BLOCKCHAIN

Blockchain is digitized, decentralized, public ledger of all cryptocurrency transactions.It's a distributed database.Blockchain can not be controlled by any single entity.It's transparent and provides us trust.

ETHEREUM BLOCKCHAIN

Ethereum is an open-source, public, blockchain-based distributed computing platform featuring smart contract (scripting) functionality.Ether is ethereums inbuilt native cryptocurrency used for paying for smart contracts to run.

Smart Contracts:

A smart contract, also known as a cryptocontract, is a computer program that directly controls the transfer of digital currencies or assets between parties under certain conditions. Programming languages for smart contracts are solidity,serpent,LLL.

Gas Limit,Gas Price,Transaction Fees:

To do anything on the Ethereum platform, you need to pay for it, and the payment (or fee) is calculated in Ether (ETH) via an intermediary benchmark called *gas limit* and *gas price.*

Ether = Tx Fees = Gas Limit * Gas Price

Gas Limit:

Usually, when one is talking about "gas" in Ethereum, they are referring to "gas limit". This simply means some amount of fuel is required to execute that operation or run that particular smart contract code.Gas limit depends on the size of code we are deploying on blockchain.

Gas Price:

If you want to pay less for your transaction, you can do so by varying the other variable which also determines the final cost (or Tx fee) of the transaction. This variable is called "gas price".

We can vary the gas price.But lowering gas price will take more time for contract to be executed and validated.

Ethereum Virtual Machine:

Ethereum's core innovation, the Ethereum Virtual Machine (EVM) is a Turing complete software that runs on the Ethereum network. It enables anyone to run any program, regardless of the programming language given enough time and memory. The Ethereum Virtual Machine makes the process of creating blockchain applications much easier and efficient than ever before.

Uncles and Orphans:

Ethereum's rate of block generation is much higher than Bitcoin's (250 blocks per hour on Ethereum vs 6 blocks per hour on Bitcoin). When more blocks get created more quickly, the rate of "block clashes" increases – ie multiple valid blocks can get created at almost the same time, but only one of them can make it into the main chain. The other one

“loses”, and the data in them is not considered part of the main ledger, even if the transactions are technically valid.

In Bitcoin these non-mainchain blocks are called *orphans* or *orphaned blocks* and they do not form part of the main chain in any way and are never referenced again by any subsequent blocks.

In Ethereum they are called *uncles*. Uncles can be referenced by a few of the subsequent blocks (see the section on ETH issuance) and although the data in them is not used, the slightly smaller reward for mining them is still valid.

Ethereum software: geth, eth, pyethapp:

The official Ethereum clients are all open source.geth(written in Go),eth(written in C++),pyethapp(written in python).

What can ethereum used for?

Ethereum can be used to build and deploy decentralized applications.Decentralized applications (dApps) are applications that run on a P2P network of computers rather than a single computer.

Benefits of Ethereum decentralized platform:

Because decentralized applications run on the blockchain, they benefit from all of its properties.

- **Immutability** – A third party cannot make any changes to data.
- **Corruption & tamper proof – Apps are based on a network formed around the principle of consensus, making censorship impossible.**
- **Secure – With no central point of failure and secured using cryptography, applications are well protected against hacking attacks and fraudulent activities.**
- **Zero downtime – Apps never go down and can never be switched off.**

Ethereum blockchain is based on proof of work and they are planning to shift to proof of stake.

1)Proof of work:

proof of work is a requirement to define an expensive computer calculation, also called mining.When you want to set a transaction this is what happens behind the scenes:

- **Transactions are bundled together into what we call a block;**
- **Miners verify that transactions within each block are legitimate;**
- **To do so, miners should solve a mathematical puzzle known as proof-of-work problem;**
- **A reward is given to the first miner who solves each blocks problem;**
- **Verified transactions are stored in the public blockchain**

2)Proof of stake:

Proof of stake is a different way to validate transactions based and achieve the distributed consensus.It is still an algorithm, and the purpose is the same of the proof of work, but the process to reach the goal is quite different.

In the proof of stake, the creator of a new block is chosen in a deterministic way, depending on its wealth, also defined as stake.

No block reward.Also, all the digital currencies are previously created in the beginning, and their number never changes.This means that in the PoS system there is no block reward, so, the miners take the transaction fees.This is why, in fact, in this PoS system miners are called forgers

ETHEREUM SETUP

Genesis Block Information:
Genesis block is the first block in the blockchain.It contains following information.

mixhash A 256-bit hash which proves, combined with the nonce, that a sufficient amount of computation has been carried out on this block: the Proof-of-Work (PoW). The combination of nonce and mixhash must satisfy a mathematical condition described in the Yellowpaper, 4.3.4. Block Header Validity, (44). It allows to verify that the Block has really been cryptographically mined, thus, from this aspect, is valid.

