MTFS: Merkle Tree based File System

1st Jia Kan
Department of Electrical and Electronic Engineering
Xi'an Jiaotong-Liverpool University
Suzhou, China
Jia.Kan17@student.xjtlu.edu.cn

2nd Kyeong Soo Kim
Department of Electrical and Electronic Engineering
Xi'an Jiaotong-Liverpool University
Suzhou, China
Kyeongsoo.Kim@xjtlu.edu.cn

***Abstract*—The blockchain technology has been changing our daily lives since the cryptocurrency *Bitcoin* [1] was invented and released as open-source software by an unidentified person or group using the name of Satoshi Nakamoto in 2009. Of many applications which can be implemented based on the blockchain, storage is an important one, a notable example of which is the InterPlanetary File System (IPFS). IPFS is a distributed Web based on a peer-to-peer hypermedia protocol to make the web faster, safer, and more open and focuses on public accessible files. To provide a solution for private file storage in the blockchain way, in this thesis we propose a Merkle Tree [2] based File System (MTFS). In MTFS, the blockchain is more than a trust machine; it is an abstract of a cluster system. Distributed random nodes form a tree network [12] cluster without a central controller to provide a secure private storage service and faster message propagation. Advance Proxy Re-Encryption (PRE) [10] algorithm is applied to guarantee secure file exchanges under permission. Merkle Tree [2] will make sure the files are distributed balanced among the service nodes. The proposed MTFS can be used not only for personal file storage and exchange but also for industry requiring mutual trust in file uploading and downloading in making contracts like insurance.**

***Index Terms*—Blockchain, Private file system, P2P Network**

I. Introduction

The early Internet was created and used by academia mainly for research purpose: The email was used to exchange ideas, and the file transfer protocol (FTP) was used to exchange data like software packages and experiment results. Note that centralized file servers then couldn't meet user demands all the time due to their limitations in network bandwidth, input/output (I/O) speed, and/or storage capacity. There are two major use cases for FTP: Public file distribution and private access. In the following years, hypertext transfer protocol (HTTP) replaced FTP for public file distribution as the world wide web (WWW) and browser technology became widely popular. The concept of mirror was also used to better serve users with higher speed from mirror servers geographically nearer to users than the original server. Later the peer-to-peer (P2P) technology brought another revolution, where a client downloading contents also serves other clients with downloaded contents, named a peer.

On the other hand, the development of private data storage has been taking a different path: FTP/secure FTP (SFTP) can be used for private data exchanges, but, because they are not supported by Windows File Explorer, the common internet file system (CIFS), a dialect of server message block (SMB) protocol, is used in private file storage purpose more often, especially in enterprises and commercial organizations. Due to the insufficient performance of CIFS/SMB, more powerful and user-friendly commercial applications like Dropbox, Google Drive, and Baidu Yun appeared, and similar services based on cloud also were directly integrated into operating systems like Microsoft OneDrive and Apple iCloud. BitTorrent Lab surprised us by introducing BitTorent Sync (BTSync; now Resilio Sync), which is a Dropbox-like application without the requirement of a centralized server. Using existing P2P network, BTSync gets files synchronized among personal devices.

The blockchain technology brought us into a new era in private data storage as well. InterPlanetary File System (IPFS) [8] intends to build a new distributed web. It could be a grand new architecture based on P2P network, and contributor would be rewarded with actual compensation, e.g., tokens. The public file will be hosted in a P2P network.

Merkle Tree [2] based File System (MTFS), which we propose in this thesis, is to provide a solution for private file storage based on P2P network and the blockchain technology. It uses the asymmetric cryptography including Proxy Re-encryption (PRE) [10] technology to provide secure and reliable private storage under permission. Random nodes form a cluster without a central controller to provide a private storage service. Merkle Tree [2] will make sure that the stored contents are distributed in a balanced way among the service nodes. A tree network [12] technology is also used in broadcasting blockchain messages, content storage, and playing the role of Distributed Hash Table (DHT) network for routing. This replaces the Kademlia (KAD) algorithm [3] to provide a similar function, but nodes are assigned in more static position than KAD [3]. Also the convergence of message broadcasting in the tree network is faster than the gossip protocol [4].

II. Use Case

Cloud service for private files is common used in enterprise, school, government and even personal. Maybe most of people is using the cloud service now without awareness.

Let me take a example about specific use case in industry: insurance. People buy insurance to cover the accident loss in life. However insurance company is required user to provide information. After risk evaluation, insurance company decided to offer the contract. However, those submitted file is only kept

by insurance company's private server. If any important file missing, it will benefit the insurance company side, unless the applicant kept the original reception. Blockchain based private file can provide this level trust for both side. When applicant request for the insurance service, all the supported material are provided in a restricted network space, it's immutable and reliable.

