**Wk4**

**What is blockchain?**

Blockchain is a distributable, immutable ledger that records transactions and tracks assets in a network; stores data in a way that allows multiple parties to access it reliably without having to trust one another. Cryptocurrency is a form of currency that’s stored completely digitally, and isn’t issued by a central authority; made secure with cryptography, distributed consensus, and economic incentive alignment.

We trust banks with some very sensitive services – transferring and redeeming money; accurately recording account history; storing our personal information. Bitcoin is a fight for privacy and self-governance using cryptography. Bitcoin components are identity, transactions, distributed ledger, trustless consensus.

**Distributed ledger**

Record-keeping by blockchain is data is stored by, and updates are broadcasted to, everyone; transparent and fault-tolerant by design; transactions are bundled into blocks with backlinks to enforce an ordering.

**Trustless consensus**

Motivating question is that now that I’ve recorded my transaction, how does everyone know it’s valid? Consensus is the agreeing on changes to a ledger of transactions. Solution 1 – naïve consensus concept is everyone accepts transactions as they come around without individually verifying that they are legitimate; failure is double spend attack. Solution 2 – democratic consensus concept is have proposers that broadcast transactions, and voters that choose whether or not to include them; failure is sybli attack (a bunch of accounts to vote in one’s favor). Solution 3 – Nakamoto consensus concept is voters do a bunch of pointless brute-force computation to be able to cast a vote.

**Identity**

Public and private keys are each identity with a unique public key, a corresponding private key acts as a key to unlock the public key and your money; unique private key generated randomly, public key derived from private key; public encrypts, private decrypts. Note: 1. No personal information is required. Anonymous? 2. No limit to how many accounts of bitcoins. Affect the security of bitcoin? 3. No restrictions on keys that have been taken. Same private key? 1. What’s the significance of anonymity? The fact that no personal information is included makes bitcoin pseudonymous, anonymity is needed for consumer privacy and fungibility; fungibility allows us to ignore the history of consumers (exchangers won’t know your background); censorship resistant so you can post important information without fear. 2. Public key security. Practically impossible for any two keys to overlap using random generation of addresses.

**Transactions**

Validity: 1. Proof of ownership. 2. Available funds; 3. No other transactions using the same funds. UTXO model is each account holds a set of unspent transaction outputs (UTXOs) – quantities of bitcoin sent to this account that have not been redeemed yet; UTXOs can only be redeemed once. 1. Proof of ownership - UTXOs can be unlocked by a signature generated by a private key. 2. Available funds – aggregate all UTXOs belonging to a public address.

**What is bitcoin?**

Key characteristics of a blockchain are decentralized control, tamper-evidence (it’s immediately obvious if data stored on the blockchain has been tampered with), consensus (one has to provably spend resources to update the blockchain, those resources vary). Can bitcoin be a currency? Durability – persists forever on the blockchain. Portability. Divisibility – exchangeable in arbitrary quantities as little as 0.00000001 BTC. Fungibility – no distinguishing characteristics that would make one bitcoin worth more than another. Limited supply. Acceptability. Summing up bitcoin is identity (use public key to receive money, and private key to redeem it); distributed ledger; trustless consensus (transactions are approved via proof-of-work, an expensive voting process, to deter double spend attacks).

**Wk5**

**The rise of Ethereum**

Bitcoin is coin centric, primary purpose is alternative to existing currency. Ethereum is a turing-complete protocol that uses its coin ether as fuel, primary purpose is platform for decentralized applications and smart contracts. Ethereum powers advanced application logic to replace centralized institutions. Where is Ethereum heading now? surge: begin incorporating a new technology called sharding into the Ethereum blockchain; verge: named after a new cryptographic technology called verkle trees; purge: Ethereum begin purging any unnecessary data clogging the network, reach a speed of 100000 transactions per second; splurge: a splurge of new innovative applications building in Ethereum’s ecosystem.

**Enterprise blockchain**

Permissionless vs permissioned - no central authority vs controlled by one authority or group.