nonce A 64-bit hash, which proves, combined with the mix-hash, that a sufficient amount of computation has been carried out on this block: the Proof-of-Work (PoW). The combination of nonce and mixhash must satisfy a mathematical condition described in the Yellowpaper, 4.3.4. Block Header Validity, (44), and allows to verify that the Block has really been cryptographically mined and thus, from this aspect, is valid. The nonce is the cryptographically secure mining proof-of-work that proves beyond reasonable doubt that a particular amount of computation has been expended in the determination of this token value. (Yellowpager, 11.5. Mining Proof-of-Work).

difficulty A scalar value corresponding to the difficulty level applied during the nonce discovering of this block. It defines the mining Target, which can be calculated from the previous block's difficulty level and the timestamp. The higher the difficulty, the statistically more calculations a Miner must perform to discover a valid block. This value is used to control the Block generation time of a Blockchain, keeping the Block generation frequency within a target range. On the test network, we keep this value low to avoid waiting during tests, since the discovery of a valid Block is required to execute a transaction on the Blockchain.

alloc Allows defining a list of pre-filled wallets. That's an Ethereum specific functionality to handle the "Ether pre-sale" period. Since we can mine local Ether quickly, we don't use this option.

coinbase The 160-bit address to which all rewards (in Ether) collected from the successful mining of this block have been transferred. They are a sum of the mining reward itself and the Contract transaction execution refunds. Often named “beneficiary” in the specifications, sometimes “etherbase” in the online documentation. This can be anything in the Genesis Block since the value is set by the setting of the Miner when a new Block is created.

timestamp A scalar value equal to the reasonable output of Unix time() function at this block inception. This mechanism enforces a homeostasis in terms of the time between blocks. A smaller period between the last two blocks results in an increase in the difficulty level and thus additional computation required to find the next valid block. If the period is too large, the difficulty, and expected time to the next block, is reduced. The timestamp also allows verifying the order of block within the chain (Yellowpaper, 4.3.4. (43)).

parentHash The Keccak 256-bit hash of the entire parent block header (including its nonce and mixhash). Pointer to the parent block, thus effectively building the chain of blocks. In the case of the Genesis block, and only in this case, it’s 0.

extraData An optional free, but max. 32-byte long space to conserve smart things for ethernity.

gasLimit A scalar value equal to the current chain-wide limit of Gas expenditure per block. High in our case to avoid being limited by this threshold during tests. Note: this does not indicate that we should not pay attention to the Gas consumption of our Contracts.

Setup Instructions:

1)Commands to install ethereum client:
1.sudo apt-get install software-properties-common
2.sudo add-apt-repository -y ppa:ethereum/ethereum
3.sudo apt-get update
4.sudo apt-get install ethereum

2) Create a main account for your network and specify other node parameters
geth --networkid 500 --identity node1 --verbosity 3 --nodiscover --nat none
--datadir=/home/fractaluser/blockchain/privatenet account new

Address: {77f4dc4db3025ec9a9637939a1822d1a0bd9f168}

3) Use custom genesis block and initialize blockchain
geth --networkid 500 --identity node1 --verbosity 3 --nodiscover --nat none
--datadir=/home/fractaluser/blockchain/privatenet init
/home/fractaluser/blockchain/custom_genesis.json

4) Start mining:

geth --networkid 500 --identity node1 --verbosity 3 --nodiscover --nat none --datadir=/home/fractaluser/blockchain/privatenet --mine --ipcpath /home/fractaluser/blockchain/privatenet/geth.ipc.

5)**Download ethereum mist wallet available here:**
https://github.com/ethereum/mist/releases

6)**Start mist client and connect to the private blockchain:**
./mist --network 500 --rpc ~/blockchain/privatenet/geth.ipc.

Frameworks to deploy smart contracts are ethereum wallet,geth javascript console,truffle.

1)For deploying smart contracts on ethereum wallet,go to the wallet ,then click n new contract and paste the solidity code and hit on deploy button.

2)For deploying smart contract on geth Javascript console:
1.First on terminal go to directory of you code,then run the following command:
echo "var testOutput=`solc --optimize --combined-json abi,bin,interface piracy_detection_1.sol`" > test_1.js

This will create an ABI (Application Binary Interface) of the code and store it in “test_1.js” file. (ABI is the interface between two program modules, one of which is often at the level of machine code. The interface is the de facto method for encoding/decoding data into/out of the machine code.)

2.Following commands are executed on geth console:
1.geth --networkid 500 --identity node1 --verbosity 3 --nodiscover --nat none --rpc --rpcport=8545 --rpccorsdomain=* --port=30303 --datadir=/home/fractaluser/Test/privatenet console

This will start the geth javascript console

2.loadScript("test_1.js").

3.var detectionCompiled=testOutput.

4.var detectionContract = eth.contract(detectionCompiled.contracts["piracy_detection_1.sol:detection"].abi).

