(1) 21→09 OCT 2022 #21

Blockchain Agenda

Bitcoin White paper

NodeJS - Agenda

1) What is npm + Why is it used
2) Modules in nodeJS
3) Events
4) http server

White paper

1) New research
2) New technology

Back story of BTC

How BTC comes into existance?

2008

1998-2002 - Internet companies → Start a information

2002-2008 → Boom IT Sector

Quantum Computing

__/__/__

(2)

2008 – Very Big Global recession (USA)

Big Bank financial recession ⟹ Bank currupt

Growth Stoppage ✓

why – Loans during time financial sector inflation

bail out

Too much spending

2008 Global financ crashes ⇒ Creation of Bitcoin

2008-2009 — "Satoshi Nakamoto"

2009 → published whitepaper (Research paper)

→ Alternative to traditional banking

✗ Bank

funds → lending borrowing currency transaction Nation Wise } Alternative idea of problem

Smart people elon Musk

BTC Whitepaper

IT Companies

→ Crypto currency → Blockchain Technology

2012 → becoming popular

white paper

peer to peer

Digital signature — private key & public

(3)

peer to peer → senders to Recivers

__/__/__

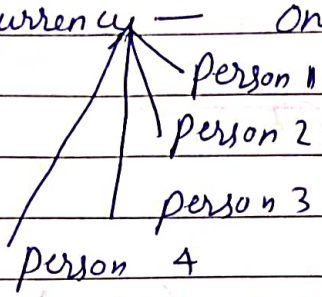
double Spending

image → multiple people

binary

text / audio / video / Graphics

digital currency — only transaction



double spending problem

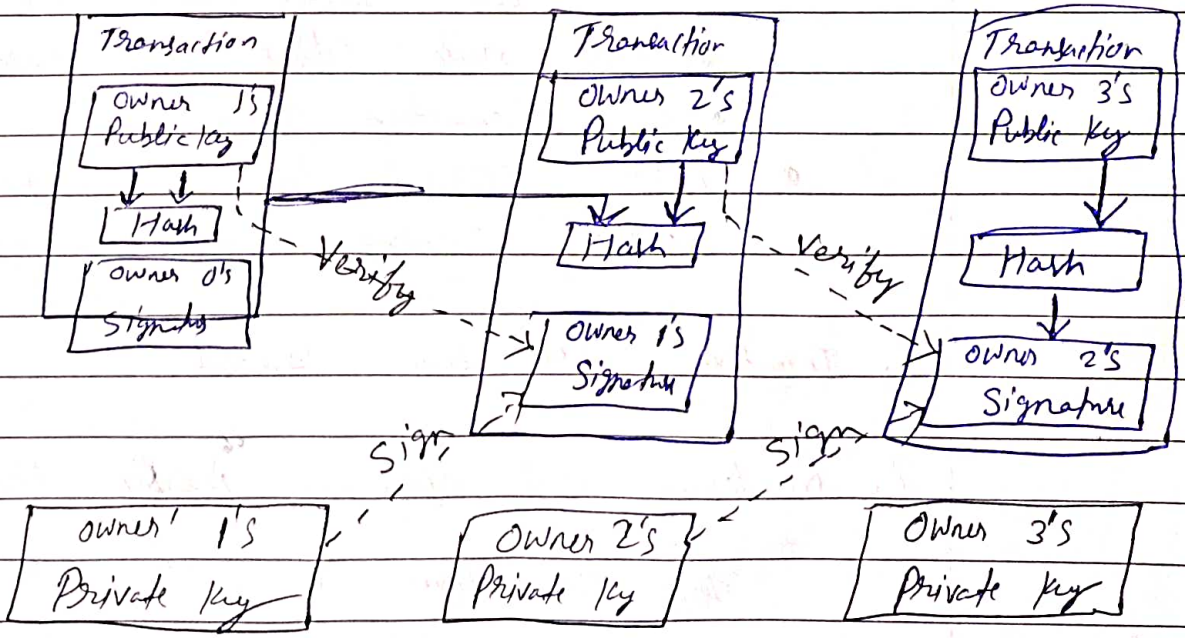
1. Introduction

non-reversible — not money back

- Bank closed → urgent money Solve
- The system is secure as long honest node

2. Transactions

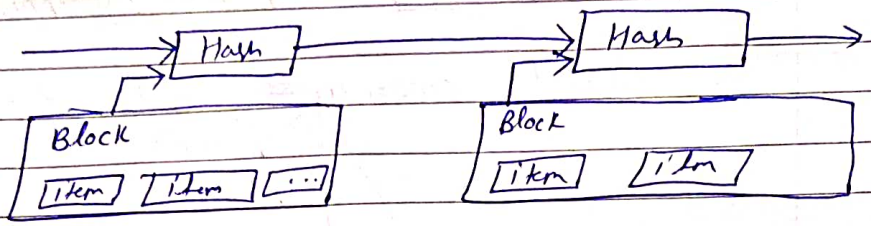
- electronic coin digital signature.



