Blockchain Technology and Systems

(SEZG569/SSZG569)

Dr. Ashutosh Bhatia
Department of Computer Science and Information Systems

BITS Pilani
Pilani Campus

innovate achieve lead

BITS Pilani
Pilani Campus

QUIZ, HYPE & FACTS

innovate achieve lead

BITS Pilani
Pilani Campus

QUIZ

Q0

What is BITCOIN

a) A Cryptocurrency
b) A decentralized peer-to-peer network
c) A public transaction ledger
d) All

Q1

Who created Bitcoin?

Satoshi Nakamoto

Satoshi Nakamoto is the name used by the presumed pseudonymous person or persons who developed bitcoin, authored the bitcoin white paper, and created and deployed bitcoin's original reference implementation. As part of the implementation, Nakamoto also devised the first blockchain database. Nakamoto was active in the development of bitcoin up until December 2010. Many people have claimed, or have been claimed, to be Nakamoto.

source: https://en.wikipedia.org/wiki/Satoshi_Nakamoto

Q2

Where do you store your cryptocurrency?

Crypto Wallets

In the cryptocurrency ecosystem, the term “wallet” refers to software, online or offline, that allows a cryptocurrency owner to access their cryptocurrency holdings.

Q3

What is a miner?

Computers that validate and process blockchain transactions

Q4

Where can you buy cryptocurrency?

- A private transaction
- An exchange
- A Bitcoin ATM

Q5

What is a blockchain?

a) A distributed ledger on a peer to peer network
b) A type of cryptocurrency
c) An exchange
d) A centralized ledger

Q6

What is a DApp?

A decentralized application that is developed over blockchain platform

Q7

What is the term for when a blockchain splits?

Fork

A bitcoin hard fork refers to a radical change to the protocol of bitcoin's blockchain that effectively results in two branches, one that follows the previous protocol and one that follows the new version. It is through this forking process that various digital currencies with names similar to bitcoin have been created, including bitcoin cash and bitcoin gold. Bitcoin cash remains the most successful hard fork of the primary cryptocurrency; as of June 2021, it is the eleventh-largest digital currency by market cap.

Q8

What incentivizes the miners to give correct validation of transactions?

A block Reward in form of Bitcoins

Q9

What is a hash function?

Takes an input of any length and returns a fixed-length string of numbers and letters

Q10

What does IPFS stand for?

Interplanetary File System

The **InterPlanetary File System** (**IPFS**) is a protocol and peer-to-peer network for storing and sharing data in a distributed file system. IPFS uses content-addressing to uniquely identify each file in a global namespace connecting all computing devices.[4]

InterPlanetary File System - Wikipedia

Q11

What is the maximum number of bitcoins that can be created?

21 million

Q12

What was the highest the Bitcoin price ever reached?

₹ 54,404,923

Q13

What is Altcoin?

Altcoins are cryptocurrencies other than Bitcoin.

Altcoin - Bitcoin Wiki

- Binance Coin (BNB)
- Cardano (ADA)
- Chainlink (LINK)
- Ether (ETH)
- Litecoin (LTC)

As of today, **over 5000** of these "alternative" currencies have been created worldwide.

Q14

What is meme coin?

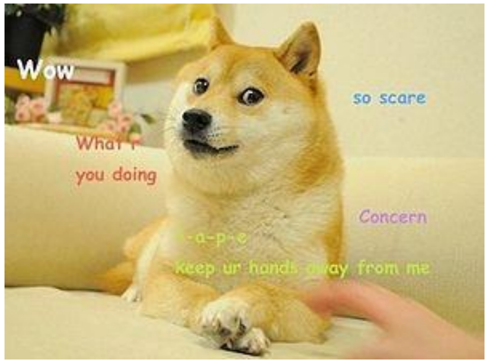
A **meme coin** (also spelled **memecoin**) is a cryptocurrency that originated from an Internet meme or has some other humorous characteristic.[1] It may be used in the broadest sense as a critique of the cryptocurrency.

In late 2013, Dogecoin was released after being created as a joke on the Doge meme by software engineers. This sparkled the creation of several subsequent meme coins. In October 2021, there were about 124 meme coins circulating in the market. Notable examples include Dogecoin and Shiba Inu,[2]

Meme coin - Wikipedia

Q15

What is This?



Dogecoin (/ˈdoʊ(d)ʒkɔɪn/ *DOHJ-koyn* or *DOHZH-koyn*,[2] code: **DOGE**, symbol: **Ð**) is a cryptocurrency created by software engineers Billy Markus and Jackson Palmer, who decided to create a payment system as a "joke", making fun of the wild speculation in cryptocurrencies at the time.[3] It is considered both the first "meme coin", and, more specifically, the first "dog coin". Despite its satirical nature, some consider it a legitimate investment prospect.

Doge (meme) - Wikipedia

Q16

What is Stablecoins?

Cryptocurrency but usually centralized and pegged with some fiat money or asset class

Stablecoins are cryptocurrencies where the price is designed to be pegged to a cryptocurrency, fiat money, or to exchange-traded commodities (such as precious metals or industrial metals).[1]

Stablecoin - Wikipedia

Q17

What is Token?

Token is a unit of value issued by a tech or crypto start-up, intended to be a piece in the ecosystem of their technology platform or project. Tokens are supported by blockchains. They only physically exist in the form of registry entries in said blockchain. Initially, most tokens were based on the ERC20 protocol by Ethereum.

Tokens are different from bitcoins and altcoins in that they are not mined by their owners nor primarily meant to be traded (although they may be traded on exchanges if the company that issued them becomes valuable enough in the eyes of the public), but to be sold for fiat or cryptocurrency in order to fund the start-up's tech project.

Token Definition – Cryptocurrency – BitcoinWiki

Q18

What is MetaVerse and Omniverse?

Facebook is changing its name to Metaverse to highlight the vision of a future that will be lived in the cyberspace (along with life in the physical space). Augmented Reality and Virtual Reality, along with sensors, displays, artificial intelligence and Digital Twins, are the enabling technologies.

Metaverse vs Omniverse – IEEE Future Directions

Q19

What is Sandbox and Dreamland

The Sandbox is a sandbox game for mobile phones and Microsoft Windows, developed by gamestudio Pixowl and released on May 15, 2012. It was released for PC on Steam on 29 June 2015. The brand was acquired by Animoca Brands in 2018, and its name used for a blockchain-based 3D open world game.

SAND is an ERC-20 Ethereum-powered utility token that will be the medium of exchange within **The Sandbox**. Facilitates the purchase or sale of LANDs or game ASSETs (LANDs are portions of the Metaverse open to player ownership, while ASSETs are tokens created by players).

Metaverse vs Omniverse – IEEE Future Directions

Q20

What is NFT?

A non-fungible token (NFT) is a and non-interchangeable unit of data stored on a blockchain, a form of digital ledger. NFTs can be associated with reproducible digital files such as photos, videos, and audio. NFTs use a digital ledger to provide a public certificate of authenticity or proof of ownership, but do not restrict the sharing or copying of the underlying digital files. The lack of interchangeability (fungibility) distinguishes NFTs from blockchain cryptocurrencies, such as Bitcoin.

Non-fungible token - Wikipedia

Q21

What is (DAO) ?

A company or group of like-minded entities that operate based on the rules set forth in a smart contract. DAOs are used to transform business logic into software logic recorded on a blockchain. A company whose funds are locked in a multisignature wallet that is controlled by a smart contract is an example of a DAO. In that same example, board of directors decisions might be voted on, recorded, and effected through a smart contract rather than by holding physical board meetings.

Q22

What is ETHEREUM ?

Ethereum is a decentralized Blockchain 2.0 chain. It was the first major smart contract platform and has widespread support from Fortune 500 companies through the Ethereum Enterprise Alliance (EEA).

Ethereum currently uses a Proof-of-Work (PoW) consensus algorithm, but future changes to the protocol will update it to a more scalable algorithm, most likely based on Proof-of-Stake (PoS).

Q23

What is HASHRATE ?

The rate at which a particular machine can perform a specific hashing function. Hashrate is similar to general CPU speed, but where processor speed is measured based on the number of arbitrary instructions a machine can carry out per second, hashrate is measured based on the number of times a machine can perform that specific function per second, allowing application-specific integrated circuits (ASIC) to have a much higher hashrate than a processor with the same clock speed.

Q24

What is MAINNET ?

The largest blockchain network a specific protocol runs, or the most valuable chain as decided by the community. Mainnets are typically where real value is derived and represent the truest intent of the core developers.

Q25

What is ORACLE ?

Services that connect real-world data with blockchain applications. Oracles are necessary to provide input that cannot be independently verified, such as temperature measurements. Oracles typically rely on the security of a trusted source rather than the security of trustlessness.

Q26

What is SOLIDITY ?

A smart contract programming language built for the Ethereum Virtual Machine. Syntactically it resembles C++ and Javascript and compiles to eWASM.

Q27

What is SOLIDITY ?

A smart contract programming language built for the Ethereum Virtual Machine. Syntactically it resembles C++ and Javascript and compiles to eWASM.

Q28

What is TOKENIZATION ?

The concept of translating business strategies, goods, or services into discrete, tradeable units that are recorded on a blockchain or other system.

Physical goods can be tokenized by associating their unique identifiers with on-chain references.

Q29

innovate achieve lead

What is TOKENIZATION ?

The concept of translating business strategies, goods, or services into discrete, tradeable units that are recorded on a blockchain or other system.

Physical goods can be tokenized by associating their unique identifiers with on-chain references.

Q30

What Is a "51% Attack"?

A 51% attack refers to a malicious actor (or group acting in concert), controlling over 50% of the total mining power of the blockchain network and disrupting the integrity of the blockchain.

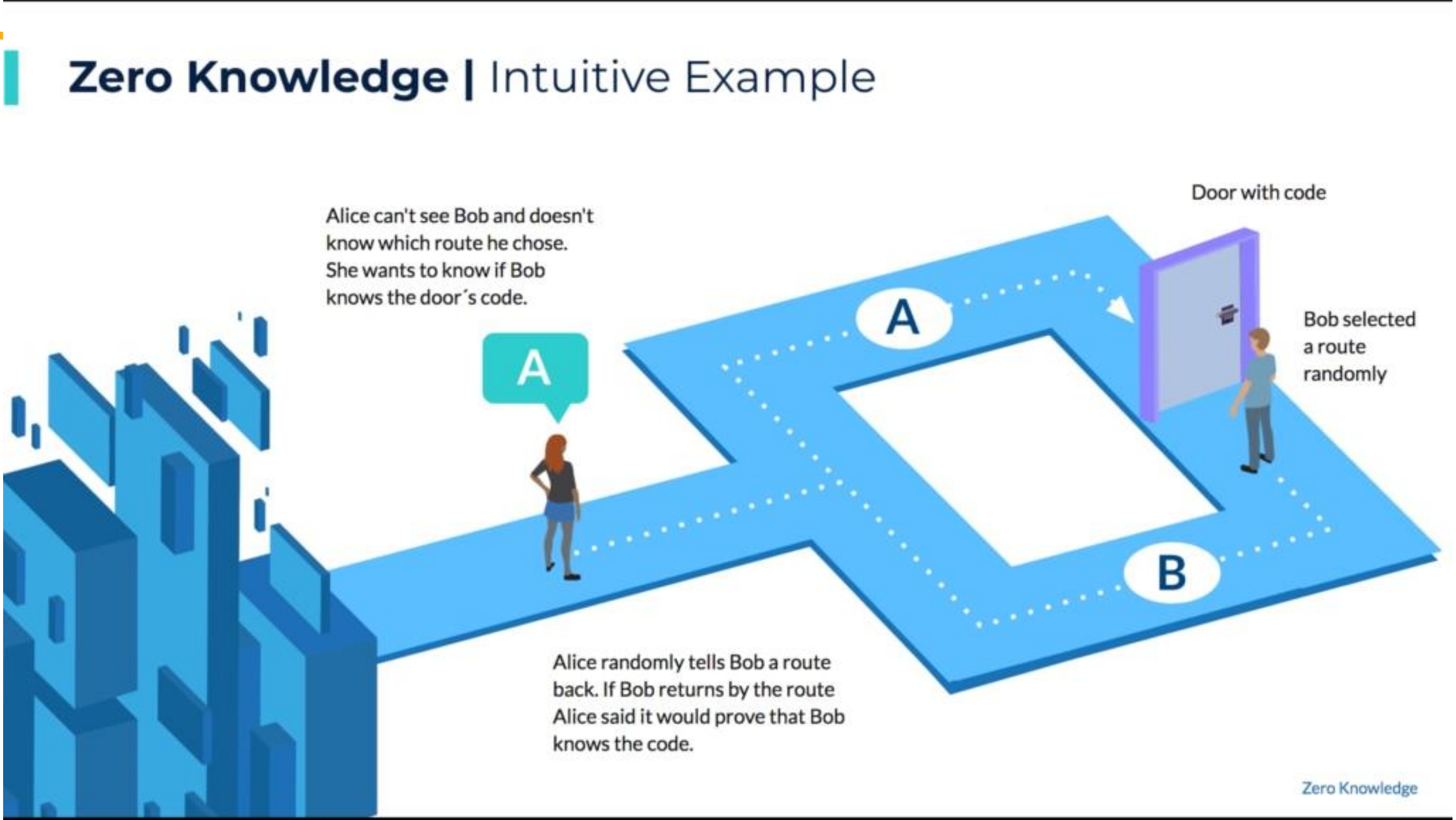
An example of a 51% attack happened in January 2019 on the Ethereum Classic blockchain.

Q31

ZERO-KNOWLEDGE (ZK) PROOF

A mathematical representation of an assertion whose output value can be determined without the input information.

Zero-knowledge proofs are used to prove that an actor is in possession of certain information without actually revealing that information. They are especially useful in cryptocurrencies because they can be used to show that a transaction is valid without revealing the sender, recipient, or amount of the transaction. ZK research is still in its infancy.



Zero Knowledge | Intuitive Example

Alice can't see Bob and doesn't know which route he chose. She wants to know if Bob knows the door´s code.

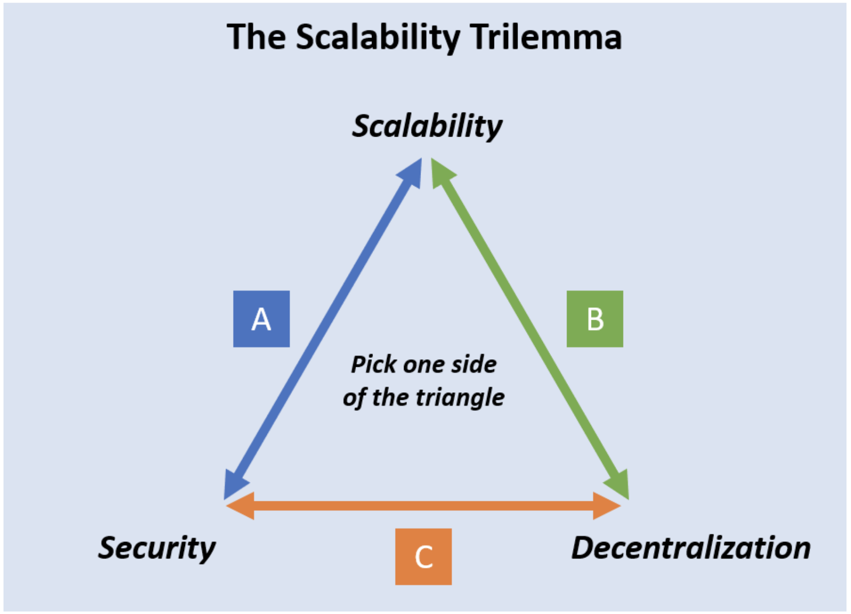
Bob selected a route randomly

Alice randomly tells Bob a route back. If Bob returns by the route Alice said it would prove that Bob knows the code.

Q32

What Is the Blockchain Trilemma?

The blockchain trilemma is a concept coined by Vitalik Buterin that proposes a set of three main issues — decentralization, security and scalability — that developers encounter when building blockchains, forcing them to ultimately sacrifice one "aspect" for as a trade-off to accommodate the other two.



Beeple's, *Everydays " The First 5000 Days"* – $69 Million

Beeple's *Everydays, the First 5000 days* is basically one of the most iconic NFT sales that ever happened in the history of the NFT world. It not only broke the world record but also made. ***Beeple, one of the richest artists in the world.***

The artwork was auctioned at NFT platform, Christies on March 11, 2021. Bascially, the Everydays 5000 consists of Beeple's entire collection of 5000 artworks that he created since May 1, 2007. The bid on this artwork started with $100 but it soon rose to millions ultimately settling for $69 Million dollars.

