**Blockchain and Digital Currency Final Project Report**

**Personal File for Public Usage via Block Chain Architecture**

1. Abstract: Key words:
2. Directory
3. Introduction
4. Development background and significance
5. Demand analysis
6. Project design
7. System test
8. Design conclusion
9. Acknowledgement
10. Appendix
11. Reference

Abstract

With the rapid growth of blockchain technology, government departments desire the application of blockchain technology that help them to solve efficiency and authenticity difficulties. Blockchain unique characteristic offer many well solution for government affairs reformation. In this project, we aim to design a blockchain-based system that can record and update personal file information for government department. We firstly introduce the development environment of blockchain technology. Then we list the reasons and feasibility of developing a blockchain-based government attestion system for personal file instead of traditional system. We introduce the unique advantage in security, reliability and availability using the blockchain technology. We introduce the whole ecosystem of our project, including specify the work of common users, miners and validators, clarify the types of transactions, clarify the structure of the blocks, illustrate two types of accounts and design incentive system. Subsequently, we introduce the model of our two types of accounts management, whole routine of how our project works and other protocol like consensus protocol, incentive system, etc. We build a java back end to complete our project’s function. And we design a React front end for function presenting. We then explain the limitation of our project and future expectation.

Key words: blockchain technology, personal file management, historical reputation system

Directory

Introduction

Development background and significance

Demand analysis

Project design

Application layer design

Code design

System test

Test class and specific functions

Front unit design

Design conclusion

Differences between application layer and code

Problems and limitation

Future expectation and improvement

Acknowledge

Appendix

Reference

Introduction

Blockchain is originated from bitcoin. Based on P2P net-work layer technology, encryption technology, timestamp, and electronic cash system architecture concept, blockchain and digital currency technology grow rapidly since January 3,2009, the birth of genesis block, with its unique characteristic of decentralization, security, publicly visible and efficiency in many given occasions.

Blockchain and digital currency develop supplement of each other. Bitcoin, with the application of blockchain technology, also known as “digital gold”, is a decentralized digital currency. Bitcoin is created after a person calling himself Satoshi Nakamoto published the bitcoin white paper "bitcoin: a peer-to-peer electronic cash system" on the website of the P2P foundation, stating his new vision for electronic money -- the introduction of bitcoin. Bitcoin has the unique currency characteristic of decentralization, pseudonym and tamperability. Supporters claim that bitcoin implementation value transfer all over the world. Other voices argue that bitcoin has the ambitions to redistribute wealth. No matter how, bitcoin wins a huge success in digital currency world. Bitcoin’s success drive Ethereum’s creation. Ethereum, also known as “blockchain 2.0”, optimizes bitcoin’s hardcore shortage and firstly introduces the concept of smart contract. It increases protocol extensibility by smart contract and replace the bitcoin script language with solidity language, which is Turing-complete. These changes enable Ethereum to be more diversity, many complicated value exchange or protocol execute become possible.

Ethereum’s diversity hatch many interesting applications. Such as decentralized risk investing organization (The DAO), decentralized share options market (Etheropt), farm-to-table application or decentralized forecast market, etc. The above application both prompt us to the idea and design of our project.

However, blockchain technology is not limited in altcoins system, on the contrary, altcoins transactions have been prohibited by law in many countries for their negative usage in property transfer and money laundering. Instead, Chinese government and some corporations show strong interest in the landing of blockchain technology. Many attempts are under way. People’s Bank of China began the digital currency DCEP（Digital Currency Electronic Payment） research in 2014 and mostly possible to become the first central bank that issue digital currency. Ant Finance, a Chinese fintech corporation, also develop their product traceability with the help of blockchain technology: blockchain application scenario was officially launched in November 2017 for the cross-border traceability service of Tmall overseas goods. In this scenario, Alipay users from Tmall for 26 goods from Australia, New Zealand, dairy products, provide about each bottle of milk products id tracking source service. Alipay user scan tracing the source code can clearly see the whole logistics of the dairy products from the manufacturer to the whole logistics, including quality inspection, overseas warehouse to domestic warehouse, including the whole link of end-to-end commodity circulation records of production date and circulation date. In addition to this, platform information share system, digital assets, supply chain finance, cross-border payments also build in the form of public chain, league chain or private chain.

Chinese president, Jinping Xi, emphasized that we should take blockchain as an important breakthrough in independent innovation of core technology to accelerate the development of blockchain technology and industrial innovation in the 18th study of the Political Bureau of the CPC Central Committee. He also specially emphasized the use of blockchain data sharing model and encourage government department to reform their inefficient work model by blockchain technology.

Above events both indicate the bright future of the development of blockchain technology in China, also is one of the motivations of our project design.

Development background and significance

We design the project mainly because following reasons:

1) traditional personal file system adopts the form in combination with offline paper file and online digital file back up in database.