**Blockchain community and politics**

Controversial topics are block size, confirmation times, centralization in third party companies.

Protocol governance is blockchain protocols can be updated through a formal proposal process, bitcoin has bitcoin improvement proposal, bip, system to update the bitcoin core, network participants can create and vote on proposals.

**Where are we now**

Global library, publishing, databank. Scalability is an important requirement in blockchain networks as it refers to the network’s ability for supporting higher transaction throughput (sharding, hard forks, sidechains, lighting network). Zero knowledge proofs, zkp, let you convince me that you know something without revealing to me what your secret thing was.

**Wk6 – What’s so special about bitcoin? How does it work? A technical overview**

**Cryptographic hash functions**

Ensure trust in communication in a trustless environment, used highly in digital signatures. Message to output (digest, tag, hash), any size to 256 bits. Cryptographic hash functions three special properties: computationally efficiently, collision resistance, hide information. Avalanche effect is a small change in the input produces a pseudorandom change in the output. SHA-256 is a cryptographic hash function designed by NSA, bitcoin uses SHA-256^2.

**A temper evident database**

Block header: merkle root, prev block hash, nonce. BlockID=H(blockHeader)=H(merkleRoot || prevBlockHash || nonce); prevBlockHash=H(prevBlockHash || merkleRoot || nonce); bitcoin’s proof of work consensus requires miners to solve a computationally difficult puzzle, hash puzzles need to be: computationally difficult, adjustable, easily verifiable; bitcoin’s partial preimage hash puzzle: a problem with a requirement to find a nonce that satisfies the following inequality: H(prevBlockHash || merkleRoot || nonce) < target, here difficulty is a representation of the expected number of computations required to find a block, which is implemented as requirement of leading numbers of 0s.

**Mining**

1. Download the entire bitcoin blockchain.

2. Verify incoming transactions - listen to bitcoin network for transactions; pending transactions sit in the mempool for a miner to include it in a block.

3. Create a block - gather transactions into a block: choose whichever transactions you want (most transaction fees) & include a coinbase transaction to pay yourself with new bitcoin; get previous block hash and other necessary metadata.

4. Find a valid nonce – find proof of work; expend computational power; increment header nonce first, then coinbase nonce as necessary to change puzzle.

5. Broadcast your block – mining is a competition.

6. Profit.

**Profit**

Mining revenue = block reward + tx fees, mining cost = fixed costs + variable costs.

1. Block reward: miner recerives bitcoin for every confirmed block, currently 6.25 per block, halves every 210000 blocks, bitcoin supply cap is 21000000; reward the honest nodes, proof of work ensures that miners are dedicated to the network (they are willing to pay money for electricity and hardware just to earn bitcoin).

2. Tx fees: motivating question is how are miners incentivized to be honest in their mining? Why isn’t it in their best interest to mine empty blocks? When block reward becomes 0, tx fess will become the only source of revenue for miners; fee estimation algorithm is that input of recent history of txns via algorithm, output is transaction fee; fee bumping is bumping up the transaction of an already broadcast transaction because transaction can get stuck in the mempool because if the fee associated with a transaction is too low, no miner will want to pick it up, which is caused by naïve fee estimation algorithms, a spike in minimum fees, etc; fee bumping solution is RBF, CPFP: RBF, i.e., replace by fee is that user signs a replacement transaction with the same inputs but pays additional fees, the additional fee may come from the change output; CPFP, i.e., child pays for parent is that user creates a new transaction which spends one or more of the outputs of the stuck transaction, attaches a large fee, this works because miners package together ancestor and descendant transactions.

3. Fixed costs: cpu, gpu, fpga, asic – hashes/second increases, time to block(yrs) decreases.

4. Variable costs.

**Real world mining**

Mining pools allow individual miners to combine their computation power together, miners in a pool submit shares (near-valid blocks) to the pool manager (take a cut of the mining rewards), producing shares implies computational power being expended, pool manager pays for valid shares, rewards distributed proportional to # of shares submitted, individual who finds the valid block is not awarded any extra coins and receive same as everyone else. Pool rewards: pay-per-share (pool pays out at every share submitted, more beneficial for miners), proportional (pool pays out when blocks are found, more beneficial for the pool).