III. Literature Review

The architecture of system changes back and forth. At the beginning computer is rare and huge machine for scientists. It's required to shared the computation resource, and users use console to connect to the machine with their own console. It's a pure centralize architecture. Personal Computer(PC) changes this, as PC gets popular and powerful, it's used for both client and server. Usually client requests for service and server provides service, which is called C/S mode. P2P changes this too. In PC era, software like BitTorrent proposed a way to boost download speed. In the scarcity of server resource, a lot of clients want to download the content, they can exchange partial downloaded content to speed up the whole downloading process. It's a decentralized architecture works for PC Internet era. Story changes again when the mobile Internet gets popular. The power supply of mobile Internet is limited while the data rate of wireless communication is expensive. In this case most of mobile downloading prefers HTTP way rather than P2P. It's observed that C/S model comes with the advantage of energy saving. To build a P2P network, it's not necessary to make every client as a contributor. P2P technology can be used to get servers connected for the better balance allocation of resource.

A. Proxy Re-encryption

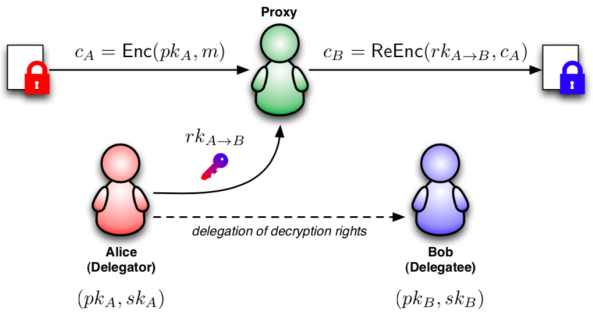

Fig. 1. Umbral proxy re-encryption

Proxy Re-encryption (PRE) [10] is the advance topic from Asymmetric cryptography. A proxy re-encryption [10] is generally used when one party, say Alice, wants to reveal the contents of messages sent to her and encrypted with her public key to a third party, Bob, without revealing her private key to Bob. Alice does not want the proxy to be able to read the contents of her messages. Alice could designate a proxy to re-encrypt one of her messages that is to be sent to Bob. This generates a new key that Bob can use to decrypt the message. Now if Alice sends Bob a message that was encrypted under Alice's public key, the proxy will alter the message, allowing Bob to decrypt it. This method allows for a number of applications such as e-mail forwarding, law-enforcement monitoring, and content distribution.

Umbral [11] is NuCypher's proxy re-encryption scheme. With Umbral, Alice(the data owner) can delegate decryption rights to Bob for any cipher text encrypted for her, through a re-encryption process performed by a set of semi-trusted proxies. When a threshold number of these proxies participate by performing re-encryption, Bob is able to combine these independent re-encryptions and decrypt the original message using his private key.

When the amount of proxy is set to one, Alice can generate the cipher text and the capsule. The cipher text and the capsule can be store on node in the cluster. When Alice decides to reveal the file content to Bob, she is going to re-encryption the capsule with Bob's public key. The capsule is small in size. Bob is able to decrypt the cipher text with the new capsule and his own private key. Actually, Alice can give file to a batch of users with the existing cipher text on the cluster, she only needs to perform limited computation and cost little bandwidth to finish the tasks. Compare to the asymmetric cryptography way, much energy and network traffic is saved.

According to Figure 1, Capsule B is generated from Capsule A with rk A to B, rk A to B is generated by Alice, and Bob is able to retrieve the content from Proxy with his private key and Capsule B. PRE [10] technology is very useful with mobile device and cloud service. As we know, mobile device use wireless network and battery, it has limited bandwidth and power. Because of this, the cloud service is a strong supplement for mobile device. PRE [10] enables remotely controlling of file content permission using pure cryptography method.

B. Git

Git is a distributed version control system. It's the most widely used collaboration tool in modern software development, and is very famous for the distributed system design. It's a classic design reference for the file system and version control.

Version control is a system that records changes to a file or set of files over time so that you can recall specific versions later [13].

In git internal, there are two types of objects: blob and tree. The blob object represents the file's content, while the tree object represents the folder. The object file name is named by content's file hash. When running command 'git init' to initialize the git repository, the current folder's tree object is not created yet. The first tree object is created when you committing the first file into the version control system, as the first blob object creating. Figure 2 shows the tree and blob objects consist the folder and file structure.