2) it is difficult for external users(citizens) to call their record when they handle business in different places and different apartments.

3) entity files face unsafe factor like losing and damaging during the transport process.

4) the recording of traditional digital file faces the problem of time latency because of the entity limitation.

5) each department has to reconcile the file frequently and face the risk of recording version differences.

6) tradition personal file system takes larger cost in storage and maintenance information.

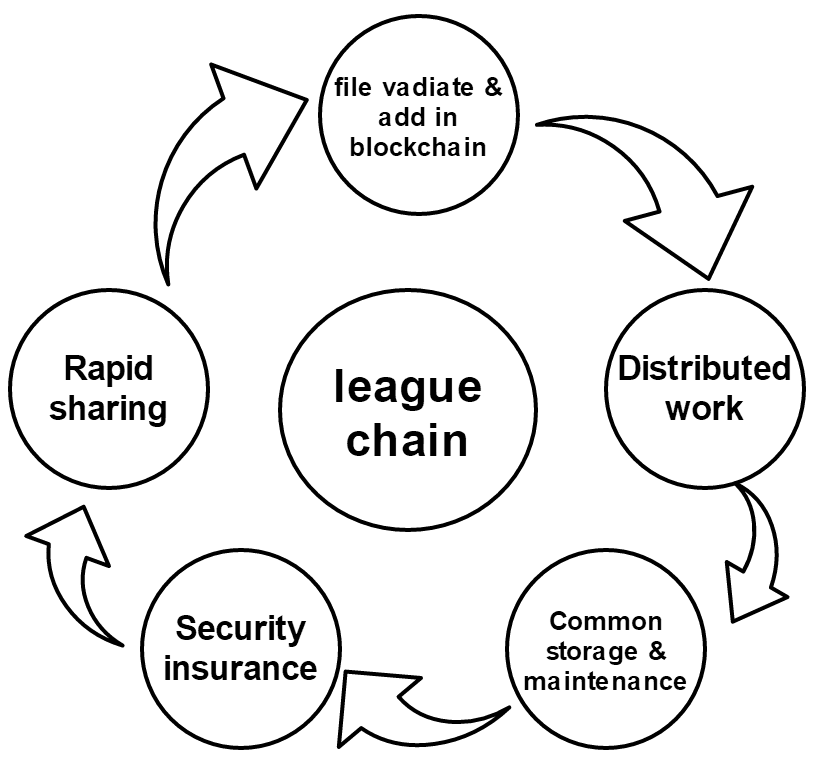
The introduction of “Personal File for Public Usage via Blockchain Architecture” increase the following characteristic:

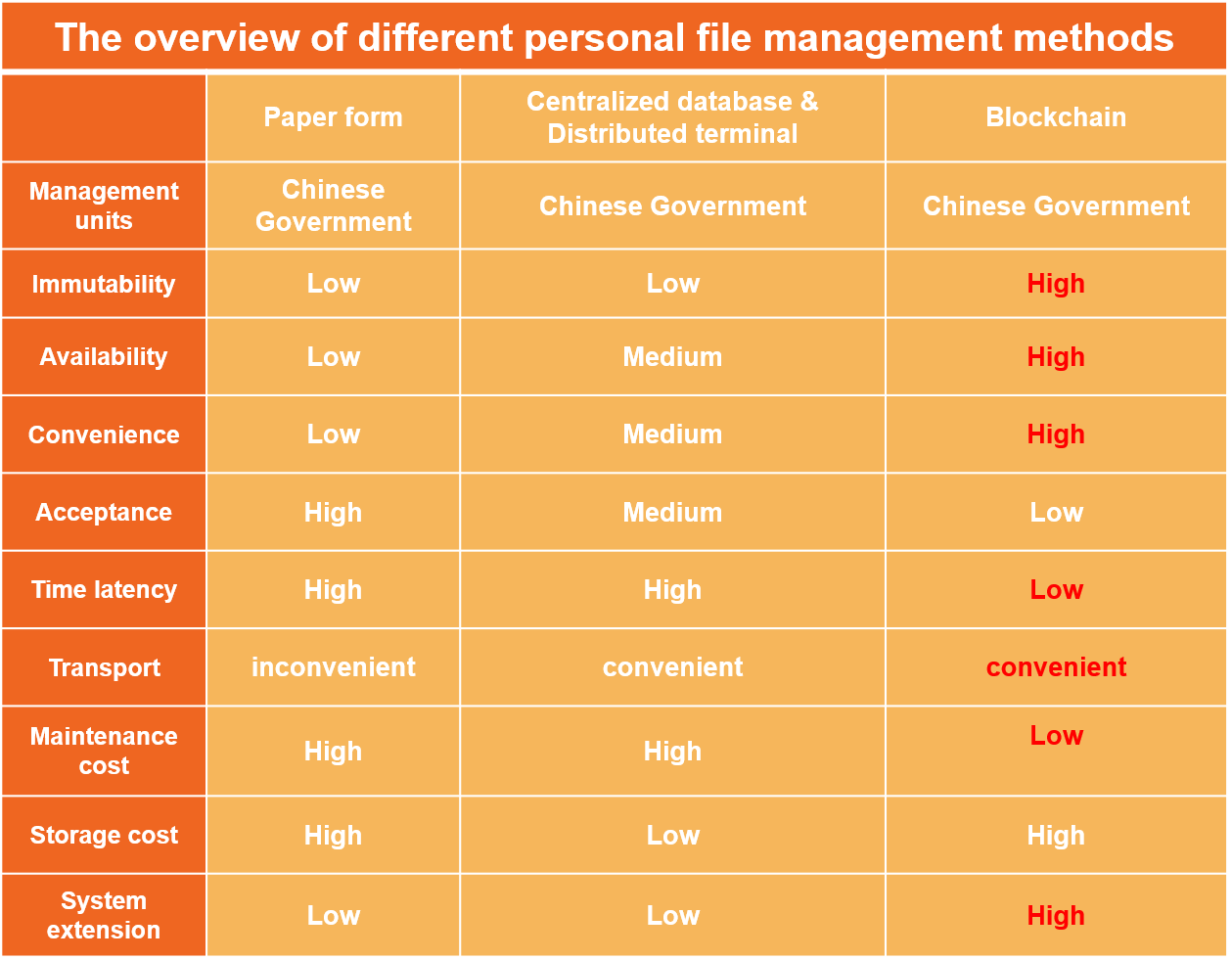
1) the authenticity of personal file. As blockchain has the characteristic of append-only ledger and irrevocable ledger, personal file’s authenticity and reliability increase through usage after file is stored.