**Wk7 - Interacting with bitcoin – wallets and mechanics**

**Types of users in bitcoin**

Full nodes vs. light nodes (SPV nodes, SimplePaymentVerification, is a method for verifying if particular transactions are included in a block without downloading the entire block, keep track of your transactions only, assumption: incoming block headers are not from a false chain, long term chain is probably honest), can’t afford to put entire blockchain on a phone, having a thin client is a decent tradeoff.

**What are wallets?**

With wallets manage all of our keys (to secure our identity, we need to secure our private key). Wallets provide a user interface to the blockchain; keep track of private key; store, send, receive, and list transactions, etc. Hot vs. cold wallets, decentralized exchanges vs. centralized.

**SIGS, ECDSA, AND ADDRESSES**

Being identifiable without identity – intro to digital signature algorithms. Dilemma: when sending transactions to other users, we want 2 seemingly contradictory things to happen: 1. Tie user identity to a transaction; 2. Have no sensitive identifiable characteristics associated with a particular transaction. Solution: public key cryptography, a cryptographic system that allows for secure dissemination of identity and authentication of valid messages. Public key vs. private key – information about a user that can be distributed widely vs. sensitive information about a user that should be only known by the user. ECDSA, EllipticCurveDigitalSignatureAlgorithm, to generate public keys and verify transactions, ECDSA summary: recipients given the (message, signature) pair should be able to verify: 1. Message origin – the origin sender has authorized this message/transaction; 2. Nonrepudiation – the original sender cannot backtrack; 3. Message integrity – the transaction cannot have been modified since sending.

**Create a wallet**

- SWORD: instead of relying on a smart contract with trusted guardians, SWORD distributes shares of an encryption key to a group of shareholders, k of n shares are required to reconstruct the original secret; the encryption key creates an encrypted version of the wallet, whenever a user wants to regain access to their wallet, the shares are reassembled, and the wallet is decrypted.

- Kryptik: network agnostic (use any blockchain you want), quick recovery, multi-factor authentication, predictable fees, no setup cost.

**Wallet mechanics**

**-** Multisig.

- Best practice is to never reuse pseudonyms or addresses (wallet software will handle this), because someone should determine how much bitcoin you own and keys are computationally easy to generate.

- Hierarchical Deterministic Wallets (hierarchical since you can organize the keys in a tree-like structure with levels; deterministic since all child keys are generated from a seed in the same way every time): fewer points of failure - store/backup one master private key, then derive the entire tree of child keys; access control – the tree-like structure allows the owner to allow someone access to only a part of a wallet by delegating them a certain branch. 1. Start with a master private key; 2. Generate child private keys by hashing private key+index; 3. Use the public key of the master private key to generate child public keys; 4. By the nature of the hash functions (same input always means same output), we only need the master key.

- Private key generation: private key on bitcoin is 256-bit unsigned integers; private key on bitcoin is not generated using regular random number generators; chances of two different individual generating the same private key is extremely low: (1/2^256)\*(1/2^256).

- Public key generation: bitcoin uses ECDSA to produce public keys; the elliptic curve is defined by some mathematical function – secp256k1; use a private key as an input, we can generate a public key by performing pint multiplication/elliptic curve scalar multiplication (point multiplication is a trapdoor function, this means calculating the public key is a one-way function).

- Public key to bitcoin address: public key ->(SHA256 -> RIPEMD160, i.e., double hash or HASH160) public key hash -> (BECH32) bitcoin address. SHA256, SecureHashingAlgorithm, used extensively in bitcoin scripts; RIPEMD160, RACEIntegrityPrimitivesEvaluationMessageDigest, produces 160-bit number.

- Conversion summary: (random number generator) -> private key -> (ECDSA) public key -> (Hash Functions, SHA256, RIPEMD160) public key hash -> (Bech32) bitcoin address

**Bitcoin script**

- Bitcoin uses a UTXO model, transactions map inputs to outputs; transactions contain signature or owner of funds; spending bitcoin is redeeming previous transaction outputs.

