Block chain Technology.

Blockchain :-

Blockchain is a System of recording Information in a way that makes it difficult or impossible to change, hack or cheat the System.

Transactions are recorded with an immutable cryptographic Signature called as hash.

Blockchain not only allows to add new data to the database but it also ensures that all the Users on the network have Exactly the Same data.

A Blockchain is a distributed and decentralised linked data Structure for data Storage and retrival which also ensures that the data is resistant to any modification.

The transactions are recorded in a distributed ledger and ledger is distributed across the Entire network.

There is no Central authority which manages the database. Instead the database is decentralized and

managed by multiple participants across the network.
Each user has the copy of ledger and data is encrypted by complex Algorithm.

* Distributed Database:-

Distributed System:-

It is a collection of independent computers connected via a network that appears to its user as a single coherent system
The distributed system looks alike a single computer rather than a collection of seperate computers.

Features:-

i) No shared Memory

ii) Each runs on its own local OS

iii) Heterogeneous

Characteristics :-

i) Present as a Single System image
   a) Hides internal organization & Communication details.
   b) Provide Uniform Interface

ii) Easily Expandable
   a) Adding new Components are hidden from the user.

iii) Continous Availability.
   a) Failures in one Component can be covered by other Components.

Decentralized Database

The decentralized database managed by multiple participants is known as Distributed Ledger Technology (DLT)

It promises benefits in Traceability, Collaboration, Organization, identification, Credibility and transparency.

Properties :-

1. Distributed :-
   Provides transparency by making copy of transactions to every participants ledger.

2. Immutable :-
   All validated records are irreversible. Hence cannot be changed.

3. Time-stamped :- The Node records the time-stamp of the transaction.

4. Unanimous :- All the participants agree to the validity of each record.

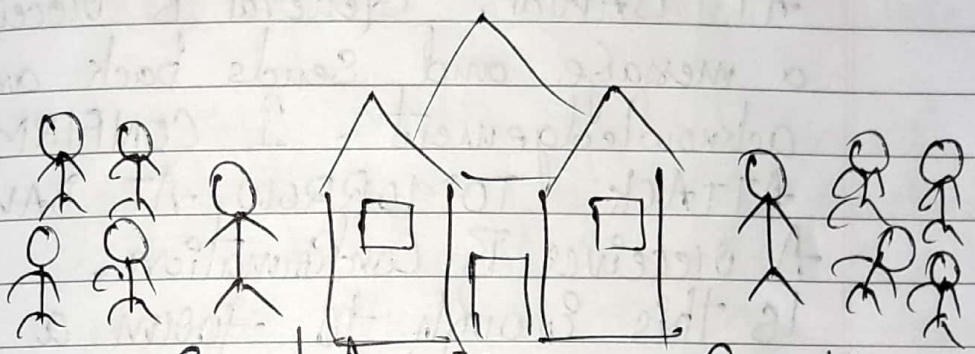
5. Anonymous :- The identity of participants is anonymous.

6. Secure :- All the records are individually encrypted.

* Two Generals Problem :-

The two Generals Problem also known as two general Paradox or two Armies Problem is a classic computer Science and Computer communicated Experiment.

The Story of two General's



Imagine two armies led by two General's Planning an attack on Common enemy. The two generals have to communicate with Each other to plan a Synchronized attack as this is the only chance to win.

The only problem is that to Communicate with each other they have to Send a messenger across the enemy's territory. If the messenger is captured then message he's carrying is lost.

Each general want's to know that the other general should know when to attack. Attacking alone is pointless.

1st Scenario:-
lets call our general A & B. Let's assume everything goes perfectly.
General A who is the leader Sends a message ATTACK TOMORROW AT DAWN. General B receives a message and sends back an acknowledgement - I CONFIRM ATTACK TOMORROW AT DAWN.
A receives B Confirmation.
Is this Enough to form a Consensus between the generals.
Unfortunately not, as General B is not sure about the confirmation received by General A. So, If General A confirms General B's Confirmation then of course the Confirmation has to be Confirmed as we end up with an Infinite Exchange of Confirmations.

2nd Scenario:-
Assume General A Sends a msg to General B. Some time has passed but no Confirmation came from General B. There are two possibilities here Either the messenger A who was carrying msg was captured or messenger B who was carrying Confirmation was captured. We ended up in inconsistent State

We quickly realise that we cannot guaranteed that Consensus is reached. and Each general is certain that his army will attack at the same time. The Problem remain Unsolvable.

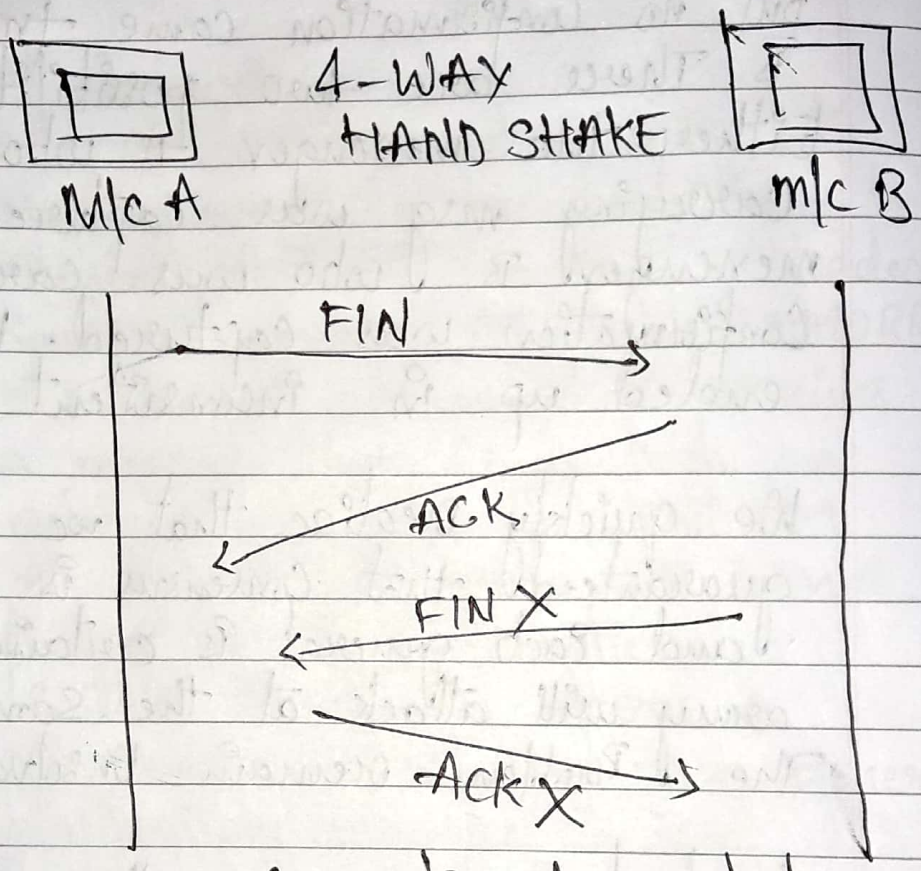
Similarly we can see the analogy to Computers Communication.

