## Lecture #3: Mechanics of Bitcoin

***Recap: Bitcoin consensus***

Bitcoin consensus gives us:

- Append-only ledger
- Decentralized consensus
- Miners to validate transactions

***Lecture 3.1 Bitcoin Transactions***

An account-based ledger (not Bitcoin)
A transaction-based ledger (Bitcoin)

***Lecture 3.2 Bitcoin Scripts***

**Bitcoin scripting language('Script')**

*Design goals*

- Built for Bitcoin (inspired by Forth)
- Simple, compact
- Support for cryptography
- Stack-based
- Limits on time/memory
- No looping

*Bitcoin script instructions*

256 opcodes total (15 disabled, 75 reserved)

- Arithmetic
- If/then
- Logic/data handling
- Crypto!
  - Hases
  - Signature Verification
  - Multi-signature verification

Should senders specify scripts?
Idea: use the hash of redemption script

***Lecture 3.3 Applications of Bitcoin scripts***

*Example 1: Escrow transactions*
*Example 2: Green addresses*
*Example 3: Efficient micro-payment*

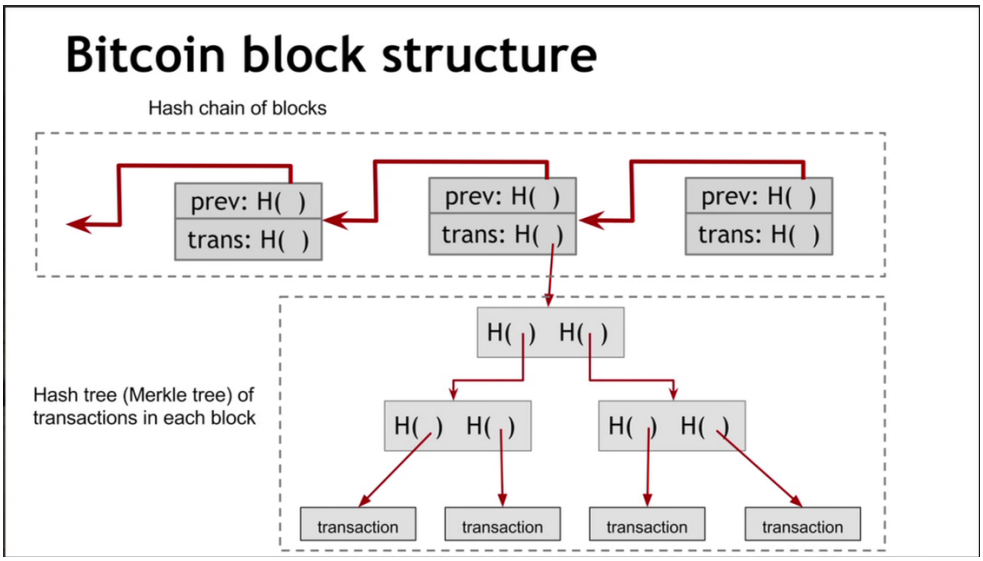
More Examples:
• Multiplayer lotteries
• Hash pre-image challenge
• Coin-swapping protocols
"Smart contract"

***Lecture 3.4 Bitcoin blocks***

Why bundle transactions together?

- Single unit of work for miners

- Limit length of hash-chain of blocks
  - Faster to verify history




***Lecture 3.5 Bitcoin network***

*Bitcoin P2P network*

- Ad-hoc protocol (runs on TCP port 8333)
- Ad-hoc network with random topology
- All nodes are equal
- New nodes can join at any time
- Forget non-responding nodes after 3 hr

**Transaction propagation**

- Transaction valid with current blockchain
- (default) script matches a whitelist - avoid unusual scripts
- Haven't seen before - avoid infinite loops
- Doesn't conflict with others I've relayed - avoid double-spends

**Race conditions**

Transactions or blocks may conflict

- Default behavior: accept what you hear first
- Network position matters
- Miners may implement other logic

**Block propagation** nearly identical

Relay a new block when you hear it if:

- Block meets the hash target
- Block has all valid transactions
  - Run all scripts, even if you wouldn't relay
- Block builds on current longest chain - avoid forks

**How big is the network?**

- Impossible to measure exactly
- Estimates-up to 1M IP address/month
- Only about 5-10k ‘fully-validating-node'
    - Permanently connected
    - Store entire block chain (20GB 3 years ago)
    - Hear and forward every node/transaction
- This number may be dropping

**Tracking the UTXO set**

- Unspent Transaction Output

**Thin/SPV clients (not fully-validating)**
**Idea: don’t store everything**
• Store block headers only
• Request transactions as needed
  • To verity incoming payment
• Trust fully-validating nodes
1000x cost saving!!! (20GB - 23MB)

Background Knowledge:
About 90% if nodes run “Core Bitcoin” (c++)
BitcoinJ (Java), Libbitcoin(c++), bcd(Go)

***Lecture 3.6 Limitations & improvements***
**Hard-coded limits in Bitcoin**

- 10min. Average creation time per block
- 1M bytes in a block
- 20,000 signature operations per block
- 100M satoshi per bitcoin
- 23M total bitcoins maximum
- 50, 25, 12.5 .. bitcoin mining reward

**Throughput limits in Bitcoin**

- 1 M bytes/block(10 min)
- > 250 bytes/transaction
- 7 transactions/sec

**Cryptographic limits in Bitcoin**

- Only 1 signature algorithm(ECDSA/P256)
- Hard-coded hash functions

**"Hard-forking" changes to Bitcoin**
*PROBLEM*: Old nodes will never catch up

*Changes require hard-fork: (Currently seem very unlikely to happen)*

- New op code
- Changes to size limits
- Changes to mining rate
- Many small bug fixes - eg. Multi-sig bug

**Soft-fork**
*Observation:* we can add new features which only limit the set of valid transactions

Need majority of nodes to enforce new rules
Old nodes will approve
*RISK:* Old nodes might mine now-invalid blocks

*New Possibilities*

- New signature schemes
- Extra per-block metadata
    - Shove in the coinable parameter
    - Commit to UTXO tree in each block