Who is sender reciver

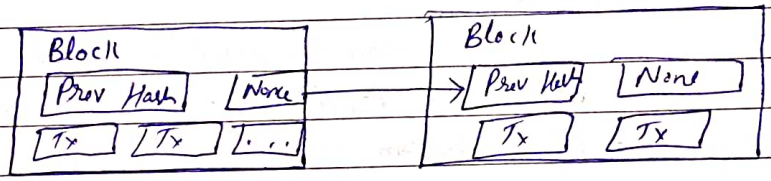
__/__/__

3. Time stamp Server • Hash of Block containing transaction to be timestamped and publish on the network.



4. proof of Work (Nonce)

Such as SHA-256



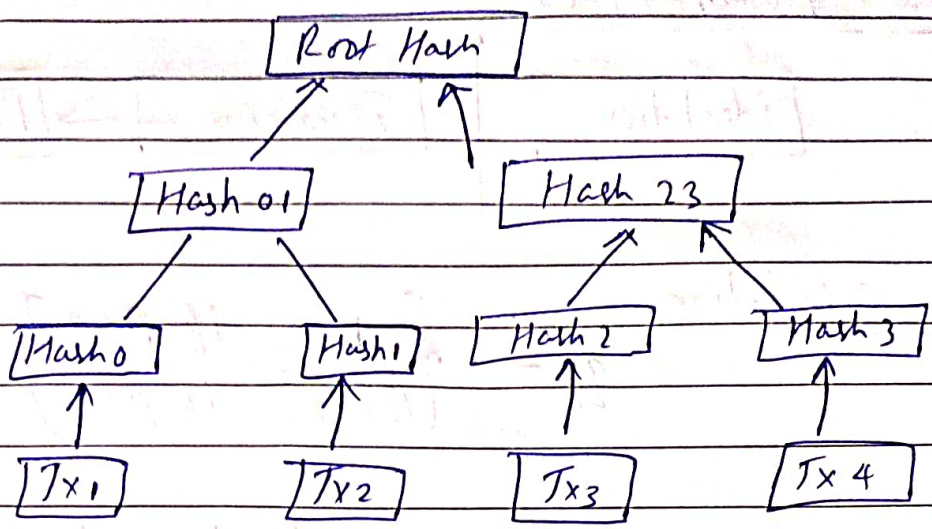
5. Networks

1. New transaction are broadcast to all nodes (Memespool)
2. Each node collect new transaction into a block.
   - All transaction valid
   - Nodes expres their acceptance of the block working creating next bloc in the chain.

6. Incentive — Coin base reward

7. Reclaiming Disk space — "Merkle tree"

two transaction combined like as binary tree.

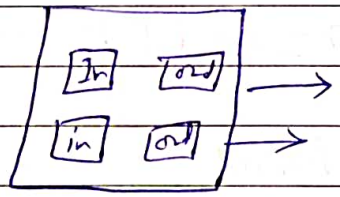


8. Simplified Payment Verification

Same as but check

9. Combining and Splitting Value

- Split and Combined, transaction contain multiple input and output



physical currency note

1500

⇒ 3x5000

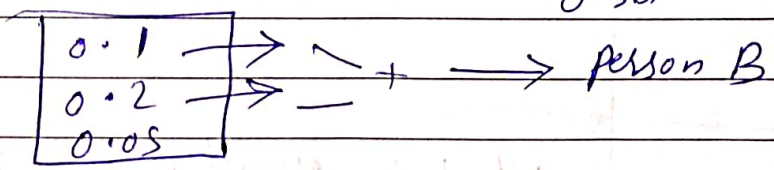
2000 + change

Person A

0.1 BTC

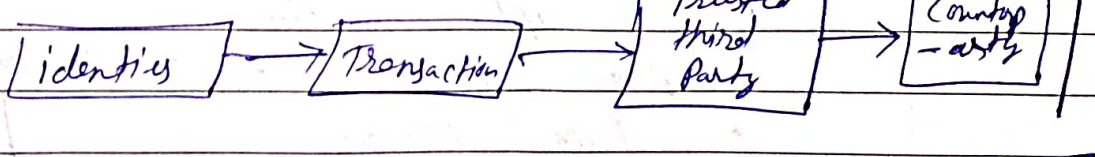
0.2 BTC

0.05 BTC



0.3BTC

10. privacy (Traditional privacy Model)



