OBG Onboarding Week3

Joe

February 15, 2023

Outline for today:

- Homework Check
- Week2 refresher
- Why we care about consensus and cryptography
- Consensus Overview: Theorems and Models
- Revisiting the Bitcoin protocol
- Cryptography Overview
- Putting it all together

Homework Check

Who did the homework

What did you learn from BTC independence day? Did you find it interesting? What questions do you have?

Week2 Refresher

1. Who remembers what we talked about last week?

Week2 Refresher

1. Who remembers what we talked about last week?
2. Global view of network vs. local view

Week2 Refresher

1. Who remembers what we talked about last week?
2. Global view of network vs. local view
3. Why the state transition function makes blockchains different

Week2 Refresher

1. Who remembers what we talked about last week?
2. Global view of network vs. local view
3. Why the state transition function makes blockchains different
4. Social/out of protocol forks

Week2 Refresher

1. Who remembers what we talked about last week?
2. Global view of network vs. local view
3. Why the state transition function makes blockchains different
4. Social/out of protocol forks
5. Stayed fairly high-level, today going to get into the nitty gritty of how different parts of the protocol work!

Why do we care about consensus and cryptography?

Consensus: a process in computer science used to achieve agreement on a single data value among distributed processes or systems. These algorithms are designed to achieve reliability in a network involving multiple users or nodes

Cryptography: the study of secure communications techniques that allow only the sender and intended recipient of a message to view its contents.

These are the most basic ingredients of a blockchain!

Consensus $\longrightarrow$ the "engine" of blockchains (how block producers agree on the next block)
Cryptography $\longrightarrow$ the "oil" of blockchains, helps everything work smoothly!

Why do we care about consensus and cryptography?

The upper bounds of what we can do with any technology is determined by how well our understanding of the fundamental principles is

Same reason that physics breakthroughs push the envelope for what modern technology and other sciences can do

Consensus and cryptography have the same fundamental value to blockchains as physics does to the real world!

Why do we care about consensus and cryptography?

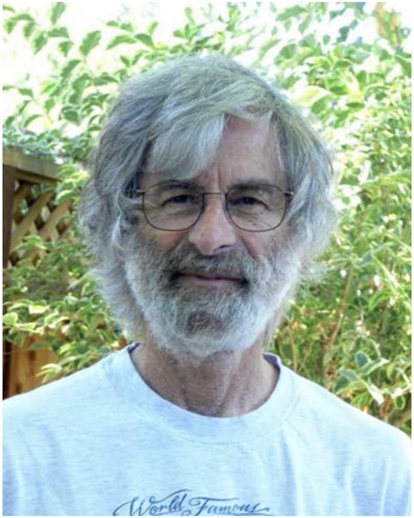
Today is about teaching you the first principles of blockchains!

Disclaimer: Modern protocols are *much* more complicated than the topics we are covering. Today is meant to give a basic understanding so you can continue the research if you want.

Today is about showing you big concepts from the type of research I want you guys to do while you are in OBG! Can find thousands of papers about these topics on arxiv and eprint.iacr

Consensus overview

Figure: Leslie Lamport



Consensus Overview

Lamport et. al wrote a paper titled *The Byzantine Generals Problem* in 1982

Used an allegory of military generals from the Byzantine empire attempting to coordinate an attack on a city

The catch is that one of the generals is a traitor! Generals only succeed if they can get all the honest generals to attack or retreat together. Generals fail if the traitor can convince them to split up between attacking or retreating

Consensus Overview: big concepts

Byzantine failure: When a computer node has an arbitrary failure, can be a simple computer malfunction or a malicious attack on the network

Byzantine fault tolerant protocol: A protocol that lets nodes come to agreement despite the existence of byzantine failures.

Consensus Overview: The communication model

Consensus protocols must make an assumption about communication between nodes during consensus rounds:

$\rightarrow$ Synchrony: Nodes have perfect information about other nodes by time Δ (bitcoin)

$\rightarrow$ Asynchrony: Δ does not exist, consensus round doesn't end until nodes have responses from all other nodes (no blockchain uses this model)

$\rightarrow$ Partial-synchrony: If a node does not respond by time Δ, other nodes assume that the node is malicious (Solana, tendermint chains)

For blockchains Δ is the time in-between blocks.

Why does the type of communication model matter?

The communication model determines how much power an adversary has! To disrupt consensus, need to cause disagreement between the honest nodes

Adversary can't fake messages, but can delay them by DDoSing honest nodes. The proportion of nodes an adversary needs to delay messages for to disrupt consensus depends on the communication model.