[36 MOST EXPENSIVE NFTs EVER SOLD (Ranked) - NFT's Street (nftsstreet.com)]

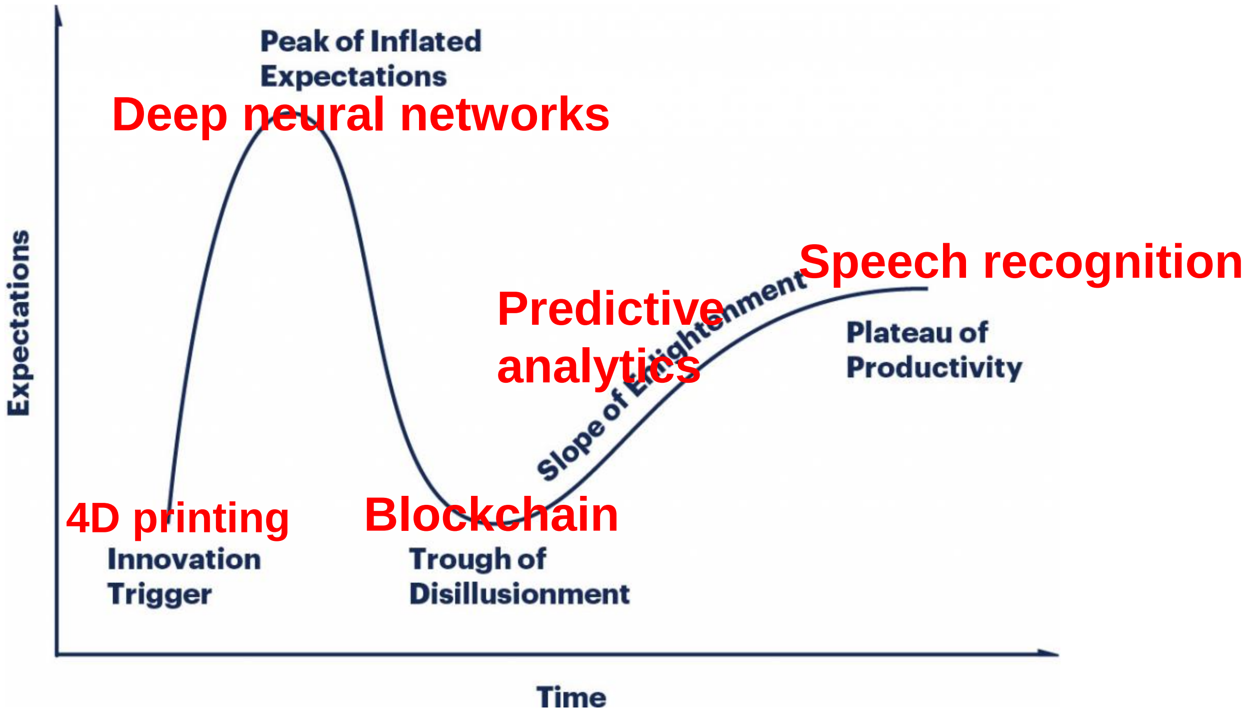
innovate achieve lead

BITS Pilani
Pilani Campus

HYPE

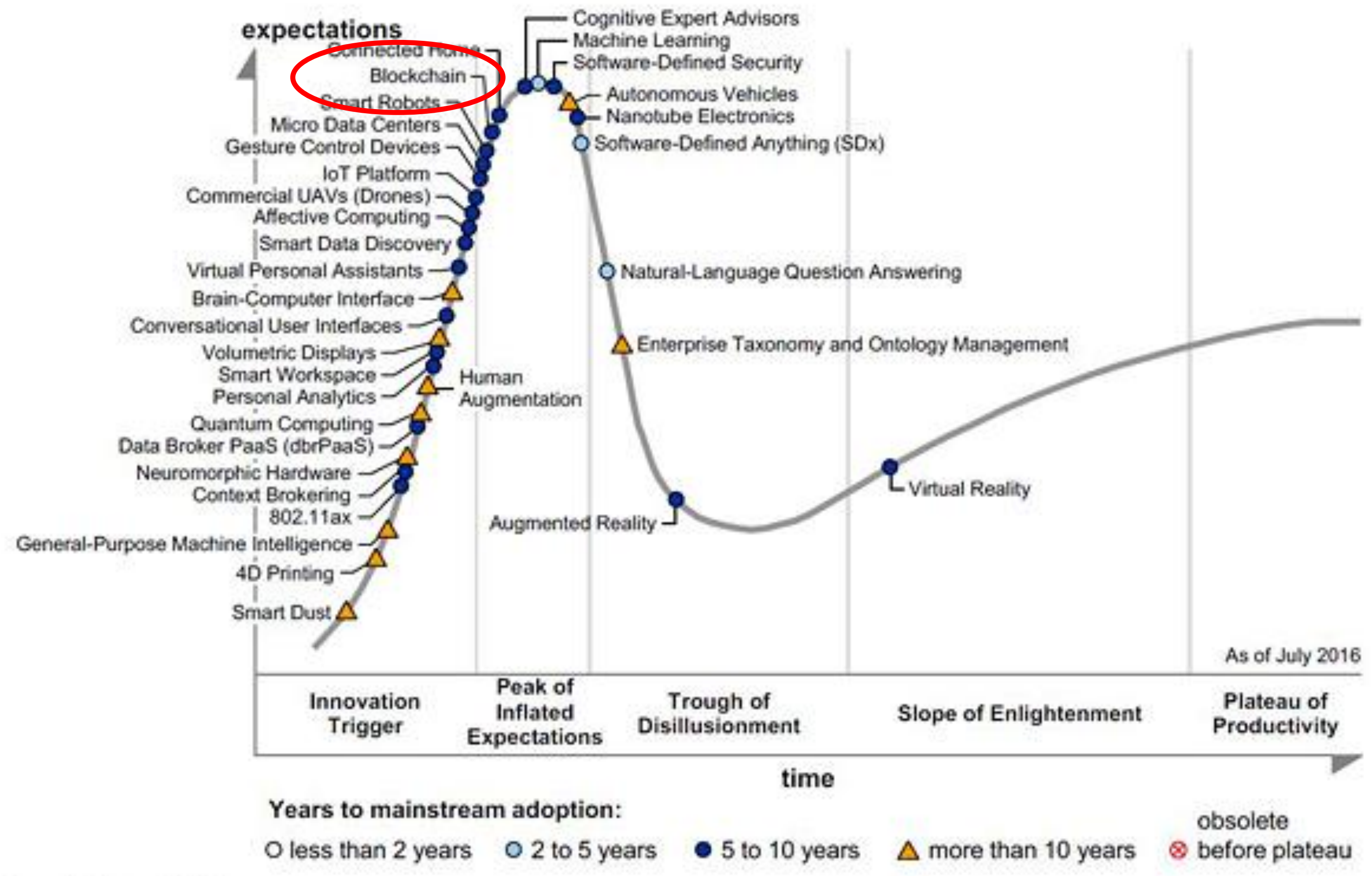
Gartner Hype Cycle

- The Gartner Hype Cycle is a graphical representation of the perceived value of a technology trend or innovation—and its relative market promotion.
- The cycle can help you understand how the perceived value of a given technology evolves over the course of its maturity lifecycle.



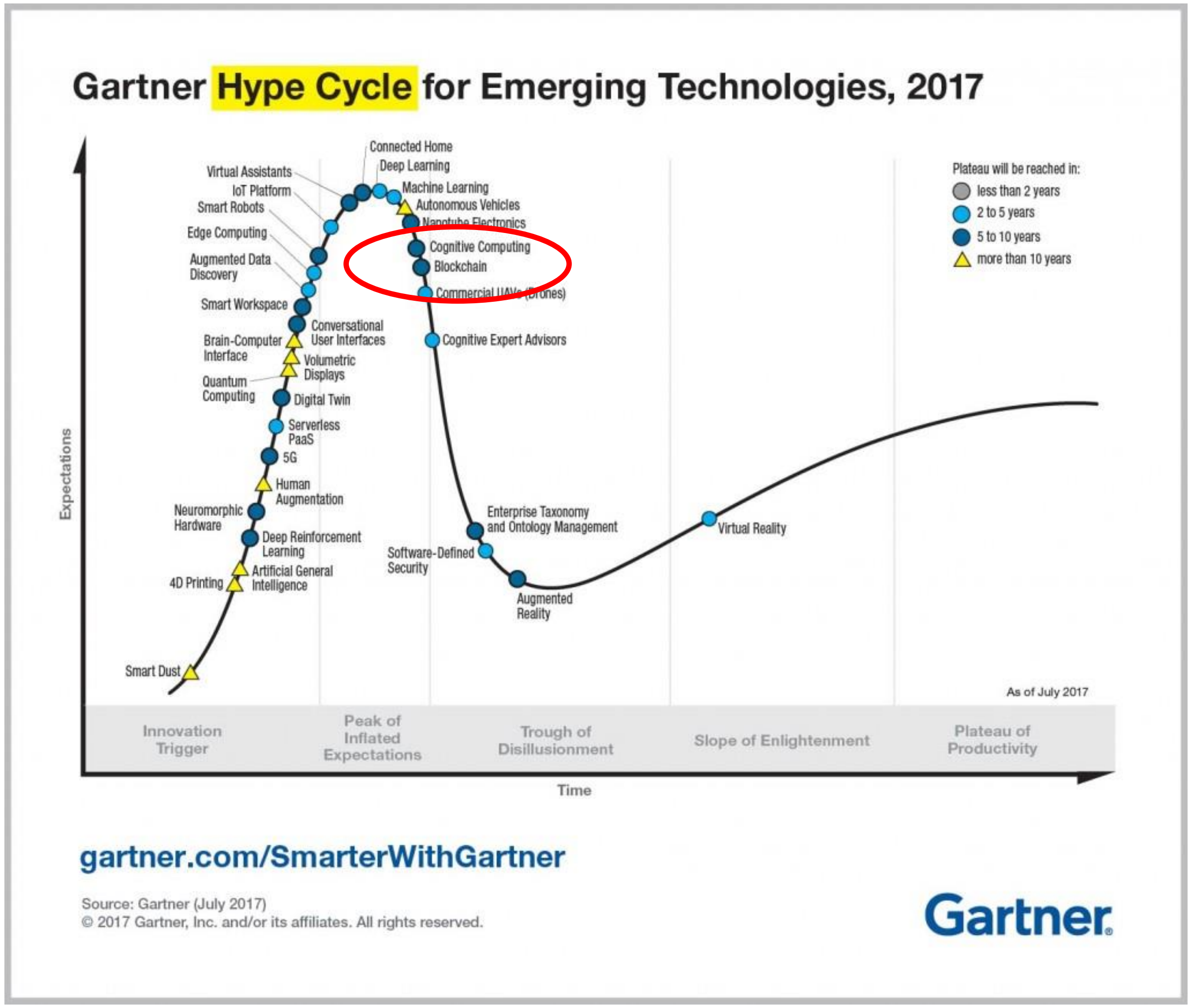
[Introduction to the Gartner Hype Cycle – BMC Software | Blogs]

Hype Cycle for Emerging Technologies, 2016

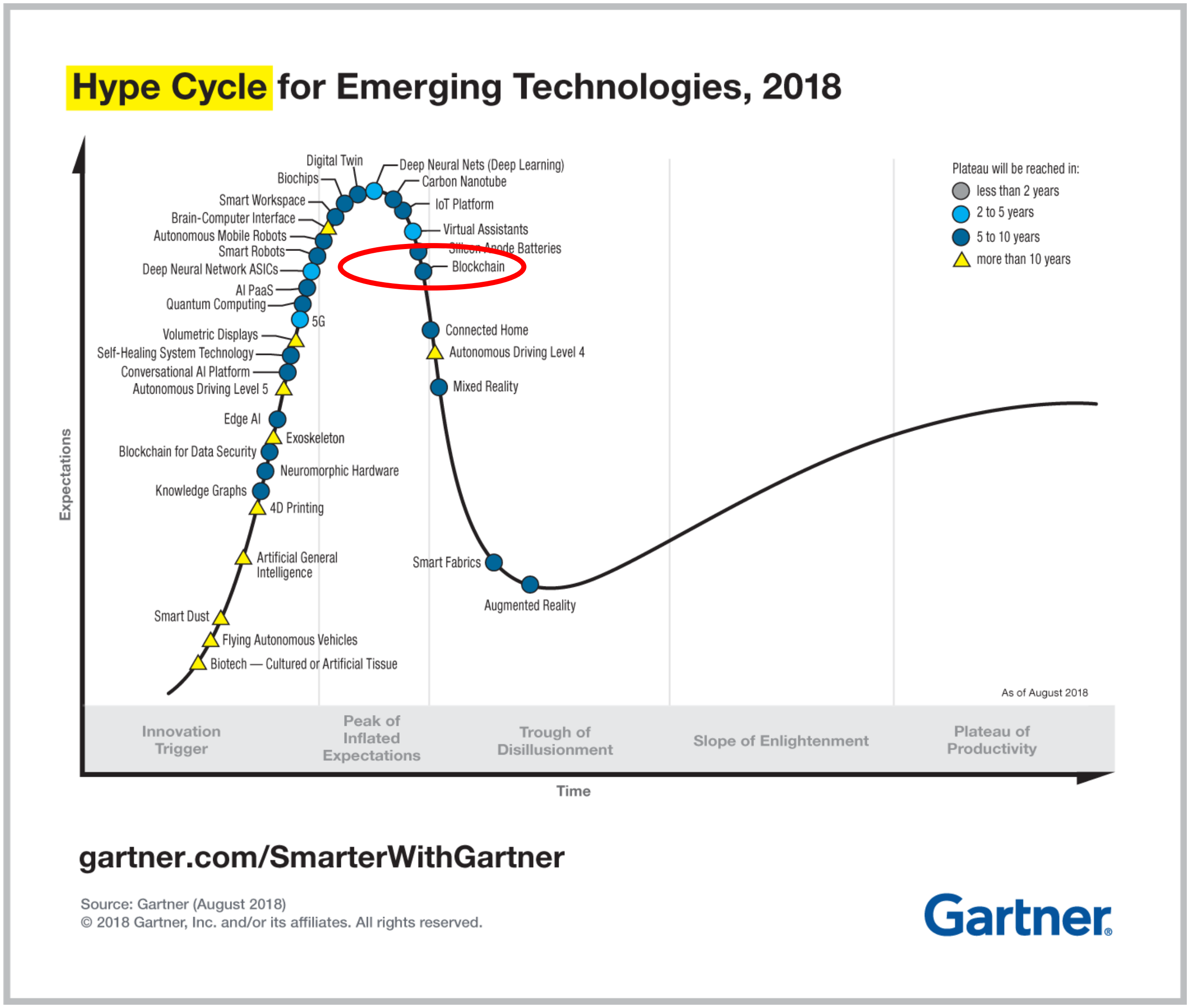


Source: Gartner (July 2016)

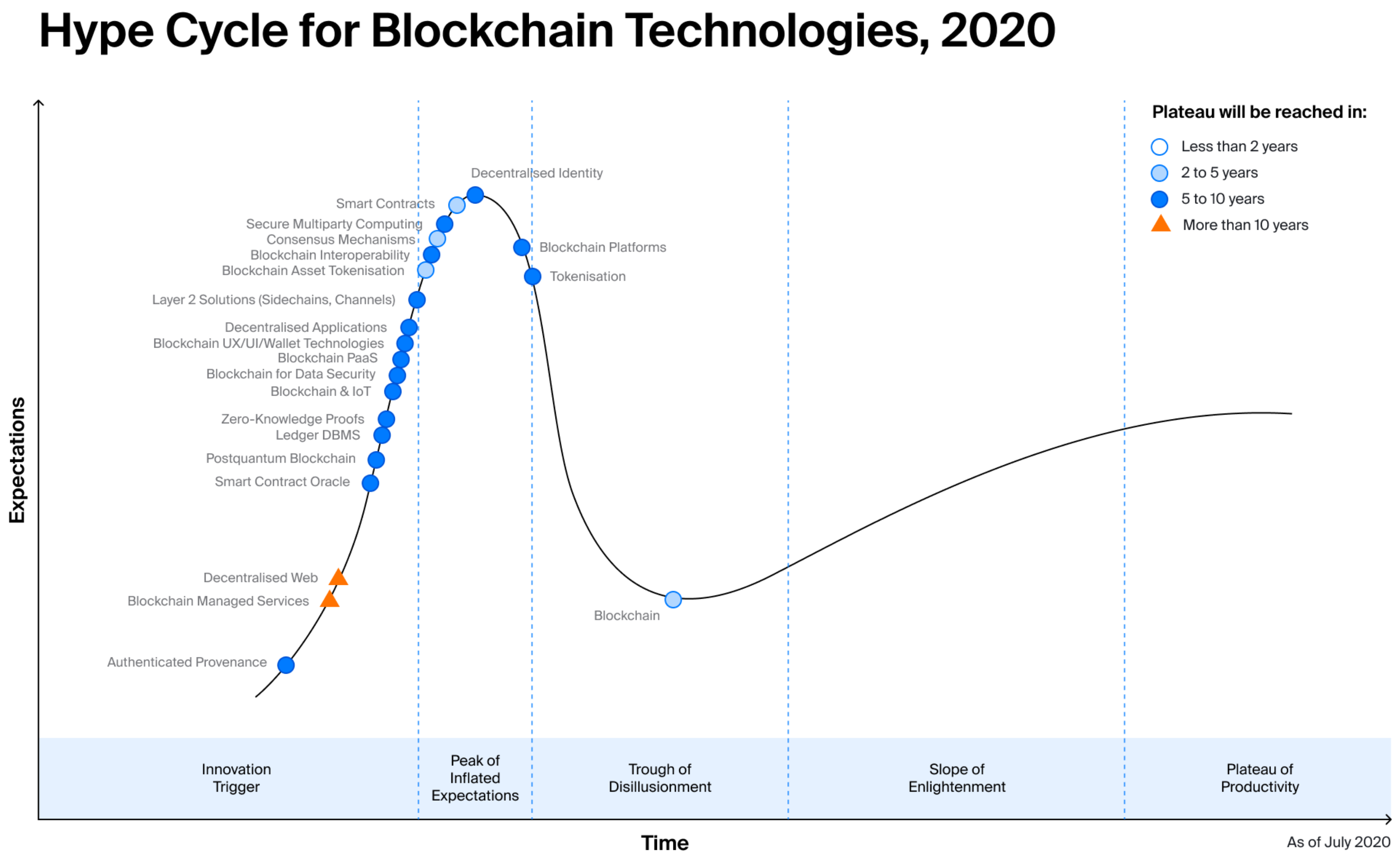
Hype Cycle for Emerging Technologies, 2017