2) the security of file storage. In the process of file storage, blockchain help reduce the risk of illegal access, tampering and theft.

3) the availability of file information. Under the premise of security and authenticity, blockchain system let file information flow through management units, using units and public efficiently. Blockchain consensus protocol also meet the requirement of “at most one run” in government department reformation. Both characteristics increase the using value of personal file information.

General design of our project and its comparison between tradition system:





Demand analysis

1. For government management departments and decision-making departments, they need more reliable national information and more efficient updated to modify management formulation and policy issuance.
2. For local archives bureau, local recording units and validation units (serve as miners, common users and validators), this project reduces their multifarious affairs and increase their working efficiency. This project conforms to their desire of department reformation. They can also get extra revenue by further reliable work.
3. For external users (citizens), blockchain system reduce time and expense costing when they handle affairs related to personal files. Blockchain trust mechanism also strengthen external users’ confidence that their personal profile will not be maliciously modified.

Project design

**Common users:** upload information with smart contract.

Scenario 1: diploma

Specific transaction:

|  |  |
| --- | --- |
| Message -> event | Somebody got diploma |
| tag | education |
| Series of events | false |
| Previous hash | null |
| signature | Signed by Tsinghua University |

Explanation:

1. Event means the modified content of your personal file.
2. Tag is a classification of the event, help us classify and verify the information.
3. Series of events is a Boolean value used for judging whether it is series of events or not.
4. If event has antecedent events, then previous hash is previous event hash. Otherwise, it is null.
5. Signature is provided by local recording units. Used for verification and accountability.

Verification (before add in transaction pool by miners):

1. Signature is true

2. Signer have authority to publish the tag

Verification should first be classified by seeing the tag.

Scenario 2: a pair of events: census register transfer

Event a: sb apply to transfer his household (send out, from shanghai to shenzhen).

Event b: shenzhen consent & not consent for event a.

Transaction 1:

|  |  |
| --- | --- |
| Message -> event | hyz send out from sh |
| tag | household |
| Series of events | true |
| Previous hash | null |
| signature | Signed by sh xx bureau |

Transaction 2:

|  |  |
| --- | --- |
| Message -> event | hyz send in to sz |
| tag | household |
| Series of events | true |
| Previous hash | eventa.gethash() |
| signature | Signed by sz xx bureau |

Scenario 3: health care

Specific transaction:

|  |  |
| --- | --- |
| Message -> event | rjx health check |
| tag | health |
| Series of events | true |
| Previous hash | null |
| signature | Signed by xx hospital |

Scenario 4: position appointment and removement

Event c: sb outgoing xx position (departure from public security department).

