File Chunking Towards On-Chain Storage: A Blockchain-Based Data Preservation Framework

Muhammed Tmeizeh Author[1,2*], Second Carlos Rodríguez-Domínguez[2†] and Third María Visitación Hurtado-Torres[2†]

[1*]Faculty of Engineering and Information Technology, Palestine Ahliya University, Bethlehem, 1041, West Bank, Palestine.
[2]Department of Software Engineering, Higher Technical School of Computer and Telecommunications Engineering, University of Granada, Granada, 18071, Andalusia, Spain.

*Corresponding author(s). E-mail(s): mohammedt@paluniv.edu.ps ;
Contributing authors: carlosrodriguez@ugr.es; mhurtado@ugr.es;
†These authors contributed equally to this work.

Abstract

The growing popularity of the most current wave of decentralized systems, powered by blockchain technology, which act as data vaults and preserve data, ensures that, once stored, it stays preserved, considered to be one of the most promising safe and immutable storage methods. The authors of this research suggest an on-chain storage framework that stores files inside blockchain transactions using file transforming, chunking, and encoding techniques. This study investigates the performance of on-chain file storage using a simulated network blockchain environment. Test files of varying sizes were deployed. Performance metrics, including consumed time in chunking, encoding, and distributing chunks among block transactions, were measured and analyzed. An analysis of the collected data was conducted to assess the framework's performance. The result showed that selecting the appropriate chunk size significantly influences the overall performance of the system. We also explored the implications of our findings and offered suggestions for improving performance within the framework.

Keywords: On-chain, File chunking, Blockchain, Immutable storage

1 Introduction

One of the most important research topics in recent years has been the widespread adoption of the most recent breed of decentralized systems, powered by blockchain and decentralized file storage, that act as data vaults, protecting data to ensure that when it is stored, it remains unchanged and can be used as secure storage, for instance, for archiving files or data, property rights, medical health records [1], and smart city data[2]. Some studies such as [3] examined open difficulties and unresolved problems in data security and the potential of integrating blockchain technology with many aspects, such as IoT and health data.

To ensure that an organization or business maintains its reputation and thrives, data security must be guaranteed. Significant consequences from data breaches could include financial losses,

reputational harm, and legal obligations. Consequently, it is imperative that businesses implement strong safeguards to protect their data integrity. The integrity of electronic files is growing in significance at a time when electronic data is the foundation of many vital systems and activities, from financial records and transactions to medical records, legal documents, intellectual property, and scientific research. We aim to address the ongoing problems of fraud and data tampering by giving files immutability. Immutability offers a strong base for preserving open and auditable documentation, safeguarding against malicious modifications, and enhancing trust in digital ecosystems.

The authors undertook a previous study [4] in this field because they believe that using blockchain technology for data storage is a promising solution. That study shows that many proposals have been made to take advantage of the immutability feature of blockchain technology. A large number of those employ blockchain-connected external storage to save a hashed value that serves as a means of preventing the original saved data from being altered or forged [4]. Furthermore, other proposed works sought to save the data directly on the blockchain ledger; the metadata, which is very small in nature and includes things like gene names, light-weight sensor data, or even changes made to specific network nodes to enable the saving of data which is linked to blockchain ledger [4]. Our goal in this study is to present a framework that makes possible to store data directly to the blockchain world-state ledger, which is regarded as an on-chain file storage solution. Moreover, data stored off-chain in Decentralized File Storage (DFS) connected to its hash value within the blockchain is not entirely safe because some DFS has security or immutability flaws [5].To the best of our knowledge the existing on-chain approaches remains limited by its inability to handle the full file storge inside the ledger, which is increasingly relevant in organization and business that seeks for tamper proof storage. This work introduces an innovative approach to extend the on-chain capabilities to adapt the file to be stored inside the ledger of the blockchain, addressing a critical gap in decentralized storage solutions.

Many blockchain-based solutions, such as off-chain approaches and non-blockchain-based solutions such as the Interplanetary File System (IPFS), introduce tamper-resistant storage, which aim to demonstrate the reliability that stored data has not changed [4], since those approaches serve as verification and checksum systems it cannot prevent participating nodes that store data fragments from updating or deleting it [5]. However, in this paper, we aim to demonstrate with high reliability that data has not and also cannot be changed after being saved. As data and transactions that are saved inside the blockchain ledger are tamper-resistant and immutable and cannot be changed after being appended to the chain ledger, we aim to utilize this concept to save the file itself inside the distributed ledger. This approach empowers file storage by providing a secure and efficient environment in terms of data integrity and immutability.

