School of Electronic Engineering

"Blockchain and its role in file storage"

Project Report

Student Name: Michael O'Hara
ID Number: 16414554

August 2021

MSc in Electronic and Computer Engineering (IoT)

Declaration

I hereby declare that, except where otherwise indicated, this document is entirely my own work and has not been submitted in whole or in part to any other university.

Signed: Michael O'Hara **Date**: 02/09/21

Blockchain and its role in file storage (August 2021)

Michael B. O'Hara, Michael.ohara29@mail.dcu.ie, 16414554, Dublin City University

***Abstract*—File storage is an integral part of every electronic systems architecture. This is how the users of the system will store and retrieve data that has been introduced to the system. The methods for storing files such as a centralized database have remained largely unchained as they have shown to be effective. However, this technology is slowly becoming obsolete and replaced by new methods. One of the newer methods for file storage is using blockchain to store the files and their data. Can blockchain offer a reliable replacement for the traditional database? If so what facets of blockchain lend themselves to being a reliable replacement for the older technologies.**

***Index Terms*— Internet of Things, Blockchain, Data Storage, Confidentiality, File Storage**

I. Introduction

Electronic systems rely heavily on file and data storage in order to main peak operating efficiency, but what is data storage? *"Data storage is the use of recording media to retain data using computers or other devices and the most prevalent form is file storage"* [1]. Now that the definition has been derived, how is this data stored on in relation to big web application and computer systems? The most prevalent way to store the data as mentioned in the definition above is file storage. *"File storage is a hierarchical storage methodology used to organize and store data on a computer hard drive or on network-attached storage (NAS) device"* [2]

The methods used to store data and files can vary from system to system. The most common method of storage adopted by systems is a centralized database kept on premises or on the cloud. However, there is a problem with these methods of file storage. Due to the centralized nature of Databases, if one becomes compromised, all data contained within will also be compromised. This means that security is the main priority for these systems.

Inadequate data protection may lead to personal or private information to be leaked to outside parties. If a system is not capable of keeping data secure, it will cause the user to distrust the system and turn to other methods of data storage. The way the data is attained by the outside parties is though a system attack on the system using software known as malware attacks.

One of the more recent instances of this type of attack was carried out on the Health Service Executive (HSE) on Friday the 28th of May. This breach saw data relating to approximately 520 patients along with many corporate documents were compromised. A small portion of this data has appeared on the dark web and requires another special program to access the data. This caused the HSE to have to rebuild their file and data storage from the ground up and rigorously test the new systems integrity.

II. Prior work

One strategy to combat attacks like these could be the implementation of Blockchain as a file storage mechanism. While it is technologically feasible to store the entire file on-chain, it is a difficult task as there many limitations such as large data demands and access latency as the number of files stores grows.

One of the more advanced solutions that tries to tackle this problem is built using IPFS (Interplanetary File system) alongside Blockchain. [3] This system does accomplish the goal of providing a method for storing files however, the bulk of the work is carried out by IPFS by "*storing transaction data in IPFS and storing the hash returned by IPFS into the block of blockchain*". This was done to reduce the amount of storage required for the transactions on the chain, however from the paper it is evident that the system has been designed for the storage of image and video files. The purpose of this project, however, is to store which is why the IPFS was not adopted as Hyperledger fabric which will be discussed later, suited the needs of the project. One of the concepts utilized by IPFS could be incorporated into the system. This was the storage of the hash of the file to reduce the size of each transaction on the chain.

Another factor that needs to be addressed is that of the failure of a node on the network and what if anything will happen to the data that is storage on said node. There are two methods of redundancy for when there is a failure of a data node. These being "*the original file can be reconstructed using the data present in the surviving nodes, or the process to create adequate redundant data so that the failed nodes can be repaired."* [4]. The solution devised by the authors of this paper was a mathematical model that would work in a *(n, k, d)* regenerative code, designed to break the source file into M data blocks and coded into *n* nodes which each contain α nodes. The receiver can recover the original data entirely through *k* nodes. If a node fails however, it can be repaired by connecting and $d\ (d \geq k)$ surviving nodes and downloading β blocks. This was a key factor in deciding upon which blockchain technology is used, the Hyperledger Fabric network has a built-in redundancy for the failure of a node on the network and allows for automatic and secure recovery of data.