Event d: sb ingoing xx position (entry Ministry of Public Security, PRC).

|  |  |
| --- | --- |
| Message -> event | Sb departure |
| tag | Appointment & removement |
| Series of events | true |
| Previous hash | null |
| signature | Signed by xx public security depart ment |

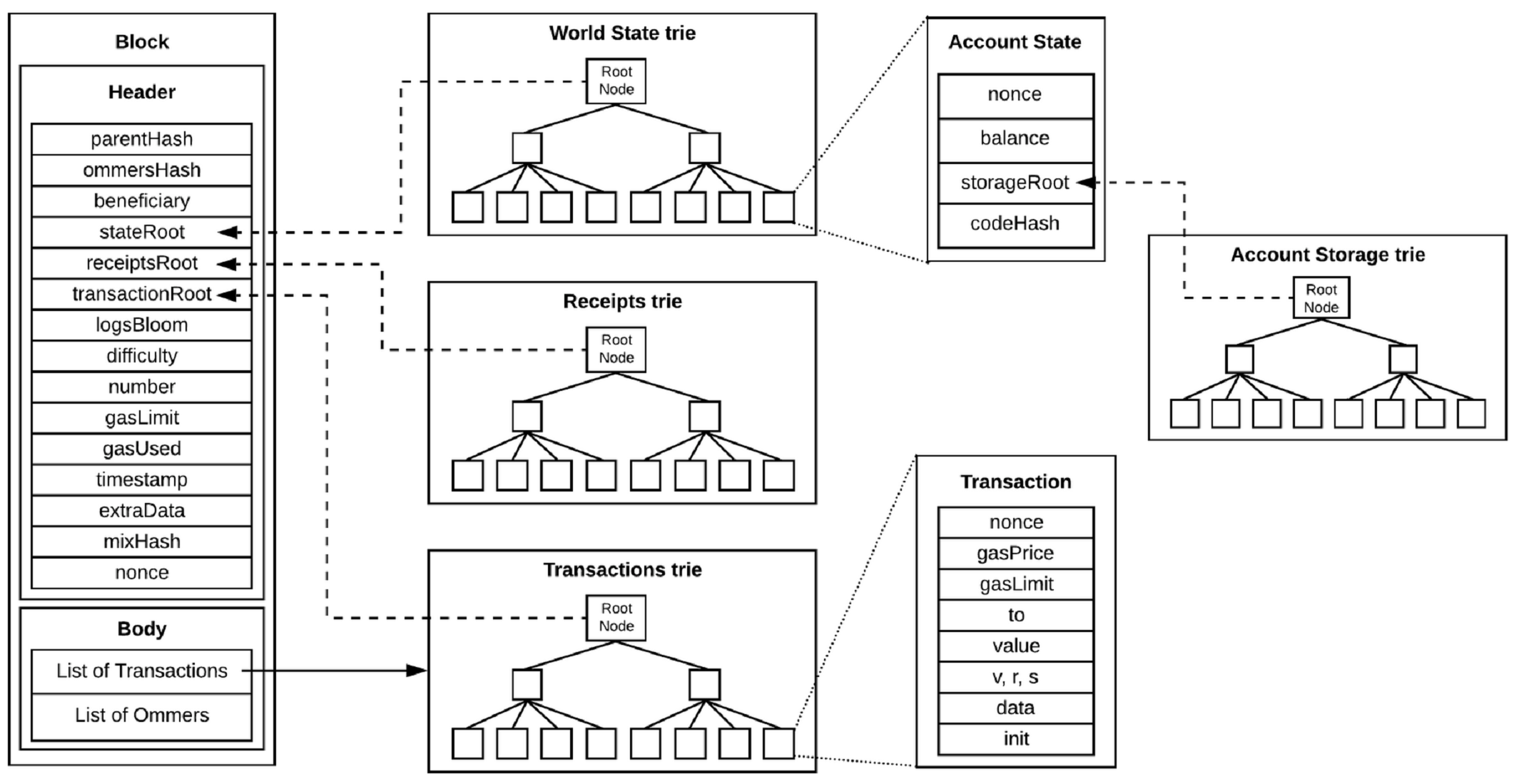
|  |  |
| --- | --- |
| Message -> event | Sb entry |
| tag | Appointment & removement |
| Series of events | true |
| Previous hash | Eventc.gethash() |
| signature | Signed by xx PRC |