Hype Cycle for Emerging Technologies, 2018



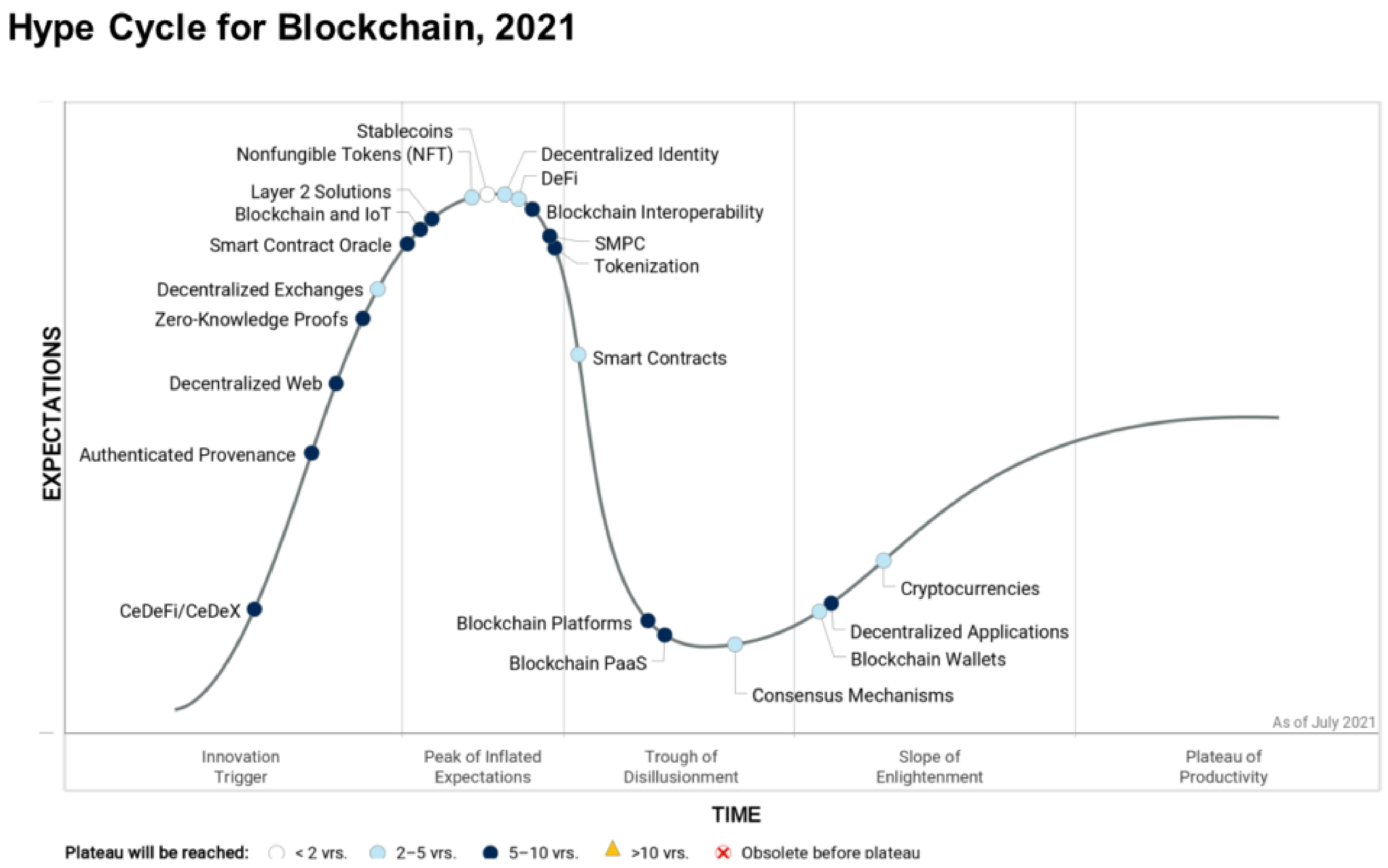
Hype Cycle for Blochchain Technologies 2020



Source: Gartner (2020)

Chainstack C Act, 1956

Hype Cycle for Blockchain 2021: More Action than Hype



Source: Gartner (July 2021)

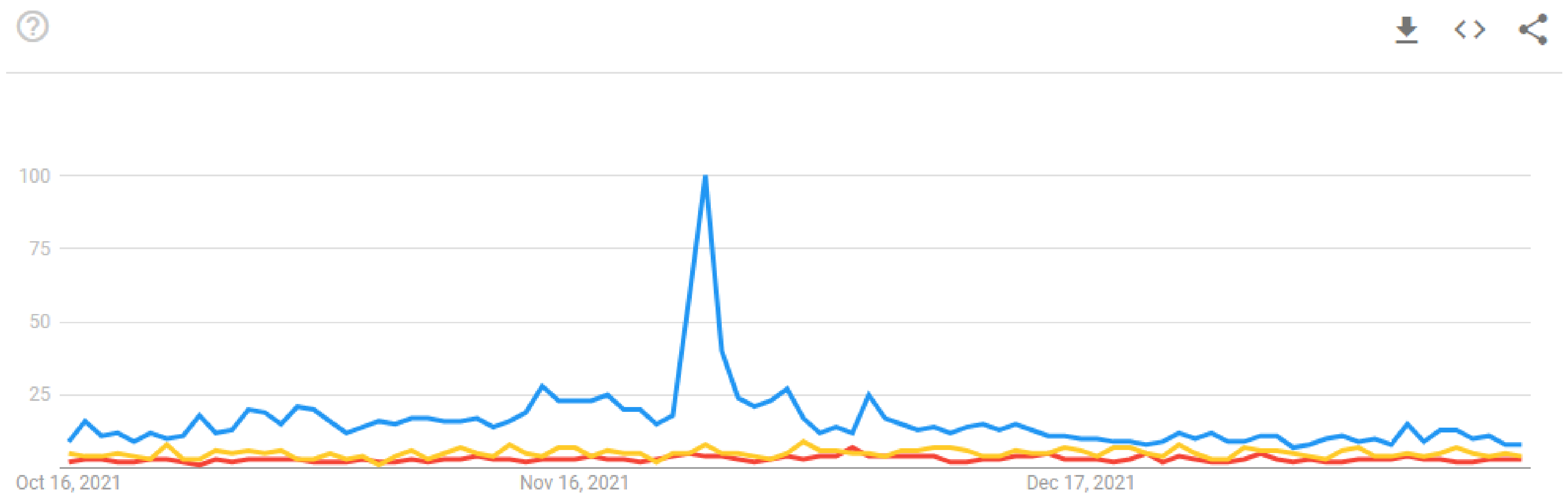
747513

innovate achieve lead

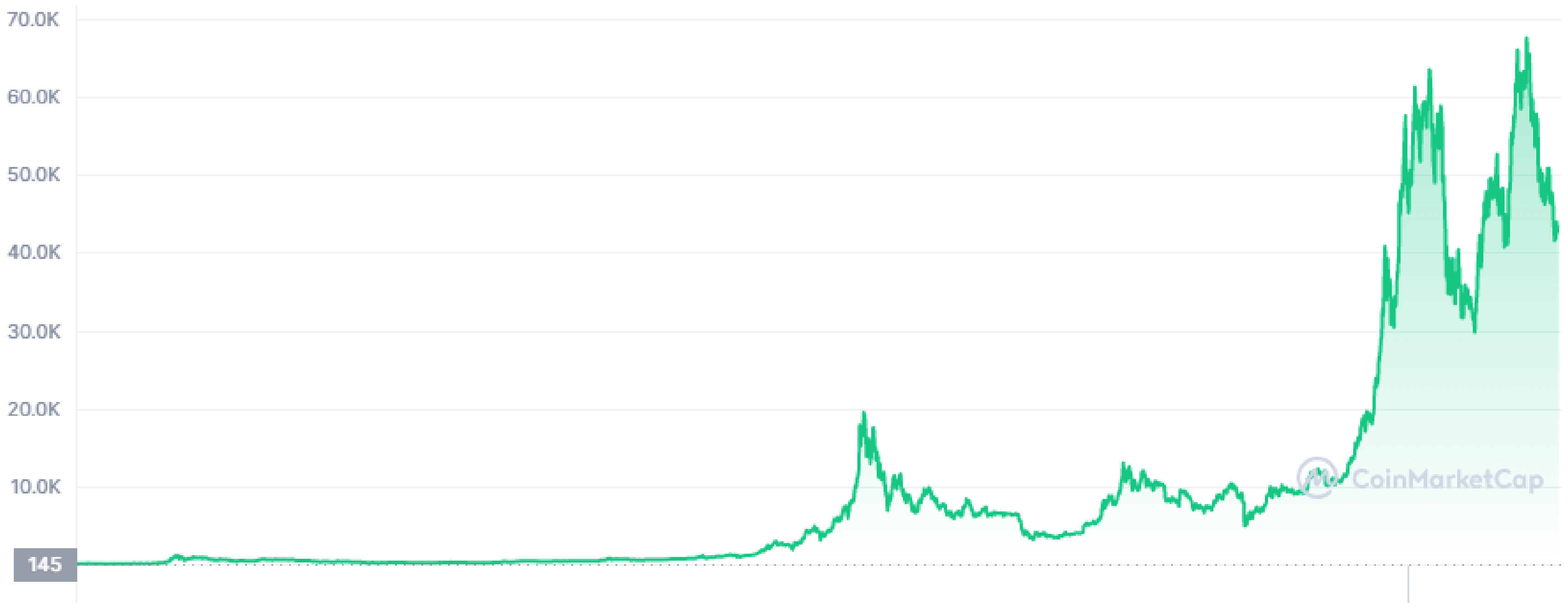
BITS Pilani
Pilani Campus

Trends and Facts

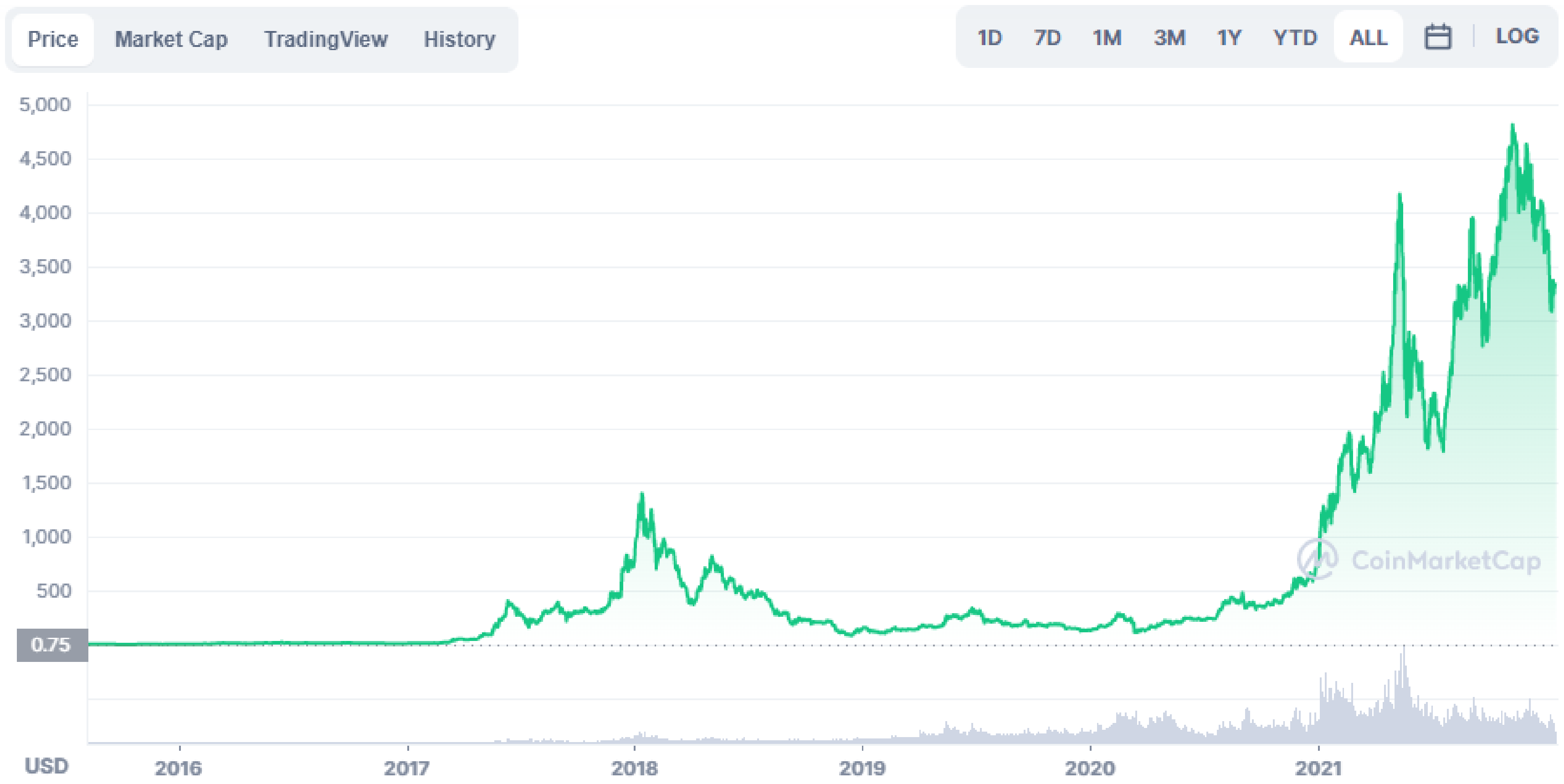
Google Trends: Cryptocurrency, Blockchain ans Machine Learning



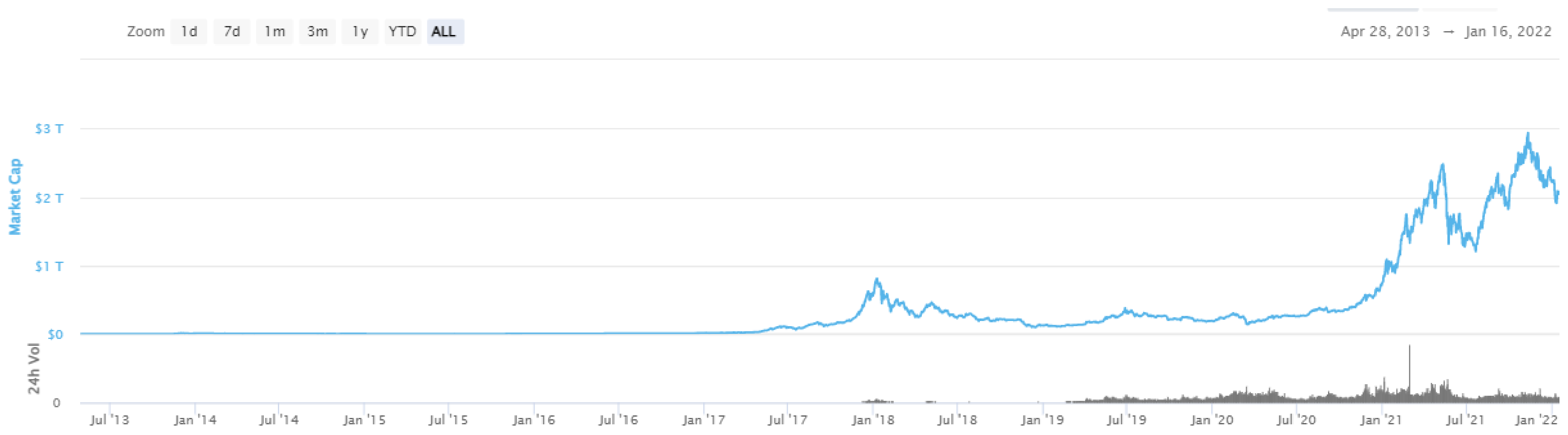
BITCOIN Growth



Etherium Growth



Cryptocurrency Market Cap



Key adoption drivers

- Mainstream adoption of Bitcoin, including El Salvador’s adoption of Bitcoin as legal tender in June 2021.
- Payment network, banking and social network adoption of distributed ledger technologies (DLTs) for money movement, with the expected deployment of central bank digital currencies (CBDCs) being a key influencer.
- Decentralized finance (DeFi) applications offer substantially greater financial rewards than traditional finance. Centralized firms like hedge funds already take advantage of this.
- Tokenization of assets, including explosive growth of NFTs and DeFi tokens, and the promise of tokens linked to physical assets in the future.
- Blockchains such as Binance, Cardano, and Solana offering viable cost-effective alternatives to Ethereum chain transactions.
- Monumental progress in blockchain interoperability, including gateways and abstraction middleware, already used today by DeFi applications.
- Blockchain migration from the proof-of-work (POW) consensus method (still used for Bitcoin) to more energy-efficient consensus methods such as proof of stake (PoS). The ongoing upgrade of Ethereum leads this trend.

