Foundations of FinTech

Blockchain

Eshwar Venugopal

UCF

Blockchain

Nuts and Bolts

Blockchain: Nuts and Bolts

- To understand how a Blockchain works, we need to understand the following:
  - Cryptographic hash functions
  - Hash pointers
  - Digital signatures
  - Identity management in anonymous systems
  - Distributed consensus
    - Incentive engineering for honesty

Cryptographic Hash Functions

- A hash function is a mathematical function with the following properties:
  - Its input can be any string of any size.
  - It produces a fixed size output.
  - It is efficiently computable.
    - Means that for a given input string, you can figure out what the output of the hash function is in a reasonable amount of time.
    - Computing the hash of an n-bit string should have a running time that is O(n)

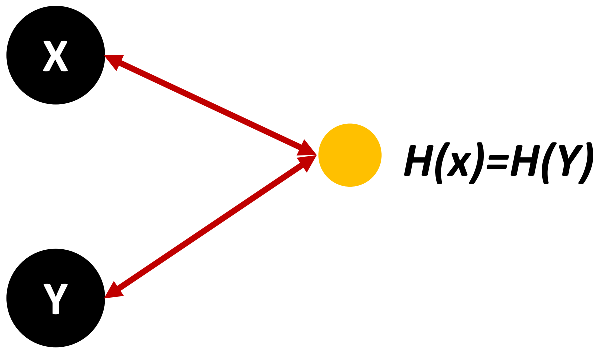
Cryptographic Hash Functions

- For a hash function to be cryptographically secure, it has the following additional properties:
  - Collision-resistance
  - Hiding
  - Puzzle-friendliness

- Hash functions in Blockchain:
  - https://www.youtube.com/watch?v=2BldESGZKB8
  - https://www.youtube.com/watch?v=lik9aaFIsl4

Cryptographic Hash Functions: 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 ≠ y , yet H(x) = H(y).*
- A collision occurs when two distinct inputs produce the same output.



- A hash function H(.) is collision-resistant if nobody can find a collision

Cryptographic Hash Functions: Collision-resistance

- Given that inputs can be of any length and the output length is fixed, most hash functions are guaranteed to have a collision input pair
- Assume that we have a 256-bit output size
  - Pick $(2^{256} + 1)$ distinct inputs and calculate hash values (Inputs>outputs)
  - We are bound to find a colliding pair
  - If we randomly input $(2^{130} + 1)$, probability of collision is 99.8%
    - Birthday paradox
- It takes one octillion (10^{27}) years to calculate 2^{128} hashes at 10,000 hashes/sec rate.
- At such small odds, we can be confident that nobody will find a collision for this 256-bit hash function.
- Bitcoin uses SHA 256

Cryptographic Hash Functions: 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.*
- That is, If we're given the output of the hash function y = H(x) , there's no feasible way to figure out what the input, x , was.
  - This is easier said than done!
- Assume we are hashing results of a "head or tails" game.
  - If an adversary knew what game is being played, he can figure out the results simply by observing a few hash outputs

Cryptographic Hash Functions: Hiding

- To avoid this, we can concatenate a small variable (r, also called nonce) to the original message (input to the hash function)
- ‘r’ must be chosen from a probability distribution that has high min-entropy
  - In information theory, Min-entropy is a measure of how predictable an outcome is.
  - High min-entropy means that the distribution is spread out
  - E.g., if ‘r’ is chosen uniformly from all strings of 256-bit length
    - Then choosing any one value has a probability of $1/2^{256}$

Cryptographic Hash Functions: Puzzle Friendliness

- *H(.) is puzzle-friendly if for every possible n-bit output value y, if k is chosen from a distribution with high min-entropy, then it is infeasible to find x such that H(k || x) = y in time significantly less than* 2^n.

- If someone wants to target the hash function to come out to some particular output value 'y', that if there's part of the input that is chosen in a suitably randomized way, it's very difficult to find another value that hits exactly that target.
  - That is, there's no solving strategy for this puzzle which is much better than just trying random values of x .
  - So, if we want to pose a puzzle that's difficult to solve, we can do it this way as long as we can generate 'k' in a suitably random way.