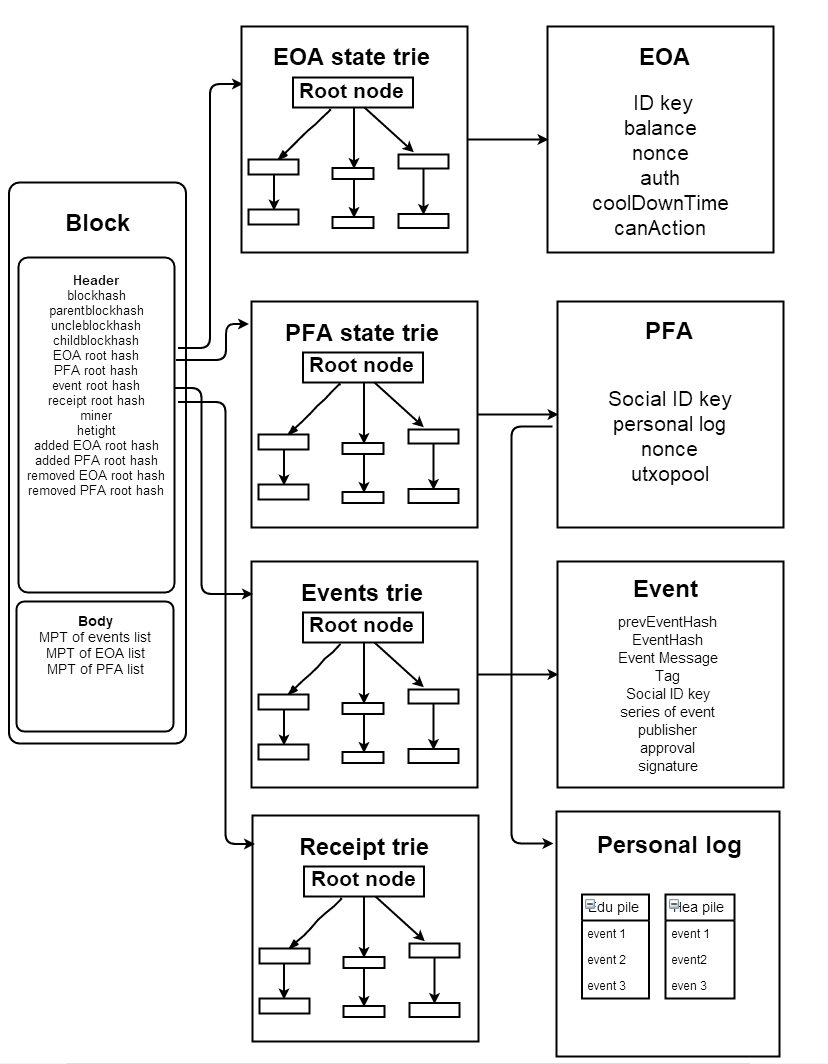
**Miners:** pack transactions to install a block

Scenario 4: Block structure

|  |
| --- |
| Block header:  Version number  POS rules  Block header hash  Previous block header hash  Uncle block header hash  Block height  Coinbase share  Dividend  MPT root hash of EOA: users’ information  MPT root hash of transactions: different kinds of transactions  MPT root hash of receipt: validation  MPT root hash of personal file accounts: main information |
| Block body:  MPT structure of events  MPT structure of EOA  MPT structure of PFA  … |

**Comparison between Ethereum and our project general structure**





Verify: valid signature, valid transactions, valid insertion position, if valid, update PFA and EOA.

miners local maintain: transaction pool, blockchain, accounts information (in modified MPT tree)

**Validator:**

Role of validator is to spot check blocks (50 blocks a time, cooling time) and re-broadcast the events that due to state fork and pass validator’s verification. We can regard this role as a special type of common users with the justification of verification.

Scenario 5: pseudo double spending; indeliberate state fork

**Accounts-based management:**

**Pesonal file accounts (PFA):**

Identification(name, birth date)

Education pile(scenario 1)

Household pile(scenario 2)

Healthcare pile(scenario 3)

Series of events(operator)

nonce

**EOA (miner,common users validator):**

balance(100 altcoins)

Authority:common users/miner/validation classification

cooldown time: 100 min

nonce(incentive): 10 times

**accounts access and exit**

1) accounts access, included external owned accounts and personal file accounts, follow the rule that activities related to new accounts must be recorded after their creation recording. In another words, a block is forbidden to record both an account’s creation and related events.

2) accounts exit must have designated approval. Personal file accounts exit should be approved by common users. External owned accounts exit should be approved by miners. Accounts exit also follow the latency mechanism introduced in accounts access.

**accounts storage:**

1) Personal file accounts: like Ethereum, merkle patricia tree structure, use stack without pop function, called it pile, to store specific events.

2) EOA accounts: like Ethereum, merkle patricia tree structure.

**whole process:**

1. citizens occur an event that needs to record in personal file
2. recording units (common users) broadcast the event after finishing offline verification and recording affairs.
3. miners accept the information after verification. They insert the transaction into their transaction pool and begin to install block.
4. the rights of charge follow the rules of PoS distribution. After a validated block inserts in blockchain, other full validating nodes should modify their local blockchain structure, local transaction pool and two local merkle patricia tree structure, PFA and EOA.
5. next mining procedure.
6. validators will randomly check continual 50 blocks to avoid pseudo double spending and unintentional state fork problem. They have cooling time restriction to prevent them from overwork.