Instead of two generals, let's imagine two Computer Systems talking each other The problem here is again untrustable and inconsistent states.

→ A very Common Example is TCP PROTOCOL

TCP - Transmission Control Protocol

TCP



Connection terminated.

TCP Uses 4-way handshake mechanism to terminate the Connection In this the System wants to terminate Sends FIN msg. the other System replies with ACK with its FIN msg to the System who initialized the termination The Initialized system send its ACK msg & terminates when rec correctly.

In above Scenario there is no problem. But, what if the 2nd FIN msg is lost. we end up with half open Connection

Pragmatic Approach:-

Approach 1:-

Going back to Generals:- What If General A sends 100 msg instead of 1 with serial no.s assuming general B will receive atleast 1. On other hand General B based on missing no. in Seq would be able to guess how reliable the communication channel is & reply with appropriate no. of Confirmation

Approach 2:-

lets assume it takes 20 mins to cross the valley & come back. General A starts sending msg every 20 mins until he gets Confirmation from B. when Confirmation arrives General A stops sending msgs. Meanwhile General B after sending ack waits for other messenger coming from A. But this time absence of messenger Builds up General B Confidence.

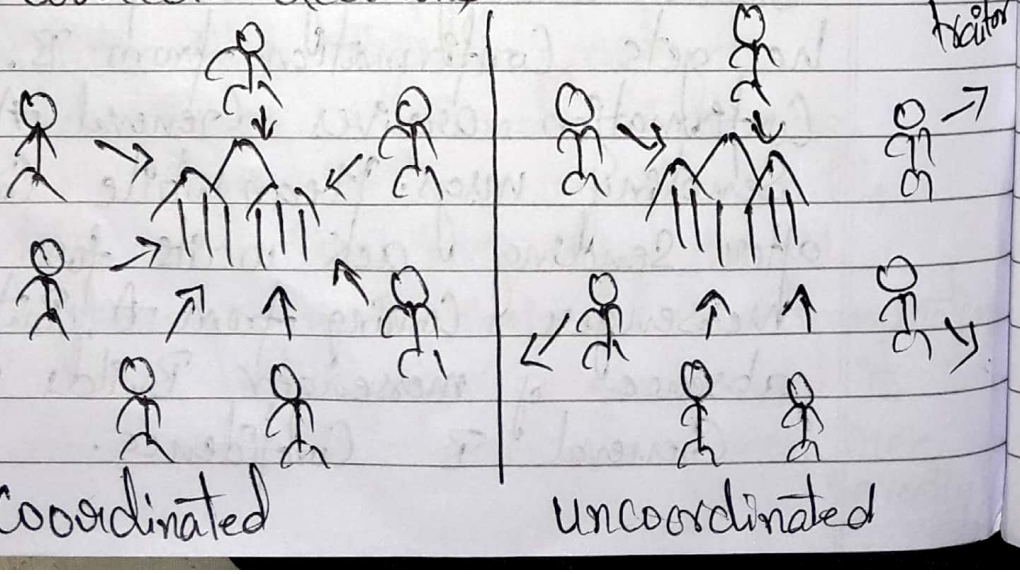
In this case we have a clear Speed vs Cost tradeoff and its up to us which approach is more suitable to our problem.

* Byzantine General's Problem:-

First theorized by the Mathematicians
Leslie lamport
Marshall Pease
Robert Shostak

The Generals are a metaphor for nodes in decentralized networks.
The core idea behind this thought Experiment is this.

How do you ensure that a peer-to-peer distributed network with no central authority can make correct decisions.



We have a castle in the middle which is Extremely well Stocked & fortified

The lieutenants have surrounded the castle. The general plans to lauch an attack, Since the army is scattered the general doesn't have centralized control.

The only way that the castle could be defeated is if the byzantine army lauches a planned & synchronized attack. If there is any miscommunication the plan will fail.

Speaking of Communication. The only way that general can synchronize a Stroke is by sending messages via messenger.

Failure Scenario :-

1. Imagine that the messenger are traitors and they convey misInformation

2. Presume that one messenger out of four is traitor and he changes a msg to RETREAT Instead of ATTACK.

3. What if the messenger gets Captured & message gets tampered.

4. What if general himself is corrupted

Byzantine General Algorithm:-

There are total n nodes
There are m malicious nodes
$N > 3m$ ensures that 2/3 rd majority is maintained.

Stage 1:-

This is data collection Stage, which has m+1 rounds. During this stage the nodes are continously sending and receiving messages

Stage 2:-

This is a decision making Stage. Based on data collected in stage 1 the system as a whole comes to a decision.

classmate
Date 22/6/21
Page

* Fault Tolerance :-

Byzantine Fault Tolerance (BFT)

BFT is the property of a System that is able to resist the 1 class of failures derived from the Byzantine General Problem.
This means that BFT System is able to continue Operating even if some of the nodes fail or act maliciously.

There is more than one possible solution to BGP & therefore multiple ways of building BFT System.

Likewise There are different approaches for a blockchain to achieve BFT and this leads us to so called Consensus Algorithm.

Consensus Algorithm is the mechanism through which blockchain network reach Consensus.
The most Common Implementations are
i) Proof-of-work (PoW)
ii) Proof-of-Stack (PoS)

In Bitcoin, PoW Consensus Algorithm defines how rules will be followed in order to reach the Consensus.

Although the Concept of PoW is older than Cryptocurrency, Satoshi Nakamoto developed a modified version of PoW that Enable the creation of Bitcoin as BFT System.

PoW is not 100% tolerant to BF. But due to its Cost-Intensive mining process & underlying Cryptographic techniques.

It is one of most Secure & reliable implementation for BCN : PoW.
It is one of the most genius solution to BF.

Therefore, PoW & PoS are very intresting approaches as BFT System, and potential applications are Certainly Inspiring widespread Innovation.

* Hadoop Distributed File System:-

Hadoop:- Hadoop is a software platform that makes it possible for users to manage huge amount of data
Its two primary functions are
i) Storage
ii) processing.

Hadoop Comprises of four primarily Components.
1. Hadoop distributed file system [HDFS]
2. Yet Another Resource negotiator [YARN]
3. MapReduce
4. Hadoop Common.