Hash Function: SHA-256

- Bitcoin uses SHA-256 for:
  - Mining as Proof of work algorithm
  - Creation of bitcoin addresses to improve security and privacy

- Recollect that a good hash function has the following properties:
  - Its input can be any string of any size.
  - It produces a fixed size output.
  - It is efficiently computable.
  - Collision-resistance
  - Hiding
  - Puzzle-friendliness
- Note: not all properties are necessary for every use of hash functions

Hash Function: SHA-256

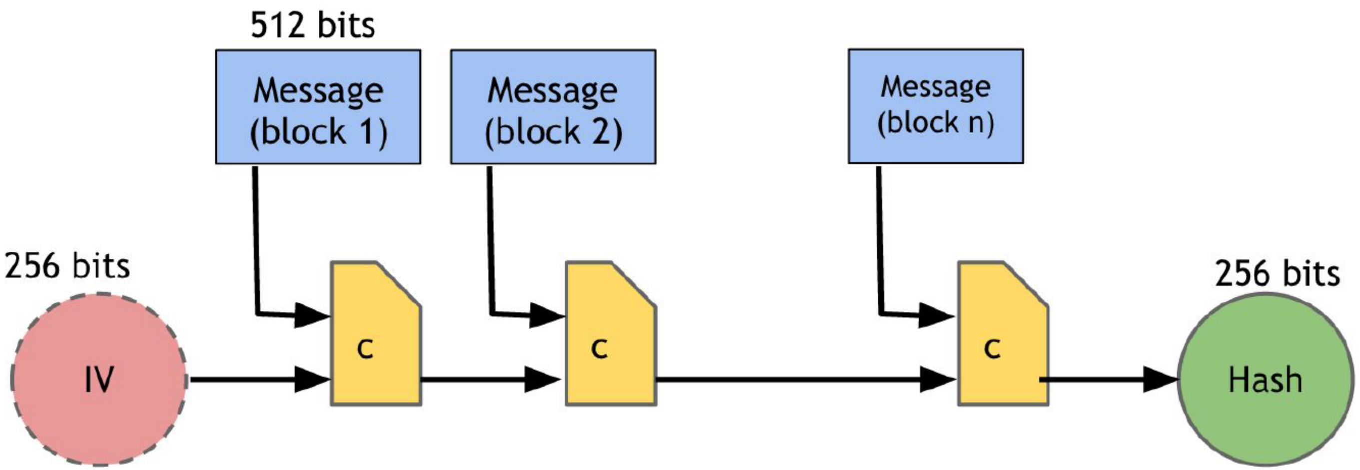
- Sample hash outputs:
  - This is a test for FinTech class at UCF
    - Output: db1793a70838f7aec96b52fec1578d39692357430cc989998c5d9deddce8b044
  - This is a test for finTech class at UCF
    - Output: 09005cad4898515eeee1510c04b827cbbbf3a042eb108d49509754a9ffe7c852
  - This is a test for FinTech class at UCF 00
    - Output: 178a2b625f463da74b3cd3dfc652248983c67f650cd74fe3a8203e8f36d5269e

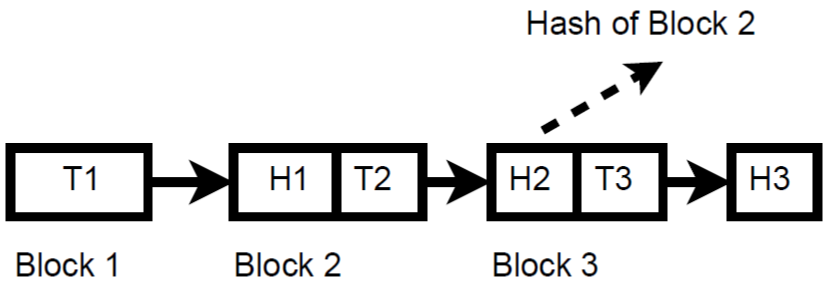
- SHA-256 satisfies required properties all *except the first one.*
- SHA-256 can only take inputs of fixed length (768-bit)

Bitcoin's Hash Function: SHA-256

- Merkle-Damgard Transform can be used to convert arbitrary length inputs into fixed length.
- Example: SHA-256 takes 768-bit input and produces 256-bit outputs
  - Divide the input into (768-256) 512-bit length blocks
  - Pass each 512-bit length block with the 256-bit hash output of the previous block
    - Total length is 768-bit and each block is chained to the previous one
    - For the first block, pass an Initialization vector (IV)