As user modify the file content(showed in Figure 3), new blob is generated. The tree object updated new blob, also

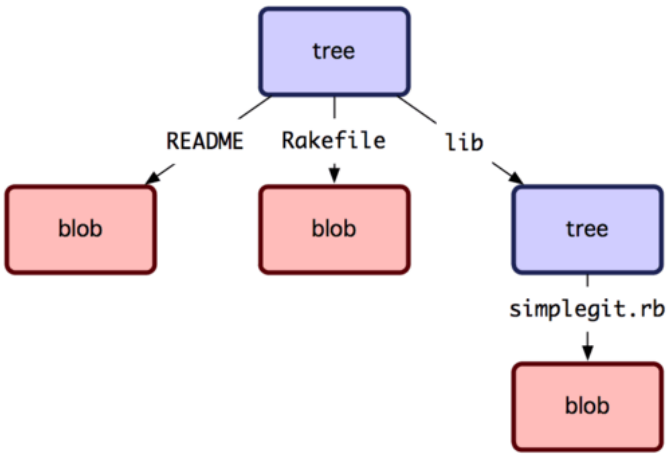

Fig. 2. Git internal objects

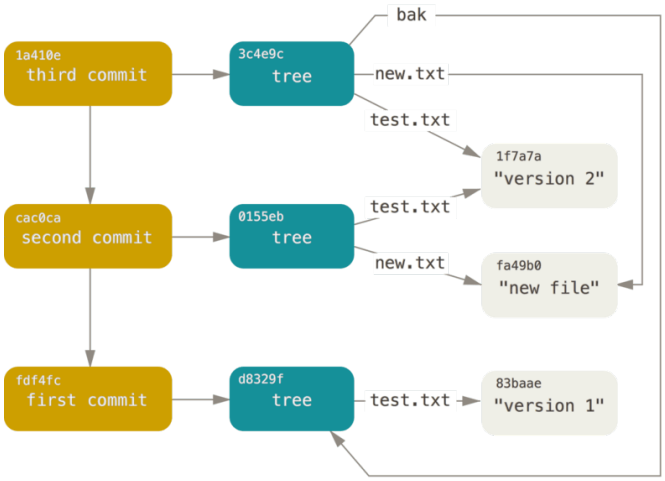

Fig. 4. Git commit objects

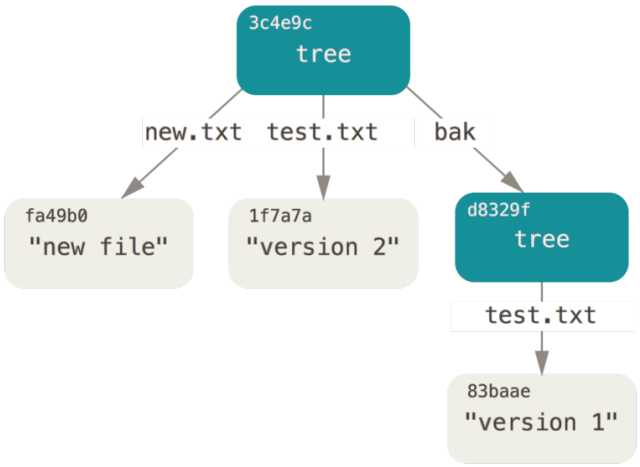

Fig. 3. Git Version with tree and blob objects

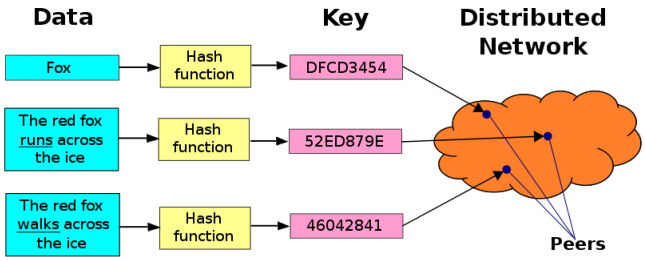

Fig. 5. Distributed Hash Table

new tree object is generated. So far there is snapshot of each version, but there is no relationship between versions. Commit object is another git hash-object type(showed in Figure 4), which link different version of snapshots, contains information about the root tree object, commit user information, and changed files overview.

C. DHT/KAD

DHT stands for Distributed Hash Table(showed in Figure 5), Kademlia (KAD) [3] is the implementation of DHT from NYU. DHT network make BT trackless. From the name it's showed that DHT works like a Hash Table. Hash Table is collection of data structures to hold data as key-value pairs. DHT is the distributed version of data structure, the data is stored on more than one computers connected by network.

BitTorrent uses a 'distributed sloppy hash table' (DHT) for storing peer contact information for 'trackerless' torrents. In effect, each peer becomes a tracker. The protocol is based on KAD [3] and is implemented over UDP.

In BitTorrent, each peer is randomly assigned an identity(160 bit binary). It's same length as the hash to download. Rule was set that the identity which is most close to hash will keep the host info, acting as the tracker for file. After several query, the downloader will find the peer with file downloading information.

In DHT network it provides a key/value database for short information. The message must be easily to copy or transfer. Peer identity is not equality well-distributed balanced at the identity name space. It's impossible to store big amount of data on the peer.

D. Gossip protocol

A gossip protocol [4] is an algorithm that social networks disseminate information or how epidemics spread. Peer-to-peer communication use gossip protocol [4] to broadcast message in the network.