Synchrony

Synchrony: Nodes have perfect information about other nodes by time Δ

Agreement requires greater than 50% honest nodes (2f + 1)

All consensus protocols must assume honest majority to work. Because of the perfect information assumption, synchronous protocols can tolerate up to 49.999% faults

Problems with synchrony

Δ must be very large because nodes are distributed around the world (bitcoin block time is 10 minutes)

If honest nodes do not have perfect information about other honest nodes by Δ, synchrony assumption is violated and the 51% honest assumption fails.

Synchronous protocols have bad notions about finality as there is no way to tell if nodes are communicating synchronously. In the real world, must program ways to recover from situations where nodes were not behaving synchronously.

Partial synchrony

Partial synchrony: If a node does not respond by time Δ, other nodes assume that the node is byzantine

Agreement requires greater than 66% honest nodes (3f + 1)

1/3 dishonest can cause disagreement by delaying 1/2 honest messages until Δ, and then conflicting with the remaining 1/2 honest.

This doesn't work in synchronous protocols because they assume all messages from honest nodes have been delivered by Δ

Communication models

What are the trade-offs?
- Synchrony: Can tolerate 1/2 faults, bad notions about finality. Standard for people to wait multiple bitcoin blocks before they believe their tx is safe, this is because of the synchrony assumption

- Partial synchrony: Can only tolerate 1/3 faults, good notions about finality

This finality tradeoff has many second order effects!

CAP Theorem

Theorem: a belief from theoretical computer science that distributed systems can only provide two of the three: consistency (finality), availability, or network partition

Blockchains are built for resiliency, must assume that a network partition will happen at some point!

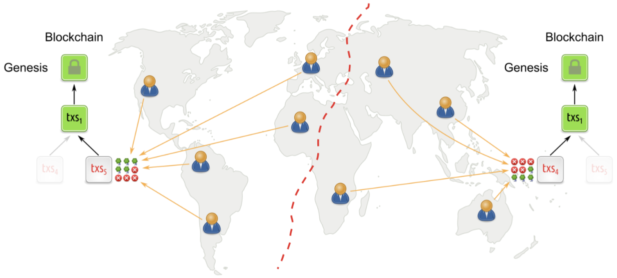
Synchronous protocols: Probabalistic finality but this lets them optimize for availability! Nodes are allowed to join in and drop out of consensus in between rounds because of the assumption that a no vote is not malicious. This allows proof-of-work to exist! These protocols are known as longest-chain protocols.

Partially synchronous protocols: "stricter" assumptions about communication which allows instant finality. Even if some nodes do not cast votes, as long as 2/3 honest come to agreement a block can not be reverted even if communication is not synchronous. These protocols are known as BFT protocols.

CAP Theorem

Recall: synchronous protocols can't determine if network is in synchrony (no vote = no vote). Under network partition, network needs a way to resolve which side of the network is "correct". Partially synchronous protocols simply stop making progress if 66% agreement is not reached

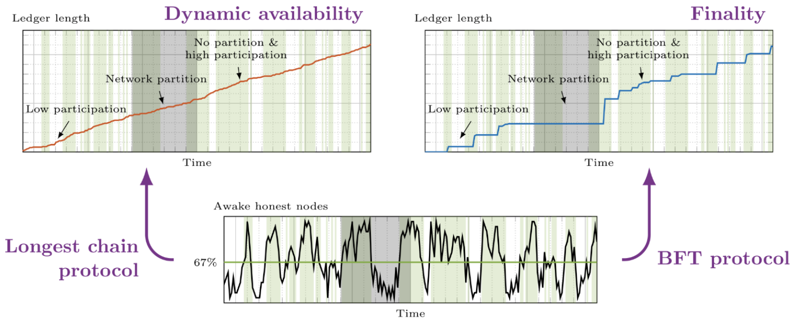
Figure: Network partition



Propose-and-vote style protocols under network partition

Communication model visualized

Figure: Blockchain progress



Nakamoto Consensus

Nakamoto's longest chain protocol took advantage of how synchrony allows for availability under network partitions. Key breakthrough was that voting does not happen via a quorum! Quorum requires $O(n^2)$ communication complexity which doesn't scale well.

Rather, voting happens by spending computing power (proof-of-work). Synchrony naturally allows for forks of the network, this lets nodes cast votes by allocating their computing power to the fork they support! This style of voting is only possible because of the synchrony assumption.