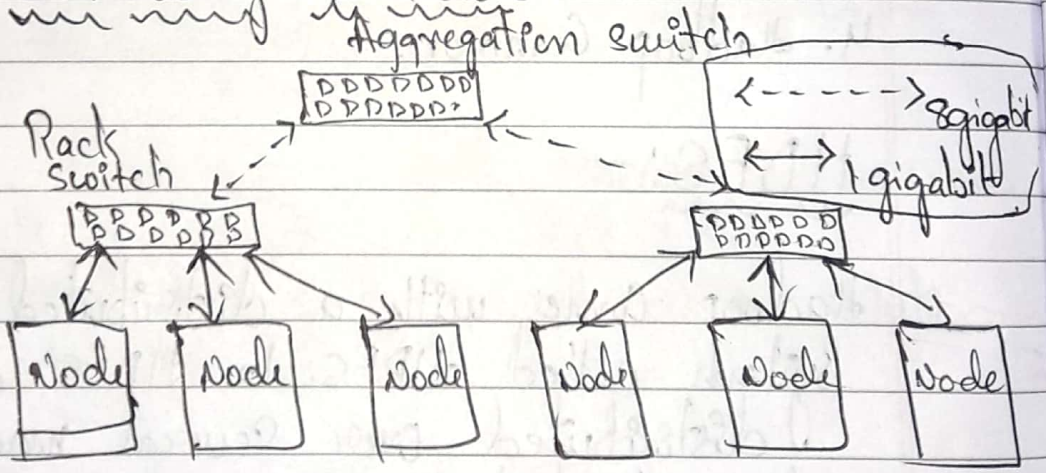
HDFS:-

Hadoop Comes with a distributed file System called HDFS. In HDFS data is distributed over several machines and replicated to ensure their durability to failure and high availability to parallel applications.

Uses:-

1. It is used to process very large files (hunderds of MB's, GB's & more)
2. Very expensive to build reliability into each application
3. Nodes fails every day
   The no. of nodes in a cluster is not constant
4. Need a common Infrastructure
   Efficient, reliable & easy to use.
5. Commodity Hardware.
   works on low cost

Commodity Hardware:-

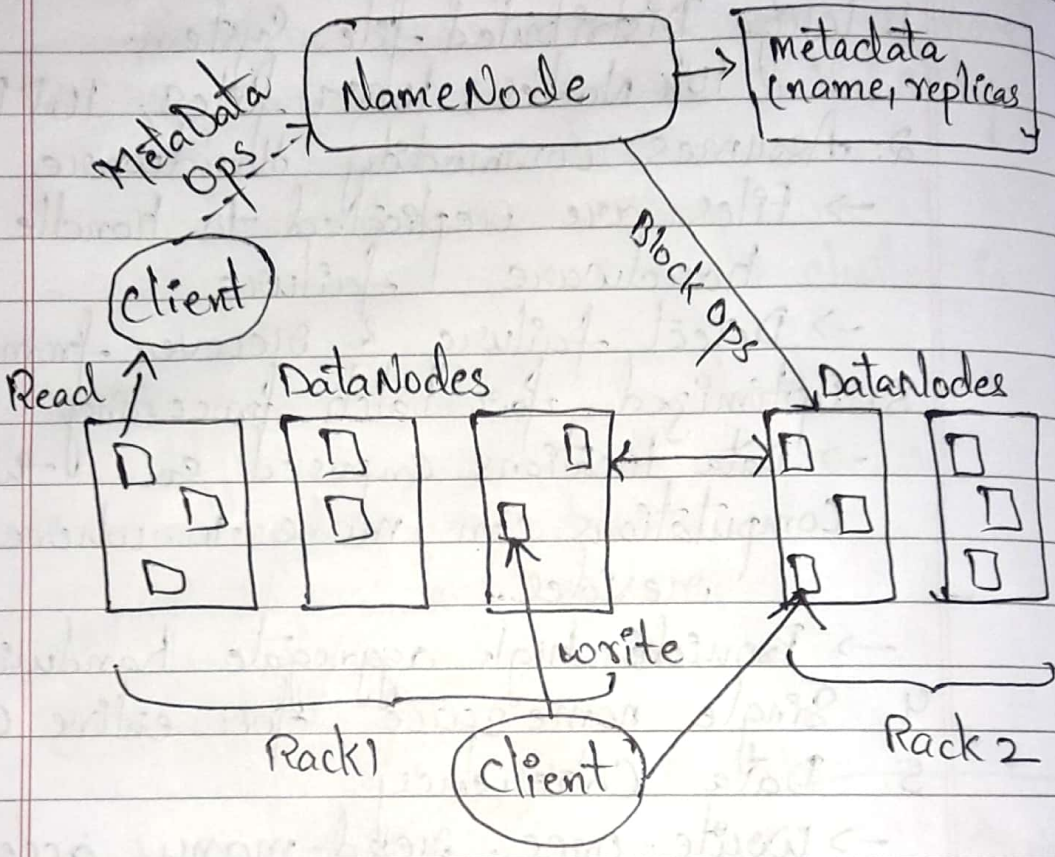


→ Typically a 2-level architecture
→ Nodes are Commodity PC's
→ 30-40 nodes per rack
→ Uplink from rack is 3-4 gigabit
→ Rack internal is 1 gigabit

Goals of HDFS :-

1. Large Distributed file System.
   10k nodes, 100M files, 10PB
2. Assumes commodity hardware
   -> Files are replicated to handle hardware failure
   -> Detect failure & recover from them
3. Optimized for batch processing
   -> Data locations exposed so that computations can move to where data resides
   -> Provide high aggregate bandwidth
4. Single name space for entire cluster
5. Data Coherency.
   -> Write-once-read-many access models
   -> client can only append to existing file
6. Files are broken up into blocks
   -> Typically 128MB or 64MB block size
   -> Each block replicated on multiple DataNode
7. Intelligent client
   -> Client can find location of blocks
   -> client access data directly from DataNode.

HDFS Architecture :-



Block :- A block is the minimum amount of data that it can read or write. HDFS blocks are 128MB. HDFS are broken into block-sized ya chunks.

NameNode :- HDFS works in master-worker pattern where NameNode works as a master. NameNode is Controller & manager of HDFS as

it knows the status & metadata of all files in HDFS.

DataNode :- It stores and retrieve blocks when they are told to by client or name node. They report back to name node

Functions of NameNode :-

-> Manages file System NameSpace
   1. Maps a file name to set of blocks
   2. Maps a block to DataNode where it resides
-> cluster Configuration Management.
-> Replication Engine for blocks.