Most of P2P network is interested in how to find the node(routing) and how to find the file(group-wise). Blockchain application, like Bitcoin [1], use P2P technology for communication. Differently, blockchain is more interested in how to get the information spreading around the world [7], to get everyone agree on the same, to form the consensus. Gossip protocol [4] is used by Bitcoin [1] on P2P network, to

propagate the transaction and new block discovery information worldwide.

E. Proof of replication

IPFS [8] is the Distributed Web, for public file hosting. Filecoin based on IPFS, try to solve the incentive problem. In technical report Proof of Replication [9] it lists kinds of proofs including Provable Data Possession(PDP), Proof-of-Retrievability(PoRet), Proof-of-Storage(PoS), Proof-of-Replication(PoRep), Proof-of-Work(PoW), Proof-of-Space (PoSpace) and Proof-of-Spacetime(PoSt). Each proof has its advantage and disadvantage, so far Proof of Replication is used to verify contributor(miner)'s commitment.

In the verification process, the verifier challenges the prover, and the prover responses the proof of data replication. The honest contributor(miner) can answer the response. The cheater could playing the following attacks: Sybil Attack, Outsourcing Attack, Generation Attack.

With hash digests and proper designed algorithm, Sybil Attack can be prevent. Outsourcing Attack requires to designed in architecture, it will be ideal if the contributor has no idea about who else has the same data block. Generation Attack is quite difficult to perform.

IV. Design

A. Private Data Storage

The private data storage requires security, speedy and stable. Unlike the public files are usually shared among the users with common interest, like movies, album, open courses and software packages, the private files are unique to user: personal documents, photos. Safety is critical to the private storage.

B. Architecture

In P2P, peer is defined as both uploader and downloader. Peer combines the role of client and server. The trade off of peer is the performance. Lots of peers is behind the firewall connected with slow home connection like ADSL. Uploading bandwidth can not be ensured.

In MTFS' design, node is consisted of a batch of servers with professional connection sitting in the data center. The host is assigned with the public IP address and accessible open port, the power supplied in 7x24. To build a modern and efficiency service, the fundamental infrastructure must be stabilize. The nodes connects with each other one by one, forming a tree based network for message broadcasting, which will be discussed in the following chapters.

The nodes sits in the cloud data center. They play the role of service unlike the peer defined in P2P network. That's why we use term 'node' instead of 'peer'. As mobile device take a large proportion, the role of server and client need to be separated again, wireless devices has limited resource which requires the strong cloud service to help.

C. Cryptography

IPFS [8] comes without built-in cryptography. However, it's still possible to put a file in public place with the content encrypted safely. Asymmetric cryptography algorithm can be applied to make sure only the private key owner can decrypt the file content. In application level, OpenPGP or GPG tool can be used to encrypt and decrypt the file content.

With the public and private key pair, it is also possible to send the content to another user. Plain file content can be encrypted with receiver's public key and uploaded to a public address, the receiver download the encrypted content and decrypted with its own private key.

Although with the asymmetric cryptography tool chain, the file can be send to other users, the built-in cryptography mechanism is still necessary in MTFS' design.

Now days mobile users take a large proportion. Limited bandwidth and battery should be considered by solution design. And limited storage is on the mobile device as well. Imagine the user case of sending the encrypted file content to multiple users, the following actions should be take: 1. collection every user's public key 2. encrypt plain content with each user's public key 3. upload each encrypted content to server for multiple times. The step 2 costs too much computing power and local storage, the step 3 costs too much bandwidth.

Proxy Re-encryption [10] is used here to re-encrypt the decoding capsule without modifying existing cipher text. With this technology it will be very convenient for the mobile user to send content to any receiver under protection.

D. Broadcast network

Inside blockchain, P2P network is a mandatory component. Among the three types of P2P communication models: pair-wise, group-wise and broadcast, broadcasting is the most common requirement as every transaction or new block discovery requires to be announced in the whole network as efficiency as possible.

The broadcast network can be implemented with gossip protocol [4]. Gossip protocol [4] is extreme reliable, but it can not ensure the message can be arrived in every corner of network within fixed time. The propagation convergence time could last very long.

We proposed a tree based network for broadcasting use case(show in Figure 6). In tree based network topology, node join to tree as a leaf one by one, protocol is set to ensure the tree is constructed as balance as possible. In tree network, the longest distance between two nodes will be less than 2 times of tree height. That means even in a binary tree based topology, the message can be passed to the farthest node with less than 2 times tree height hops. It might be the most efficiency way for message broadcasting in logic.