- Script programming language properties are no loops, stack based, several operators.

- P2PK, Pay to Pub Key: first script type sending directly to connected nodes; P2PKH, Pay to Pub Key Hash: first address type transaction script, starts with 1, introduced the idea of hash in scriptPubKey to hide full Pub Key & UTXO; P2SH, Pay to Script Hash, starts with 3, receiver in charge of sending policy and should pay for increased fee size. Ex: script: <signature><publickey>OP\_DUP OP\_HASH160 <PKHash>OP\_EQUALVERIFY OP\_CHECKSIG.

**Wk8**

**Distributed Systems & Consensus – Trust Without Trust**

**Distributed Systems**

Distributed System is a collection of independent components that share messages with each other in order to achieve common goals. Distributed System characteristics are concurrent components (nodes may perform operations at the same time), individual component failures (individual nodes may fail), no global clock (no method of keeping every node on the same page). Consensus is a general agreement between many parties. Correctness of a distributed system depends on if the system is able to achieve its intended goal, ex, google drive should correctly retrieve your file when requested, a consensus algorithm is used to ensure correctness by achieving – validity (any value decided upon must be proposed by one of the nodes), agreement (all non-faulty processes must agree on the same value), termination (all non-faulty nodes eventually decide). Correctness property is that to prove the correctness of any system, we must prove both safety (bad things will not happen, ex, google drive won’t lose our data even if a server fails) and liveness (good things will happen, ex, google drive must return your data within a reasonable timeframe when requested).

**CAP Theorem**

CAP theorem is the fundamental theorem of distributed systems that states that any system cannot achieve Consistency (every node provides the most recent state, or does not provide a state at all), Availability (you will get a response if you talk to any node, unless it has failed), Partition tolerance (the system works despite being split into groups that can’t communicate with each other). CAP theorem is not black-and-white tradeoffs, but on a spectrum; partition tolerance is almost a given for any system, tradeoff is between consistency and availability.

**Byzantine Fault Tolerance**

“Reliable computer systems must handle malfunctioning components that give conflicting information to different parts of the system. This situation can be expressed abstractly in terms of a group of generals of the Byzantine army camped with their troops around an enemy city. Communicating only by messenger, the generals must agree upon a common battle plan.” A general and their lieutenants are planning to attack a city, everyone is waiting for the general to give an order to attack/retreat, but they can’t trust each other, so the general must issue an order and the lieutenants will share the order amongst each other in an effort to reach honest consensus, here our consensus is correct if – all loyal lieutenants obey the same order; if the general is loyal. 1/3 Bottleneck is that no solution for >= 1/3 traitors. Connection to blockchain is generals->nodes, traitor generals->faulty/malicious nodes, geographic distance->distributed network, attack/retreat->consensus on history(ex, transaction log). PBFT handle f Byzantine faults in a system with 3f+1 nodes.

**Voting-based Consensus**

- Paxos. Consensus protocol named after a legislative system on the island of Paxos; roles are: proposer (advocates for citizens’ requests and tries to turn requests into law by convincing acceptors), acceptor (can cast votes for proposals from proposers), learner (announces the outcome for all citizens of Paxos); how it works: prepare (proposer asks all working acceptors whether anyone has already received a proposal, if not then propose a law to acceptor), accept (if a majority of acceptors agree to this proposal, then the learner reports back to the citizens of Paxos); real world use cases: only works for fail-stop, i.e., no Byzantine failures, faults, many variants of Paxos, generally used to replicate large sets of data.

- Raft. Easy to implement consensus algorithm; roles are: leader (the only node that can interact with the client, ensures all other nodes follow that request), candidate (can ask for votes to become the leader), follower (responds to the candidates or the leader); how it works: a leader is elected at each term here term numbers are maintained by every node, if any follower node times out while waiting for a heartbeat, elect a new leader through majority vote.