NameNode MetaData :-

-> MetaData in Memory. (resides in main memory)
-> Types of metaData
   1. list of files
   2. list of blocks for each file
   3. list of DataNodes for each block
-> A Transaction log.
   Records file creations, file deletions etc

classmate
Date
Page

DataNode:-

→ A Block Server
Stores data in local file system
Stores meta data of blocks
Serves data & meta data to clients
→ Block Report
periodically sends report to name node
→ Facilitates pipelining of Data
forwards data to other specified DataNodes

Block Placement:-

→ Current Strategy.
1 replica on local node
2 replica on remote rack
3 replica on same remote rack
Additional replicas are randomly placed
→ Clients read from nearest replicas
→ would like to make this policy pluggable

HeartBeats:-

-> DataNodes sends heartbeat to the namenode once every 3 seconds
-> NameNodes uses heartbeats to detect datanode failure

Replication Engine:-

-> NameNode detects DataNode failure
choose new DataNodes for new replicas
Balances disk usage
Balances communication traffic to DataNode

Data Correctness:-

-> Use checksum to validate data (CRC32)
-> File Creation
1. Client computes checksum per 512 bytes
2. DataNode stores the checksum
-> File Access
1. Client retrieves the data & checksum from DataNode
2. If validation fail, client tries other replicas.

NameNode Failure :-

-> A Single point of failure
-> Transaction log stored in multiple directories
   1. A directory on local file system
   2. A directory on remote " "
-> Need to develop real HA Solution

Data pipelining :-

-> Client retrives a list of DataNodes on which to place replicas of block
-> Client writes block to first DataNode
-> The first DataNode forwards the data to next node in pipeline
-> When all replicas are written, the client moves on to write the next block in file

* Distributed Hash table :-

A Distributed Hash table is a distributed System that provides a lookup Service similar to hash table. key-value pairs are stored in a DHT and any participating node can efficiently retrieve the value associated with a given key.

The main advantage of a DHT is that nodes can be added or removed with minimum work around re-distributing keys.

Distributed Hash tables are used when no. of keys increases, memory becomes too big and time Complexity increases.

Ex:- Get 1000 Computers.
Creating hash table on each
To determine which computer owns
Object O. number h(O) mod 1000

Request is send to Computer to search/ modify in local hash table.

Problems:-
Computer sometimes breaks.

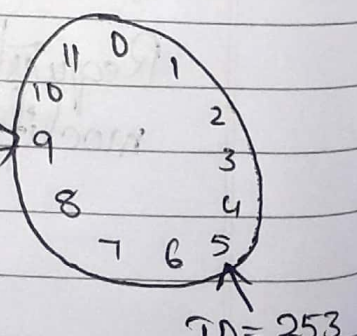
So Several copies of data is stored and need to relocate the data from broken Computer.
So new Computers are added then h(D) no longer woorks.

So, we go with Consistent hashing method!

1. Choose a hash function with cardinality m and put numbers from 0 to m-1 on a circle clock-wise
2. Each object O is then mapped to a point on the circle with with number h(O)
3. Map Computer IDs to Same Circle

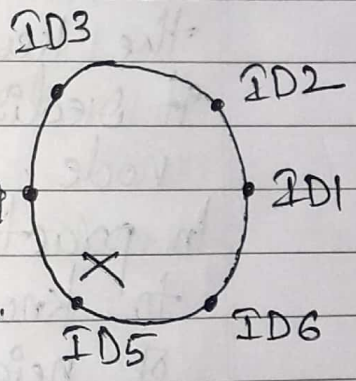
CompID → point number h(compID)

Ex:- m=12
If hash value of Steve is 9
Then it is mapped to point 9.
Similarly for ID.



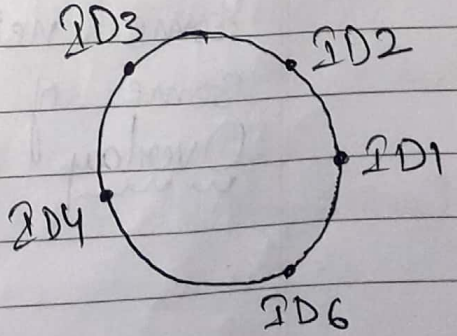
We make a rule that each object is stored on the so-called closest computer. closest in terms of distance along the circle. In this case, each computer stores all objects falling on some arc, which consists of all objects which are closer to this computer than to any other computer.

Consider 6 computers mapped to some points on circle and the arcs of the same color as the computers near them.



When new computers are added or if computer are broken its neighbour takes its data. So, it has two neighbour and arc is divided into parts.

If ID5 goes away than ID4 & ID6 will divide the arc amongst themselves



Another problem still needs to be solved is that when some Computers breaks, we need to copy or relocate the data.
For this the rule is that for any key each node will either store the key itself or it will be acquainted. It will know some other Computers which is closer to this key interms of distance on Circle.

And in that way if a request comes to some node, it either can find the key inside its own storage or it redirects the request to another node.
In practice we can make each node to know its neighbour & its neighbour of neighbour.

The network of nodes which knows some neighbours and they know some of their neighbour is called Overlay Network.

classmate
Date 23/6/21
Page

* ASIC Resistance:-

ASIC: Application Specific Integrated Circuit

ASICs are integrated Circuits that are created to serve a specific use case, performing a particular Computing task. In the world of cryptocurrencies, ASIC devices are designed to participate in the process of mining Bitcoin (or other Cryptocurrencies).
Bitcoin is an example of a cryptocurrency that cannot be considered ASIC-resistant

ASIC-Resistant is the property of a CryptoCurrency that is immune to ASIC mining.

An ASIC-resistant cryptocurrency has its protocol and mining algorithm configured in such a way that using ASIC machines to mine the coin is either impossible or bring no significant benefit when compared to traditional GPU mining.

In Some Cases, Using ASICs on ASIC-Resistant cryptocurrencies may be even worse than using the more Conventional hardware.

Mining Involves multiple attempts of finding a solution for a sort of mathematical problem, So the job of an ASIC is to perform as many attempts as possible. This means that using ASICs to mine Bitcoin or other PoW cryptocurrencies is much better than using a general purpose piece of hardware such as GPU card.

However, the process of making a cryptocurrency ASIC-Resistant is a defensive game which requires continued development & modifications.

It is worth noting that blockchain network that rely on other methods of achieving Consensus (such as PoS, dPoS) are ASIC-Resistant by Design.
In regards of PoW cryptocurrencies, Some of them are resistant & some not. (depends on mining Algorithm Implemented).

classmate
Date
Page

* Turing Complete :-

Turing Complete refers to a machine that, given enough time and memory along with the neccessary instructions, can solve any Computational problem, no matter how complex it be.

The turing machine is a theoretical, abstract idea given by Alan Turing which can simulate any algorithm that can be logically constructed. This mean that a Turing machine can solve any problem if it can be coded out.

Most of the programming languages are turing Complete.

A device or programming language is considered to be turing Complete when it can replicate a TM by solving a problem that turing machine can solve.
If it is not able to do so then it is said to be turing Incomplete.

Ex:- Simple Calculator → Turing Incomplete
Scientific Calculator → Turing Complete.

Blockchain & turing Completeness.

Some applications of BCT is turing Complete/ others are turing Incomplete, this varies according to the Scripting technology Implemented.