Still, the picture is not all rosy. There are plenty of challenges

- Adoption of permissioned blockchains is moving much more slowly. Some use cases — especially around supply chain and authenticated provenance — are benefiting from ledger technology. However, most users are stuck trying to align use cases to the technology.
- Global regulations and accounting standards need clarification before most enterprises adopt cryptocurrency
- China continues to clamp down on crypto activities as they work on making their own CBDC the world’s dominant currency.

Blockchain Technology
(BITS F452)

BITS Pilani
Pilani Campus

Dr. Ashutosh Bhatia, Dr. Kamlesh Tiwari
Department of Computer Science and Information Systems

BITS Pilani
Pilani Campus

Introduction to Crypto and Cryptocurrency

LECTURE OUTLINE

- Crypto Background
  - Hash Functions
  - Digital Signatures and its Applications
- Introduction to cryptocurrency
  - Basic digital cash

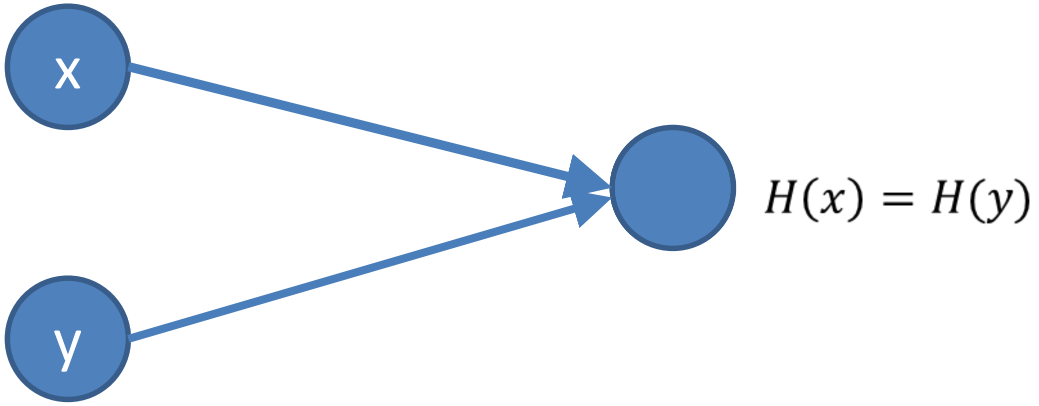
Hash Functions

- Takes arbitrarily length of string as input
- Produces a fixed sized output
- Efficiently Computable

- Security Properties
  - Collision Free
  - Hiding
  - Puzzle friendly

Hash Properties 1: Collision Resistant

- A collision occurs when two distinct inputs produce the same output
- **Collision-resistance**: A hash function H is said to be collision resistant if it is infeasible to find two values, x and y, such that $x \neq y$ and $H(x) = H(y)$



- However, collision do exist

How to find a collision?

- Try 2^{130} randomly chosen inputs and assuming that hash output is 256 bits, 99.8% chance that two of them will collide

- This works, no matter what the hash function is. (**Birthday Paradox**)

- However, 2^{130} is a so large number and any computer ever made by the humanity was trying to find a collision since the beginning of the universe till now, the probability of it finding a collision is infinitesimally small.

How to find a collision?

- Try 2^{130} randomly chosen inputs and assuming that hash output is 256 bits, 99.8% chance that two of them will collide

- This works, no matter what the hash function is. (**Birthday Paradox**)

- However, 2^{130} is a so large number and any computer ever made by the humanity was trying to find a collision since the beginning of the universe till now, the probability of it finding a collision is infinitesimally small.

Birthday Paradox

Find the probability that at-least two people in a room have the same birthday

Event A: at least two people in the room have the same birthday

Event A' : No people in the room have the same birthday

$$\Pr[A] = 1 - \Pr[A']$$

$$\Pr[A'] = 1 \times \left(1 - \frac{1}{365}\right) \times \left(1 - \frac{2}{365}\right) \times \left(1 - \frac{3}{365}\right) \cdots\cdots \left(1 - \frac{Q-1}{365}\right)$$

$$= \prod_{i=1}^{Q-1} \left(1 - \frac{i}{365}\right)$$

$$\Pr[A] = 1 - \prod_{i=1}^{Q-1} \left(1 - \frac{i}{365}\right)$$

$$\sqrt{Q \approx 2M \ln \frac{1}{1-\epsilon}}$$

M = 365, ϵ is the desired

if $\epsilon = .5\ then\ Q \approx 1.17\sqrt{M}$

Thus to achieve 128 bit security against collision attacks, hashes of length at-least 256 is required

Is there a better way?

- For some possible Hash functions, YES
  - Example H(x) = x mod 2^{256}
- For others we don't know one
- No Hash Function is <u>proven</u> to be collision resistant

Application: Hash as a message digest

- If we know that H(x) = H(y)

  it is safe to assume that x = y

- To recognize a file that we saw before

  just remember its hash

- Useful as the hash is small

Hash Property 2: Hiding

- We want something like this

  given H(x) it is infeasible to find x.

- The problem is that this property can not be true in the stated form if the number of possible input values is small

- **Hiding:** A hash function H is hiding if: when a secret value r is chosen from a probability distribution that has **high min-entropy**, then given H(r | x) it is infeasible to find x.

- High min-entropy means that the distribution is very spread out and no particular value is chosen with negligible entropy.

Application: Commitment

We want to ”seal a value” in the envelop and ”open the envelop” later

Commit to a value and reveal it later

Commitment API

(com, key) := commit(msg)

  match := verify(com, key, msg)

To seal msg in envelop

  (com, key) := commit(msg), then publish com

To open envelop

  publish key, msg

Anyone can use verify() to check the message

Commitment API

(com, key) := commit(msg)

match := verify(com, key, msg)

Security Properties

Hiding: Given com, infeasible to find msg

Binding: Infesible to find msg != msg' s.t.

verify(commit(msg), msg') = true

Commitment API

commit(msg) := (H(key | msg), key))

`where key is a random 256 bit value

verify(com, key, msg) = (H(key | msg) == com)

Security Properties

Hiding: Given H(key | msg), infeasible to find msg

Binding: Infeasible to find msg != msg’ s.t.

H(key | msg) == H(key | msg’)

Hash Property 3: Puzzle friendly

For every possible out put value y,

if k is chosen randomly from a distribution with high min entropy,

then it is infeasible to find x such that H(k | x) = y

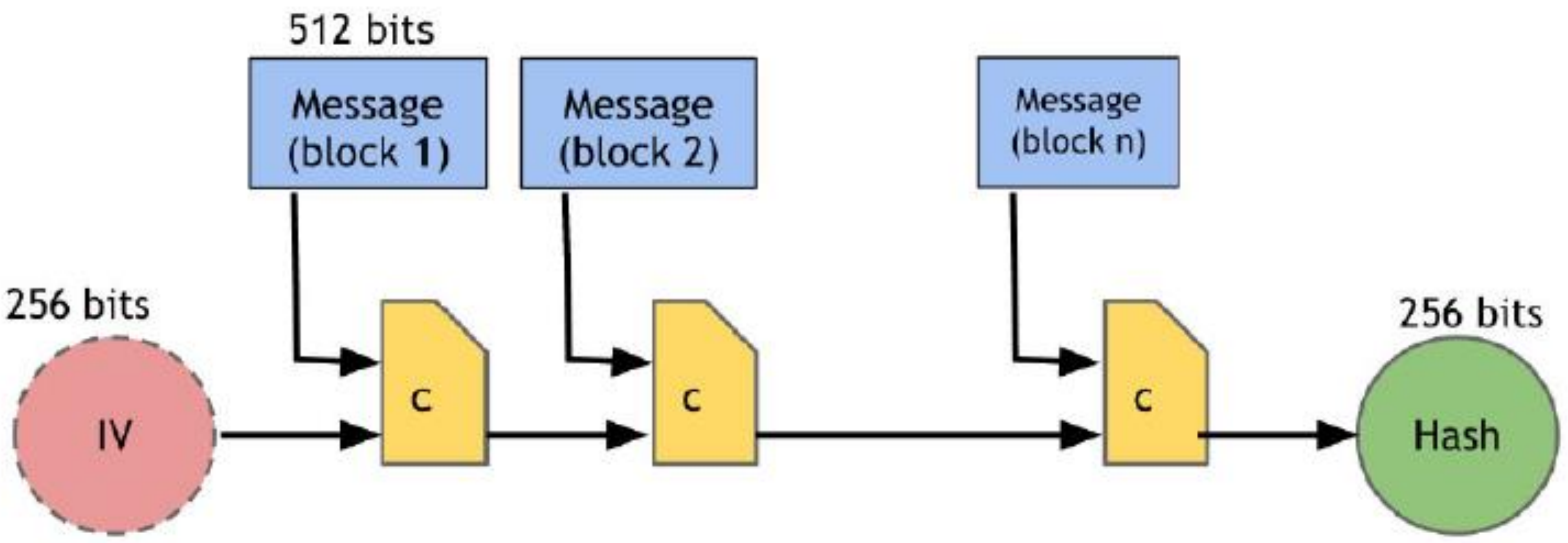
Application: Search Puzzle

Given a puzzle ID, id (from high min-entropy dist.) and a target set Y

Try to find a solution x, such that H(id | x) ∈ Y)

Puzzle friendly property implies that no solving strategy is much better than trying random values of x.

SHA 256 hash function



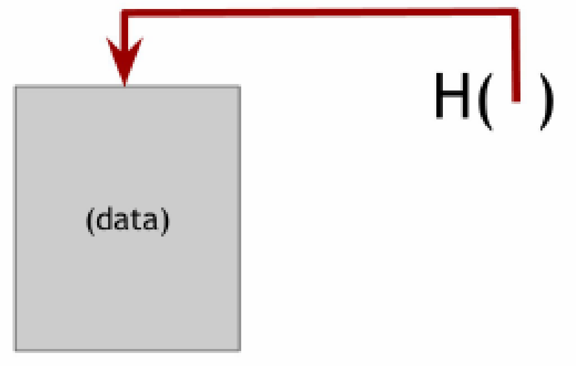
Theorem: If c (the compression function) is collision-free than SHA-256 is collision free

Blockchain Demo (andersbrownworth.com)

Hash Pointer

- Hash Pointer is :
  - pointer to where some information is stored
  - cryptograhic hash of the information

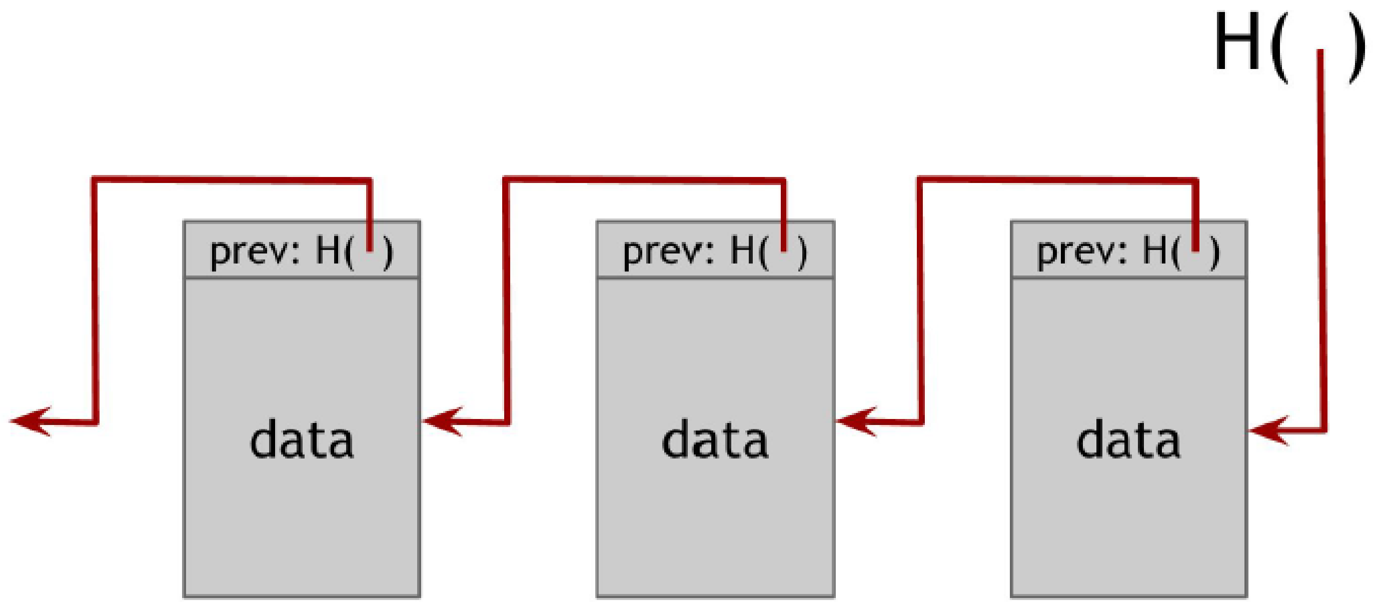
- If we have a hash pointer, we can
  - ask to get the info back
  - verify that it has not changed



Key Idea

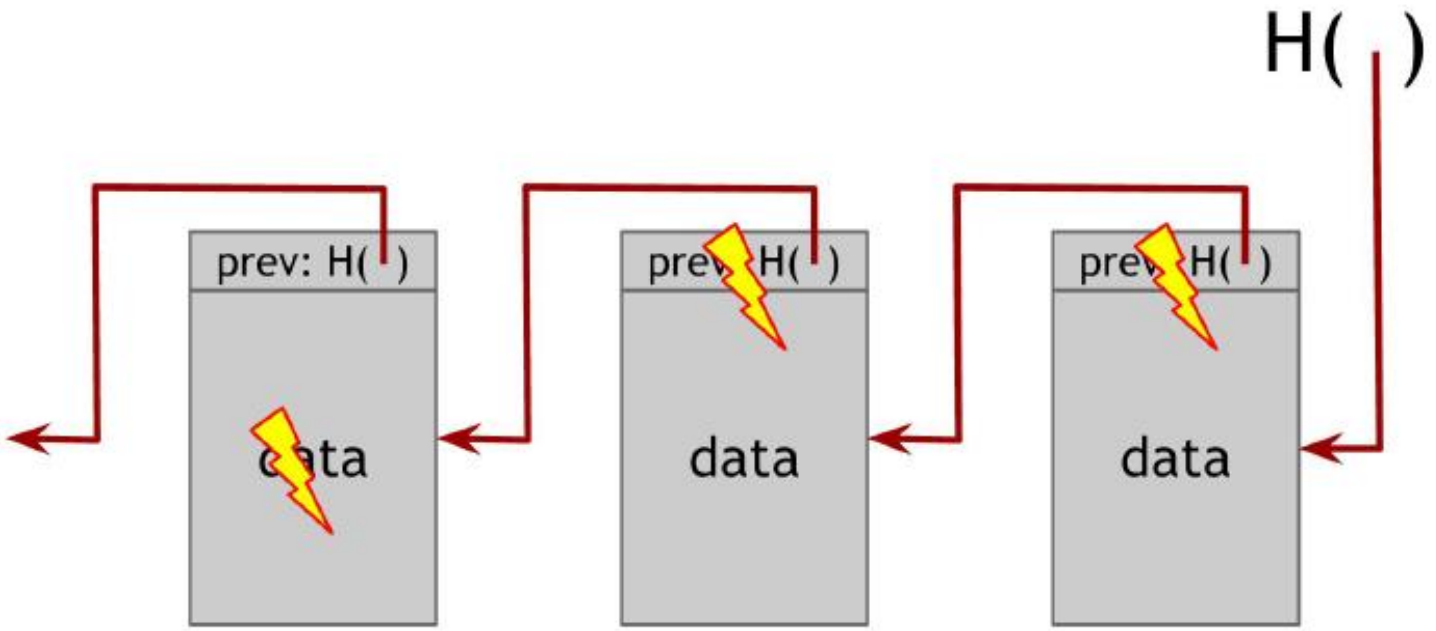
Build Data Structures with Hash Pointers