Arguably the most important factor, is the multitude of open challenges to using blockchain as a replacement for a traditional file storage. The key challenges to this are scalability, application, and big data issues. The scalability issues show that bottlenecks can still occur when the network is performing high latency I/O tasks and downloading and resolving transactions. [5]. This will be a key challenge of this project as if the system cannot operate efficiently at peak load times it cannot be deemed as a suitable replacement.

III. Software And Methods Used

As mentioned in the previous section a method which could be used as a more secure and suitable option for file and data storage, is to implement Blockchain. This means that the content, interactions, and log of edits of these files will be stored on-chain. This section will serve to describe the underlying technologies and the methods used in this project to try to achieve this goal.

The first major component used in the implementation of the project is a Linux based virtual machine (VM). This would allow the running of a Linux machine inside of a windows-based machine. This is done as the second required technology, the blockchain network, which is built using Hyperledger Fabric. As it is more suited to running on this Operating System (OS). It is possible to run Hyperledger on a windows machine however the installation process has significantly more steps, which is why the VM was opted for.

The virtual machine that was chosen for this project was the Virtual Box VM provided by the Oracle Corporation. It is a general purpose full virtualizer for x86 hardware, targeted at server, desktop, and embedded use. [6]. In order to get this machine up and running there is a number of smaller tasks that need to be carried out on the host OS, in this case windows. The first option that needs to be enabled is "VT-x/AMD-V" in the BIOS of the windows device, this allows the host OS to run a 64-bit guest OS.

Next, virtual box needs to be installed along with an ISO file of a Linux distribution. For this project an ISO of Debian was used. [7] Now the actual VM can be initialized, this includes applying the ISO file and configuring the specifications of the machine. The machine used for the implementation of this project needed to have a large storage and memory capacity. This is due to the fact that in the project implementation the blockchain network will be hosted on the machine and the network needs to remain running for long periods of time during testing and usage.

Once the machine has been created, the code prerequisites also need to be installed on the machine. Some of these prerequisites include the GO [8] and the node.js [9] programming languages. These are a requirement for the Hyperledger Fabric network to run successfully. A text editor is also installed in order to facilitate the editing of chain code and node.js files to allow for the interaction with file and storing of file contents on-chain.

Hyperledger fabric is an open-source, non-cryptocurrency based distributed ledger software. It serves as the foundation for the development of applications with a modular architecture. This allows the network to deliver confidentiality, resiliency, flexibility, and scalability [10]. This open-source software was selected for this project, for the fact that it allows for "chain code" to be packaged and applied on the network and invoked when needed. This would allow for the chain code to be written in such a way that it would be able to accommodate files and their content.

The Hyperledger Fabric network requires a few dependencies to be installed. As mentioned previously Go is required. It can be installed by downloading and unpacking the tar.gz archive. This is then added to the path of the VM to allow it to be visible to programs on the VM. Go language by Google is required by Hyperledger fabric for the packaging of the chain code for the network. The chain code is the main business logic for applications and how they can interact and add to the ledger.

Another dependency that needs to be installed prior to the network being operational is Docker. This is a tool required for deploying and managing containers. This is required to deploy the Fabric network, as it is packaged as set of docker images and then is run as Docker Containers. [11] When the network is being instantiated, the outdated containers will be removed and the new set of docker images is packaged.

The final dependency for this can be bundled as a number of dependencies, Node.js and its package manager need to be installed in order to get the required packages to allow the code to interact with the files to run. The required package manager that is associated with node.js is NPM. This will be used to install the required packages for the network such as the package for reading the file's content and converting it to a string to be added to the network and the package for hashing his content to be stored securely.