Scripting language used in bitcoin is intentionally designed as turing Incomplete because it serves its purpose and increased Complexity would potentially introduce problems. By keeping it simple the developers can predict with high accuracy how it is going to react in finite number of situations in which it is used.

Ethereum on other hand, is built as a turing complete blockchain. This is important because it needs to understand the agreements which make up Smart Contracts.

Ethereum has the ability to understand and implement any future agreements even those that have not been thought of yet.

Ethereum's turing Completeness means that it is able to use its code base to perform virtually any task.

Turing machine as an I/o device

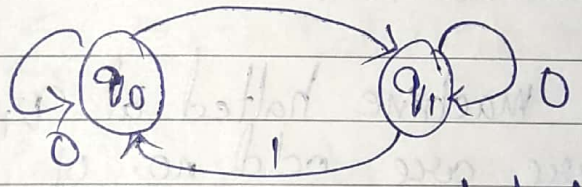
P.S:- Check if number of 1's in string is even or odd.

Let us take an infinitely long tape with random string on it.

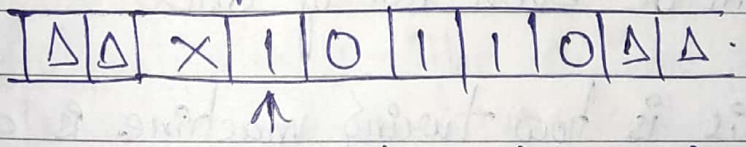
Let the string be $w = 010110$

Print '0' if its 'odd else 'e'.

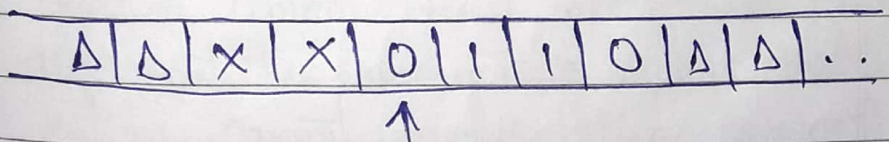
Rules governing machine behaviour.



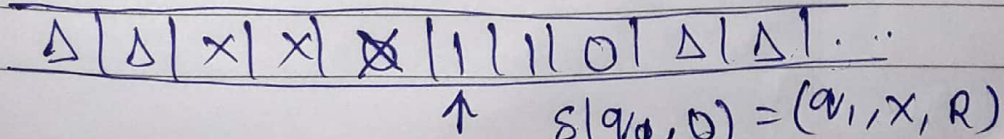
The turing machine will look like:



$\delta(q_0, 0) = (q_0, X, R)$



$\delta(q_0, 1) = (q_1, X, R)$



$\delta(q_1, 0) = (q_1, X, R)$

Δ	Δ	x	x	x	x	1	0	Δ

$\delta(q_1, 1) = (q_0, x, R)$

Δ	Δ	x	x	x	x	x	0	Δ

$\delta(q_0, \Phi) = (q_1, x, R)$

Δ	Δ	x	x	x	x	x	x	Δ	...

$\delta(q_1, 0) = (q_1, x, R)$

$\delta(q_1, \Delta) = \text{halt}$

The machine halted at q_1 state so, there are odd no. of ones.
If machine halted at q_0, so there will be even no. of ones.

This is how turing machine is an I/O device.

Cryptography:-

* Hash function:-

Cryptography is the most important Component of Blockchain. It is a Science of keeping things Confidential Using Encryption techniques.
There are various Usage of Cryptography as mentioned below.

→ Confidentiality:- Only the intended or authorized receipients can understand the msg. It can also be referred as privacy.

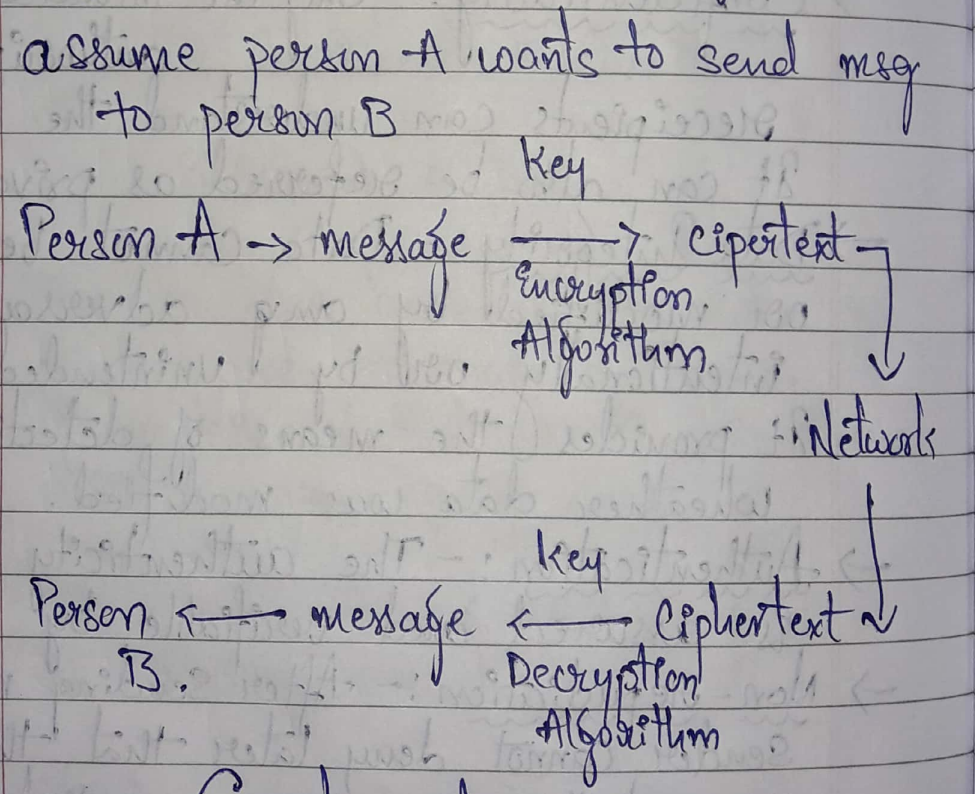
→ Data Integrity:- Data cannot be forged or modified by any adversary intentionally or by unintended errors. It provides the means of detecting wheather data was modified.

→ Authentication:- The authenticity of Sender is assured and verifiable by receiver.

→ Non-repudiation:- After Sending msg, Sender cannot deny later that they sent the sent the msg. This means that an entity cannot refuse the ownership of previous Committed action.

Any Information in the form of a text, numeric data, or a Computer program can be called plaintext.
The idea is to Encrypt the plaintext Using an Encryption algorithm and a key that produces ciphertext.
The ciper ciphertext can then be transmitted to the intended recipient, who decrypts it using decryption algorithm and the key to get plaintext.

assume person A wants to send msg to person B



Cryptography.