- Tendermint. Byzantine fault tolerant, i.e., 1/3 of validators can be malicious and the protocol will still work; Tendermint has two parts: Tendermint Core (the consensus layer, is responsible for sharing blocks and transactions between nodes and establishing an immutable order of transactions, i.e., the blockchain), ABCI (a general application. Interface, allows the specific application logic to be implemented), ex, if we were tryting to implement bitcoin, Tendermint Core would provide the consensus mechanism to ensure that blocks are processed securely and reliably, ABCI would handle the creation, validation, and storage of bitcoin transactions and blocks.

**Resource-based Consensus**

Nakamoto Consensus is set of rules that verifies the authenticity of a blockchain network; proof-of-work consensus mechanism is that nodes support the system with work (computing power, i.e., solving a mathematical puzzle), designed to prevent double spending (spending more bitcoin than a node has); Nakamoto Consensus Process is: elect a leader through a lottery process, leader creates the next block, other nodes agree implicitly by including a block in their chain. Other than work, some other resources that can be spent in consensus are: currency, time, space, reputation; proof-of-stake is that validators instead of miners, locking up stake, resource consumed is native currency; proof-of-stake flavors are: 1. randomly choose a validator based on the proportional stake invested, 2. the chosen validator creates a block, 3. protocol continues forward with no explicit notion of votes, (or, 2. the chosen validator proposes a block, 3. protocol ensure 2/3+ votes or start over), 4. the chosen validator gets the block reward and the transaction fees; proof-of-space is that use disk space to solve challenge, can also use for file storage, resource consumed is storage space.

**Federated Consensus**

In a distributed system a quorum is a set of nodes sufficient to reach agreement, what if you don’t necessarily trust certain nodes in the quorum, how can we still achieve consensus? i.e., how do we choose quorums in a decentralized way? Solution is to introduce quorum slices – subset of a quorum that can convince one particular node of agreement, and individual nodes decide on other participants they trust for information. What happens when multiple quorum slices join together? We get a quorum intersection – quorum slices that come together will slowly convince other quorums slices and form a larger quorum, otherwise we get disjoint quorums that agree on different things. Federated Byzantine Agreement is decentralized control, low latency, flexible trust (nodes choose who they trust, don’t have to trust the entire network).

**Wk9**

**Ethereum & Smart Contracts – Enabling A Decentralized Future**

**Smart Contracts**

Code that facilitates, verifies, or enforces the negotiation or execution of a digital contract, trusted entity must run this code.

**Ethereum**

A decentralized platform designed to run smart contracts; distributed computer to execute code, account-based blockchain, transactions i.e. state transition function; Ethereum has a native asset called ether (the basis of value in the Ethereum ecosystem). Bitcoin vs. Ethereum: bitcoin the gold standard of blockchain vs. Ethereum smart contract blockchain platform; asset is bitcoins vs. asset is ether; simple and robust vs. complex and feature rich; stack-based, primitive scripting languages, not Turing-complete vs. Turing-complete; UTXO-based vs. account-based (bitcoin bob owns private keys to set of UTXOs vs. Ethereum alice owns private keys to an account, UTXOs are easy to make transactions and prevent double spending vs. account is space-efficient to update balances instead of storing UTXOs, as well as easier to look up balance and transfer between accounts when programming). Ethereum account types are: externally owned accounts(owned by some external entity, can send transactions to transfer ether or trigger, contains address and ether balance) and contract accounts (owned by contract, code execution triggered by transactions or function calls i.e. msg, contains address, associated contract code, and persistent storage). Smart contracts in Ethereum are like autonomous agents that live inside of the Ethereum network: react to external world when poked by transactions (which call specific functions); have direct control over internal ether balance and internal contract state; 4 purposes are store and maintain data (data represents something useful to users or other contracts, ex, a token currency or organization’s membership), manage contract or relationship between untrusting users (ex, financial contracts, insurance), provide functions to other contracts (serving as a software library), and complex authentication.