The required packages are 'SHA256' and 'READLINE'. These packages are required for creating the hash of the files content and reading in the file contents respectively. The READLINE package provides an interface for reading data from a readable stream one line at a time. [12]

The 'SHA256', also known as the Secure Hash Algorithm – 256 is a key requirement to be able to store the contents of the file securely on chain. It does this by creating a 256 bit or 32-byte hash which is almost unique. It is almost unique in the sense that the same input will consistently produce the same hash. However, hashing is a one-way function which means it is suitable for being used in a project such as this as it can be used to ensure the integrity of the data. This can be ensured as it would require a brute force strategy to decode the hash.

After the dependencies are installed and the Hyperledger Fabric network can be cloned onto the virtual machine and can be initialized. The initial network is designed to be ran using invoke.js which will run through several tasks such as calling the chain code and passing it arguments which create a car object and add it to the network. This would be the starting point for allowing the network to interact with and manage files.

To begin altering the network to be able to handle files, the chain code needs to be changed as this is the main business logic for the application. The chain code needs to be modified to handle file objects. The accompanying attributes of the file class are the name, file extension, owner, content and finally timestamp. These attributes are set upon the calling of the create File method which in turn submits a transaction to the chain

```
type File struct {
        Type string `json:"filetype"`
        Name string `json:"name"`
        Owner string `json:"owner"`
        Content string `json:"content"`
        TimeStamp string `json:timestamp`
```

Another change that needs to be made to the chain code is that the methods for editing the file owner, name and extension need to be created. Each of these methods will take in a key which will be a file number, this is assigned when the file is initially added to the chain. Using this file number, it will query the chain to check if the file exists. If the file does exist on chain the then the corresponding attribute will be assigned the value that was passed into the method. After the new assignment the chain code will then read the timestamp on the VM and assign it to the timestamp attribute of the file to reflect when the change was made to the file on chain. The code below is for changing the owner of a file on chain.

```
file, err := s.QueryCar(ctx, fileNumber)

if err != nil {
        return err
}

file.Type = newName

timestamp,e := ctx.GetStub().GetTxTimestamp()
t := time.Unix(timestamp.Seconds, int64(timestamp.Nanos)).String()
if e != nil {
        return fmt.Errorf("Failed to fetch time. %s", e.Error())
}
file.TimeStamp = t
```

The next key file that needs to have changes made to it is the invoke.js file. This file is used for invoking transactions on the chain. Firstly, the file will check that a command line argument is passed to be added to the chain. If there is no user added file, the code will display a message on how to add another file and then will run through creating a few dummy files and then run the abovementioned chain code functions on them. This file needs to have the ability to read in command line arguments, which in this case are the files to be added to the network. This is done by using the 'READLINE' package that was installed as one of the dependencies to read in the files content and convert them to a string. File to be read in are passed as command line arguments.

When a file is passed in, it will call a method which will read the file name and extension to be used in the transaction payload. The command line argument which contains the filename will be taken and split into two separate strings. The split will occur at the '.'. The result will be two strings, one being the filename and the other being the file extension.

```
function inputFile(filetohash){
        var file = filetohash;
        var fileparts = file.split('.',2);
        var name = fileparts[0];
        fileparts[1] = "." + fileparts[1];
        var extension = fileparts[1];
        return fileparts;
```

A similar function is also used to attain the username of the user calling the invoke.js code based on the way Linux handles command line arguments.

Once the content of the file has been read in successfully and converted to a string the next step is to convert it into a hash to prepare it for storage on chain. This is where the 'SHA256' package is used. The package will create the 256-bit hash of the content of the file. This hash was chosen as up until recently there was no way to decode it, only recent improvements to CPU and GPU hardware have allowed for it to be possible to decrypt a block of SHA256. However, the hashing algorithm used by this code can be substituted with another algorithm, the prototype of this system used 'SHA1'. The resulting hash is then passed back to be added to the transaction payload and added to the chain.

```
function hashFile(stringtohash) {
    var x = stringtohash;

    var hash = sha256(x);
    //console.log(hash);

    return hash;
```

After all elements of the file have been attained in the format that is conformant to the payload of the transaction, the items will be added to an array and returned to the main function of 'invoke.js' and then the create transaction method from the chain code is invoked and the values from the array are passed into it. It will also generate the Unix timestamp at the point of the methods invocation. As mentioned above once this has taken place a number of hardcoded dummy files will demonstrate the methods for editing the other attributes and storing the subsequent edited file on the chain

.