The authors of this study suggest an architecture that uses permissioned blockchain technology to allow file storage inside the blockchain ledger. This on-chain solution provides an immutable guarantee for the files since the whole file is being stored inside the ledger. Using a permissioned blockchain network such as Hperledger Fabric, besides the access roles and authorizations offered, will facilitate file sharing between parties or organizations in a secure and immutable manner. The main contributions and benefits of the suggested framework are:

- providing a comprehensive on-chain solution for file storage;
- reducing the processing load of saving data on the blockchain side by dividing tasks across the edge and core blockchain network;
- applying compression techniques lessens the amount of data in memory by shrinking the file, which optimizes ledger storage capacity and enhances storage effectiveness;
- leveraging the concept of storing files within a database as Binary Large Object (BLOB) to facilitate the storage of files within a blockchain ledger using a similar approach.

The remainder of this paper is organized as follows: Section 2 introduces an overview of related works by classifying, summarizing, illustrating related articles. Section 3 introduces an overview of the proposed framework while also providing a

comprehensive background on relevant subjects in order to put the suggested framework into context. Section 4 shows testing of the proposed framework. Section 5 discusses in-depth the core elements of the framework, which also explain each component's presence within the framework and justify its utilization for each function.. Finally, section 7 concludes the paper and outlines some future work.

2 Related Work

This section presents the related works of researchers that contributed to utilize blockchain as an immutable storage medium. Immutable storage using blockchain technology has been addressed by several approaches in recent years, two main classification approaches can be considered: on-chain and off-chain approaches.

Within the realm of off-chain, a diverse array of studies has been conducted. Each contributing in saving the data outside the blockchain while corresponding data are being stored in blockchain to verify the original off-chain data. For instance, Babu et al. [6] used MongoDB to store electronic healthrecords (EHRs) while its hash value with some patient information will be stored in the blockchain. Yang et al. [7] proposed a solution to save data of the products into two parts, the first part called public data which will be saved in relation database where the private part of the data will be stored in the blockchain.

Another approaches in off-chain that use IPFS for instance, in [7] the authors use IPFS to save the data generated by IoTs in smart city applications then linked the data with Ethereum blockchain. Additional study by V. Mani et al. [8] uses a similar approach for saving data in IPFS which is connected to a Hyperledger blockchain that used as index in the form of key-value to reach the date in IPFS. F. Liu et al. [9] provided a way in his work to save images in IPFS and save the associated hash for the images inside the blockchain.

Furthermore, some studies accomplish the off-chain storage using cloud storage which is connected with blockchain. Y. Chen et al. [10] used blockchain to index the cloud data as well as saving some metadata of the cloud records. additionally, N. Alrebdi et al. [11] used cloud storage with blockchain, the role of blockchain is to serve as access control and indexing system for the cloud data.

On the other hand, many studies conducted using on-chain data storage for instance, P.K. Sharma et al. [12] used blockchain to store a transactions of IoTs of smart city application in the form of hashed values which aimed to increase the security of data. Loss et al. introduce and integration between FIWARE and blockchain to store data in blocks, the authors used Ethereum in their work. As well Khalaf et al. [13] introduced a traceable way of storage using blockchain, the study provides framework for saving the wireless network sensors broadcasting data as a transactions inside the blockchain.

Additionally, Arslan et al. [14] introduced a new approach to store multimedia data inside blockchain node by making some modification on the miners behavior. G. Gürsoy et al. [15] store pharmacogenomics as a string inside the transactions associated with transaction ID in the form of key value pairs which is related to some of gene attribute.

As illustrated most approaches and studies are focused on off-chain approach, since it offers many benefits such as the ability to store a variety of data formats and files as well as the storage capacity. Whereas fewer papers delve into on-chain approach due to it's nature that poses some restrictions on the format on the type of data that can be saved in addition to some other factors which is related to the blockchain capacity. Our previous study [4] illustrated this point with more details.

3 Overview of the Proposed Framework

Our proposed framework is an on-chain storage framework, using consortium blockchain between members that want to store their files in an immutable storage by get the benefits of tamper proof characteristic that blockchain offers. Figure 1 shows the general structure of the proposed framework, in this framework any participating member or organization has nodes called peers which are configured to be invoked from an authorized clients using Application Programming Interface (API) and Software Development Kit (SDK). Authorized and authenticated clients

or peers are eligible to store and retrieve files to and from the network.