- Hashing process

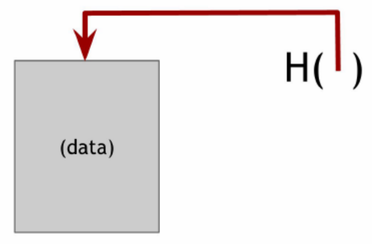




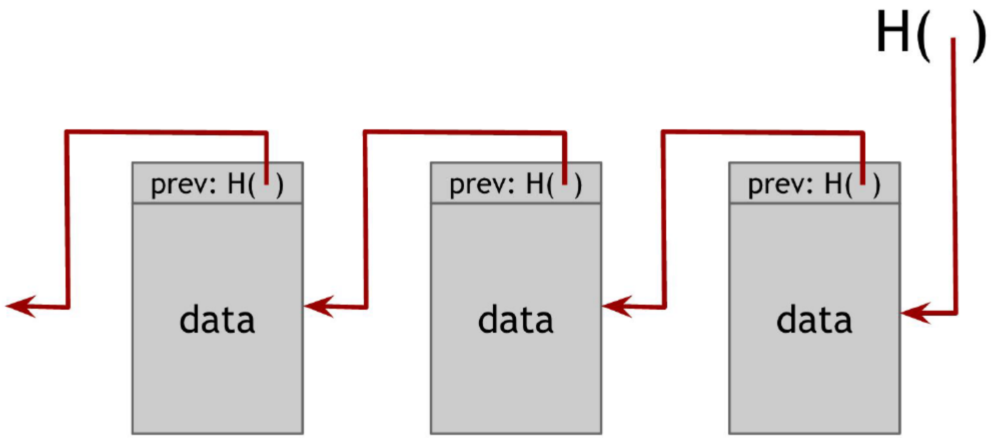
  - Take a look at: https://www.youtube.com/watch?v=s7arHByjSOw

Data Structure: Hash Pointers

- *Hash pointer is a pointer to where data is stored with a cryptographic hash of the value of the data at some point in time.*



- Blockchain is a series of blocks, where each block has data and a hash pointer to the previous block.



- *Essentially, a blockchain is a linked list that is build with hash pointers instead of pointers.*

Data Structure: Hash Pointers

- This method of appending data to the end of a log with hash pointer to previous block makes blockchains “tamper-evident” .

- Suppose someone tampers with the data in block ‘b’
- Then the hash in block ‘b+1’ will not represent the data in block ‘b’

Digital Signatures

- Online equivalent of physical signature
- Properties required:
  - Unique to a person/account, so it can be verified
  - Specific to a document, so copy-paste can be avoided

- Digital signatures are based on asymmetric cryptography.
- They provide evidence of:
  - Origin
  - Identity
  - Status of electronic documents – transactions, messages, etc.

Digital Signatures

- Digital signatures consist of three algorithms:
  - *(sk, pk) := generateKeys(keysize)* [Randomized algorithm]
    - The generateKeys method takes a key size and generates a key pair.
    - The secret key 'sk' is kept privately and used to sign messages.
    - 'pk' is the public verification key that you give to everybody. Anyone with this key can verify your signature.
  - *sig := sign(sk , message)* [Randomized algorithm]
    - The sign method takes a message and a secret key, 'sk' , as input and outputs a signature for message under 'sk'
  - *isValid := verify(pk , message , sig)* [Deterministic algorithm]
    - The verify method takes a message, a signature, and a public key as input.
    - It returns a boolean value, isValid , that will be *True if sig is a valid signature* for message under public key 'pk' , and false otherwise.

Digital Signatures: Properties

- We require that the following two properties hold:
  - Valid signatures must verify. *verify (pk , message , sign(sk , message)) == true*
    - People should be able to verify that the signature is yours
  - Signatures are *existentially unforgeable*
    - It should be computationally infeasible to forge your signature on new messages/documents
    - Remember that attackers can only observe public keys
    - Sign algorithm need to have a good source of randomness
    - In blockchains, it is better to sign a hash pointer

Digital Signatures: ECDSA

- Elliptic Curve Digital Signature Algorithm (ECDSA)
  - A U.S. government standard and believed to be secure after considerable testing
  - Bitcoin uses “secp256k1” with ECDSA that provides 128-bits of security
    - Need to perform 2^{128} symmetric-key cryptographic operations (hash function calculations) to break
  - More common curve is “secp256r1”

Private key	256 bits
Public key, uncompressed	512 bits
Public key, compressed	257 bits
Message to be signed	256 bits
Signature	512 bits

Size of variables

Identity Management: Public keys

- Public keys (pk) of a digital signature are observable
- To use a public key, the user (say U1) must also have the secret key (sk)

- So, any message signed with U1's pk is a message from U1
    - 'pk' is the digital identity of user U1
- In the context of Bitcoin, your "address" is just the hash of your public key

- This is *decentralized identity management*
    - You do not have to inform a central server of your username
- Note that this is not complete anonymity
    - Based on history, people can tie messages to a 'pk'; but less likely to a person