IV. Testing Of Solution

Once the system is fully the developed and operating successfully, the results gathering can begin. This is to benchmark how well the system works. A number of different types of tests were carried out on the network, from benchmarking performance to testing the functions relating to files.

The first set tests to benchmark the performance of the system would be classed as unit tests to ensure that the functions of the system are performing the correct actions. These functions include the reading in of the file from a command line argument and getting the details of the user interacting with system, reading the content of the file, hashing the content, and performing edits on the file.

These tests are run on a dummy file that is passed in with the content being that of the readme of the Hyperledger fabric network which explains how to operate the network. It will print to the screen an array containing the file name, extension and content and the user who is interacting with the system, which in the case of this test will be the user logged into the VM. This array is the payload which will be stored on the chain.

```
Filename: Hyperledger
File extension: .txt
Current User: mike
Resulting hash: 9dbf35c7ab9bd5319f56dd602766ad2a28bcf8d0a4117d24eecb5fe301448793
Payload content:
[
  'Hyperledger',
  '.txt',
  'mike',
  '9dbf35c7ab9bd5319f56dd602766ad2a28bcf8d0a4117d24eecb5fe301448793'
]
```

The next test that is carried out on the chain is a test to verify the integrity of the content of the file by comparing the hash of the content to the hash of a modified version of the content. As mentioned previously the hashing function is a one-way function which will produce an almost unique hash when passed a string. In essence it means that it the string passed to the function is unchanged the resulting hash will always remain the same. This means that the hash generated when the file is added to the chain can be compared to the hash when the file is being interacted with. If the two hashes are identical this ensures that the integrity of the content of the file remains intact. This is carried out by calculating the hash of a file when adding it to the chain, while also calculating the hash of the file with minor edits made to it to show that the newly acquired hash is completely different to the original. If and only if the hashes are the same, then it can be said that the data on chain has not been tampered with.

The next test that is carried out on the chain is to time how much time elapses from the invocation of adding a file to the chain to the completion of the function. This will be checked with a number of files of varying lengths of content. This would allow for the benchmarking whether the system would have a large degree of latency when trying to add a large file to the chain.

The expected results of these test were that it would show that the larger the file being added to the chain would take a longer time to complete due to the time needed for the buffer to read in the entire content and hash the resulting string. Leading to a reduction in network performance until the larger file operation has been completed.

However, the results of the test show that files two different lengths, one being significantly longer than the other both were added to the chain in very similar timeframes.

The first file was the readme of the Hyperledger fabric network used in the previous test. As can be seen below the file finished reading in and was added to the chain in a time of 2.530s.

```
Filename: Hyperledger
File extension: .txt
Reading in the file, time elapsed: 2.530s
```

The second file was a text file containing 6500 lines of Linux kernel source code stored as a text file which is significantly longer and larger in size than the previous file. As can be seen below this file was read in and added to the chain in a time of 2.448s. This file was marginally faster than the previous file.

```
Filename: linuxcode
File extension: .txt
Reading in the file, time elapsed: 2.448s
```

The results of the tests performed show that the decision to implement blockchain as a file storage solution lends itself particularly well to files which comprise of text only. It also lends itself well to file which do not need to be edited overly often (the reason for this will be discussed in the next section).

The final test on the solution was to venture into an even more secure method of storing files on the chain than previously described. This method would involve creating a hash of not only the content of the file being stored but also the file name and extension and the user adding the file to the chain as well. This would then be split in to four even parts each 16 characters long and these would be the content of the transactions payload.

This was done to test if it could add an extra level of security to the storage of the file as the resulting splits on the hash would be stored in a different order to the original hash. The user that creates this transaction would be the only one able to verify the original hash as there would be 24 combinations of the hash when trying to recreate it from the four parts. In order to compromise the file, the correct hash would need to be found first and then broken. However, attempting an attack like this would be time consuming and not guaranteed to even succeed.