**crypto principle:**

We simply adopt the method that bitcoin used. Common users and miners create an account by generating a key pair similar to bitcoin system. The signature model and encryption & decryption model follow the method used in bitcoin system.

**data structure:**

As the graph shown above, Ethereum-based modified merkle patricia trie.

**bottom network structure:**

**incentive system:**

Every creation of a block will create a coinbase event. Specific amounts of altcoins will send to miners. Miners will pre-allocate portion of altcoins to the common users contribute in this block in a given way. The altcoin itself has no value, but for common users, they can show their altcoins amounts to their superior to get extra bonus (nonce & altcoins amounts). For miners, altcoins can increase their stake in a POS system, they also get reward monthly like the common users. For validators, they get reward by check nonce and quality work.

**consensus protocol:**

We adopt Ethereum-based consensus protocol, ghost protocol, to solute state fork problem. The specific solution is as follow: First, when a state fork occurs, miners add their block to the block observed first. Second, each current block has a right to accept one uncle block. The accepted uncle block will get a portion of coinbase reward in a given proportion.

**differentiation of node status**

**code design & analysis:**

Crypto

Simply follow the method using in bitcoin system, return true when the signature is a valid digital signature with corresponding message and public key, use RSA signature

Pile

Pile is a data structure that is similar to stack but can only add string element and no pop function.

PersonalFileAccount

Personal file account are the accounts being managed, they can only be viewed but not edit. It is the digital personal file for external users.

**Attributes:**

Private String social\_id\_key------------------------social identification number (i.e. id card number)  
private HashMap<String, Pile> personalLog----------hashmap, dictionary illustrates your personal file specific events, in the code, we simply add “education” and “household”. But the log can be expanded as needed.

private BigInteger nonce---------------------------a paired number that makes search space more sparce  
private HashMap<byte[], Event> utxoPool---------arraylist of HashMap<EventHash, Event>, used to store personal event with attribute series\_of\_event = true

**Construct functions:**

public PersonalFileAccount()----default method

public PersonalFileAccount(String social\_id\_key, BigInteger nonce)---given method

**functions:**

get methods & set methods

functions for printing and adding personalLog

ExternalOwnedUser

EOA are common users/miners/validator in p-chain, they're processor instead of owner of each personal file. Thus, they undertake the work of broadcast events/mine block/validate.  
**Attributes:**  
private PubicKey id\_key-----institute identification code, public key  
private int balance-----------------altcoin owned by EOA  
private string auth------------type of EOA, miner/common\_user/validator  
private int coolDownTime--------for miner/validator, restriction constant mining/validating  
private int nonce------------counter, an incentive, major usage is to work with "coolDownTime", can also be used as an incentive  
private boolean canAction------judgement for whether you can mine/validate or not

**Construct functions:**

public ExternalOwnedUser()----default method

public ExternalOwnedUser (PublicKey id\_key, String auth) ----given method

**get methods & set methods & add methods**

**Handler functions:**

public void EOAHandler()--------cool down time operator for miner and validator. Change canAction attribute.

Event

Event is a class used to record message proposed by common users

**Attributes:**

private byte[] prevEventHash------- will only be useful when series of event is true  
private byte[] signature----------publisher signature  
private String social\_id\_key-------strings indicate that whose personal file are going to be changed  
private String eventMsg---------recording detail message of the event  
private String tag---------------classification of EOA accounts  
private boolean series\_of\_event-----whether it is series of events or not  
private byte[] hash---------------the hash of whole event  
private PublicKey publisher--------which EOA publish the event  
private String approval--------- a simplified version of receipt state, reveal whether the action is approved or not.

**Construct functions:**

public Event()----default method

public public Event(String social\_id\_key, String eventMsg, String tag, boolean series\_of\_event, byte[] prevEventHash, PublicKey publisher)---given method

**get methods & set methods & sign methods**

EventPool

A class map Event with it proposer EOA, used for fasten searching efficiency

**Attributes:**

private Event event  
private ExternalOwnedUser externalOwnedUser  
private HashMap<Event, ExternalOwnedUser> proposedEvents----- hashmap, matching table illustrates the event is proposed by specific EOA

**Construct functions:**

public EventPool()----default method

public EventPool (EventPool eventPool) ----given method

**get methods & add methods & remove methods & contain method**

PersonalFileAccountUTXOPool

A class automatically combines with EventPool, to connect with EOA, event and PFA.

**Attributes:**

private HashMap<Event, PersonalFileAccount> H------ hashmap, matching table illustrates the event is related to specific PFA

**Construct functions:**

public PersonalFileAccountUTXOPool()----default method

public PersonalFileAccountUTXOPool(PersonalFileAccountUTXOPool p\_utxoPool)----given method

**get method & add method & remove method**

EventHandler

Event handler is used to process multiple events in one time, which is prior to packing a block.