**EVM**

EVM Compilation and Process is: solidity contract -> solidity compiler + vyper contract -> vyper compiler -> EVM Code. EVM is a mini computer on computer that runs contract code, contract code that actually gets executed on every node is EVM code (low-level, stack based bytecode language, i.e., JVM bytecode), every Ethereum node runs EVM. What if contract has an infinite loop? Every node on the network will get stuck executing the loop forever, by the halting problem, it is impossible to determine ahead of time whether the contract will ever terminate, which leads to Denial of Service Attack. Solution is: every contract requires gas, which fuels contract execution; every EVM operation-code requires some gas in order to execute; every transaction specifies startgas (max quantity of gas it is willing to consume) and gasprice (fee in ether it is willing to pay per unit gas); at the start of the transaction, startgas\*gasprice are subtracted from the sender’s account, if contract successfully executes, the remaining gas is refunded to the sender, if contract execution runs out of gas before it finishes then execution reverts and startgas\*gasprice are not refunded; purchasing gas i.e. purchasing distributed trustless computational power; an attacker looking to launch a DoS Attack will need to supply enough ether to fund the attack. Ethereum network state: state transition function – (block\_state, gas, memory, transaction, message, code, stack, pc) –(EVM)> (block\_state’, gas’). Ethereum conclusions are: Ethereum is not about optimizing efficiency of computation; its parallel processing is redundantly parallel, way to reach consensus on the system state without needing trusted third parties; contract executions are redundantly replicated across nodes, expensive slow and memory-intensive, but creates an incentive not to use the blockchain for computation that can be done off chain.

**Mining with Proof of Stake**

- Consensus mechanism where nodes put down a stake of native currency as collateral; lower barrier to entry, less energy consumption; deposit currency to become a validator and earn rewards; here, validator is a node responsible for confirming and producing blocks (similar role to miners on bitcoin, i,e. nodes have an execution, consensus, and validator client, validators chosen at random to make/propose blocks), validators no special hardware needed, just 32 ETH and a computer, earn rewards from gas fees by fulfilling responsibilities, if you act dishonestly you lose money.

- Creating new blocks: download the Ethereum blockchain and clients (don’t need to download everything from start, stay up to date), verify and execute incoming transactions (as users make transactions nodes verify and execute them with execution client, if valid add to mempool), propose the next valid block (for every slot one node is randomly chosen to propose the next block, proposer executes bundle of transaction to generate a state change), broadcast your block (each slot a randomly chosen committee of validators votes on the proposed block, the block proposer broadcasts a beacon block for the committee to re-execute and verify), profit or burn (if new block passes vote and gets finalized participating validators get rewarded, otherwise if the proposer made a mistake or tampered with transactions their stake gets burned).

- Implications of proof of stake: nodes have a financial incentive to be honest (more malicious validators = higher penalty), since block proposer is randomly chosen, there is no race to find next valid block (less computational power is wasted, time between blocks can be fixed).

**Wk10 Game Theory and Attacks – Vulnerabilities of Blockchain**

**Bitcoin Attacks**

- Forking and double spends. Hard fork and soft fork. Double spending is successfully spending the same value more than once, by race attack – suppose bob simply checks that the transaction he sees is valid and immediately sends alice the iphone, bob is vulnerable to a race attack, solution is confirmations (the number of blocks created on top of the block a transaction is in). 51% attack is what if alice controls more than 50% of the total network hash power? Whenever alice’s chain is behind the honest network’s chain, she will always be able to catch up and out-produce the honest miners, the probability is 100%. If the rest of the network detects the double spend, it is assumed that confidence in the cryptography and exchange rate would plummet; alice might not physically control the mining hardware to perform a double spend, instead alice can bribe miners or even entire pools to mine on her withheld chain.