Double-spending and Centralization

- Two issues that have plagued cryptocurrencies before Bitcoin are:
  - Double spending
  - Requirement of a central party to verify transactions

- Making a blockchain append-only and immutable partly solves the double-spending problem
  - People cannot remove previous records
  - So, it is easier to look at the history and check if someone is trying to double-spend

- But, who is going to perform this verification?
  - Older solutions appointed a trusted central player
  - That is too much power in the hands of one entity

Decentralization

- What is required to move away from centralization?
  - A way for all users to agree on history of the ledger
  - A way to agree which transactions are valid
  - Ability to incentivize users to participate in verification

- Decentralization is the end result

Decentralization in Bitcoin

- Let us break down the question of how the Bitcoin protocol achieves decentralization into five more specific questions:
  - Who maintains the ledger of transactions?
  - 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?

- Let us focus on the first three questions that reflect the technical details of the Bitcoin protocol
- The last two are market-based questions

Distributed Consensus Protocol

- Decentralization means that data is stored/distributed across different servers.
  - Facebook, for example, stores data in numerous servers across the world
- There has to be a way for these servers to sync with each other and maintain a reliable copy of the data.

- Distributed consensus protocol: There are n nodes that each have an input value. Some of these nodes are faulty or malicious. A distributed consensus protocol has the following two properties:
  - It must terminate with all honest nodes in agreement on the value
  - The value must have been generated by an honest node

Distributed Consensus Protocol: Objective

- Nodes have to reach a consensus on:
  - Which transactions were broadcast
  - The order of transactions (which form a block)

Distributed Consensus Protocol: Process

At the beginning:

- Each node will have a copy of the ledger (i.e., list of blocks on which they have reached consensus)
- Users broadcast their requests across the P2P network
  - So, in addition to consensus blocks, nodes will have a list of transactions that are not part of any block
    - These transactions have to be assembled into a block
    - And, consensus has to be reached before the block can be added to the blockchain
- Each node will propose its own set of outstanding transactions to be the next block
- Nodes will execute a consensus protocol and a valid block will be added to the chain
- Even if some transactions were not included in the current block, they will added in the following blocks

Distributed Consensus Protocol: Issues

- Reaching consensus is not easy:
  - P2P networks are imperfect (not all nodes are connected)
  - Latency is an issue
  - Hardware could crash
  - Deliberate attacks have to be overcome (not all nodes may be honest)

- Latency in the network means that there is no global time
  - Makes if-then kind of transactions difficult to execute
- Dis-honest nodes can lead to breakdown
  - *Byzantine Generals Problem:* The Byzantine army is divided into several divisions and commanded by generals. They communicate via messengers. Some generals may be traitors and compromise a joint attack plan. If one-third of the generals are traitors, there can never be a unified plan (https://www.youtube.com/watch?v=kZXXDp0_R-w)
  - *Sybil Attacks: https://www.youtube.com/watch?v=-EKhIBUQjcA*
  - *Fischer-Lynch-Paterson impossibility result:* showed that there can never be consensus if even one node is corrupt.

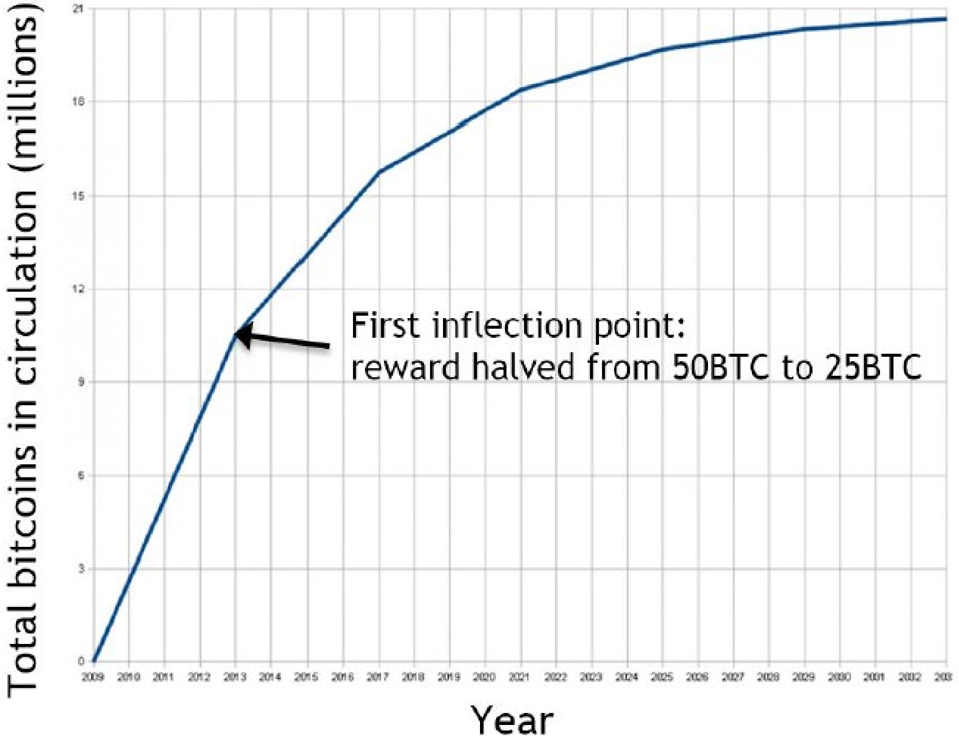
Distributed Consensus Protocol: Bitcoin solution