**Attributes:**

private EventPool eventPool  
private PersonalFileAccountUTXOPool personalFileAccountUTXOPool

**Construct functions:**

public EventHandler()----default method

public EventHandler(PersonalFileAccountUTXOPool p\_utxoPool)----given method

**Functions:**

public boolean isValidEvent(Event event)----validation for whether the event is proposed by authorized unit, whether the proposer is common users, whether the series of event can match each other.

public ArrayList<Event[]> handleEvent(Event[] waitingEvents)------ process events, select those are valid to be included in a block;the length of Arraylist<Event[]> is of length 2, where 0-th event[] is the valid Events, and 1-st event[] are those invalid one, and will be broadcast to validator

Block

For simplicity, we use list instead of MP tree in this class

**Attributes:**

public static final int COINBASE = 10----altcoins send back to miners and common users involved for reward  
private byte[] hash-------hash of whole block  
private byte[] prevBlockHash-----hash of previous block  
private ArrayList<Event> events----- simply replace modified MPT structure with list in code; analogue to transaction list in canonical cryptocurrency block  
private ExternalOwnedUser miner-----the miner of the block

**Construct functions:**

Block’s EOA should be designated to miners.

public Block()----default method

public Block(byte[] prevBlockHash, ExternalOwnedUser miner)----given method

**Get methods & sign methods**

BlockNode

Expand block's attributes, make it convenient to connect previous and forwarding block (enclosed in block node), help maintain 2 pools (personalFileAccountUTXOPool and eventPool)

**Attributes:**

public static final int CUT\_OFF\_HEIGHT = 1------only an indicator, no actual usage, implies max number of uncle BlockNode to be included  
private static final double COINBASE = 10------same as block class  
private static final double FRACTION\_NUMBER = 0.1------reward distribution portion to common users  
private Block block--- enclosed block information, inheriting from block class  
private int height-----record height of a block, counting from 1  
private BlockNode parentNode-----a block node, current node's parent node, unique  
private ArrayList<BlockNode> childNodes------ current node's child nodes, not unique  
private ArrayList<PersonalFileAccount> p\_accounts---an ArrayList<PersonalFileAccount>, maintaining status of personal file account

private ArrayList<ExternalOwnedUser> eoa\_accounts-------- an ArrayList< ExternalOwnedUser >, maintaining status of external owned account  
private EventPool eventPool----- a pool, recoding existing events with their publisher(EOA) in current block

private PersonalFileAccountUTXOPool p\_pool;  
private ArrayList<ExternalOwnedUser> awaitEOAs;  
private ArrayList<PersonalFileAccount> awaitPersonalFileAccounts-------- these PFAs are in waiting list to add into current PFAs’ MPT  
  
private ArrayList<ExternalOwnedUser> removeEOA;  
private ArrayList<PersonalFileAccount> removePersonalFileAccount;  
private BlockNode uncleBlockNode;

**Construct functions:**

public BlockNode(Block block) ----default method

private BlockNode(Block block, int height, BlockNode parentNode, ArrayList<BlockNode> childNodes, ArrayList<PersonalFileAccount> p\_accounts, ArrayList<ExternalOwnedUser> eoa\_accounts, EventPool eventPool, PersonalFileAccountUTXOPool p\_pool, ArrayList<ExternalOwnedUser> awaitEOAs, ArrayList<PersonalFileAccount> awaitPersonalFileAccounts, ArrayList<ExternalOwnedUser> removeEOA, ArrayList<PersonalFileAccount> removePersonalFileAccount)-------given method

**Get methods & set methods**

**Functions:**

public boolean addBlock(BlockNode parentBlockNode,Block block,ArrayList<ExternalOwnedUser> proposedAddEOAs, ArrayList<ExternalOwnedUser> proposedRemoveEOAs,ArrayList<PersonalFileAccount> proposedAddPAccount, ArrayList<PersonalFileAccount> proposedRemovePAccount)------- while adding block, first, check the following facts:  
1. if the block's prevBlockHash equal to parent block's hash  
2. if all events in the new block are valid  
2.1. additionally, check whether personal account and EOA exist or not  
2.2. also, check potential removes or adds  
3. if the block's height > current max height - CUT\_OFF\_HEIGHT(not shown here, depend on genesis block node, checked in {@class BlockChain})  
Second, if validation test passed, update newest blockNode, including:  
0. update parent node's child node  
1. update block's parentBlockNode  
2. update block's height  
3. update event pool  
4. update personal file account utxoPool  
5. according to above information, update EOA and personal file accounts  
6. update awaiting account list and remove account list

BlockChain

BlockChain is the structure of a chain, it should contain a genesis block, addblock will extend a block chain automatically. For the aim of experiment, we introduce manual fork function. More explanation on genesis block; genesis block is special, because it has no valid miners/other users based on our rule in genesis block, prevBlockHash = null; events is empty, miner is arbitrary; its height is 1, it removal list is empty, only has entry list.