Broadly there are two kinds of Cryptography
1) Symmetric key Cryptography
2) Asymmetric key Cryptography

If the same key is used for both Encryption and decryption, it is called Symmetric key Cryptography.

Assymetric Cryptography known as public key cryptography is a revolutionary concept introduced By Diffie and Hellman. With this technique they solved the problems of key distribution in Symmetric Cryptography System by introducing digital signatures.

Cryptographic Hash function are mathematical function that are most important cryptographic primitives and are an integral part of blockchain data structure. They are widely used in cryptographic protocols, information security applications such as Digital Signatures and Message Authentication Codes (MACs).

classmate
Date 24/6/21
Page

* Digital Signature :- ECDSA

ECDSA :- Elliptic Curve Digital Signature Algorithm.

Digital Signature :-

Digital Signature is the second cryptographic primitive along with hash function that we need as a building block for cryptocurrency.

Digital Signature is supposed to be just like a signature on paper only in Digital form. What we want from digital signature is two things.

1. It is just like an ideal paper signature, only you can make your signature but anyone who see your signature can verify that it's valid.

2. The signature should be tied to a particular document. So that somebody can't take your signature and snip it off from one document and glue to it to the bottom of another document.

It signifies your agreement or endoorsement of a particular document.

Digital Signature Scheme

Digital Signature Scheme Involves three Algorithms:

1. (sk, pk) := generatekeys(keysize)

The generatekeys methods takes keysize as a parameter and generates a key pair. The Secret key sk is kept private and is used to sign msgs. pk is public key used for verification that is given to other to verify your signature.

2. sig := Sign(sk, message)

The Sign method takes a message and a Secret key. sk as Input and outputs a signature for msg under sk.

3. isValid := verify(pk, message sig)

The verify message method takes a message, a signature and public key pk as input. It returns a boolean value. That will be true if sig is valid signature for message under pk and false otherwise.

Requirements for signature

1. Valid signature must verify.

verify(pk, message, sign(sk, message)) == true

If i sign a message with sk, with my secret key, that if someone later tries to validate that using my public key pk and same msg, then that will validate correctly.

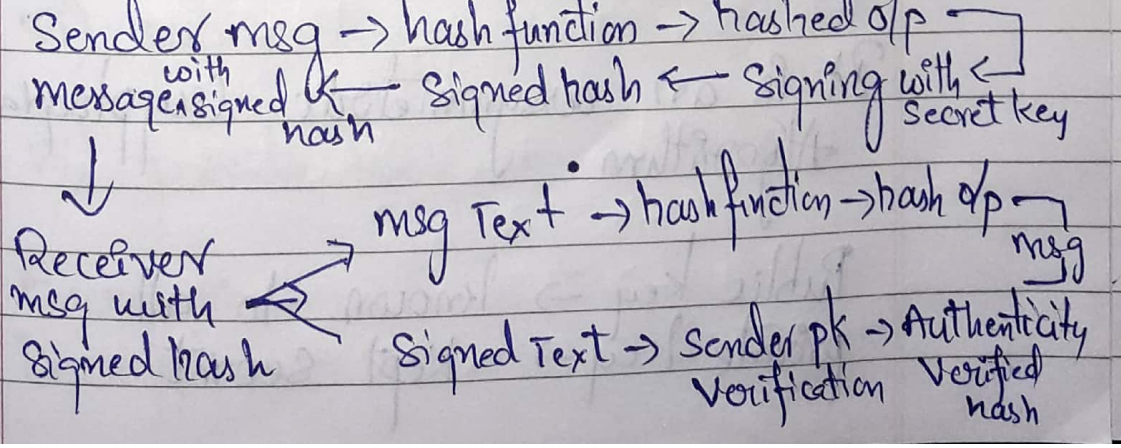
2. Signature are Ententially Unforgeable.

That is, an adversary who knows your public key, who knows your verification key and gets to see signature on some other msg, can't forge your signature. Because hash functions are collision free, it's safe to use the hash of the message as the input to the digital signature scheme rather than the message.
And if we sign a hash pointer then signature covers or protects the whole structure.

ECDSA:-

Bitcoin uses a particular digital signature scheme that is ECDSA Elliptic Curve Digital Signature Algorithm. It's a U.S government standard.

Digital Signature Algorithm:-



Modern Cryptography is founded on the idea that the key that you use to Encrypt your data can be made public while the key that is used to decrypt your data can be kept private.
This System are known as public key Cryptography
Eg:- RSA Algorithm

Public key cryptography system algorithm are easy to process in one direction but difficult to undo This type of Algorithms are known as Trap door functions.
Encryption with the public key can only be undone by decrypting with private key.

* R. S. A Algorithm

RSA named after three man who publicly described the Algorithm in 1978 Ron Rivest, Adi Shamir, Leonard Adleman

It is an assymmetric Cryptographic Algorithm.

Public key → known to all
private key → Kept secret

If public key of User A is used for Encryption then we have to use private key of same User for decryption

The RSA scheme is a block cipher in which plaintext & cipher are integer between 0 to n-1 for some n.

1. Key Generation.

i) Select two large prime numbers p & q

ii) find $n = p * q$

iii) Calculate $\phi(n) = (p-1) * (q-1)$

iv) Choose some value for e such that $1 < e < \phi(n)$ &
$\gcd(\phi(n), e) = 1$ (co-prime)

v) Calculate
$$d = e^{-1} \bmod \phi(n)$$
i.e $ed = 1 \bmod \phi(n)$
$ed \bmod \phi(n) = 1$

vi) public key = $\{e, n\}$

vii) private key = $\{d, n\}$

Encryption:- $C = M^e \bmod n$

Decryption:- $M = C^d \bmod n$

Example:-

1. lets take two prime no. 13 & 7

2. $n = 13 \times 7 = 91$

3. $\phi(n) = (13-1) \times (7-1)$
   $12 \times 6 = 72$

4. let $e = 5$ ($\because 1 < e < \phi(n)$,
   $\gcd(e, \phi(n)) = 1$)

5. $d = e^{-1} \bmod \phi(n)$
   $ed \bmod \phi(n) = 1$
   $(5 * d) \bmod 72 = 1$
   $d = 29$ ($\because$ Extended Euclidean Algorithm)

6. public key = {5, 91}

7. private key = {29, 91}

let the message M = "HI"

lets take the ASCII value
H → 72
I → 73
72 & 73 < n

we Encrypt & decrypt Individually.

Encrypt. $C = M^e \bmod n$

for H

$C = (72)^5 \bmod 91$

$C = 11$

for I

$C = (73)^5 \bmod 91$

$C = 47$

Decrypt. $M = C^d \bmod n$

for H $M = (11)^{29} \bmod n$

$M = 72$ (ASCII of H)

for I $M = (47)^{29} \bmod n$

$M = 73$ (ASCII of I)

∴ for the Message is "HI".