In tree based broadcast network, message could be initialed from any tree branch node. A binary tree node will pass the message to its parent and two children from the original node. Any neighborhood receives the message, the message will be forward to other neighborhoods except the incoming one. For example, original node's parent receives the broadcast

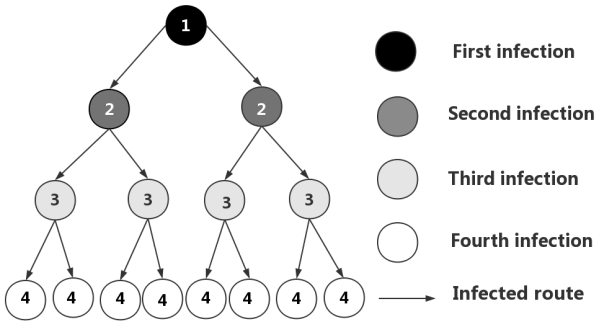

Fig. 6. Tree based broadcast network

message, it will forward the message to its parent node and another leaf node. If the original node's child receives the message, the message will be forward to both its children leaves.

However, tree based network also has its disadvantage compare to gossip protocol [4]: any single node failure will block the message propagation, because there is only one way for message to travel. To fix this issue, redundancy is required to add in this tree based broadcast network design.

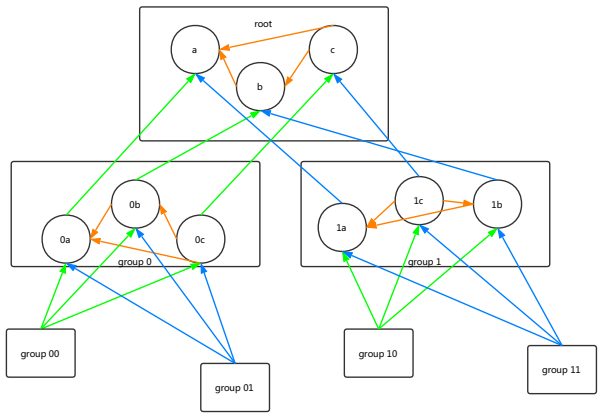

Fig. 7. Tree based broadcast network with cluster

1) Tree based network with cluster: The idea of adding redundancy to the tree based network, is to extend the single node to a cluster group. For the experiment purpose a group contains 3 nodes at most. Inside the group, the nodes interlink with each other. The broadcast message is passed to any node of group if node receives message from either parent or child. The child node still connect to one parent, if the parent node is connected by other node of current child group buddies, child node will try to connect another node in parent group. As a result, there will be several parallel connections between parent and children groups(Figure 7).

Each group should contains the fixed number of nodes, it's a conventional number agreed by every node in the cluster. The other nodes inside current group are called buddies. Besides the up-link and connection from children, there are also connections among the buddies. Any node within group received new message from parent or children, will forward the message to buddies. If the message from buddies is the new message that node hasn't heard from others, the node will also forward to parent and children nodes.

E. Blockchain

The last generation of P2P application like BitTorrent lacks of the incentive mechanism. User can join the file downloading any time. After managed to complete the download and verify the content, user can either choose to continue uploading as a server or close the application. In MTFS, it's required for the contributor(miner) stay online as long as possible. In this situation, the blockchain became a necessary part of the system, to make sure once the contributor takes the job, it's recorded in blockchain, not amendable.

The blockchain technology will play a very important part in MTFS design. User requests for certain file storage, a contract would be signed between user and system. A subscription relationship will be set up, unless user decides to cancel or node is about to quit. The contract here is not the typical blockchain smart contract, which doesn't require user to coding in programming language.

V. Implementation

In MTFS' design, Merkle Tree [2] is used to split encrypted content to small pieces of objects(less or equal to one mega bytes) and verify the completeness. With all the existing design including Proxy Re-encryption, GraphChain [6], Tree based network [12], we're able to see the big picture of a private storage system now.

User will be able to start using MTFS by generating a public/private key pair. To store a file on the network, the content needs to be encrypted with the user's public key first. After the cipher text generated, a storage contract between user and resource contributor will be signed, the contract is recorded in blockchain. MTFS will keep the encrypted content for the user.

A. Tree network construction

The tree network [12] is the backbone of MTFS system. It interlinks the distributed system's resource, plays the role of communication infrastructure and implements self governance. Each node in the network should be a server with public IP address and open accessed port. High computing capacity is not necessary, but good Internet connection with sufficiency bandwidth and enough available storage space is recommended.

Tree network [12] is consisted with nodes. The first node in network becomes root node, whose group id is tagged as an empty string. Following the protocol any node seeks for other nodes before they decide to play the root role. Expect the first root, all the rest nodes have their parent node in the network. Almost every node needs to connect to their parent.

Any node in the tree allows at most two children connecting with it. After the node connecting to the parent node, the parent node will assign the child a group id. When a node connecting to the parent node, meanwhile, the child itself became a parent available for another two new nodes. Occasions, it's possible for several new nodes attaching to a parent at almost same time, the parent node usually accept the first one and reject the others with a force disconnection message, responding them to try other available connection points, which is called open branches.