**Construct function**

public BlockChain(Block genesisBlock, ArrayList<ExternalOwnedUser> entryEOAs,  
ArrayList<PersonalFileAccount> entryP\_accounts)----default method

**Functions:**

public BlockNode getMaxHeightBlockNode(BlockNode blockNode)-----get max height block node

public boolean automaticExtendBlockChain(Block block, ArrayList<ExternalOwnedUser> entryEOAs, ArrayList<ExternalOwnedUser> removeEOAs, ArrayList<PersonalFileAccount> entryP\_accounts,ArrayList<PersonalFileAccount> removeP\_accounts)-----extend the block chain automatically, return Boolean value for whether success or not

**Prints Methods**

System test

testPile

Simply test the implementation of pile data structure (no pop function stack).

testValidSignature

Simply test the implementation of our encrypted algorithm signature.

testEvent

Simply test the implementation of our event class.

testAuto\_p\_Chain

In this test class, we simulate the scenarios we present in application design part. We test all the blockchain functions and the scenario with print functions. This part will be mainly shown in front end webpages.

Front end presentation

The whole progress of testing will be presented as a video and the related front end webpages. We will upload them to our github. In addition, we will provide our back end code completely in our github.

Design conclusion (problem unsolved + optimize 展望)

Compared the project design in application layer and the code design, several simplified measures are as follow:

**PersonalFileAccount**

1. In the personal file account code design, we simply add two types of tag, “education” and “household”, in the personal log. But as the types of log can be expanded, we can add new types of log in our code at any time.
2. In our code, we build a hashmap (key,value) pair utxopool for quick searching of the link between event and event hash.

**PersonalFileAccountUTXOPool**

In this class, we aim to build a fast channel for matching event and its related personal account. It will not affect our application design.

**Event**

1. In our code, we simply separate personal id from event message to realize the classification function (put event in corresponded personal account).
2. We also separate publisher id from signature for verification.
3. We simplify the receipt MPT tree structure to a attribute approval in our code. It can be optimized if we have adequate time.

**EventPool**

We build this class to set a fast channel for the link of event and the propose EOA. It build for fast matching, no conflicts with our application design.

**EventHandler**

We build this class for mainly two reasons. First, we build a fast channel to link event and its related PFA and EOA. Second, we encapsulate a class for validating a series of events and return valid events set and invalid events set. Validating function before events are packed and added in a block by miners.

**Block & BlockNode**

1. These two classes realize the function of a block in our project system. In our code, we add the miner account in the class directly. This differs from our application design.
2. For better realizing the block function, we add following addition attributes to perfect the block: childnodes which contain the information of children blocks, eventpool and p\_pool which respectively indicates link between event and EOA & PFA, changes of EOA & PFA including await and remove list.
3. The addblock function in our code have some characteristic that meet the application design in another way. After validation success, the block automatically add the parent block’s EOA & PFA, await list of parent block’s EOA & PFA, and remove the remove list of parent block’s EOA &PFA. In the meantime, the current block updates the newest await list and remove list of EOA & PFA. At last, we update three state to finish addblock function. First, the function update miner’s balance and status after adding a block. Second, the function inherits the status of our MPT trees (simply arraylist). Third, the function updates the series of events.

**BlockChain**

1. In BlockChain class, we construct one attribute genesisNode belongs to BlockNode class. It is different to the nature of our application design. This feature does not affect our application design concept.
2. In BlockChain class, we construct many print functions, including printing EOA & PFA information, event information, personal log information and the whole block information.

We design the code in this way to help us build front end.

**Problems unsolved & limitation**

1. Take a look at our blockchain file management application scenarios, it can be seen that our blockchain file management is still a partial change of the management mode based on the traditional mode, or a simplified private chain demonstration sample, rather than a real blockchain file management application with the characteristics of extensive distribution without trust.
2. Several application problems still exist: first is file quantity is little, second is specific catalog not classified, third is the application scope is limited, fourth is have authority for its endorsement and decentralization is not complete, fifth the application is all in the pilot phase, sixth is making use of the archives and information content should be timely feedback users types, has the certain passivity.
3. Some other external limitation likes: traditional trust concept may hard to convert; the potential risk of leaks via public held system; technique rule may depress the motivation for specific units; lack of technical standard in the industries and the lack of law built already.

**Future expectation**

As the maturity of blockchain technology, more success application samples of blockchain technology, more trust built with civilian, government and blockchain-based application and industry standard and law standard built, government can build the whole ecosystem of government affairs public chain to absorb all the league chains serviced for government departments.

Acknowledgement

Appendix （code）

Reference