Linked List

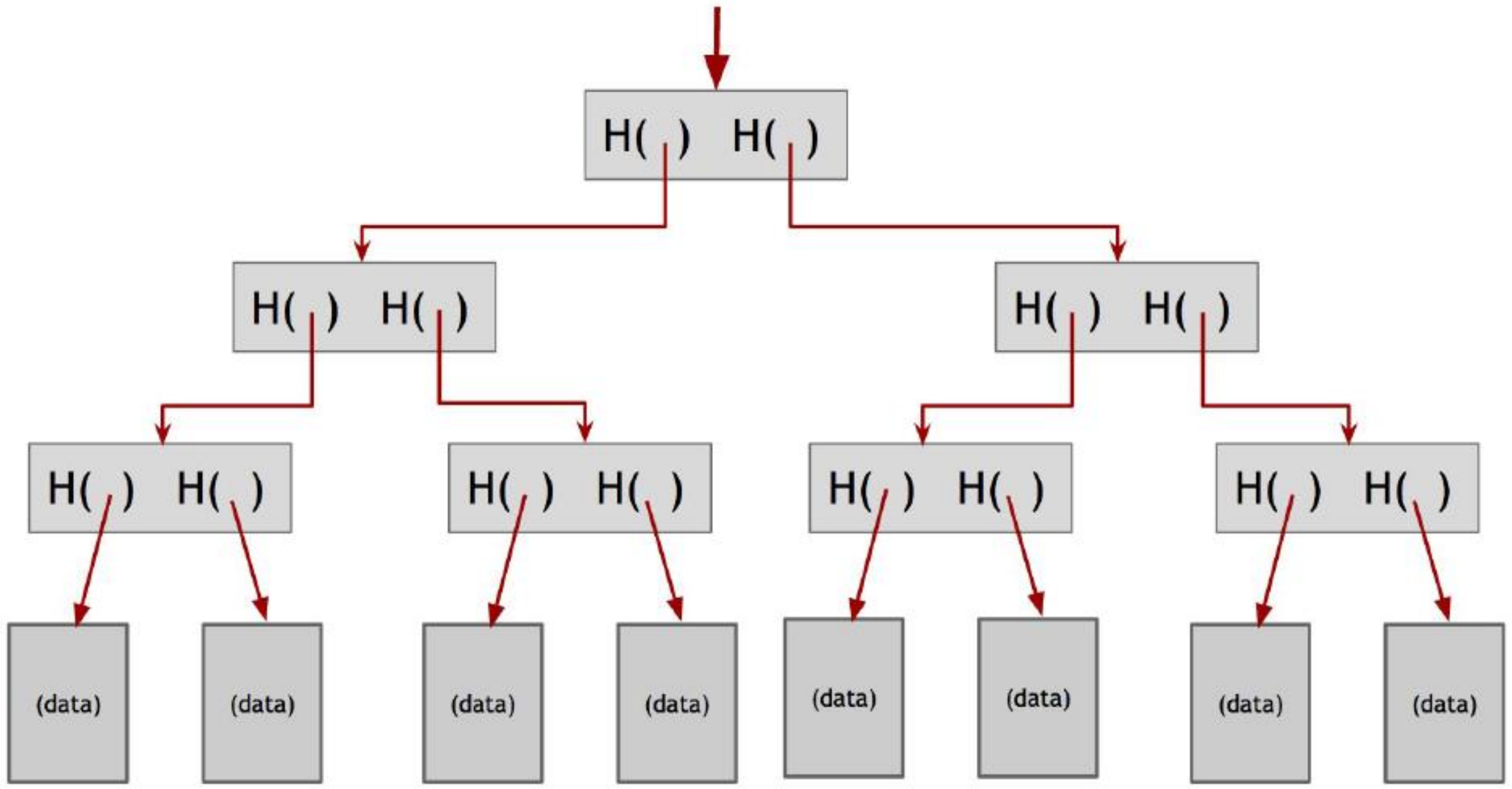


A **blockchain** is a linked list that is built using hash pointers instead of pointers

Linked List: Tampering Detection

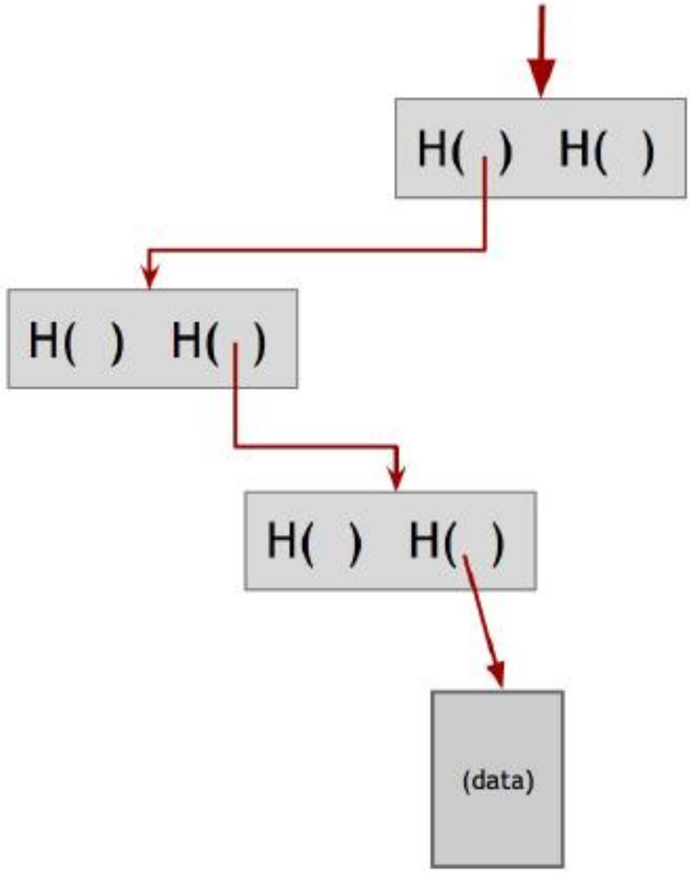


Binary Tree With Hash Pointers: Merkle Tree



https://prathamudeshmukh.github.io/merkle-tree-demo/

Proof of Membership in a Merkle Tree



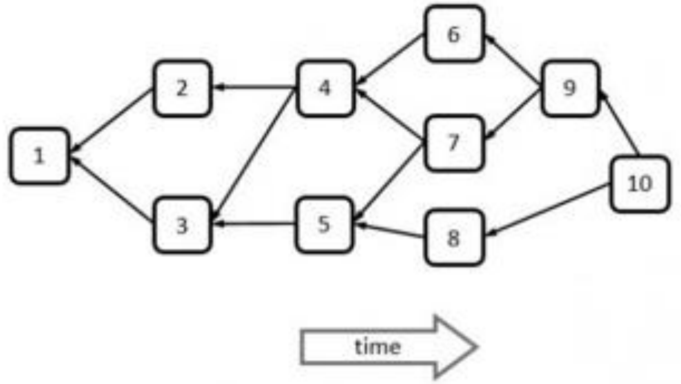
Advantages of Merkle Tree

Tree holds many items but just need to remember the root hash

Can verify the membership in $O(\log\ n)$ time

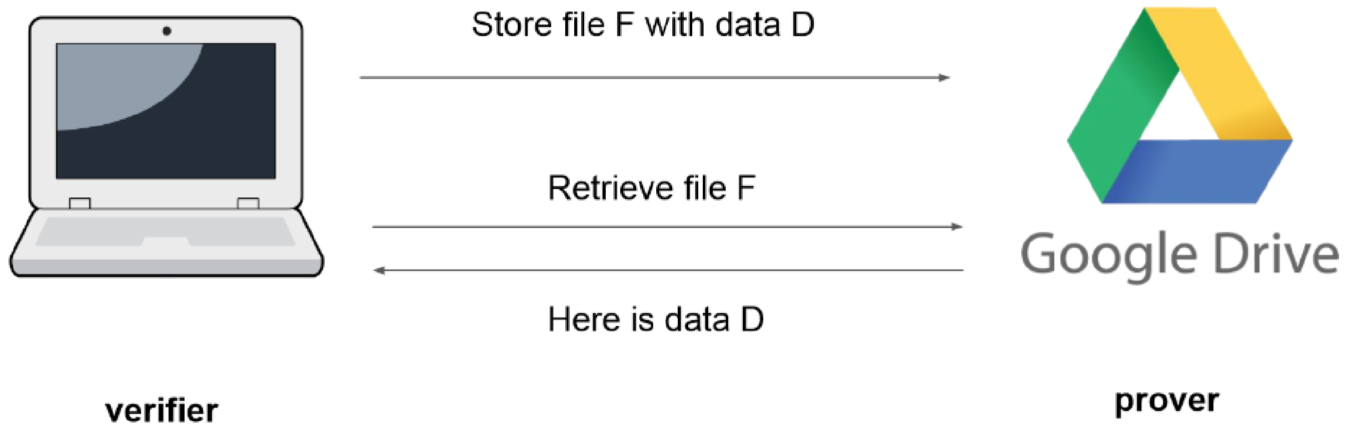
More generally we can use hash pointers in any pointer based data structure that has no cycle.

In case of cycles there is no node to start



The storage problem

- Client wants to store a file on the server
- File has a name **F** and data D
- Client wants to retrieve F later



The storage: Basic Protocol

- Client sends F with Data D to server
- Server stored (F, D)
- Client deletes D
- Client requests F from server
- Server returns D
- Client has recovered D

The storage protocol Against Adversaries

- What if server is adversial and returns D’ != D

- Trivial solution
  - Client does not delete D
  - Whenever server return D’ client can compare D and D’

What is client does not have memory to store data for a long time?

The storage : Hash based protocol

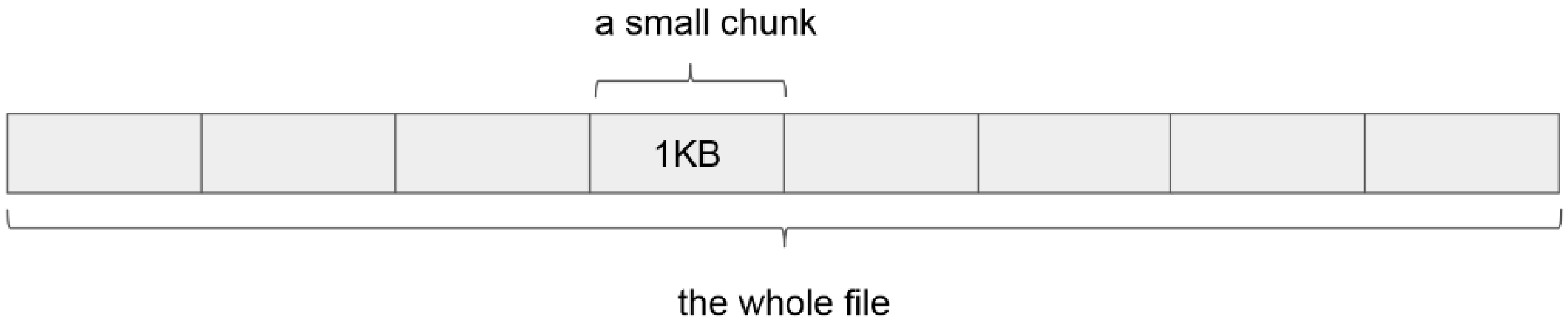
- Client send file F with Data D to the server
- Server stores (F,D)
- Client stored H(D), deletes D
- Client requests F from server
- Server returns D’
- Clinet compares H(D) = H(D’)

The storage : File chunks

- What if client wants to retrieve the 19007th byte of the file
- Must download the whole file
- Merkle tree to rescue.

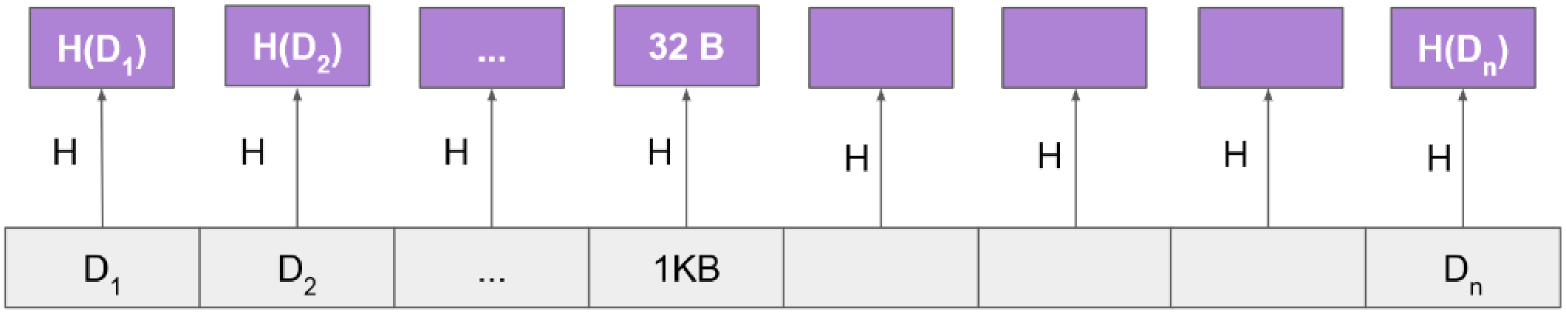
Merkle Tree:

- Splits file into chunks (say 1 KB)



Merkle Tree:

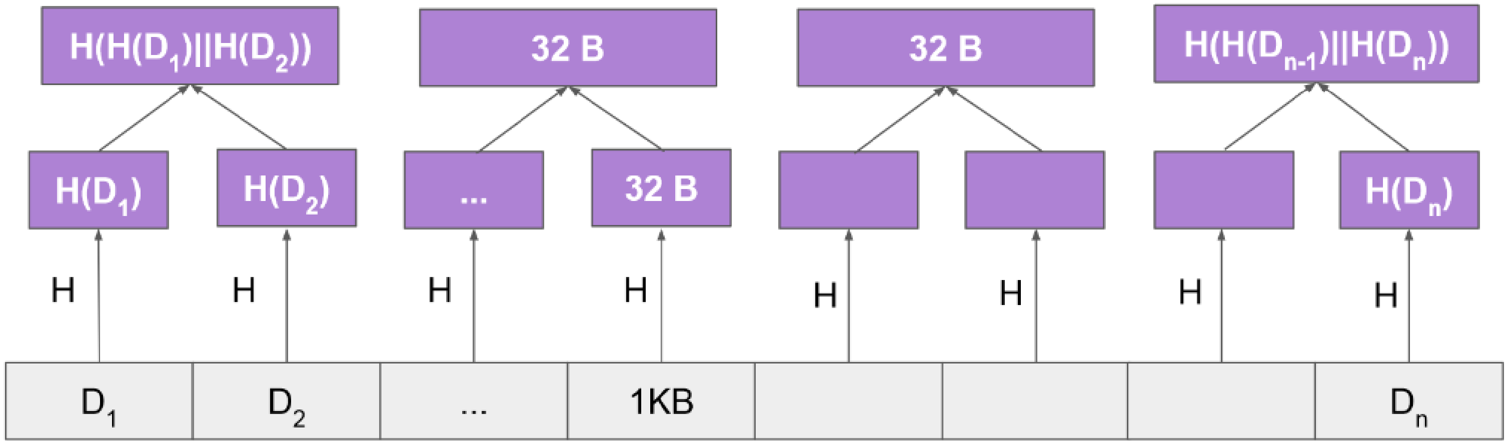
- Hash each chunk using cryptographic hash function



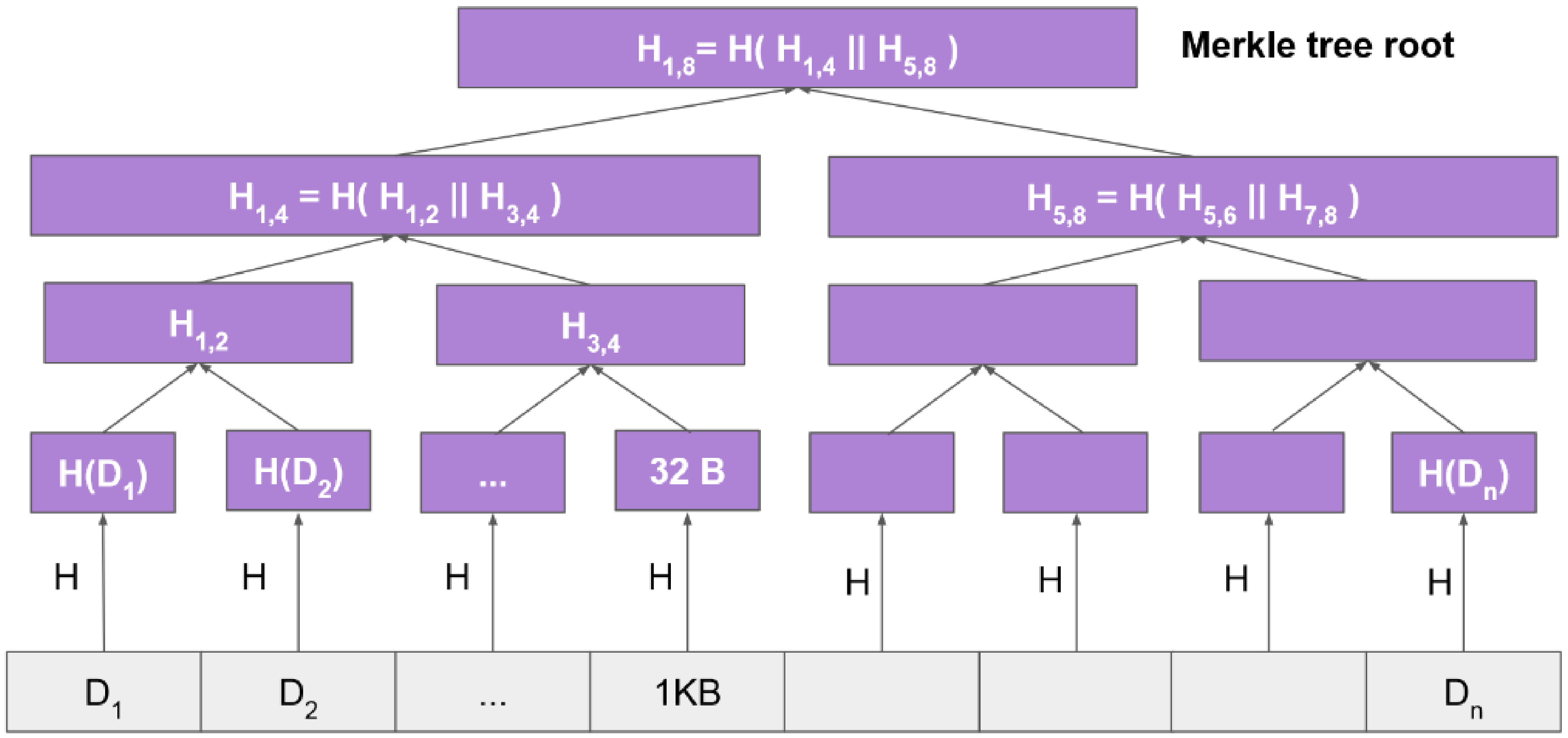
Arrows show direction of hash function application