- Bitcoin achieves consensus by violating assumptions in traditional consensus models:
  - It introduced the concept of incentives
  - It does away with the notion of specific starting and ending point for consensus
  - It relies on randomization

- Simplified Bitcoin consensus algorithm:
  - New transactions are broadcast to all nodes
  - Each node collects new transactions into a block
  - In each round a *random* node gets to broadcast its block
  - Other nodes accept the block only if all transactions in it are valid (unspent, valid signatures)
  - Nodes express their acceptance of the block by including its hash in the next block they create

Bitcoin solution: Incentives

- The incentive for adding blocks (verifying transactions) to the blockchain are:
  - Block rewards
    - At the beginning, the reward was 50BTC
    - It halves every 4 years or 210,000 blocks
    - It is currently 6.25BTC
  - Transaction fees
    - Creators of transactions can choose to pay fees
    - Given that block rewards will soon be too low, fees will become a prominent incentive for miners to include a transaction in a block



- See paper- *StableFees: A Predictable Fee Market for Cryptocurrencies*, by Basu, Easley, O'hara & Sirer (2020)

Bitcoin solution: Long-term consensus

- Even though consensus has to be reached for each block, the transaction is not considered as confirmed immediately.
- Typically, a transaction is said to be confirmed after the block it contains has received *multiple confirmations*
  - That is, multiple blocks have to be added after the current block
  - 6 confirmation is the current heuristic
  - So, typically prudent service providers will wait for multiple confirmations before providing the service
- Similarly, *a node will get its block reward only after it has received multiple confirmations and is part of the longest chain*
  - Note that since blocks are sequentially added, the probability of a block becoming part of the longest chain increases exponentially after every confirmation

Bitcoin solution: Randomization

- There are multiple issues here:
  - A node has to be randomly chosen to propose a block
  - Incentives can lead to a rat race
  - Attackers with multiple nodes will try to subvert consensus

- Mining is the solution for all these issues:
  - Proof-of-work
  - Proof-of-stake
  - Casper
  - Gas
  - Steem

- For more Bitcoin solution see https:// bitcoin.org/en/developer-documentation

Bitcoin solution: Randomization & Mining

- Instead of explicitly selecting random nodes, mining allows us to approximate randomization
- Nodes are selected in proportion to the resource a node commands
  - Computing power: proof-of-work
  - Ownership: proof-of-stake

- In proof-of-work, nodes are competing against each other to solve a moderately difficult problem (hash puzzles in bitcoin)
- In bitcoin
  - The node that is proposing a block is required to find a number (aka nonce) that satisfies the following:
  - $H(nonce||prev_{hash}||tx||tx \ldots ||tx) < target$
  - Target is determined by the prevailing difficulty levels
- *This process of repeatedly trying to solve the above puzzle is called mining.*

Bitcoin solution: Randomization & Mining

- The mechanism behind proof-of-work simultaneously solves two problems.
  - First, it provides an effective consensus algorithm, allowing nodes in the network to collectively agree on a set of updates to the state of the Bitcoin ledger.
  - Second, it provides a mechanism for allowing free entry into the consensus process, solving the political problem of deciding who gets to influence the consensus, while simultaneously preventing Sybil attacks— that is, attacks where a reputation system is subverted by forging identities in peer-to-peer networks.

- An alternative approach is proof- of-stake, calculating the weight of a node as being proportional to its currency holdings and not its computational resources.
  - Ethereum contemplated a shift to Proof-of-Stake by mid-2019.
  - The merge was postponed multiple times and finally happened on September 15, 2022.
    - For more read: https://ethereum.org/en/upgrades/merge/