(6)

New Privacy Model

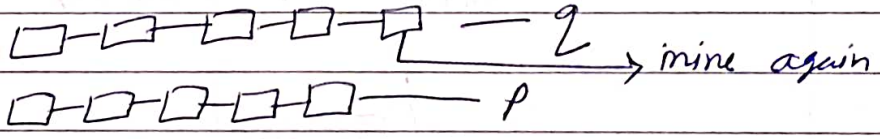
identities	Transaction → Public

11. Calculation

$$q_z = \begin{cases} 1 & \text{if } p \le q \\ (q/p)^z & \text{if } p > q \end{cases}$$

p = probability an honest node find the next block
q = probability the attackes find next block
q_z = probability the attackes will ever catch up from z block behind.

malicious attackes q.



3 block behind

$z = 3$

$p = 0.6$

$q = 0.5$

What is NPM (Node Package Manager)

- nodeJS backend framework
- javascript runtime
- prior nodeJS → JS runtime for Google chrome V8 engine
- DOM HTML

local PC offline

2016 - 2017 → Without browsers

New project

1) project idea
2) Features
3) plan → Document, Wireframe
4) Writing code

High level ↑ Low level ↓
HLD, LOL

a) directly write Website

b) to test backend first → then integrate frontend

NPM

NodeJS → read made / plug & play features

Crypto →

ReactJS → npm

Repository of backend + frontend mods + framework
java, python, R, C++ etc

Package → It is collection of file with specific functionality.
→ Reusable code
npm → Hardhat, truffle, Web3.JS

* ⇒ ⇒ npm install package-name

ex- npm install hardhat

yarn → package manages

(8)

__/__/__

What are modules?

add.JS ⇒ function { }

cart.JS
payment.JS
home.JS

Website

Any big project → Compile all files → Package

Core modules
- installed during nod installation
- third party modules.
- local modules → projects.

modules

fs →
http →
https →
os mo →
dns →
url →

main core modules

4/5 modules → different funct " "

Package → single unit

VS Code

File System Module

- It is used for reading files, writing inside files.
- Perform calculation & modification on files.
- text files
- reading files.

__/__/____

module

```
const fs = require("fs");
fs.readFile('demo.txt', 'utf-8' or (err, data) => {
    if (err) {
        throw err.name;
    }
    else {
        console.log(data);
    }
})
```

→ or (data.toString())

fs.readFile — return a binary ⇒ hexadecimal format
→ english

conversion

Writing file — (new file with txt form)

```
const fs = require('fs')
const path = require('path');

fs.writeFile('write.JS', "this is new file, we are are writing something", (err) => {
    console.log('successfully written to the file');
    if (err) {
        throw err.name;
    }
})
```

__/__/__

file system → core / inbuilt module

Own Module → local modules
→ Separate files
→ one's file code into another.

util.js ⇒

```
function
Constant test = function () {
Console.log (" this is just a test case to
              check local modules");
}
modules.exports = test;
```
→ multiple [test, name]
→ name

util2.js

forward slash use

```
// local module are imported by ./ reference
const utility = require ("./utils.js");
utility ();
```
path

Events

nodeJS — Single thread event driven

frontend API → Server

Click

Req / client side comm happen using event

Synchronous

Asynchronous —

(11)

__/__/____

emitting an event

emit event

↳ event listner & Asynchronous
process ↓
non blocking

⇒ events.JS

on indicates what to do when event is triggered from client side

```
Const EmitEvent = require("events");
let eventEmitter = new EmitEvent();
eventEmitter.on('mySimpleEvent', (msg) => {
    console.log(msg);
});

eventEmitter.emit('mySimpleEvent', "First event has occured");
```

Project

⇒ Adding item to cart

process payment
↳ event triggered → database
↳ bank account

response
Payment successful

React JS complete
↳ complete website with UI
→ React JS + nodeJS

example event ⇒

"learning lot of time building projects"

X X X