ECDSA:-

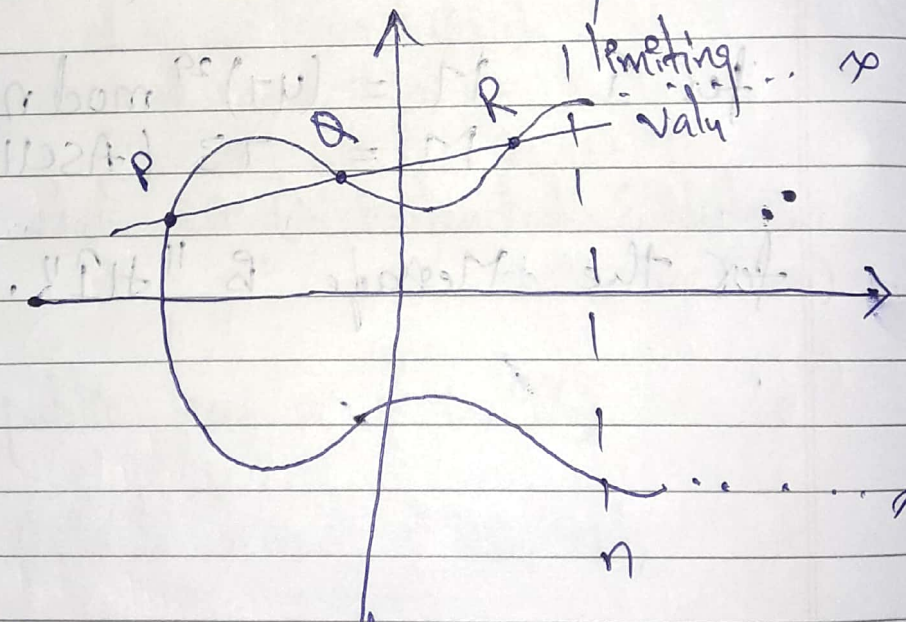
* Elliptic curve cryptography is a asymmetric cryptography.

It provides equal security with smaller keys size (unlike RSA)
High security even with small size

Elliptic curves are used for this.
It is defined by some mathematical cubic function.

$$y^2 = x^3 + ax + b$$

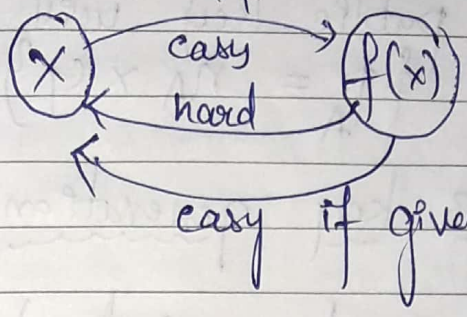
// degree = 3



It is symmetric to x-axis

If we draw a line it touches maximum of three points.

Trap door function:- The function which is easy to compute in one direction yet difficult in opposite direction.



Let $E_p(a,b)$ be elliptical curve
Consider the Eqn $Q = kp$
where P & Q are points on curve
& $k < n$

If k & p is given, its easy to find Q
but if P & Q is given its hard to find k
This is called discrete logarithmic problem for elliptic curves

ECC - key exchange :-

Global public elements

$E_q(a,b)$: Elliptic curve with parameter a,b & q
q → any prime number or in no. in form of 2^m

G :- point on curve $> n$.

User A key generation :-

Let n_A be private key of user A then public key will be

$$\{P_A = n_A \times G\}$$

User B key generation :-

Let n_B be private key of user B then public key will be

$$\{P_B = n_B \times G\}$$

Calculation of secret key by User A :-

$$\{k_A = K = n_A * P_B\}$$

Calculation of secret key by User B :-

$$\{k_B = K = n_B * P_A\}$$

(∵ key exchange).

Encryption :-

Let the message be M

First encode the message M into point on elliptic curve let the point be P_m

P_m will be Encrypted

For encryption choose a random positive integer K

The Cipher point will be

$$C_m = \{ kG, P_m + kP_B \}$$

→ Public key of B is used for encryption

Now C_m is sent to receiver.

Decryption :-

For decryption, multiply 1st point in the C_m with secret key.

$$kG * n_B$$

Then sub it from second point

So,

$$P_m + kP_B - kG * n_B$$

we know that $\boxed{P_B = n_B \times G}$

we get

$$P_m + kP_B - kP_B$$
$$= P_m \text{ (original point)}$$

So, receiver gets the same point P_m.

Elliptic Curve Cryptography is now used in a wide variety of applications. Such as

1) U.S government uses it to protect internal Communications
2) The tor project uses it to help assure anonymity
3) It is the mechanisms used to prove ownership of bitcoins
4) It provides signature in Apple's imessenger service
5) Used to Encrypt DNS Information
6) The chrome or fibrefox browser is using elliptic curve Cryptography.

* Memory Hard Algorithm:-

In Cryptography, a memory hard function (MHF) is a function that costs significant amount of memory to evaluate. MHFs find their use as a form of PoW.

There are different ways to measure the memory hardness of a function such as.

1. A commonly seen measure is CMC Cumulative Memory Complexity. In parallel model CMC measures memory hardness by summing up all the i/ps in each step.
2. Integrating memory against physical time
3. Memory bandwidth consumption on a memory bus. This category of functions are also dubbed "Bandwidth Hard functions"

Bitcoin uses repeated evaluation of SHA function as PoW, but it turned out that modern general purpose processors i.e off-the-shelf CPUs are inefficient to compute fixed function over n over.

Miners adopted ASICs and achieved 10^6 speedup. While this is fine for what bitcoin is used for, we want a more egalitarian hardness measure

In other words, we want everyone to be equally inefficient in computing the function even if they have ASICs. Because if some people can evaluate the function efficiently and some can't, then in order to make the function relatively hard for short-cut takers, then it make the function too hard for a regular user.

If everyone is inefficient, then everyone can evaluate a moderately hard function. Over time, it has been recognized that memory cost remains fairly equal across the board.

Based on evaluation pattern, MHFs can be divided into two categories

1. data dependent (dMHF)
2. data Independent (iMHF)

dMHFs are that which sometimes you don't know which pieces of information you would still need for later calculations.

pMHFs are ones that there's no such ambiguity.

dMHFs Examples → Scrypt, argon2d.

pMHFs Examples → argon2i, Catena

* Zero knowledge proof:-

It is an Encryption Scheme proposed by MIT researchers Silvio Micali, Shafi Goldwasser and Charles Rackoff in 1980s.

In this method, One party (Prover) can prove that Specific Statement is true to the other party (Verifier) without disclosing any additional information.