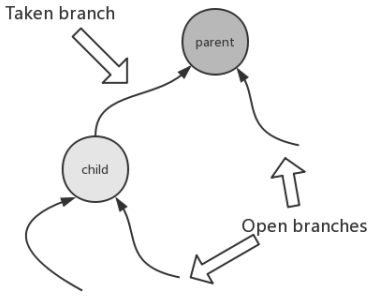

Fig. 8. Taken and open branches

1) Open branches: We can imagine that in a real world tree, node is the point where one branch forks into two branches. So we named the available connecting points as open branches, showed in Figure 8. Tree network [12]is used for message broadcasting, the information of open branches is spreading and synchronized among nodes by message broadcasting. In another word, each node keeps a copy of information listing available branches within the whole tree network [12].

Two message protocols are defined for the open branches information management: AVAILABLE_BRANCHES and DISCARDED_BRANCHES. The message AVAILABLE_BRANCHES is used to announce new open connection points to the network that new nodes can join the network. Another message DISCARDED_BRANCHES indicates that current branch already accepts a child node connecting, so the node's information will be removed from the global available open branches list.

When a child node managed to connect to a parent node, the parent node will send a DISCARDED_BRANCHES message to network due to his two available branches taken. Meanwhile, the connecting child node will send a message AVAILABLE_BRANCHES after its parent node assigns it a group id, telling the network another two branches is open and ready to be connected.

It's observed that when any node joins the network, one branch on the parent taken but two open branches on the child are added. As the height of tree and amount of nodes increase, the list of available branches will be longer and longer, then each node will have to maintenance a long list to remember the whole open branches in the network. This issue can be addressed by delaying the AVAILABLE_BRANCHES message sent. For example, when the node attached to parent, DISCARDED_BRANCHES message is sent at once, but AVAILABLE_BRANCHES is delayed until the available list is less than certain amount. This is an optimized topic for large scale of network management, details will not be discussed in this paper.

2) Group Naming: A node connects to a parent node, there might be two branches available for attaching. Binary tree has left branch and right branch. When branch is attached, the parent will response a GROUP_ID message to confirm the official group id of the new joined node. This action is to prevent a node to connect to a tree branch which is already taken by other node. In such case the parent will force to disconnect the attaching connection.

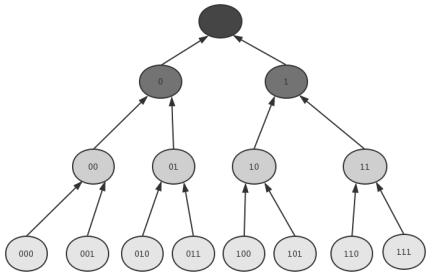

Fig. 9. Tree base network naming

In binary tree, the group id is represented in binary format with dynamic string length. The group id of root node is the empty string, the left branch child node is 0 and the right branch is 1, showed in Figure 9. Any hash hex string, digest like sha1 or md5 can be easily converted into binary represent. As the tree network [12] grows, the hash string can be distributed mapped to the outer nodes by the same prefix.

The group id is used to located the resource at the tree network [12]. The node may change the server IP or port, but group id assigned won't change unless the node decides to quit the network. If a node went offline for a long time, other node will consider it leaving and its parent can reassign the group id to another node.

B. Discover of nearby nodes

As the tree network grows, there are two things for each node to memorize: first is the available branches, second is all the nodes' host information. The second requirement will be extremely expensive when the network size grows. Instead of storing all the nodes' host information in memory or in storage, it's cheaper to remember the nearby neighbors' IP and port in memory. We can set a distance k, ask node to remember the nodes within k steps of hops. K must be equal or larger than 2, otherwise it's meaningless as any node already knows the direct neighbors' info such as its parent and children.

In Figure 10, an algorithm compares the distance between two nodes. The function parameters are two group ids. Func-

tion returns the hop distance of two group ids. Using this algorithm, each node can broadcast a special type of message with sender's group id and host information. During the message propagation, the nearby nodes will calculate how far this message passed and stop relay the message after k steps' hop. This is a locally nearby broadcasting technique, which is useful to discover the nearby nodes without overall network broadcasting.

```
def group_distance(a, b):
    if len(a) > len(b):
        a, b = b, a
    i = 0
    while i < len(a):
        if a[i] != b[i]:
            break
        i += 1
    return len(a)+len(b)-i*2
```

Fig. 10. Group distance

1) Recursive node discovery: If the encrypted file size is more than one mega bytes, the file is divided into small objects by Merkle Tree [2]. Otherwise the file doesn't need to split, it can be directly uploaded to a node. By comparing the object hash with existing group ids, user can recursively find the outermost node in the tree network [12].

2) Children and parents: It's easy to see all the parents when a new node joins the tree network [12]. But inversely, the parent is difficult to remember all the children nodes when the tree network is growing big. Take a example of root node, if root tries to remember all the children nodes' host info, the root will have to memorize the entire network, which is too much information when the tree network [12] scales up.

