^4	10^5	10^6	10^7
	Batches:	1	10	100	1000	-

The result shows that the time it takes to upload one batch is consistent per experiment. As the number of batches in a table increases from 10 to 100, the upload and verify II time also increases with a factor of 10. The only case where the linear increase is not consistent is for verify I. This higher increase could be explained by a higher search time because more items are in the database. For verify II, this is less of an issue as the querying is a more negligible influence on the run time.

For the next benchmark, the variation in performance with a changing number of records is evaluated, the results are shown in Fig. 6 (2). There are some expected results; the time of Verify I is significantly lower when there are a low number of records in a batch but increases to the same time as uploading and Verify II as the number of records inside a batch increases. This is expected because the difference between Verify I and Verify II is that for Verify I, the verification document does not have to be received. The receiving of the verification record from IPFS happens at a constant time regardless of the number of records in the batch. Further, it is notable that the duration of both uploading and verify II increases only slightly when increasing the number of records from 1000 to 10.000. The increase in time of uploading from 10.000 to 100.000 and from 100.000 to 1.000.000 is also insignificant compared to the increase in time from 1.000.000 to 10.000.000 records per batch. With this, it is possible to deduce that performance-wise it is most efficient to upload batches containing 1 million records as this takes only 100 seconds, and uploading 10 million records takes nearly 10.000 seconds. However, this would lead to a reduction of tamper traceability in comparison to uploading smaller sets. To conclude, it is possible to say that it is not efficient to upload small batches of 1000 records as this takes almost the same time as 10.000 records and large batches of 10 million as this takes 100 times as long as batches from 1 million.

To investigate this further, the time taken by the individual components and a variety of batch sizes has been recorded and plotted in Fig. 6 (3). In the figure, it is possible to see what functions cause the delay. The encryption and decryption time have not been taken into account because the file size is the same, resulting in a consistent encryption time. When having 1000 records per batch, the limiting factor is the two functions that perform an API request and have to wait for a response. Both the sending to IPFS and blockchain take almost 2.5 seconds accounting for more than 50% of the upload time.

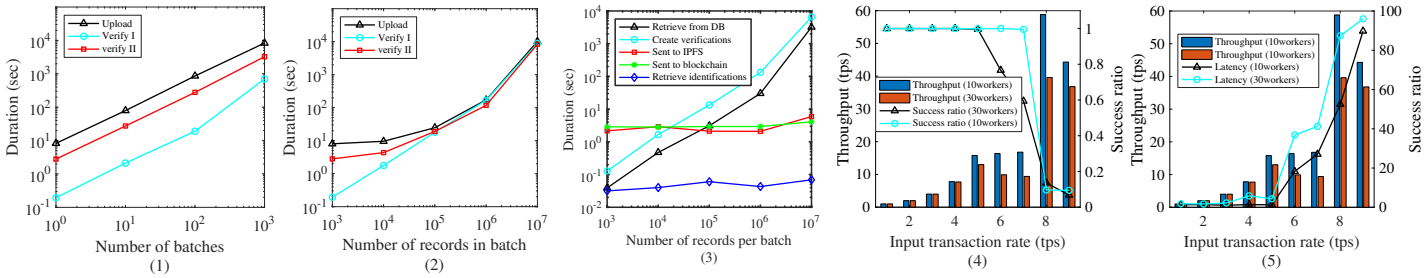


Fig. 6. Performance of DWH-DIM with (1) different amount of batches on upload and integrity verification tasks; (2) different amount of records per batches on the upload and integrity verification tasks; (3) different amount of records per batches on the individual components; (4) different workloads in throughput and success ratio; and (5) different workloads in throughput and average latency.

Both transaction functions do not increase in time when more records are added to a batch. However, it can be the case that when multiple companies use a small batch size, the number of transactions per second to the blockchain increases, causing a delay. After 100.000 records per batch, retrieving the data from the database and creating the verification attributes is the most significant factor. This can be expected as the size of 10.000.000 geographical records has a size of almost 4 GB that needs to be transferred to DWH-DIV. When a company wants to upload larger batches, a faster method needs to be found to get the data to DWH-DIV and to calculate the cryptographic hashes.

B. Throughput of the Blockchain Network

As mentioned in the previous section, the simultaneous use of blockchain techniques often leads to decreased performance. This is the limited throughput on the blockchain networks leading to high latency and failed transactions [16]. For this reason, DWH-DIM tries to limit the number of transactions done with the blockchain. Nevertheless, the blockchain network needs to handle multiple users at the same time, making a performance evaluation necessary. For this, a tool developed by the Hyperledger Foundation called Caliper[5] is used. For the evaluation, Caliper has been installed as a docker image and run inside a docker container. The performance benchmarks have been run following these configurations:

- Resources as described in the previous section.
- Invoke transactions using the create records function, can be seen as $Write$ function.
- One Hyperledger fabric framework containing two organizations with both one peer and one orderer node.
- Using the RAFT consensus and a CouchDB state database.
- Different workloads to the Hyperledger fabric framework by generation several transactions proposals per second (1, 2, 4, 8, 16, 32, 64, 128, 256).
- A different amount of workers (10 and 30) to diversify the number of clients connected to the Hyperledger Fabric network.