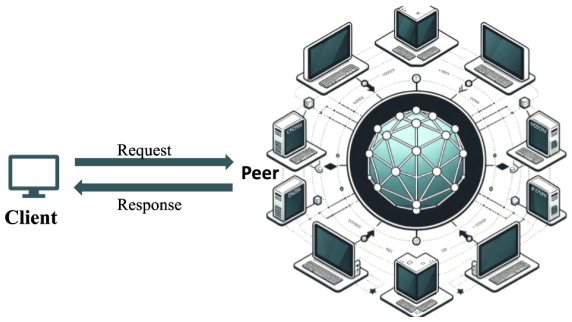

Fig. 1 General structure of the proposed framework

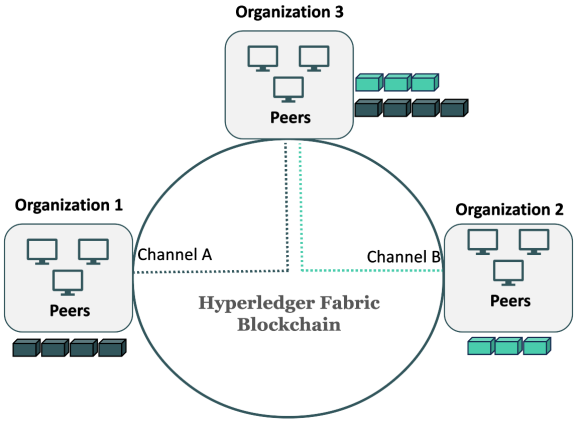

Fig. 2 Hyperledger Channels Example

In our framework, we use the Hyperledger blockchain Platform [16], which acts as a trusted infrastructure by authenticating any new member, organization, or node that wants to join the network. Besides, it's ability to offer a private channel between members or organizations that wish to make shared immutable file storage and need to impose their policy of committing or reading files to the network.

The client edge and the blockchain network share the workload in our framework. The file should be chunked and transformed by the client before being sent over the network. The data transformation and storage are handled by the core network, also known as the blockchain network. Further details regarding the framework are given in the subsections that follow.

3.1 Background

The background of the suggested work is presented in this section. Immutable data storage means that after storing the data, editing, altering, or deleting this data is not possible. In the real world, there's a need for such storage. For instance, medical trials, EHR, academic certificates, business contracts, and many other similar cases are all seeking such a solution. For instance, when someone buys a house, an ownership contract is issued, and if the same house is sold, a new contract is issued. The old contract should not be overwritten; instead, it should be appended to the existing one.

Hyperledger [16] is a general-purpose permissioned blockchain that gives peers the ability to execute smart contracts called chaincodes, which can be written using general-purpose programming languages such as Go, Java, and Node.js. It enables authenticated clients to send transactions to their peers. The transaction lifecycle goes through different entities: endorsing peers check the validity and correctness of the transaction; the ordering service uses the endorsement policy to order them chronologically into new blocks; then broadcasts the new state updates to peers and establishes consensus; finally, blocks are delivered and committed to the current state database for all peers on the channel.

Hyperledger consists of one channel or more; the channel is established between two organizations or more. Each channel has its peers. As illustrated in Figure 2, we have a Hyperledger with three organizations and two channels: channel A is between organization 1 and organization 3, and channel B is between organization 2 and organization 3. Each channel has its own distributed ledger. Each peer who joins the network must have an identity issued via a membership service provider (MSP). As illustrated, an organization can't be a member of one or more channels.

The suggested approach can offer technical remedies to deal with the aforementioned difficulties, such that if organizations wish to share immutable storage for their files, they can establish or join a blockchain network and create their own private channel. This channel will be used

to save their data in a secure and tamper-proof manner.

3.2 Edge Side

The edge or client side is responsible for preparing the file and sending it to the core blockchain network to save it directly inside the blocks of transactions. As illustrated from Figure 3 the file will be zipped and converted to BLOB, then chunked with a chunk size of 256 kilobytes,and each chunk will be converted to a Base64 string before the client sends the data to the peer in the core network.On the client side, each chunk is labeled with an index number before being sent to the blockchain core side. The process of indexing will be done using TypeScript code on the edge side; this code is also responsible for calling blockchain peers throughout the API using the SDK, which allows client applications to interact with a Fabric blockchain network. We utilize a compression strategy to minimize the file size; choosing the right algorithm is essential to utilizing the on-chain storage technique. Comparing different compression algorithms is done to find the best candidate for our work.