5.personal.unlockAccount(eth.accounts[1], "newOne", 60000000).

6.var detectionContract = web3.eth.contract(JSON.parse(detectionCompiled.contracts["piracy_detection_1.sol:detection"].abi)).

7.var detection = detectionContract.new({from:eth.accounts[1], data: detectionCompiled.contracts["piracy_detection_1.sol:detection"].bin, gas: 1000000}, function(e, contract){ if(!e) {if(!contract.address) {console.log("Contract transaction send: TransactionHash: " + contract.transactionHash + " waiting to be mined...");} else {console.log("Contract mined! Address: " + contract.address);console.log(contract);}}})

8.var detection = detectionContract.new({from:eth.accounts[1], data: "0x"+detectionCompiled.contracts["piracy_detection_1.sol:detection"].bin, gas: 1000000}, function (e, contract) {console.log(e, contract);if (typeof contract.address !== 'undefined') {console.log('Contract mined! address: ' + contract.address + ' transactionHash: ' + contract.transactionHash);}});

9.miner.start(); admin.sleepBlocks(1);miner.stop();

Web3 API

To make your app work on Ethereum, you can use the web3 object provided by the web3.js library. Under the hood it communicates to a local node through RPC calls. web3.js works with any Ethereum node, which exposes an RPC layer.

Tutorials on web3 api are available on:https://github.com/ethereum/wiki/wiki/JavaScript-API
And https://web3js.readthedocs.io/en/1.0/.

NodeJS commands to decode input data:

Download ethereum input data decoder(npm install ethereum-input-data-decoder)

1)const InputDataDecoder = require('ethereum-input-data-decoder');

2)Pass abi:

abi=
JSON.parse('[{\"constant\":false,\"inputs\":[{\"name\":\"newOwner\",\"type\":\"address\"},{\"name\":\"_productname\",\"type\":\"string\"}],\"name\":\"transfer\",\"outputs\":[],\"payable\":false,\"stateMutability\":\"nonpayable\",\"type\":\"function\"},{\"constant\":true,\"inputs\":[{\"name\":\"\",\"type\":\"address\"}],\"name\":\"balances\",\"outputs\":[{\"name\":\"\",\"type\":\"uint256\"}],\"payable\":false,\"stateMutability\":\"view\",\"type\":\"function\"},{\"constant\":true,\"inputs\":[{\"name\":\"_target\",\"type\":\"address\"}],\"name\":\"check_balance\",\"outputs\":[{\"name\":\"\",\"type\":\"uint256\"}],\"payable\":false,\"stateMutability\":\"view\",\"type\":\"function\"},{\"constant\":false,\"inputs\":[{\"name\":\"_productname\",\"type\":\"string\"},{\"name\":\"amount\",\"type\":\"uint256\"},{\"name\":\"LicenseID\",\"type\":\"uint256\"},{\"name\":\"validity\",\"type\":\"uint256\"}],\"name\":\"organization\",\"outputs\":[],\"payable\":false,\"stateMutability\":\"nonpayable\",\"type\":\"function\"},{\"constant\":false,\"inputs\":[{\"name\":\"_productname\",\"type\":\"string\"}],\"name\":\"buy\",\"outputs\":[],\"payable\":false,\"stateMutability\":\"nonpayable\",\"type\":\"function\"},{\"constant\":true,\"inputs\":[],\"name\":\"owner\",\"outputs\":[{\"name\":\"\",\"type\":\"address\"}],\"payable\":false,\"stateMutability\":\"view\",\"type\":\"function\"},{\"constant\":false,\"inputs\":[{\"name\":\"_name\",\"type\":\"string\"},{\"name\":\"_emailid\",\"type\":\"string\"},{\"name\":\"_company\",\"type\":\"string\"}],\"name\":\"Register\",\"outputs\":[],\"payable\":false,\"stateMutability\":\"nonpayable\",\"type\":\"function\"},{\"inputs\":[],\"payable\":false,\"stateMutability\":\"nonpayable\",\"type\":\"constructor\"},{\"anonymous\":false,\"inputs\":[{\"indexed\":false,\"name\":\"f

rom\",\"type\":\"address\"},{\"indexed\":false,\"name\":\"to\",\"type\":\"address\"},{\"indexed\":fal
se,\"name\":\"amount\",\"type\":\"uint256\"}],\"name\":\"Sent\",\"type\":\"event\"}]')

3)const decoder = new InputDataDecoder(abi);

4)Pass input data:
const data =
`3e83980100000000000000000000000000000000000000000000000000000800
0000000000000000000000000000000000000000000000000000027100000000000
0000000000000000000000000000000000000000000030390000000000000000000
0000000000000000000000000000000000000000100000000000000000000000000
0000000000000000000000000000000066c6170746f7000000000000000000000000000
00000000000000000000000000`;

5)const result = decoder.decodeData(data);

6)console.log(result);