C. Group Path

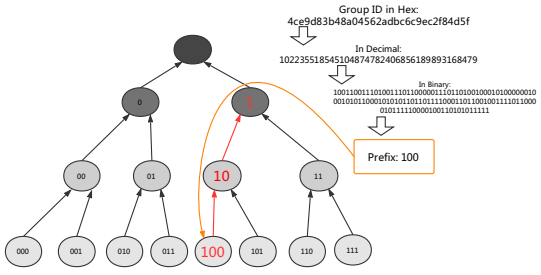

Fig. 11. Group Path

For a given hex hash string of node id, it can be represent in the binary format. Comparing with existing network, we can find the nodes whose group id is the prefix of the given hex hash string. Those nodes are defined as Group Path, showed in Figure 11.

D. Local encryption

Proxy Re-Encryption(PRE) algorithm [10] is an asymmetric encryption. When user uploads a file to the tree network [12], it's required for the user to encrypt the files with the public key. Besides the cipher text, another file named capsule in PRE [10] is generated. The encrypted cipher text can be decrypted with user's private key and the capsule.

Plain text is encrypt with PRE [10], output a cipher text file and a capsule file. The cipher text file size is usually as big as the plain text, the capsule file size is usually tiny.

E. File and folder object

The objects are generated from the encrypted content using Merkle Tree [2]. If the encrypted content is less than one mega bytes, only one object is created. In this case, root hash equals to the encrypted content hash. Otherwise, a file object named with root hash suffix "_mt" is generated which contains Merkle Tree [2] structure of file objects. Besides, there is capsule file named with root hash suffix "_capsule".

We can decode the file content with user's private key and the capsule, but the file name is lost. The file name and size is stored in the folder object.

1) Convention object size: The convention object size of MTFS is one mega bytes(1024*1024 bytes). It's the base size unit in MTFS. For small document file, usually only one unit of object requires. For user generated photos or videos, more objects can be allocated in parallel to speed up. Merkle Tree [2] algorithm splits any file larger than one mega bytes into small pieces of objects. If content is less than one mega bytes, the Merkle Tree [2] root hash is equal to the hash of the content.

F. User and nodes

1) Upload the file: The user's root folder object is created when user uploads the first file or create a sub folder in the root folder.

User is tended to choose the the edge node to store the new object, we can imagine the network like a city, in the central area it's always busy with traffic jam, in the country far from the city there is less traffic and enough land.

The user's files, got encrypted, split into objects, uploaded to tree network [12] by comparing group id and object hash, stored into MTFS. Then, the user's root folder information upload to MTFS as an object. The folder object is updated following the same process as file object.

As the folder and file objects are saved into tree network [12], a contract is set up between user and nodes. The root folder object's hash is committed into blockchain as a transaction. Besides the root hash, the transaction also includes the group id information.

2) Retrieve the file: Inversely, when the user is browsing file from MTFS, the first step is looking for the root folder information on the nodes. User's root folder object and location information(group id of storage) is committed as the transaction of blockchain. With object id and group id, it's easy to retrieve the content of root folder object. If the folder

content is larger than one mega bytes, user will follow the procedure to gather all the folder objects and decrypt them.

While user clicks the file to download from MTFS, folder's provide the information to locate the file objects. After all the file objects finish downloading, user will get the plain content decrypted with user's private key and the capsule.

In the worse case, the information in blockchain transaction and folder can not locate the file objects for us. It's because some node quit the system. However, any object has several copies on the group path. In this case, it's requires for the user to seek the group path to find the file objects.

3) Folder object schema: User check the group path on the tree network [12], there should be a node hosting the user's information(assume the network keeps 3 replications).

Folder object is different from file object. Folder object contains the encrypted folder data. Data is represented in JSON format. Entities in folder data links to other file object and folder object. File object size indicates the file size, and folder object size indicates the summed file size under the folder. This information is useful to calculate how much storage consumed by the user.

If entity of file object's size is larger than one mega bytes, the hash is a Merkle Tree [2] root hash, otherwise it's a digest hash of file content.

G. Exchange file with other

Exchange file is similar to send an email to another user. Sender should know the receiver user's address, in another word, the public key. The file sending action is a blockchain transaction operation, in the transaction it shows who and which file on MTFS will be added into receiver's folder.

MTFS is highly security system. In the file sending operation, even the system server doesn't have the permission to modify user's folder information, the objects are encrypted that only user's private key can decrypt and modify.

However, the user may interaction with MTFS from mobile device. It is too expensive to follow the progress for any operation: download, decrypt, modify, encrypt and upload. If it's required user to modify content, this progress has to be applied. In another case, when it's not required for user to modified the content, is it possible to reduce the process? PRE [10] is introduced in this case.