[5]https://github.com/hyperledger/caliper

The performance of the benchmarks is measured by the metric throughput that is defined as:

$$throughput = \frac{total\ committed\ transactions}{total\ time\ taken\ in\ seconds} \quad (1)$$

Further metrics used are latency (which is defined as the average duration of the execution of a transaction) and the ratio of successful transactions (as defined in equation 2).

$$success\ rate = \frac{successful\ transaction}{successful + failed\ transaction} \quad (2)$$

Fig. 6 (4) interprets the performance of the system under a different workload. The highest TPS for both numbers of workers can be found at an input transaction rate of 128. Above 128, the number of handled transactions per second decreases. If we look at the latency at an input rate of 128, it can be seen that it increases, especially with more workers. In Fig. 6 (5) the performance is shown with the success rate. It can be seen that at the input transaction rate of 128, the ratio is below 0.2. This means that more than 80% of the transaction fails. The highest input rate for both the number of workers the success ratio is above 0.9 is at 16 transactions per second. In Fig. 6 (4) can be seen that for this amount of transactions, the latency is low and increases fast when increasing the input transaction rate. In the computational overhead benchmarks, we concluded that when uploading a small batch of data, the latency of the blockchain is the most significant influence on the system. For small batches, this high latency is not desired.

TABLE II
OVERVIEW OF LATENCY PER BENCHMARK SETTING

	30 workers			10 workers		
input TPS	Max (s)	Min (s)	Avg (s)	Max (s)	Min (s)	Avg (s)
1	10.56	0.38	1.8	8.12	0.37	1.29
2	6.31	0.47	1.68	2.46	0.41	1.35
4	12.76	0.42	2.22	2.68	0.4	1.04
8	22.36	0.47	5.81	5.12	0.43	1.36
16	8.07	1.05	4.32	2.72	0.48	1.26
32	78.47	1.28	36.8	33.17	0.66	18.28
64	74.84	2.14	41.2	46.05	1.06	27.05
126	147.71	1.53	87.4	135.56	1.33	52.5
256	140.56	1.84	96	137.59	4.14	89.78

In Table II the latency has been shown. It can be seen that both the average as the max latency increases fast when going from 16 to 32 transactions. If we take, for example, a batch of 10.000 records, the upload time will increase from 9.6 seconds to over 25 seconds with 10 connections and 32 transactions per second with a maximum of over 80 seconds in the worst case when receiving 32 transactions per second with 30 connections. Thus, we can conclude that the preferred amount of transactions that can be written on the chain simultaneously in this setting is around 16; this will probably decrease when more peers and orderer services are connected to the network.

V. Related Work

Current related research is in multiple directions. The blockchain based data integrity verification tools focus on proving that a third-party stores the data and does not modify/read the data. Common challenges in this field are how to prevent a large computational overhead and how to minimize redundant data storage. The latter is especially important in combination with blockchain, as blockchain has limited scalability. Current research on distributed integrity data management is presented in [11], [7], [4], [17]. The other techniques with relevance to the topic are blockchain based data management tools. These are the techniques presented in [18], [19], [20]. The common challenges are the limited scalability of blockchain, the computational overhead of the integrity checks, the protection of data, and personal data regulations.

VI. Conclusion

In this paper, we propose a new decentralized database integrity verification tool called DWH-DIM. DWH-DIM is designed to replace the current expensive and semi-honest third-party auditing tasks. DWH-DIM fills the gap in the existing literature by working with voluminous, complex data stored in a traditional data warehouse without changing the system core component. Besides being decentralized, DWH-DIM is built with three objectives, namely scalability, protection over private data, and GDPR compliance.

For further research, the complete data warehouse can be placed within DWH-DIM for benchmarks. The combination with a more fine-grained access system will lead to required integrity verification and an additional possibility to share data. For example, the decentralized access system presented by [19] [17] in combination with a decentralized data warehouse storage and the use of identification attributes, would lead to a decentralized system where the right persons can find and access the correct data at any time. This could replace the current data warehouse system and be leveraged in a distributed research environment.

Acknowledgment

This work has been partially funded by the European Union's Horizon 2020 research and innovation programme by the ARTICONF project grant agreement No 825134, by the ENVRI-FAIR project grant agreement No 824068, by the BLUECLOUD project grant agreement No 862409, and by the LifeWatch ERIC. The work is also partially supported by China Scholarship Council.