V. Analyzing & Evaluating The Solution

One of the things discovered after the completion of the testing phase of the project was the fact that the blockchain network is not a suitable replacement for traditional file storage methods. There are two main reasons for this, the first is that the way a blockchain network is built is "*based on a sequence of blocks which carries a certain amount of information. This volume is limited to the framework which is being used*." [13]. This means that storing information about the transactions on the network may be feasible but storing actual files and data may be infeasible as they are larger than the volume allowed by the network

As the file sizes get larger, they will surpass the volume limit meaning that to add the file content to the network multiple transactions will be required. This means that the more transactions are required the longer it will take to complete with the possibility of several hours or several days. If every file on chain is of a certain size, trying to access any file on the network would take too long to be a viable option in a commercial landscape.

Another feature of the blockchain that was initially deemed as an advantage in the region of file storage was the immutability of the chain. In the context of blockchain, immutability means outside parties cannot gain entry to the chain and delete any of the information stored on chain. As was seen during the lifecycle of the project it had become the second disadvantage when storing large amounts of files on chain.

When accessing and editing a file on a blockchain system, every instance of the modified file and previous versions will remain on the chain. On a system with a large number of files, which are frequently edited there is the potential for the blockchain to become increasingly large and difficult to manage. This means that any privacy concerns that need to be amended in one version of a file are basically moot as the original file cannot be removed from the chain which basically leads to this being a breach in certain polices such as GDPR.

During the duration of the project, it was discovered that when the solution developed is running it can only handle the reading in of one file at a time. The network will almost pause while completing this action and no other methods can be executed. This pause is longer depending on the file size. However, the solution is relatively fast at reading in the files in comparison to other methods that were test such as developing the solution in Java or Golang.

Now if this was scaled up to be applied to a small business this delay would grow even larger if multiple people were trying to add a file to the chain at the same time. This does not suit the live and fast paced environment of a business setting as the people of the business would be able to complete any tasks the file is needed for while waiting on the file to be added to the chain.

In order for the solution offered by this project or indeed any other approach in the future to become a viable replacement for the traditional file storage methods, it needs to overcome one major issue. This issue is the amount of data required to successfully operate a network at peak load times. As it stands right now the amount of data required for operating a Bitcoin blockchain is upwards of 220GB and increasing rapidly. This level of data requirement for a small to medium business is fair too costly to be in any way feasible to maintain on day-to-day basis.

Evaluation: To evaluate the system it must be vigorously tested, and the results gathered and evaluated.

As previously mentioned, the system needs to remain running for long periods of time, in order to evaluate if the system was capable of this the VM was started, and the network was instantiated and left to run over the course of two days. During this the network ran and completed a large sample of dummy transactions such as adding a template car object (this test was prior to any development of new chain code). The network remained running and suffered only minor slowdowns but overall operated with any acceptable level of efficiency. This means the VM, and Hyperledger Fabric network run in tandem is a suitable method of hosting the system.

The next method of evaluation was the speed and ability to handle the files of the system. As seen in the testing section of this document the system was able to read in the files significantly faster when compared with reading in files in other programming languages such as Golang and Java. The results of reading in text files such Hyperledger's ReadMe were in the range of 2 to 2.5 seconds. A basic test was done to read a file in on Golang and Java which each took 4 and 6 seconds respectively. This made node.js the optimum choice for the system.

The operation of reading in of file while fast is limited to reading in one at a time and halting any other operations until the oldest operation is complete. A number of methods and packages were tested to try to overcome this design challenge to no avail, however. There was no way in the project timeframe to develop a workaround for this. However, this offers a future avenue for more work to be carried out on the system.