Merkle Tree:

- Combine them to create a binary tree
- Each node stores the hash of the concatenation of their children



Merkle Tree:

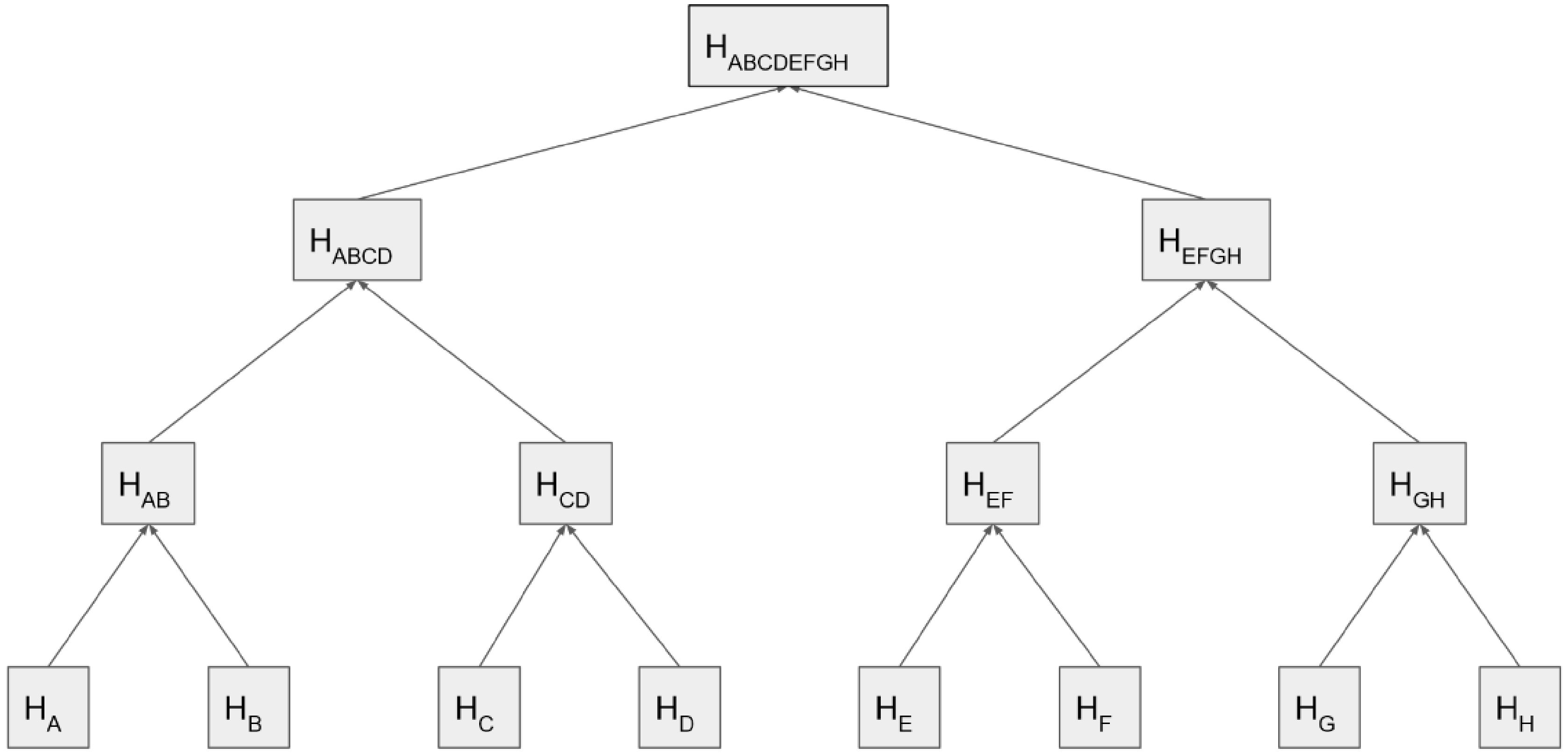


$H_{1,8} = H(H_{1,4} \| H_{5,8})$
Merkle tree root
$H_{1,4} = H(H_{1,2} \| H_{3,4})$
$H_{5,8} = H(H_{5,6} \| H_{7,8})$
$H_{1,2}$
$H_{3,4}$
$H(D_1)$
$H(D_2)$
...
32 B
$H(D_n)$
H
H
H
H
H
H
H
H
D_1
D_2
...
1KB
D_n

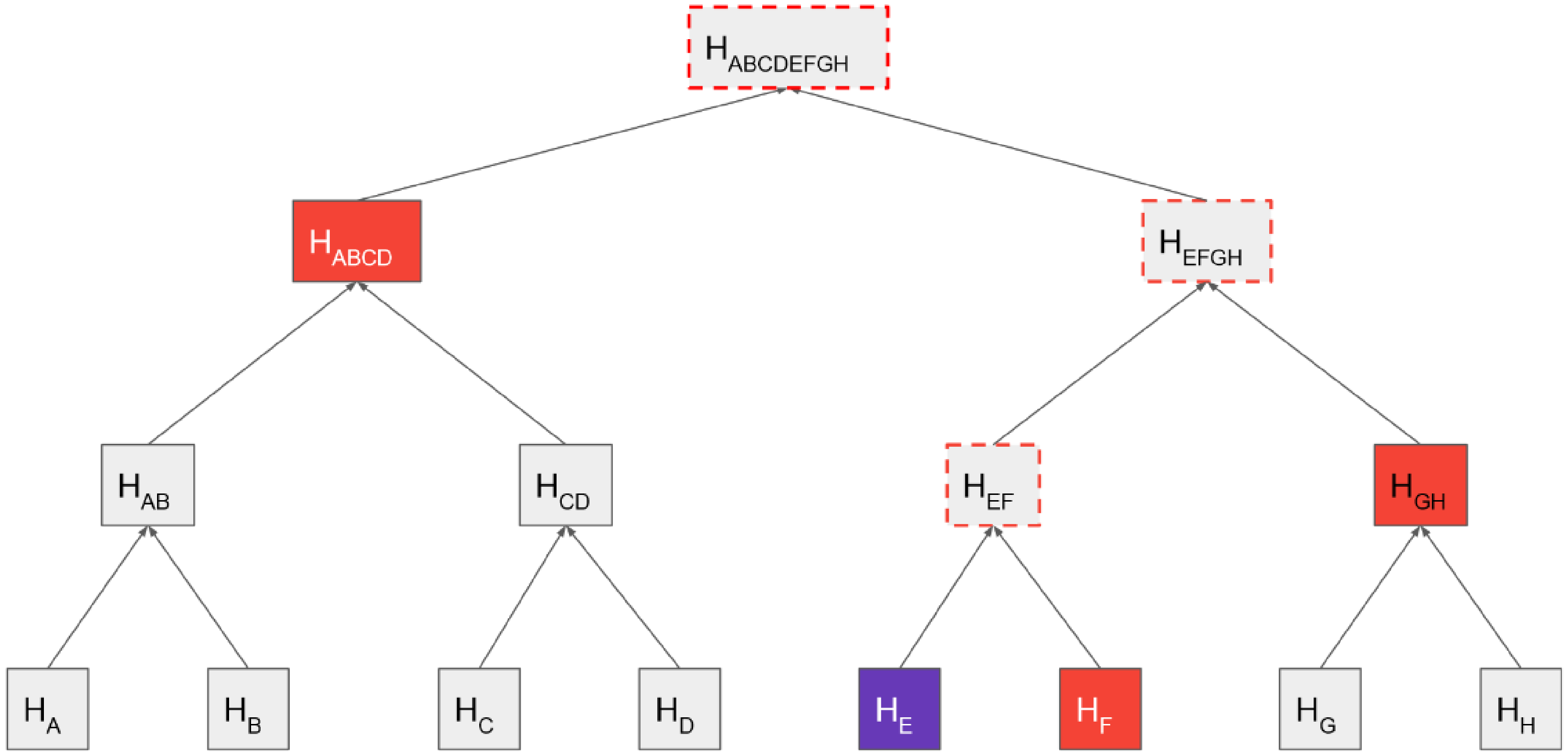
Proof of inclusion

- Client creates Merkle Tree with root MRT from file data D.
- Client send file data D to server.
- Client deletes data D but stores MTR
- Client request chunk X from the server
- Server returns chunk X and a short proof of inclusion π
- Client checks that chunk X is include in MTR using proof π

Proof of inclusion



Proof of inclusion



Proof of inclusion

- Prover sends chunks
- Prover sends siblings along path connecting leaf to MTR
- Verifier computes hashes along the path connecting leaf to MTR
- Verifier checks that computer root is = MTR
- The proof of inclusion is O(logn)
- If adversay can present proof-of-inclusion for incorrect leaf then we can break the hash function

Merkle Tree Protocol (Optional)

```
MT-Construct(D)

// Constructs a Merkle Tree with given Data D
//Return the Merkle tree root

**If** |D| = *chunk size* **then**
    MT-Construct(D) = H(D)
**Else**
    MT-Construct(D) = H(MT-Construct(D1) || MT-Construct(D2), where D = D1 || D2
```

Merkle Tree Protocol (Optional)

MT-Prove(D,x)

- Given Data D and element x in D, construct proof of inclusion
- Return the proof of inclusion π to be used with MT-construct
- Proof contains:
  - Siblings on path connecting x to root
  - A bit for each sibling indication whether the path we are taking is left or right.

Merkle Tree Protocol (Optional)

MT-Verify(r, π ,x)

- Given Merkle root r, element x and proof-of-inclusion π
- Output True/False based on whether the verification was successful

Correctness

For all D, x :

MT-Verify(MT-Construct(D), MT-prove(D,x), x) = True

Merkle Tree Applications

- Bitcoin uses Merkle Tree to store the transactions
- Bit-Torrent uses Merkle tree to exchange file
- Etheriun Blockchain uses Merkle-Patricia tries for storage and transactions

innovate achieve lead

BITS Pilani
Pilani Campus

Digital Signatures

What we want from Digital Signatures?

Only you can sign but any one can verify.

  Signature is tied to a particular document

Can’t be cut and paste to another document.

API for digital signatures

(sk ; pk) := generateKeys(keySize)

sk : secret signing key

pk : Public verification key

sig := sign(sk ; message)

isValid : = verify(pk ; message; sig)

Requirements for Signatures

Valid Signatures Verify

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

Can’t forge signatures

Adversary who, knows pk , gets to see the signature of his own choice, can’t produce a verifiable signature on another message.

Practical Stuff ...

Algorithms to generate keys need to be randomized

- So, we need a good source of randomness

Limit of message size

- fix: use Hash(message) rather than message.

Fun Trick: Sign a hash pointer

- Signature covers the whole structure

BITCOIN uses ECDSA standard for Digital Signatures

Useful trick: Public key == Identity

If you see sig such a verify(pk; msg; sig) == true

Think of it as

pk says “[msg]”

To speak for **pk** you must know **sk**

Decentralized Identity Management

Anybody can make a new identity at anytime

make as many as you want

No central point of coordination

<u>These identities are called "addresses" in Bitcoin</u>

Privacy

Addresses not directly connected to real world identity

But observer can link together an address's activity over time

BITS Pilani
Pilani Campus

Blockchain Technology
(BITS F452)

Dr. Ashutosh Bhatia, Dr. Kamlesh Tiwari
Department of Computer Science and Information Systems

innovate achieve lead

BITS Pilani
Pilani Campus

A simple Cryptocurrency

Useful trick: Public key == Identity

If you see sig such a verify(pk; msg; sig) == true

Think of it as

pk says “[msg]”