When user send the file to another, the file size might be huge, download, decrypt, and encrypt with another user's public key, then upload is too expensive. In this case, file owner can encrypt the file content into a cipher text and a capsule. The cipher text is as long as the content, but the capsule is usually very short. The capsule can be re-encrypt with sender's private key and receiver's public key. Which means, once the receiver accepted the re-encrypt capsule granted by sender, he will be easily decrypt the cipher text with the new capsule and his own private key.

When the sending transaction broadcasts, receiver need to accept the sending action, which is to update his folder object to keep the file. The file exchange process is finished.

H. Replication and verification

After the file is uploaded to system, the file must replicate itself for several copies on the group path. Because the objects on the node is protected with authentication, the other nodes on the group path doesn't have the permission to pull the objects, the replication needs the node to push objects to the replicated nodes.

Verification of the storage needs to be performed periodically, as the object content may damaged, lost, or even the node will try to cheat by clamming that they kept the objects but actually not. The verification is initiated by node quiz the replicated node for the sub range of object content. The node quiz with the range expects the response of content's hash. By comparing with local object's sub range hash, it proves the replication of object.

VI. Conclusions

In this thesis, we have proposed MTFS and discussed its many aspects from the technology requirements to the design and implementation with major focus on cryptography, blockchain, network and storage schema.

We use existing technology based on a novel design in building a secure private storage system with decent performance: BitTorrent and Bitcoin [1] allow normal computer to work as peers, while a tree network [12] is used as a backbone network for storage with high performance broadcasting. In the tree network, only servers with public IP addresses are allowed to join it as nodes to avoid the issues related with firewall and gateway NAT in achieving higher performance. Also PRE [10] encryption is introduced to re-encrypt the decoding capsule without modifying existing cipher texts. Base on these component technologies, the schema of storage is designed, and MTFS can provide a high-performance solution for a secure private storage service based on the blockchain technology.

Throughout this final-year master project, we have implemented and demonstrated the operation of a prototype for the network and the blockchain parts: The tree network is ready for message broadcast and object storage. To better understand the operation of the network, we have also implemented a visualization demo.

Building a real test bed is an important topic for further study, as it could enable performance evaluation under a more realistic environemnt and thereby allow us to identify the areas of improvement.

Acknowledgment

I would like to give special thanks to my supervisor Dr Kyeong Soo Kim for his suggestions and detailed guidance, without which I couldn't obtain these achievements presented in this thesis.

References

[1] Nakamoto, S. (2008). Bitcoin: A peer-to-peer electronic cash system.

[2] Merkle, R. C. (1987, August). A digital signature based on a conventional encryption function. In Conference on the theory and application of cryptographic techniques (pp. 369-378). Springer, Berlin, Heidelberg.

[3] Maymounkov, P., & Mazieres, D. (2002, March). Kademlia: A peer-to-peer information system based on the xor metric. In International Workshop on Peer-to-Peer Systems (pp. 53-65). Springer, Berlin, Heidelberg.

[4] Demers, A., Greene, D., Hauser, C., Irish, W., Larson, J., Shenker, S., ... & Terry, D. (1987, December). Epidemic algorithms for replicated database maintenance. In Proceedings of the sixth annual ACM Symposium on Principles of distributed computing (pp. 1-12). ACM.

[5] Eyal, I., Gencer, A. E., Sirer, E. G., & Van Renesse, R. (2016, March). Bitcoin-NG: A Scalable Blockchain Protocol. In NSDI (pp. 45-59).

[6] Kan, J., Chen, S., & Huang, X. (2018). *Improve Blockchain Performance using Graph Data Structure and Parallel Mining.* arXiv preprint arXiv:1808.10810.

[7] Bi, W., Yang, H., & Zheng, M. (2018). An Accelerated Method for Message Propagation in Blockchain Networks. arXiv preprint arXiv:1809.00455.

[8] Benet, J. (2014). IPFS-content addressed, versioned, P2P file system. arXiv preprint arXiv:1407.3561.

[9] Benet, J., Dalrymple, D., & Greco, N. (2018). Proof of replication. Technical report, Protocol Labs, July 27, 2017. https://filecoin.io/proof-of-replication.pdf. Accessed June.

[10] Wikipedia, (2017). Proxy re-encryption - Wikipedia, the free encyclopedia. Available at: https://en.wikipedia.org/wiki/Proxy_re-encryption

[11] David, N. (2018). Umbral: a threshold proxy re-encryption scheme. Available at: https://github.com/nucypher/umbral-doc/blob/master/umbral-doc.pdf

[12] Kan, J., Zou, L., Bella, L., & Huang, X. (2018). *Boost Blockchain Broadcast Propagation with Tree Routing.* arXiv preprint arXiv:1810.12795.

[13] Chacon, S., & Straub, B. (2014). Pro git. Apress.