References

[1] R. Mork, P. Martin, and Z. Zhao, "Contemporary challenges for data-intensive scientific workflow management systems," in *Proceedings of the 10th Workshop on Workflows in Support of Large-Scale Science*, 2015, pp. 1–11.

[2] N. Dedić and C. Stanier, "An evaluation of the challenges of multilingualism in data warehouse development," 2016.

[3] S. H. A. El-Sappagh, A. M. A. Hendawi, and A. H. El Bastawissy, "A proposed model for data warehouse etl processes," *Journal of King Saud University-Computer and Information Sciences*, vol. 23, no. 2, pp. 91–104, 2011.

[4] N. Lu, Y. Zhang, W. Shi, S. Kumari, and K.-K. R. Choo, "A secure and scalable data integrity auditing scheme based on hyperledger fabric," *Computers & Security*, vol. 92, p. 101741, 2020. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0167404820300274

[5] M. Belotti, N. Božić, G. Pujolle, and S. Secci, "A vademecum on blockchain technologies: When, which, and how," *IEEE Communications Surveys Tutorials*, vol. 21, no. 4, pp. 3796–3838, 2019.

[6] P. Patil, M. Sangeetha, and V. Bhaskar, "Blockchain for iot access control, security and privacy: A review," *Wireless Personal Communications*, pp. 1–20, 2020.

[7] R. Kalis and A. Belloum, "Validating data integrity with blockchain," in *2018 IEEE International Conference on Cloud Computing Technology and Science (CloudCom)*, 2018, pp. 272–277.

[8] D. Brandon, "The blockchain: The future of business information systems," *International Journal of the Academic Business World*, vol. 10, no. 2, pp. 33–40, 2016.

[9] H. Zhou, C. de Laat, and Z. Zhao, "Trustworthy cloud service level agreement enforcement with blockchain based smart contract," in *2018 IEEE International Conference on Cloud Computing Technology and Science (CloudCom)*. IEEE, 2018, pp. 255–260.

[10] D. V. Dimitrov, "Blockchain applications for healthcare data management," *Healthcare informatics research*, vol. 25, no. 1, p. 51, 2019.

[11] K. Hao, J. Xin, Z. Wang, Z. Jiang, and G. Wang, "Decentralized data integrity verification model in untrusted environment," in *Asia-Pacific Web (APWeb) and Web-age information management (WAIM) joint international conference on Web and big data*. Springer, 2018, pp. 410–424.

[12] A. A. Siyal, A. Z. Junejo, M. Zawish, K. Ahmed, A. Khalil, and G. Soursou, "Applications of blockchain technology in medicine and healthcare: Challenges and future perspectives," *Cryptography*, vol. 3, no. 1, p. 3, 2019.

[13] J. Sun, X. Yao, S. Wang, and Y. Wu, "Blockchain-based secure storage and access scheme for electronic medical records in ipfs," *IEEE Access*, vol. 8, pp. 59 389–59 401, 2020.

[14] J. Polge, J. Robert, and Y. Le Traon, "Permissioned blockchain frameworks in the industry: A comparison," *ICT Express*, vol. 7, no. 2, pp. 229–233, 2021. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S2405959520301909

[15] Hyperledger. (2021) Hyperledger fabric documentation. [Online]. Available: https://hyperledger-fabric.readthedocs.io/en/release-2.2/

[16] Z. Shi, H. Zhou, Y. Hu, S. Jayachander, C. de Laat, and Z. Zhao, "Operating permissioned blockchain in clouds: A performance study of hyperledger sawtooth," in *2019 18th International Symposium on Parallel and Distributed Computing (ISPDC)*. IEEE, 2019, pp. 50–57.

[17] H. Wang, Q. Wang, and D. He, "Blockchain-based private provable data possession," *IEEE Transactions on Dependable and Secure Computing*, 2019.

[18] C. Wirth and M. Kolain, "Privacy by blockchain design: a blockchain-enabled gdpr-compliant approach for handling personal data," in *Proceedings of 1st ERCIM Blockchain Workshop 2018*. European Society for Socially Embedded Technologies (EUSSET), 2018.

[19] S. Wang, Y. Zhang, and Y. Zhang, "A blockchain-based framework for data sharing with fine-grained access control in decentralized storage systems," *Ieee Access*, vol. 6, pp. 38 437–38 450, 2018.

[20] G. Zyskind, O. Nathan *et al.*, "Decentralizing privacy: Using blockchain to protect personal data," in *2015 IEEE Security and Privacy Workshops*. IEEE, 2015, pp. 180–184.