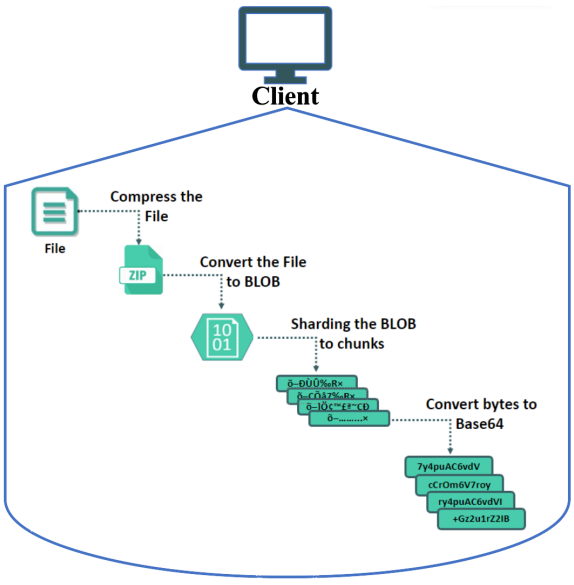

Fig. 3 Edge side: compressing, converting to BLOB, chunking, then transforming to Base64

To make sure that every component of the file was intact during the compression and decompression procedures, we employed lossless compression in our framework. Lossless compression is crucial because of the design of our system, which protects the file's integrity and entirety. We can preserve the original file throughout the saving and retrieval stages by selecting this option, which ensures that no data is lost during the compression and decompression phases.

3.3 Blockchain Side

On the core side of the blockchain network, the peer received a request to save file chunks in the blocks; as illustrated in Figure 4, the chunks are received as a Base64 datatype. The chain code will transform the received chunk into a binary datatype and put it in an array of bytes, then send it to the transaction and, ultimately, to the blocks.

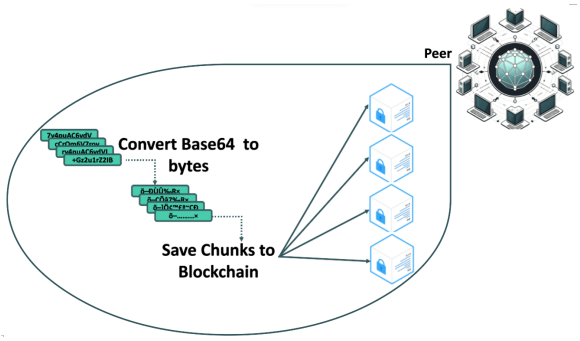

Fig. 4 Blockchain Peer side: converting to binary data, then save the chunks in the transactions

3.4 File Chunks Indexing

Figure 5 illustrates what the used chaincode anticipates receiving from the client once file chunks are sent to the blockchain peer. Since each file will consist of many chunks, the formal arguments for the chaincode are the chunk in Base64 format, the hash value of the file identifying number, and the chunk number.

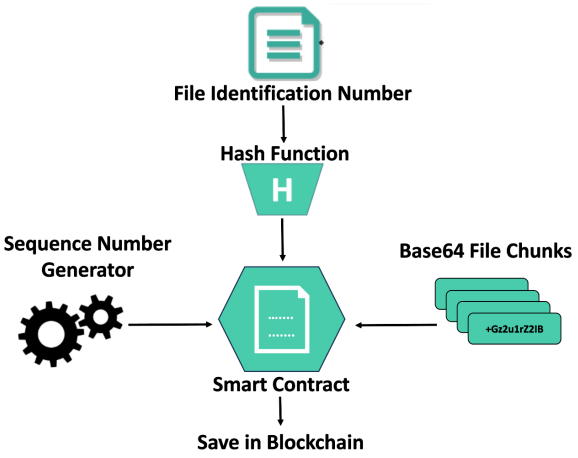

Fig. 5 File chunks indexing

4 Experiments and Results

In this section, we present the outcomes of implementing our proposed framework for saving text files as a case study. The results provide insights into the effectiveness and performance of our framework in addressing the challenges of data storage, and efficiency in decentralized environments.

First, we describe the experimental setup and methodology employed assess the functionality of the framework. The information gathered from our trials is then presented, including metrics for file size reduction, time spent chunking files, time needed to convert chunks to Base64 encoding, and transaction throughput on the blockchain network.

Following that, elements of the suggested framework have been deployed in an Oracle virtual machine (VM) running Ubuntu 22.04. 8 GB of RAM and four 1.80 GHz Intel (R) Core (TM) i7-8565U CPUs were set up in the virtual machine. The suggested system is then operated as a simulation environment on a test network running the Hyperledger Fabric-2.5 version. The simulation fabric network is implemented in Hyperledger Fabric and consists of two organizations, one peer node for each organization, a shared ledger among members, ordering service, endorser peer node, fabric certificate authorities through MSP, and chaincode.

The authenticated end-user application on the edge side interacts with blockchain peer chaincodes through the SDK to send data chunks to the ledger by connecting the endorsed peers when they need to store files in the ledger. The edge node application is divided into two sub-applications. The first is implemented in Python, which receives the file, compresses it, converts it to a BLOB, chunks the file, encodes each chunk to a Base64 string, and then saves each chunk in a distinct line in a text file. The second sub-part is implemented using TypeScript, which takes the output text file of chunks and the hash code of the original file, then connects to the chaincode to send the chunks to be stored in the ledger.