- Censorship. Say alice is a government that has jurisdiction over mining pools, and alice’s mining pools control over 51% of the network’s hashrate, objective is to censor the bitcoin address owned by bob, and prevent them from spending any of their bitcoin. Naïve strategy is blacklisting: alice tells her pools not to include bob’s transactions, doesn’t work unless 100% of the network, other miners will eventually include bob’s transactions in a block, can only cause delays and inconveniencies; second strategy is punitive forking: alice tells her pools not to mine on top of any chain containing bob’s transactions and announces this to the world, since alice controls over 51% of hashrate, so in time her pools will mine the longest chain, invalidating bob’s transactions and those in subsequent blocks, miners outside of alice’s pools will stop including bob’s transactions when mining blocks, since alice announced she would invalidate their blocks and thus mining rewards, thus, we’ve shown that an entity with over 51% of the network’s hashrate can prevent someone from accessing their funds

- Selfish mining. Miner have just found a block, instead of announcing block to the network and receiving reward keep it secret by trying to find two blocks in a row before the network finds the next one, if you succeed in finding a second block you have fooled the network, network still believes it is mining on the longest proof of work chain, you continue to mine on your own chain; if the network finds a block you broadcast, your two secret blocks and make the network block invalid, while network was working on the invalid block you got a bunch of time to mine by yourself for free, i.e., higher effective proportion of hashrate, higher expected profits; but what if the network found their new block before you could find a second one, race to propagate, i.e., if on average you manage to tell 50% of the network about your block first, malicious strategy is more profitable if you have over 25% mining power, else if you have over 33% mining power you can lose the race every time and malicious strategy is still more profitable.

**Ethereum Attacks**

- Censorship. Censorship in a PoS system would involve a validator or group of validators refusing to include certain transactions in a block; validators might do this due to various reasons such as political motivations, personal bias, or to comply with regulatory pressures. Censorship entails omission of transactions and ignoring requests, possible motivations for censorship are regulatory compliance, bias or prejudice, network manipulation. While PoS has mechanisms to deter censorship, it’s not completely immune, an entity with a significant stake could theoretically attempt to censor transactions, but they would risk losing their staked assets and potential rewards, and they’d likely face backlash from the network and community, making such attempts costly and less attractive.

- Double-signing attack. Malicious activity where a validator signs two different blocks at the same height or the same slot, in doing so they aim to have their alternate chain accepted by the network as the legitimate one, thereby invalidating the transactions on the original chain. Motivations are financial gain (short-term profits, exploiting rewards), disruption and sabotage (network instability, competitive sabotage), and exploiting for other attacks (creating vulnerabilities, facilitating additional malicious activities). Preventions are slashing, cryptographic commitments, randomized validator selection, validator churn; detections are whistleblowing protocols, advanced monitoring tools, open source development and peer review.

- Ro-org attack. An event where an attacker, who possesses a significant amount of stacked cryptocurrency, tries to create a separate blockchain history or fork, in doing so they aim to have their alternative chain accepted by the network as the legitimate one, thereby invalidating the transactions on the original chain. How could it happen in Ethereum PoS? The attacker accumulates a significant amount of ETH to gain substantial staking power, the malicious actor might then use this staking power to influence the blockchain’s validation process, in doing so they aim to have their alternative chain accepted by the network as the legitimate one, thereby invalidating the transactions on the original chain, if the attacker’s chain becomes accepted by the network it can cause the blockchain to reorganize, adopting the attacker’s fraudulent chain. To prevent long-range attacks, probabilistic on Ethereum uses heaviest chain to finality every epoch

**MEV**

Maximum Extractable Value represents excess value captured by miners from users in a cryptocurrency network, this excess value often comes from reordering users’ transactions to maximize fees or inserting new transactions that allow a miner to front-run users’ transactions. Unlike bitcoin, which has a limited set of use cases, Ethereum’s theoretically unbounded complexity causes MEV to grow exponentially; validator revenue streams from MEV can prove detrimental to the security of the consensus mechanism. MEV’s impact are: amount of MEV on Ethereum, flow of MEV (user->wallet->searcher->builder->validator), sandwich attack, proposer-builder separation (split the block construction role from the block proposal role, goal is to separate strong centralizing pressure of selecting transactions for block inclusion from the ordering of transactions within the block, preventing Exclusive-Order Flow, i.e., small set of builders with an oligopoly can fix prices and pay validators less than they would in a competitive market). Future of MEV are: OFAC Compliance, user privacy (TEEs, Trusted Execution Environments, integrity and privacy guarantees; Threshold Decryption, validator set generates shared encryption key, can only decrypt with 2/3 of validator set, keeps transactions private which prevents frontrunning and sandwiching), endgame of MEV.