In Cryptography a zero-knowledge proof or zero-knowledge protocol is a method by which one party (the prover) can prove that another party (the verifier) that

they knew a value x without conveying any information apart from the fact that they know the information.

The essence of zero-knowledge proof is that it is trivial to prove that one possess knowledge of certain information by simply revealing it, the challenge is to prove such possession without revealing the information itself or any additional information.

A zero knowledge proof of knowledge is a special case. When the statement consists only the fact that the prover possess the secret information. Interactive zero-knowledge proofs require interaction between the individual (or computers) proving their knowledge and individual validating the proof.

A protocol implementing zero-knowledge proof of knowledge must neccessarily require interactive input from the verifier. The interactive input is usually in the form of one or more challenges such that the response from the prover will convince the

verifier if and only if the statement is true i.e if the prover does possess the claimed knowledge. The verifier could also record the execution and replays it to convince someone else that they possess the secret information.

The new party's acceptance is either justified since the replayer does possess the information (which implies that the protocol leaked information, and thus, is not proved in zero knowledge) or the acceptance is spurious i.e was accepted from someone who does not actually possess the information

The Zero-knowledge proof must satisfy three properties.

1. Completeness :- If the statement is true the honest verifier (one who follows protocol properly) will be convinced of this fact by honest prover.

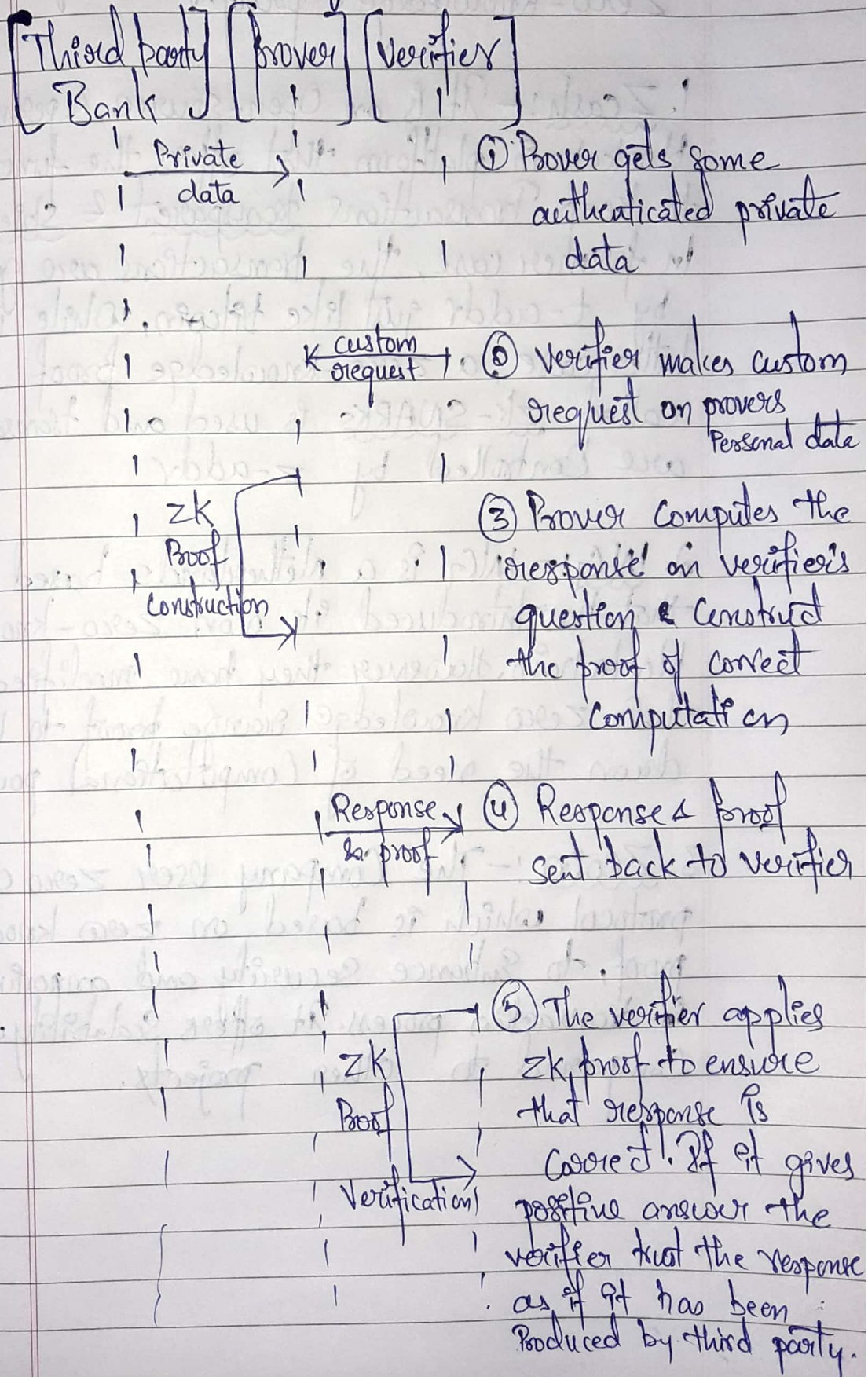
2. Soundness :- If the statement is false no cheating prover can convince the honest verifier that it is true, Except with small probability.

Zero-knowledge :- If the Statement is true, no verifier learns anything other than the fact that the Statement is true. In other words just knowing the statement (not the secret) is sufficient to imagine a scenario showing that prover knows the secret. This is formalised by showing that every verifier has some simulator that, given only the statement to be proved (no access to prover), can produce a transcript that looks like an interaction.

The first two properties are of more general Interactive proof system. The third is what makes the proof zero-knowledge.

Zero knowledge proofs are not proofs in mathematical sense of the term because there is small probability. Zero knowledge proofs are probabilistic proofs rather than deterministic.

Zero-knowledge protocol : Data Exchange



① Prover gets some authenticated private data

② Verifier makes custom request on provers Personal data

③ Prover computes the 'response' on verifier's question & construct the proof of correct computation

④ Response & proof sent back to verifier

⑤ The verifier applies zk proof to ensure that response is correct. If it gives positive answer the verifier trust the response as if it has been produced by third party.

Real-life Examples of convergence of zero-knowledge proof.

1. Zcash :- It is an Open-source & permissionless blockchain platform that offers the functionality to keep Transactions transparent & shielded. In former case, the transactions are governed by t-addr just like bitcoin, while in latter case; a zero-knowledge proof called zk-SNARKs is used and transactions are controlled by z-addr

2. ING :- ING is a Netherlands based bank that has introduced its own zero-knowledge blockchain. However they have modified their zero knowledge range proof to lower down the need of Computational power.

3. ZCoin :- The Company uses zero coin protocol which is based on zero knowledge proof, to Enhance security and anonimity in transaction proces. It offers Scalability when Compared to other projects.