Distributed File Systems dive deeply into selecting the optimal file chunk size; for instance, Swarm [17] uses a 4 kilobyte (KB) default chunk size, while IPFS [18] uses 256 KB. We conducted our studies for both sizes (4 KB and 256 KB) based on that.

We ran two separate experiments with 12 randomly PDF files for each of the two chosen chunk sizes (4 KB and 256 KB). The majority of the random files include only text or text with charts, with the exception of the five megabyte (MB) file, which is an electronic PDF book. Tables 2 and 1 present the measures and outcomes for the two experiments. As can be seen from the tables, the tests' largest file size was approximately 10.5 MB, while the smallest file size was just about 100 KB.

Figure 6 shows that it takes time to split the file into chunks using 4 KB and 256 KB chunk sizes. Figure 7 illustrates the time in seconds for the client application to transform the junk chunk of binary data into a Base64 string format, which will be used to send the data to the core network for the same chunk sizes used before.

Table 1 Data Summary: Using a 256KB chunk size

File Size (bytes)	Compressed Size (bytes)	Chunk Count	Chunking Time (seconds)	Encoded Size (bytes)	Encoding Time (seconds)	Save Time (Seconds)
102602	77985	1	0.0000	103980	0.0003	2.103
266603	187461	1	0.0000	249948	0.0010	2.135
359075	253832	1	0.0000	338444	0.0006	0.155
422552	196232	1	0.0000	261644	0.0008	0.124
512088	311796	2	0.0002	415732	0.0033	2.196
829048	741722	3	0.0005	988968	0.0016	2.102
1052352	881284	4	0.0005	1175056	0.0021	2.480
1664464	466725	2	0.0003	622304	0.0010	0.243
1821766	1280137	5	0.0007	1706860	0.0028	0.609
4293938	3354594	13	0.0016	4472824	0.0084	1.466
5842628	5402908	21	0.0027	7203932	0.0132	4.659
11081517	1694115	7	0.0009	2258836	0.0044	2.798

Table 2 Data Summary: Using a 4KB chunk size

File Size (bytes)	Compressed Size (bytes)	Chunk Count	Chunking Time (seconds)	Encoded Size (bytes)	Encoding Time (seconds)	Save Time (Seconds)
102602	77985	20	0.0001	104032	0.0001	40.805
266603	187461	46	0.0002	250068	0.0005	93.688
359075	253832	62	0.0002	338608	0.0011	126.077
422552	196232	48	0.0002	261768	0.0005	97.605
512088	311796	77	0.0003	415932	0.0014	156.531
829048	741722	182	0.0005	989448	0.0020	370.133
1052352	881284	216	0.0006	1175620	0.0023	438.965
1664464	466725	114	0.0004	622604	0.0021	231.815
1821766	1280137	313	0.0008	1707684	0.0034	636.022
4293938	3354594	819	0.0019	4474976	0.0099	1664.697
5842628	5402908	1320	0.0030	7207396	0.0144	2679.468
11081517	1694115	414	0.0010	2259924	0.0045	840.943

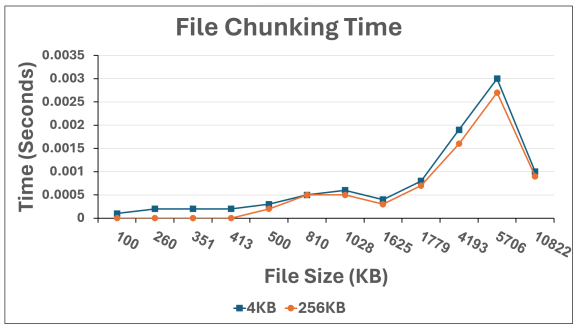

Fig. 6 Time consumed to chunk the file using 4KB and 256KB chunk sizes

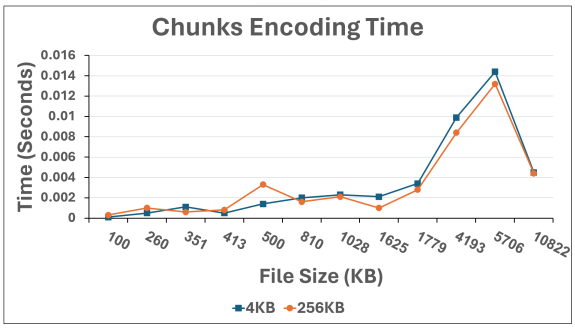

Fig. 7 Time consumed to encode the chunks to Base64 using 4KB and 256KB chunk sizes

As a key goal of our suggested framework is to minimize the file size before storing it in the blockchain ledger, we accomplish this task by applying the Zstandered compression algorithm. Figure 8 shows the size of the original file, the file size after compression, and the size of the encoded chunks following conversion to Base64 string formats.