The reason you wait multiple btc blocks for tx confirmation is to increase probability your block was mined under synchrony

Bitcoin's consensus protocol is grounded in the 30+ years of academic work before!

Consensus Overview

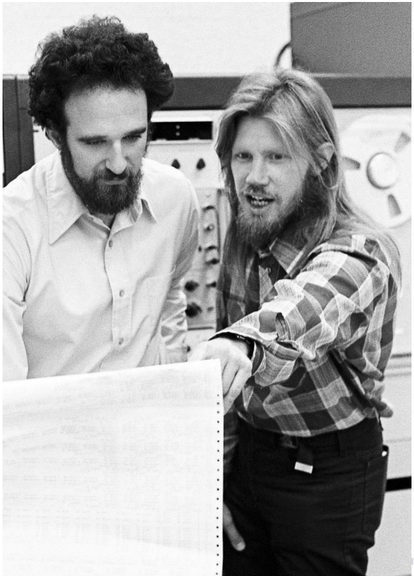
That rounds out what we are covering for consensus

The communication model is one of the most important elements of a consensus protocol, hopefully gave you intuition of where the 51% fault tolerance and reason you have to wait k-deep blocks for tx confirmation comes from

For more advanced topics and people looking to understand protocols in depth I can send more resources!

Cryptography overview

Figure: Diffie & Hellman 1976



Types of cryptography

Symmetric cryptography: What you traditionally think of, used to hide info from eavesdroppers and requires shared key between two parties. Practice of symmetric crypto dates back to ancient egyptians!

Asymmetric cryptography (PKI): Inspired by diffie & hellman in 1976. Requires a public key and private key, can be used to authenticate messages online via digital signatures!

Variations of symmetric and asymmetric crypto: lattice cryptography, PQC, RLWE (basically just different hardness assumptions)

Others: hash functions, quantum crypto (uses photons instead of bits), zero knowledge, etc

Why we care about asymmetric crypto

If you can authenticate messages via digital signatures can create "accounts" for the internet!

Traditionally: used to authenticate email messages, VPNs, etc

For blockchains: lets us create accounts on a blockchain and custody our own cryptocurrencies! Basis for why we don't have to "trust" blockchains in the same way we have to trust traditional client/server models. We can manage our own passwords (private key) and accounts!

Today's agenda

Look at how asymmetric crypto works

If time permits, explain intuition behind ZK proofs (new type of crypto that started a 100 billion dollar industry)!

Figure:

"BENEDICT CUMBERBATCH IS OUTSTANDING"
★★★★
"A SUPERB THRILLER"
★★★★
"THE BEST BRITISH FILM OF THE YEAR"
★★★★★
"AN INSTANT CLASSIC"
★★★★★

Computational complexity theory

All of cryptography (esp. asymmetric crypto) depends on a "hardness assumption"

Q: Can we create a mathematical problem that is extremely inefficient to brute force solutions, yet efficient to verify? (P $=$ NP, sometimes known as a trapdoor one-way function)

This question created an entire subfield of math/computer science called computational complexity theory. Many of the smartest mathematicians work in this field!

Today we are going to look at the *integer factorization* hardness assumption. It is the backbone for the (relatively) simple RSA asymmetric encryption scheme. Most blockchains use ECDSA which is based on the elliptic curve discrete logarithm problem which is much too complicated to cover today.

The integer factorization problem

Remember we want two things: 1) A private key only known to us, that we can use as our password to our blockchain account and sign messages with 2) A public key which others can use to verify our signed messages that reveals nothing about the private key

Diffie and Hellman did not come up with the integer factorization problem (they created a key exchange used in symmetric crypto), but did a lot of the heavy lifting that the problem and RSA is based on

The integer factorization problem

It is very hard to break down the product of two prime numbers back into the original prime numbers if all you have is the product!

Many unique solutions to this problem and brute forcing every combination is a task that is exponentially hard as prime numbers get bigger.