**Wk11**

**What is Defi?**

Defi is the movement to build an open, global, and interconnected financial system without the drawbacks in traditional finance we’ve discussed; often build on blockchain platforms that offer security guarantees; continuously developing that has so much room for growth. The Decentralized Finance Stack: aggregation layer, application layer, protocol layer (exchanges, loans, derivatives, asset mgmt., etc.), asset layer (native protocol asset for ETH, fungible token like ERC-20, non-fungible token like ERC-721, etc.), settlement layer (native protocol asset for ETH, Blockchain/Ethereum). 5 pillars of Defi: transparency (smart contract code is open-source and publicly verifiable), experimentation (write code, fork projects, deploy quickly), accessibility (borderless system accessible to anyone with internet connection), composability (interoperability between on-chain platforms), permissibility (open financial platforms).

**Stablecoins**

- The goal is that a cryptocurrency that maintains a stable value; many types of stablecoins are: reserve-backed often sort of centralized stablecoins (USDT, USDC), collateral-backed decentralized stablecoins (DAI, FEI), partially-collateralized decentralized stablecoins (FRAX), algorithmic decentralized stablecoins i.e. uncollateralized (UST).

- USDC. The premise is that deposit fiat currency into a bank and get USDC in return, problem is centralization.

- DAI. The premise is that use volatile cryptocurrency to get loaned stable crypto, but how? By putting in more volatile crypto than stable, thus, collateralization rate > 1.0, but why should I put more, there’s real value in having stable currency, enough so to be worth it. All you have to do is pay back DAI borrowed and get your ETH, note: volatility always exists, what if your underlying ETH loses value and is worth less than the DAI generated? Liquidate, DAI contracts do everything they can to get DAI back, i.e., sell your collateral (which they own) to others for DAI, that means you lose most of your collateral, further fail safes exist, but for now that’s it.

- The uncollateralized stablecoin. The premise is that no hard collateral to maintain stablecoin peg, but how? By minting and burning governance token to keep peg closed to $1; the issues: UST needs to have traction, otherwise a death spiral can occur for the coin, i.e., need significant traction and use cases for the stablecoin otherwise the system falls apart.

**Decentralized Exchanges**

DEXes come in all shapes and sizes but we’ll choose 2 types: Order Book “the classic”, Automated Market Maker “the upstart”. AMMs with liquidity pools: rather than match orders 1:1, why not use a pool of liquidity; handles low liquidity situations beautifully, and quite adaptable at that, quite useful in the diverse and rapid world of cypto; Uniswap by using a Constant Product Function (impermanent loss: user deposits liquidity into a pool, time passes and bananas grow in value in the world, the lending pool now has fewer bananas relative to apples reflecting a higher banana price, the user withdraws their liquidity having made some money, however they would have made more money if they had just held into their bananas instead of depositing them since they withdrew fewer bananas than they originally deposited); Uniswap isn’t perfection, diversity in the DEX space is quite insane.

**Lending markets**

Purpose is to accrue interests by locking up tokens in smart contracts, issue permissionless P2P loans; reason is that trustless way to earn interest and get access to market efficient rates on a central point of failure; value is the access to interest on assets with personal collateral without sacrificing your identity or financial history. How Defi lending works: each market has suppliers and borrowers, revolve around interest rates (set by function of Utilization - % being loaned out currently, compound uses a linear model), borrowers pay interest on their crypto loans which is distributed to suppliers (to borrow must over-collateralize first), supply rate < borrow rate, because amount of supplied > amount of borrowed, and you have to distribute paid interest among all suppliers. Diversity in lending/borrowing platforms: Compound, AAVE, Rari Capital’s Fuse.