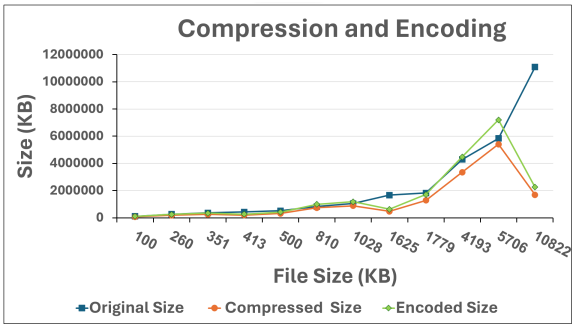

Fig. 8 File sizes before and after compression and encoding

By sending the chunks from the TypeScript client code one after another throughout invoking API procedures of the chaincode within the peer of organization 1 in the simulation network, Figure 9 illustrates how long it takes for both chunk sizes to save the file into the core network. Every file's duration, from transmitting the first chunk to committing the last, is measured in milliseconds and subsequently converted to seconds.

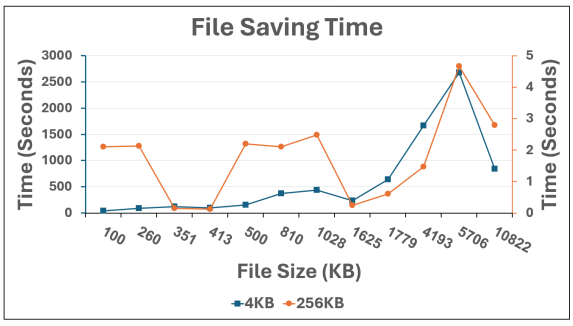

Fig. 9 File-saving time in the core blockchain network

5 Discussion

5.1 Blockchain Platform

Several blockchain platforms can be used to implement the suggested architecture. To determine which blockchain would be best for our job, we compared the Mutlichain and Hyperledger Fabric blockchains in Table 3. Transactions per second (TPS) are supported for both platforms; however, proper configuration and tweaking are mostly required. Hyperledger is an outstanding match for our work since it supports parallel transaction processing, which allows us to have a higher throughput in terms of TPS, and it offers a blockless design in terms of block size, also, supporting role-based access control (RBAC) fits our proposed work since enables organizations to grant peers access to the files based on their roles. This approach allows for the precise management of permissions and privileges. The comparison conducted through a review of Multichain white papers [19] and [16].

5.2 File Chunking

In the proposed work the file will be divided into segments or chunks in the edge side. chunking module can be either fixed size or variable size. In our framework the we use fixed size chunking. Choosing the chunk size is an important issue since (i) choosing the optimal chunk size is a critical factor in effective chunk distribution; (ii) furthermore, the performance and reducing latency in the core blockchain network while transforming and saving the chunks; (iii) It is also a key consideration in optimizing bandwidth usage; (iv) Additionally, it has a direct relationship with consistency and integrity in the chunk distribution process in the public ledger.

The chart analysis in figure 6 reveals that using 256 KB chunk size slightly outperformed 4 KB chunk size, albeit demonstrating a degree of symmetry or parity in many aspects.

5.3 Chunks Encoding

The base64 index table is used to convert the binary format that is fed into it into printable ASCII characters. It is possible to utilize many encoding algorithms to encode the file. choosing the optimal algorithm for our task based

Table 3 Mutlichain and Hyperledger Fabric Comparison

Feature Name	Multichain	Hyperledger
Scalability	Less	More scalable
Smart contracts	Yes	Yes
Code power	Simple scripting language	More powerful and flexible
Authorization	Permissioned blockchain,	Permissioned blockchain, (RBAC)
Block Capacity	Default block size 2 MB	Blockless architecture
Privacy	Greater control using private chains	Fine-grained control over data access
Appending new block	Traditional way	Support parallel transaction processing

on the highest performing methods. In terms of speed and performance, Base64 beats a lot of other algorithms, including UUEncode, XXEncode, USR, BINHEX, BTOA, BOO, and Quoted-Printable [20]. It also performs well on a variety of processor types and includes standard libraries for well-known programming languages including JavaScript, Go, Swift, PHP, Python, C#, and Java [21]. Thus, Base64 is an excellent fit for our framework since it allows us to use our framework on a variety of computers and programming environments, especially on the edge.