To speak for **pk** you must know **sk**

Decentralized Identity Management

Anybody can make a new identity at anytime

make as many as you want

No central point of coordination

<u>These identities are called "addresses" in Bitcoin</u>

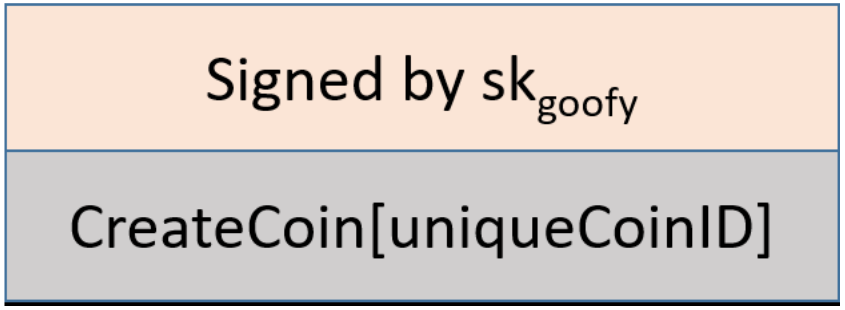
Privacy

Addresses not directly connected to real world identity

But observer can link together an address’s activity over time

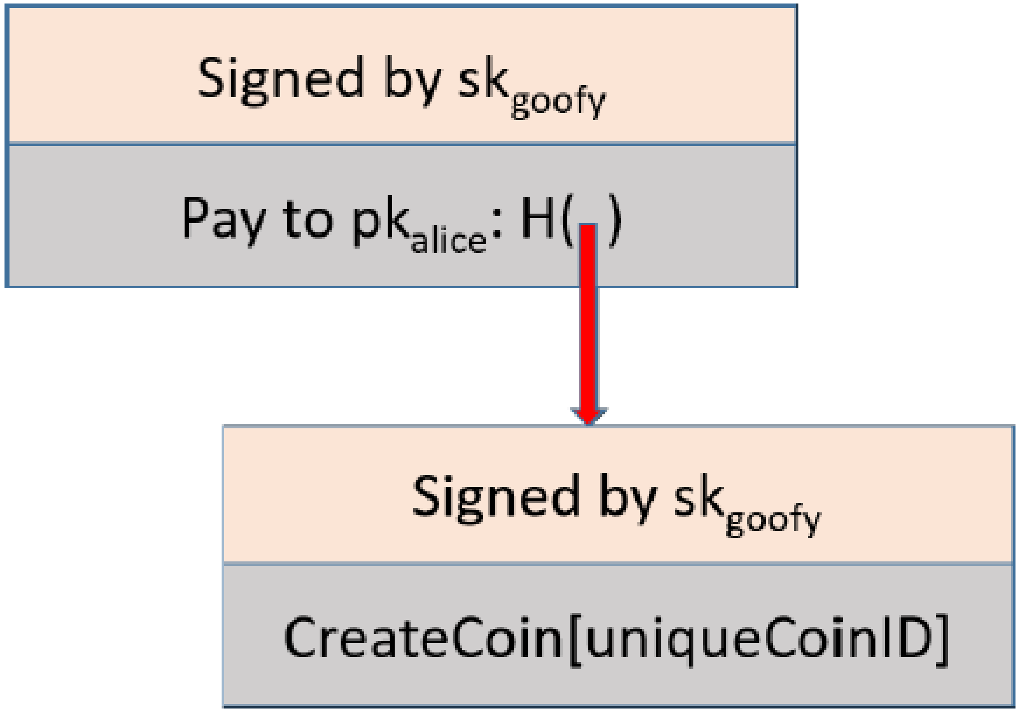
GoofyCoin

Goofy, can create new coins



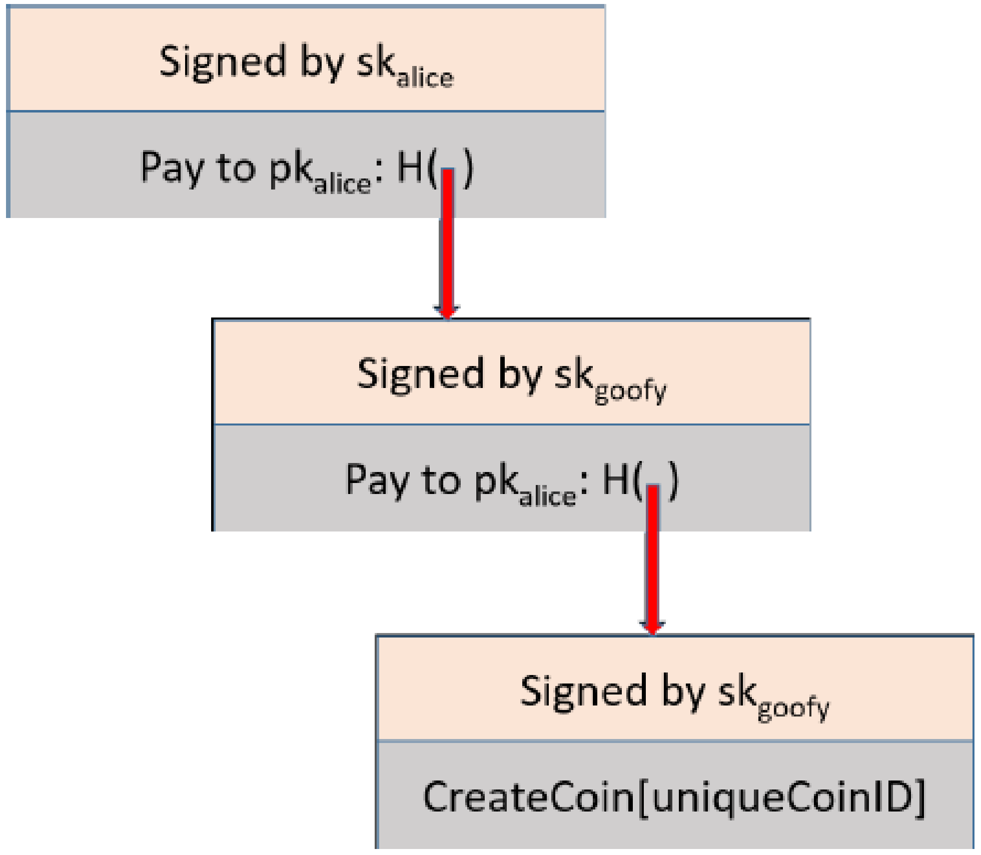
GoofyCoin

A coin's owner can spend it



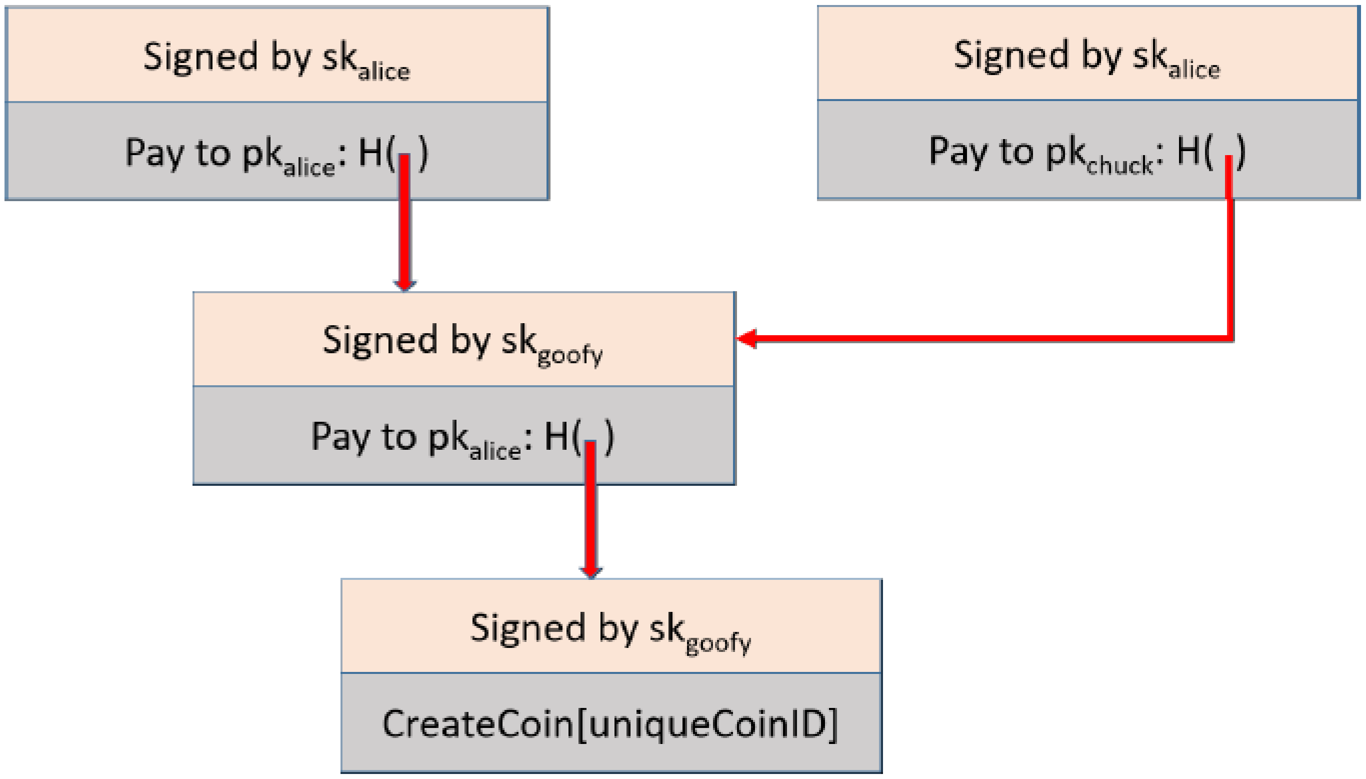
GoofyCoin

A recipient can pass the coin again



GoofyCoin

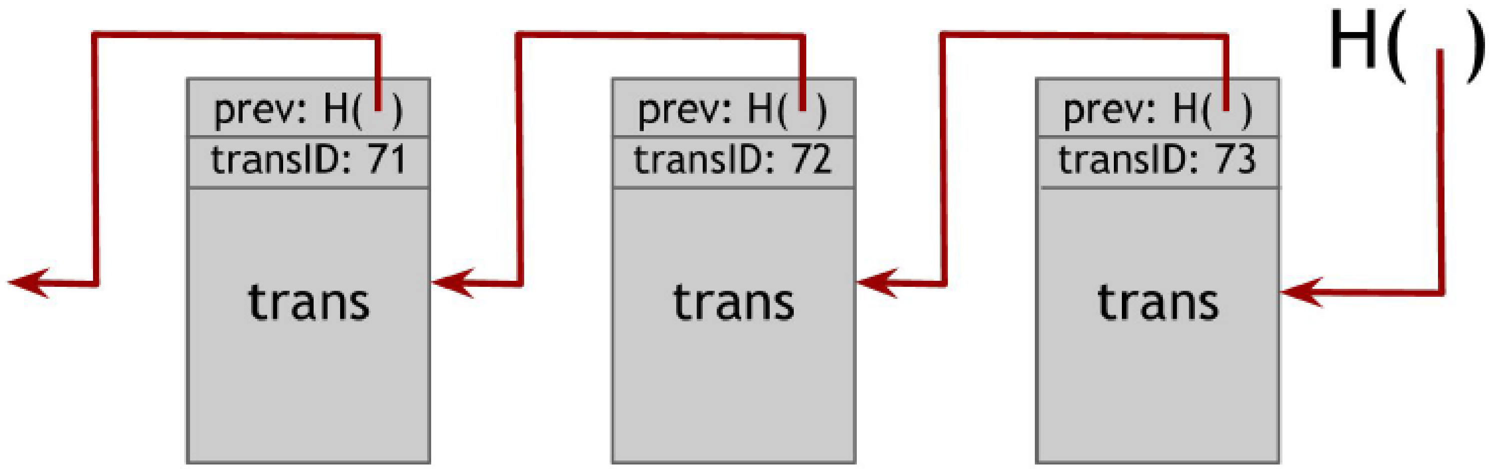
Double Spending Problem



ScroogeCoin

ScroogeCoin: Solving Double Spending Problem

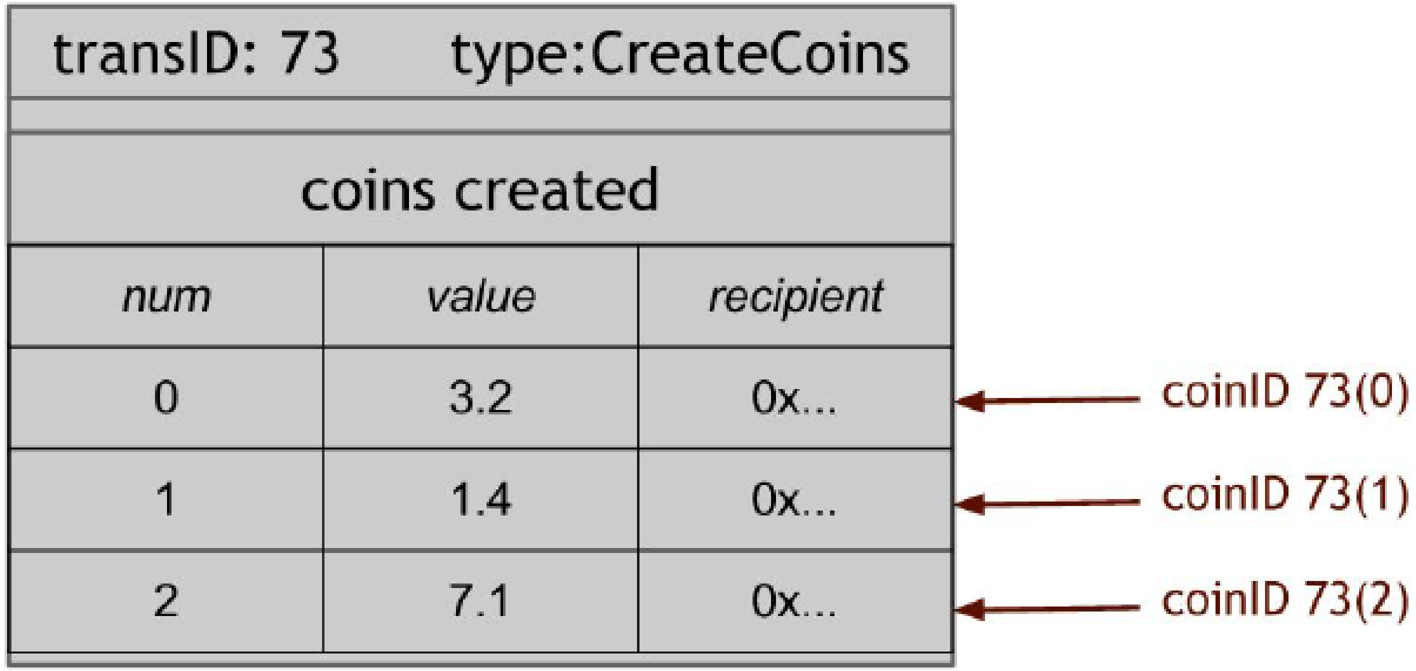
Scrooge publishes a history of all the transactions in form of a append only ledger (blockchain



Optimization: put multiple transactions in the same block

ScroogeCoin

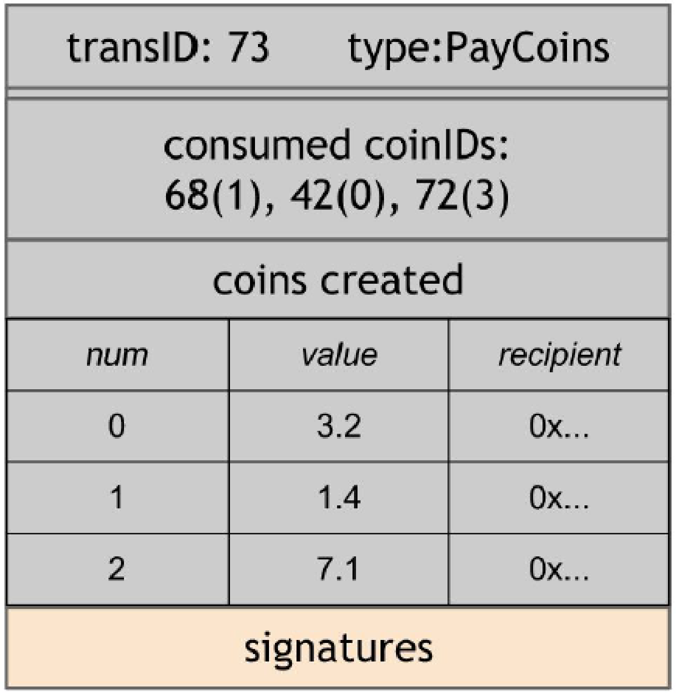
CreateCoin Transaction create a new coin



transID: 73	type:CreateCoins	
coins created		
num	*value*	*recipient*
0	3.2	0x...
1	1.4	0x...
2	7.1	0x...

ScroogeCoin

A Paycoin transaction consumes some coins and creates new coins of the same value



transID: 73 type:PayCoins

consumed coinIDs: 68(1), 42(0), 72(3)

coins created

num	*value*	*recipient*
0	3.2	0x...
1	1.4	0x...
2	7.1	0x...

signatures

Valid if

- ✓ Consumed coins are valid
- ✓ Not already consumed
- ✓ total value out = total value in
- ✓ signed by owners of all consumed coins

ScroogeCoin

Problem with the scrooge coin

- Coins can't be transferred, subdivided or combined

but you can get the same effect by using transactions to sub divide:

- create a new transaction, consume your coin and pay out two new coins to yourself.

ScroogeCoin

Crucial Question

Can we de-scoogify the currency and operate without a trusted third party

We need to figure out:

How every one agree upon a single public block chain

How every one agree upon which transactions are valid

How to assign IDs to coins in a decentralized manner.

Blockchain Technology
(BITS F452)

BITS Pilani
Pilani Campus

Dr. Ashutosh Bhatia, Dr. Kamlesh Tiwari
Department of Computer Science and Information Systems

innovate achieve lead

BITS Pilani
Pilani Campus

Decentalized Cryptocurrency

ScroogeCoin

Crucial Question

Can we de-scoogify the currency and operate without a trusted third party

We need to figure out:

How every one agree upon a single public block chain

How every one agree upon which transactions are valid

How to assign IDs to coins in a decentralized manner.

Decentralization is not all-or-nothing

Email: Decentralized protocol but dominated by centralized webmail services

Aspects of Decentralization in BITCOIN

- Who maintains the ledger?
- Who has authority over which transactions are valid?
- Who creates new Bitcoins?
- Who determines how the rules of the system change?
- How do Bitcoins acquire exchange value?

Beyond the protocol: Exchange, wallet, software and service providers

Aspects of Decentralization in BITCOIN

- **Peer to Peer Network**
  - Open to anyone, low barrier to entry
  - Currently there are several thousands of bitcoin nodes
- **Mining**
  - open to anyone but inevitable concentration of power often seem as undesirable.
- **Updates to Software**
  - Core developers trusted by the community, have great power

BITCOIN’s Key Challenge

Key technical challenge of decentralized ecash : **Distributed Consensus**

or: How to decentralize ScroogeCoin

Why Consensus Protocol ?

Traditional Motivation

Reliability in Distributed Systems

Distributed Key-Value Store enables various applications

DNS, Public-Key Dictionary, Stock Trades

Defining Distributed Consensus

There is a fix number of nodes or processes and each of these has some input value

Protocol terminates and all correct nodes decide on the same value

This value must have been proposed by some correct node

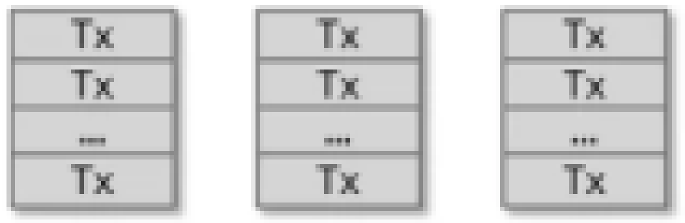
BITCOIN is a P2P system

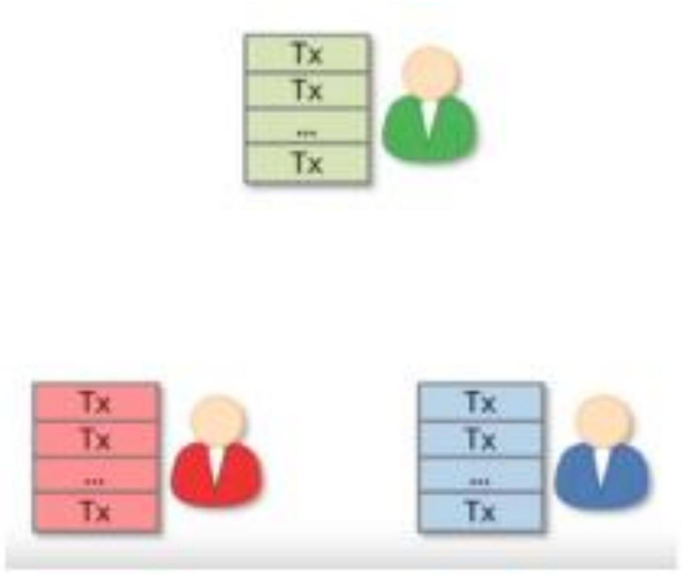
At any given time

- All nodes have sequence of <u>blocks of transactions </u>that they have consensus on

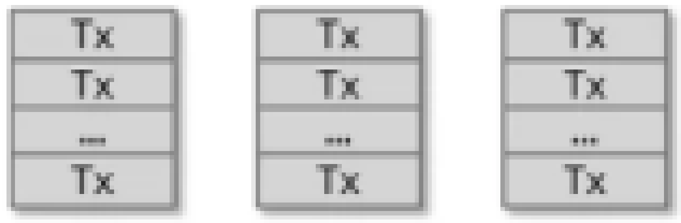
- Each node has a set of outstanding transactions that they have heard about

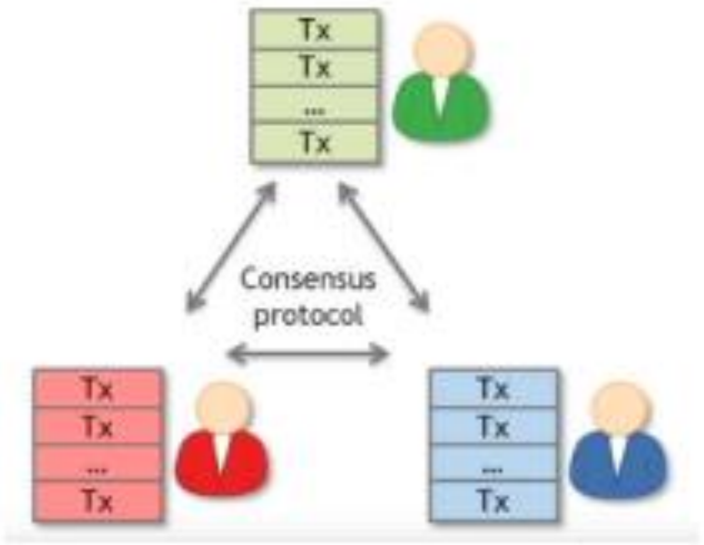
How Consensus could work in BITCOIN



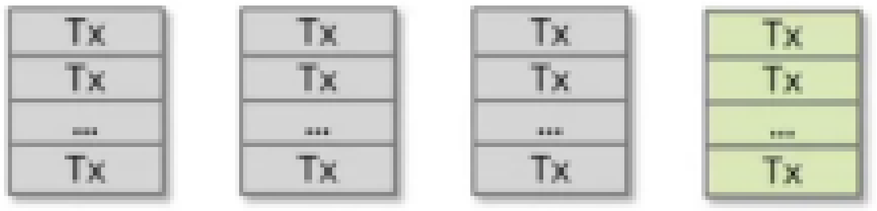


How Consensus could work in BITCOIN





How Consensus could work in BITCOIN



OK to select any valid block, even of proposed by only one node innovate achieve lead

Why Consensus is hard

Nodes may crash

Nodes may be malicious

Network is imperfect

- Not all pair of nodes connected
- Faults in network
- Latency

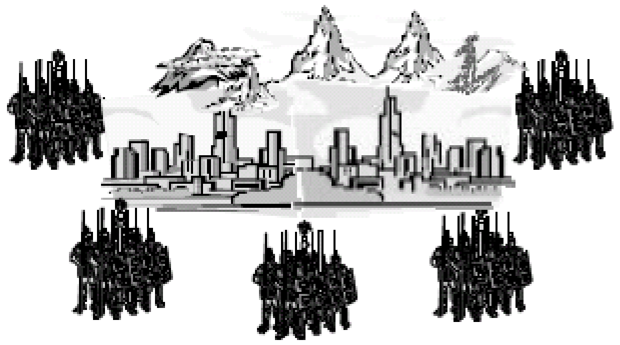
No notion of Global Time

Many impossibility results

- Byzentine generals problem
- Fischer-Lynch-Paterson (FLP) result says that you can't do agreement in an Asynchronous Message Passing system if even one crash failure is allowed, unless you augment the basic model in some way, e.g. by adding randomization or failure detectors.