VI. Conclusion

When determining if blockchain could be a viable solution for replacing the traditional methods of file storage, one thing that needs to be considered is the ability to integrate with legacy systems. The majority of companies across all major infrastructure sectors are still running old software systems, which are deemed outdated and too costly to replace. Blockchains are not able to be incorporated with legacy systems, in order to introduce one into an already established infrastructure the chain would have to be built up from scratch, which would be a major expense. The reason for this is two-fold, it is too costly to replace the legacy system as a whole as the codebase of the system is outdated and developers who work with the codebase are scarce, such as ADA or Cobalt.

The other reason it would be costly is the blockchain system is quite costly as the require a lot of power in order to run the network. "*Every time the ledger is updated with a new transaction, it means spending a lot of energy."* [14]. This would be costly for a business when compared to traditional storage methods such as a centralized database. This just isn't a feasible switch to make for every business out there no matter the size.

From the results of this project, it is believed that the technology is seemingly too new to be applied in mainstream industry as a replacement for traditional file storage. The technology is only a decade old as it stands and has not had to time to mature like other methods of file storage. Due to this lack of maturity, it is seen that there a number of major players in the market of developing the technology such as Hyperledger and Ethereum, yet they are not working together so their approaches differ widely so there is no standardized approach.

As was seen from the development and results of testing this project storing files on blockchain while possible is not as suitable as the methods currently employed in industry today such as centralized databases and cloud storage architectures. However as was seen with some of literature reviewed in this report and over the course of the project, there have been several advancements made towards making blockchain a suitable solution for storing files.

VII. References

[1] "What is Data Storage? – Enterprise IT Definitions." https://www.hpe.com/us/en/what-is/data-storage.html (accessed Aug. 19, 2021).
[2] "What is File Storage | IBM." https://www.ibm.com/cloud/learn/file-storage (accessed Aug. 19, 2021).
[3] R. Kumar and R. Tripathi, "Implementation of Distributed File Storage and Access Framework using IPFS and Blockchain," in *2019 Fifth International Conference on Image Information Processing (ICIIP)*, Nov. 2019, pp. 246–251. doi: 10.1109/ICIIP47207.2019.8985677.
[4] W. Liang, Y. Fan, K.-C. Li, D. Zhang, and J.-L. Gaudiot, "Secure Data Storage and Recovery in Industrial Blockchain Network Environments," *IEEE Trans. Ind. Inform.*, vol. 16, no. 10, pp. 6543–6552, Oct. 2020, doi: 10.1109/TII.2020.2966069.
[5] H. Huang, J. Lin, B. Zheng, Z. Zheng, and J. Bian, "When Blockchain Meets Distributed File Systems: An Overview, Challenges, and Open Issues," *IEEE Access*, vol. 8, pp. 50574–50586, 2020, doi: 10.1109/ACCESS.2020.2979881.
[6] "VirtualBox – Oracle VM VirtualBox." https://www.virtualbox.org/wiki/VirtualBox (accessed Aug. 21, 2021).
[7] "Debian -- Reasons to Choose Debian." https://www.debian.org/intro/why_debian (accessed Aug. 28, 2021).
[8] "The Go Programming Language." https://golang.org/ (accessed Aug. 26, 2021).
[9] Node.js, "Node.js," *Node.js*. https://nodejs.org/en/ (accessed Aug. 26, 2021).
[10] *Hyperledger Fabric*. Hyperledger, 2021. Accessed: Aug. 21, 2021. [Online]. Available: https://github.com/hyperledger/fabric
[11] "Tutorial: Hyperledger Fabric v1.1 – Create a Development Business Network on zLinux · CATechnologies/blockchain-tutorials Wiki," *GitHub*. https://github.com/CATechnologies/blockchain-tutorials (accessed Aug. 22, 2021).
[12] "Readline | Node.js v16.8.0 Documentation." https://nodejs.org/api/readline.html (accessed Aug. 28, 2021).
[13] "How to Use Blockchain to Store Data - Merehead 3263," *Merehead*. https://merehead.com/blog/how-to-use-blockchain-to-store-data/ (accessed Aug. 26, 2021).
[14] "Top Disadvantages of Blockchain Technology," *101 Blockchains*, Apr. 17, 2020. https://101blockchains.com/disadvantages-of-blockchain/ (accessed Aug. 27, 2021).