We encode the data to Base64 before transmitting it to the blockchain after transforming the file to binary format. Sending raw binary data has some drawbacks compared to using Base64, as a result of which our suggested work is compatible with blockchain platforms that allow text-based protocols and formats. When transmitting data over an unencrypted connection, Base64 encoding additionally helps in avoiding data modification and interception. Additionally, because JSON payloads and URLs support text data, it is more convenient to communicate data to an API using them. However, this requires adding certain overheads to the framework, such as more processing and about a 33% increase in the size of the original data. The framework's last phase will store the data as binary rather than Base64.

Performance is an important consideration; therefore, we examine the time spent on the edge side of the encoding process, considering the fact that using the compression approach helps minimize the additional file size that results from encoding the file. The encoding time, which takes about 3.4 milliseconds on average for the different chunk sizes we use in our tests for all files enrolled in the testing, is sufficient for both chunk size scenarios, as Figure 7 illustrates.

5.4 File Compression

An on-chain file saving is the goal of our endeavor. The ledger will eventually get larger because the files are kept directly on it. One useful method to take advantage of the blockchain's storage capacity is to compress files. We decide to compress the files on the client or edge side in order to minimize their size before saving them. In this study, the selection of an acceptable compression technique is primarily dependent on a few features, like minimizing the file size and maintaining the original compressed file without losing any of its contents.

Data compression algorithms come in two categories: "lossless" and "lossy". Text, strings, and programs can be compressed using lossless data compression techniques, while images, videos, and audio files can be compressed using lossy compression algorithms. We select lossless algorithms depending on the requirements indicated before.

Since our proposed framework is not meant to save a certain file type, selecting the optimal compression method will depend on the use case for the desired file type to store. Table 4 provides a list of possible techniques that can be used to compress files in the Portable Document Format (PDF), for instance. Since our goal is to minimize the size of the ledger, the compression ratio is the basis for choosing the best algorithm. Based on the results of the RasterLite2 benchmark [22], Zstandard could be one of the best options for our chosen case, which is text files.

The sizes of the original file, the compressed file, and the total size of encoded chunks in the file are all demonstrated in Figure 8. The figure shows that, despite the encoding process increasing the quantity of data, the encoding size for all used files is smaller than the original file size. This is due to

the compression process that takes place prior to the encoding process.

Table 4 Examples of lossless compression algorithms

Algorithm	Mainly Targeted File
Run Length Encoding	Simple graphic
Shannon-Fano coding	General-purpose
Zstandard	General-purpose
Lempel-Ziv-Welch (LZW)	General-purpose
Huffman Coding	Text data
Arithmetic Encoding Text data	
BZIP2	Text data
GZIP	General-purpose
PPM	General-purpose

5.5 Optimal Chunk Size Selection

Based on the aforementioned results, we infer that there is a little variation in the various performance measures that we ran while employing a 4 KB or 256 KB chunk size on the edge side. Conversely, the choice of a suitable chunk size is dependent on optimizing the file's performance efficiency during the saving process on chunks in the block transactions. According to the results, it took less time to save the file into blocks while utilizing a 256 KB chunk size as opposed to a 4 KB chunk size. According to Figure 9, saving a file in a block with a 256 KB chunk size often takes 1.75 seconds, but a file with a 4 KB chunk size typically takes around 614 seconds. This outcome derives from the concept that decreasing the chunk count is preferable to decreasing the number of calls to the chaincode's API.

6 Conclusion

In this study, we set out to investigate the potential and performance of on-chain file storage in a blockchain ledger. The proposed framework reveals that blockchain offers a promising solution for organizations that seek an immutable storage solution with the power of authentication and authorization that Hyperledger Fabric blockchain offers. Despite some limitations and constraints in terms of consumed time, performance, and leger size to deploy the framework, the overall performance demonstrates its potential as a viable alternative to traditional centralized storage systems. In addition to evaluating the performance of our framework, we also shed light on the critical importance of selecting the appropriate chunk size, and compression techniques for efficient file storage.

While our study provides valuable insights into utilizing on-chain solutions, it is not without limitations. Future research efforts should focus on addressing these limitations and exploring opportunities for further optimization and enhancement of file storage solutions, such as investigating the effects of redundancy of storage, ledger capacity, encoding and appropriate file indexing.

Finally, our research highlights the importance of blockchain technology in tackling the changing problems associated with data storage. Organizations can explore new avenues for secure, effective, and unchangeable data storage solutions by adopting secure and decentralized principles.