Byzantine Generals Problem (Optional)

- Generals = Computer Components



- The abstract problem…
  - Each division of Byzantine army is directed by its own general.
  - There are n Generals, some of which are traitors.
  - All armies are camped outside enemy castle, observing enemy.
  - Communicate with each other by messengers.
  - Requirements:
    - G1: All loyal generals decide upon the same plan of action
    - G2: A small number of traitors cannot cause the loyal generals to adopt a bad plan
  - Note: We **do not** have to identify the traitors.

Some well known protocols

Example: Paxos

Nerver produces inconsistent result but can get stuck.

BITCOIN consensus theory and practice

BITCOIN consensus works better in practice than in theory

Theory is still catching up

BUT theory is important, can help predict unforeseen attacks.

Some things BITCOIN does differently

- Introduces incentives
  - Possible only because it’s a currency

- Embraces randomness
  - Does away with the notion of specific end point
  - Consensus happen over a long time scale

BITCOIN consensus algorithm

Keep in mind that BITCOIN does this without having any long term identities which is different from classical distributed system.

Why don't BITCOIN node have identities

Identity is hard in P2P system – **Sybil Attack**

Psedoanonymity is a goal of BITCOIN

Key Idea: Implicit Consensus

In each round a random node is picked

This node proposes a next block in the chain

Other nodes implicitly accept/reject this block

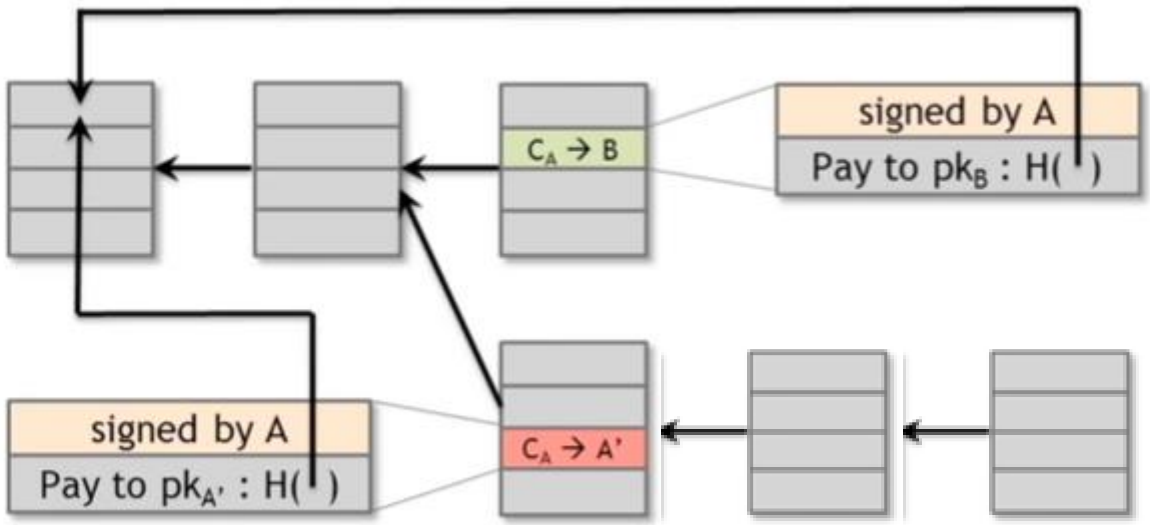
- By either extending it
- Or ignoring it and extending the chain from the earlier block

Every block contains the hash of the block it extends

Consensus algorithm simplified

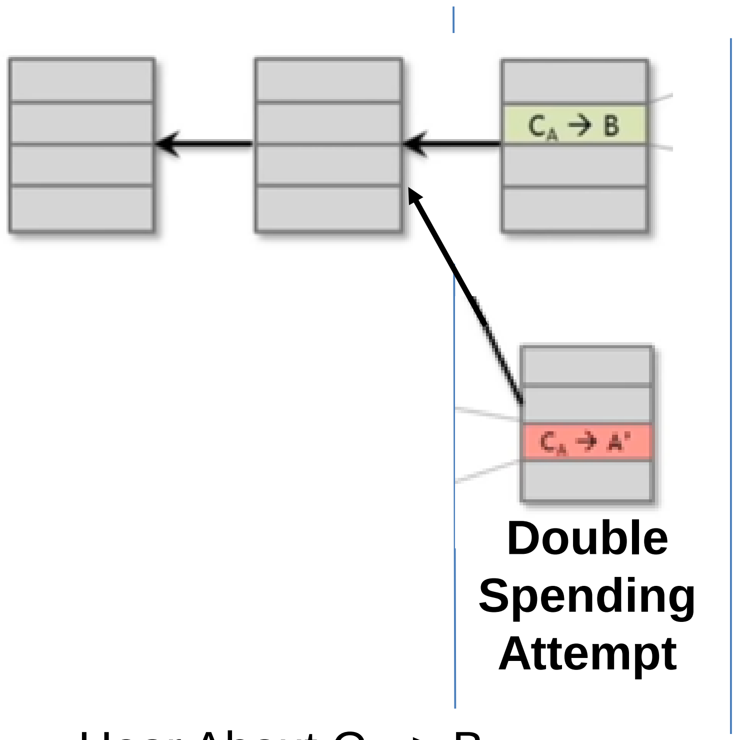
1. New transactions are broadcast to all nodes
2. Each node collects new transactions into a block
3. In each round a random node gets to broadcast its block
4. Other nodes accept the block only if all the transaction in the block are valid (unspent, valid signatures)
5. Nodes express their acceptance of the block by including its hash in the next block they create.

What can a malicious node do



Double-spending attack

From Bob the merchants point of View

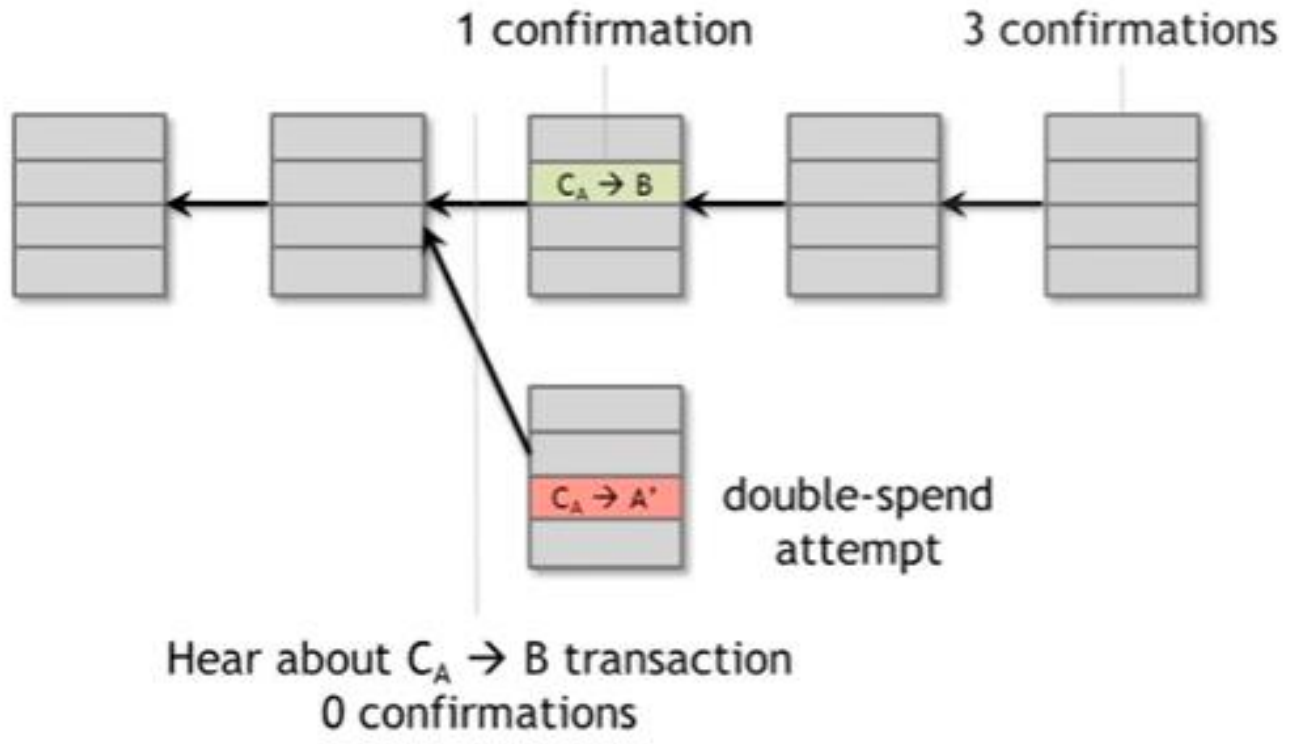


Double Spending Attempt

Hear About C_A -> B transaction in the blockchain first time (**1 confirmation**)

Hear About C_A -> B transaction over P2P network
(**0 confirmation**)

From Bob the merchants point of View



- **Double spending probability decreases exponentially with number of confirmation**
- **Most common heuristic is wait for 6 confirmations**

Recap

Protection against invalid transactions is cryptographic but enforced by consensus

Protection against double spending is purely by consensus

You are never 100% sure that a transaction is in consensus branch

Guarantee is probabilistic

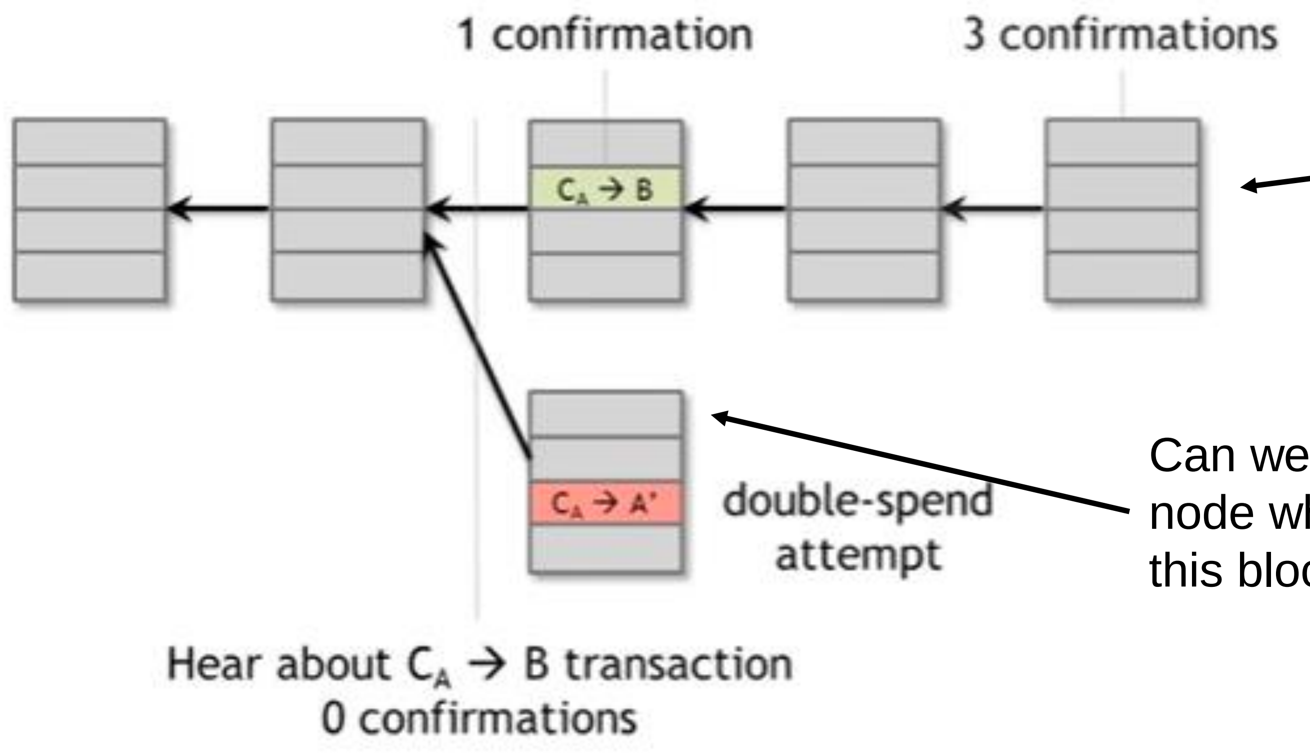
innovate achieve lead

BITS Pilani
Pilani Campus

Incentives and Proof of Work

Assumption of honesty is problematic

Can we give nodes incentives for behaving honestly.



Can we reward the nodes who created these blocks

Can we penalize the node who created this block

Incentive 1 : Block Reward

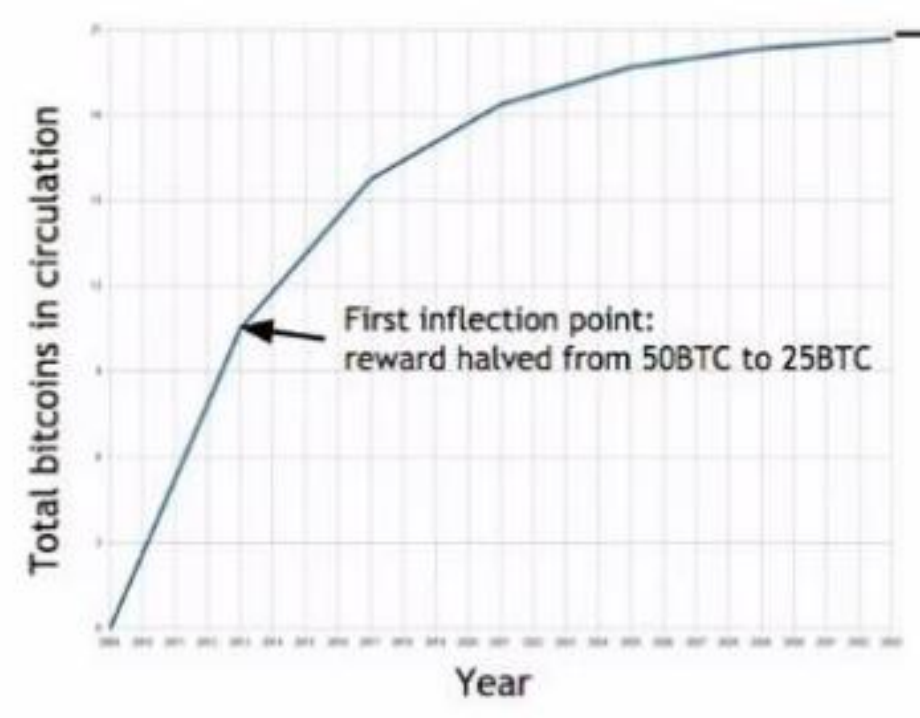
Creator of block get to

- Special coin-creation transaction in the block
- Choose receipt address of this transaction

Value is fixed currently : 6.25 BTC halves every 4 year

If the block end up on the long term consensus branch

Finite Supply of BITCOINs



Total supply: 21 million

Block reward is how new bitcoins are created

Runs out in 2140. No new bitcoins unless rules change

Incentive 2: Transaction Fee

Creator of a transaction can make its output value less than to its input value

Remainder is the transaction fee and it goes to the block creator

Purely voluntary like a tip

Remaining Problems

How to pick a random node?

How to avoid a free-for-all due to rewards?

How to prevent the Sybil attack?