RSA

1. randomly generate two distinct prime numbers p and q

RSA

1. randomly generate two distinct prime numbers p and q
2. take the product of p and q; this is known as n or *modulus*

RSA

1. randomly generate two distinct prime numbers p and q
2. take the product of p and q; this is known as n or *modulus*
3. compute the euler function: $\varphi(n) = (p - 1) * (q - 1)$

RSA

1. randomly generate two distinct prime numbers p and q
2. take the product of p and q; this is known as n or *modulus*
3. compute the euler function: $\varphi(n) = (p - 1) * (q - 1)$
4. choose e such that $1 < e < \varphi(n)$ and e and $\varphi(n)$ are coprime

RSA

1. randomly generate two distinct prime numbers p and q
2. take the product of p and q; this is known as n or *modulus*
3. compute the euler function: $\varphi(n) = (p - 1) * (q - 1)$
4. choose e such that $1 < e < \varphi(n)$ and e and $\varphi(n)$ are coprime
5. Solve for d such that $(d * e)\% \, \varphi(n) = 1$

RSA

1. randomly generate two distinct prime numbers p and q
2. take the product of p and q; this is known as n or *modulus*
3. compute the euler function: $\varphi(n) = (p-1) * (q-1)$
4. choose e such that $1 < e < \varphi(n)$ and e and $\varphi(n)$ are coprime
5. Solve for d such that $(d * e)\% \ \varphi(n) = 1$
6. public key is (e, n)

RSA

1. randomly generate two distinct prime numbers p and q
2. take the product of p and q; this is known as n or *modulus*
3. compute the euler function: $\varphi(n) = (p - 1) * (q - 1)$
4. choose e such that $1 < e < \varphi(n)$ and e and $\varphi(n)$ are coprime
5. Solve for d such that $(d * e)\% \ \varphi(n) = 1$
6. public key is (e, n)
7. private key is (d, n)

RSA

1. randomly generate two distinct prime numbers p and q
2. take the product of p and q; this is known as n or *modulus*
3. compute the euler function: $\varphi(n) = (p - 1) * (q - 1)$
4. choose e such that $1 < e < \varphi(n)$ and e and $\varphi(n)$ are coprime
5. Solve for d such that $(d * e)\%\ \varphi(n) = 1$
6. public key is (e, n)
7. private key is (d, n)
8. signing a message m: $S = m^d\%n$

RSA

1. randomly generate two distinct prime numbers p and q
2. take the product of p and q; this is known as n or *modulus*
3. compute the euler function: $\varphi(n) = (p - 1) * (q - 1)$
4. choose e such that $1 < e < \varphi(n)$ and e and $\varphi(n)$ are coprime
5. Solve for d such that $(d * e)\% \ \varphi(n) = 1$
6. public key is (e, n)
7. private key is (d, n)
8. signing a message m: $S = m^d \% n$
9. verifying signed message S: $m = S^e \% n$

RSA

1. randomly generate two distinct prime numbers p and q
2. take the product of p and q; this is known as n or *modulus*
3. compute the euler function: $\varphi(n) = (p-1) * (q-1)$
4. choose e such that $1 < e < \varphi(n)$ and e and $\varphi(n)$ are coprime
5. Solve for d such that $(d * e)\% \varphi(n) = 1$
6. public key is (e, n)
7. private key is (d, n)
8. signing a message m: $S = m^d \% n$
9. verifying signed message S: $m = S^e \% n$
10. if the signed message to the exponent of e mod n does not result in the original message, the public key does not match the private key and thus the signature is invalid

RSA continued

Demo above showed how to sign messages using a private key and verify a signature using a corresponding public key

Can also use RSA to send someone a message only they can see by encrypting with their public key. This uses a different equation

Some problems: need a way to turn messages into numbers as RSA only works on integers. Typically done by mapping letters to numbers or using hash functions

Asymmetric crypto

If you have ever generated a private key on a blockchain node or exported your key from metamask you will notice it is a number not a pair of numbers

This is because they use ECDSA where a private key is just 1 number instead of two numbers. Concepts are the same

Asymmetric crypto

Hopefully this background gave you an understanding of how we are able to create accounts on a blockchain and custody our own cryptocurrencies without relying on a trusted server to manage our passwords/accounts

Important: This stuff is used all over the internet! Not blockchain specific so it's worth learning the basics of

If time permits

Going to ramble about what ZK proofs are and the intuition behind how they work

1. Minimum number of points to find a polynomial
2. Can encode unlimited information in a polynomial using lagrange interpolation
3. If you can find a way to turn an execution trace into a polynomial, very easy to prove that computation was done without naively re-executing it
4. This is a way to outsource computation to a third party and verify it was done correctly
5. Being used to scale blockchains, make sure that machine learning models are behaving and being trained correctly, also has applications for privacy and proving knowledge of something without revealing what it is you know

Homework

Thanks for listening!

Homework: read the bitcoin whitepaper and the papers it cites (will link in telegram)