References

[1] A. Shahnaz, U. Qamar, A. Khalid, Using blockchain for electronic health records. IEEE Access **7**, 147782–147795 (2019). https://doi.org/10.1109/ACCESS.2019.2946373

[2] S. Loss, H.P. Singh, N. Cacho, F. Lopes, Using fiware and blockchain in smart cities solutions. Cluster Computing **26**(4), 2115–2128 (2023)

[3] O. Popoola, M. Rodrigues, J. Marchang, A. Shenfield, A. Ikpehia, J. Popoola, A critical literature review of security and privacy in smart home healthcare schemes adopting iot & blockchain: problems, challenges and solutions. Blockchain: Research and Applications p. 100178 (2023)

[4] M. Tmeizeh, C. Rodríguez-Domínguez, M.V. Hurtado-Torres, *A Survey of Decentralized Storage and Decentralized Database in Blockchain-Based Proposed Systems: Potentials and Limitations*, in *International Congress on Blockchain and Applications* (Springer, 2023), pp. 204–213

[5] F. Casino, E. Politou, E. Alepis, C. Patsakis, Immutability and decentralized storage: An

analysis of emerging threats. IEEE Access **8**, 4737–4744 (2019)

[6] E.S. Babu, B.R.N. Yadav, A.K. Nikhath, S.R. Nayak, W. Alnumay, Mediblocks: secure exchanging of electronic health records (ehrs) using trust-based blockchain network with privacy concerns. Cluster Computing **26**(4), 2217–2244 (2023)

[7] X. Yang, M. Li, H. Yu, M. Wang, D. Xu, C. Sun, A trusted blockchain-based traceability system for fruit and vegetable agricultural products. IEEE Access **9**, 36282–36293 (2021). https://doi.org/10.1109/ACCESS.2021.3062845

[8] V. Mani, P. Manickam, Y. Alotaibi, S. Alghamdi, O.I. Khalaf, Hyperledger healthchain: patient-centric ipfs-based storage of health records. Electronics **10**(23), 3003 (2021)

[9] F. Liu, C.y. Yang, J. Yang, D.l. Kong, A.m. Zhou, J.y. Qi, Z.b. Li, A hybrid with distributed pooling blockchain protocol for image storage. Scientific Reports **12**(1), 3457 (2022)

[10] Y. Chen, S. Ding, Z. Xu, H. Zheng, S. Yang, Blockchain-based medical records secure storage and medical service framework. Journal of medical systems **43**, 1–9 (2019)

[11] N. Alrebdi, A. Alabdulatif, C. Iwendi, Z. Lian, Svbe: Searchable and verifiable blockchain-based electronic medical records system. Scientific Reports **12**(1), 266 (2022)

[12] P.K. Sharma, J.H. Park, Blockchain based hybrid network architecture for the smart city. Future Generation Computer Systems **86**, 650–655 (2018). https://doi.org/https://doi.org/10.1016/j.future.2018.04.060. URL https://www.sciencedirect.com/science/article/pii/S0167739X1830431X

[13] O.I. Khalaf, G.M. Abdulsahib, Optimized dynamic storage of data (odsd) in iot based on blockchain for wireless sensor networks. Peer-to-Peer Networking and Applications **14**, 2858–2873 (2021)

[14] S.S. Arslan, T. Goker, Compress-store on blockchain: a decentralized data processing and immutable storage for multimedia streaming. Cluster Computing **25**(3), 1957–1968 (2022)

[15] G. Gürsoy, C.M. Brannon, M. Gerstein, Using ethereum blockchain to store and query pharmacogenomics data via smart contracts. BMC medical genomics **13**(1), 1–11 (2020)

[16] E. Androulaki, A. Barger, V. Bortnikov, C. Cachin, K. Christidis, A. De Caro, D. Enyeart, C. Ferris, G. Laventman, Y. Manevich, et al., *Hyperledger fabric: a distributed operating system for permissioned blockchains*, in *Proceedings of the thirteenth EuroSys conference* (2018), pp. 1–15

[17] V. Trón, The book of swarm: storage and communication infrastructure for self-sovereign digital society back-end stack for the decentralised web. V1. 0 pre-Release **7** (2020)

[18] J. Benet, Ipfs-content addressed, versioned, p2p file system. arXiv preprint arXiv:1407.3561 (2014). Accessed: (2024-2-26)

[19] MultiChain. Multichain private blockchain — white paper. https://www.multichain.com/download/MultiChain-White-Paper.pdf. Accessed: (2024-2-15)

[20] M. Mustapa, A. Taliang, A. Iskandar, et al., *Comparison of Encoding and Decoding Methods for Binary Files*, in *Journal of Physics: Conference Series*, vol. 1364 (IOP Publishing, 2019), p. 012024

[21] W. Muła, D. Lemire, Faster base64 encoding and decoding using avx2 instructions. ACM Trans. Web **12**(3) (2018). https://doi.org/10.1145/3132709. URL https://doi.org/10.1145/3132709

[22] Gaia-GIS. Benchmarks (2019 update). https://www.gaia-gis.it/fossil/librasterlite2/wiki?name=benchmarks+(2019+update) (2019). Accessed: (2024-2-25)