ECE 455: CYBERSECURITY

Lecture #7

Daniel Gitzel

Announcements

- **Read paper for quiz next week.**
- **Guest speaker next week.**
- **Continue work final project.**
  - **Get work done early!**
  - Project check-in tonight (during break)
  - Lab 1.5 will be distributed (cryptography)

BITCOIN / BLOCKCHAINS

Introduction

- **Cryptocurrency**
- **Ideals / Principles**
- **Protocal (blockchain)**
- **Problems and Attacks**
- **Extensions (programmable money)**
  - Ethereum
  - Smart Contracts
  - DeFi

Replacing Fiat and Central Banks

- **Basic requirements for a banking system:**
  - Identity management
  - Transactions
  - Prevent double spending
- **Can these be enforced cryptographically?**

Identity

- **How can we give a person a cryptographic identity?**

Identity

- Each user has a Public Key and Secret Key
- User referred to by PK (address)
- User users SK to sign transactions

Transactions

- **How can Alice transfer bitcoin to Bob?**
  - Alice signs transaction using her S_{KA}
  - sign S_{KA} (A transfers to B)
- **How anyone can check Alice's transaction?**
  - Assume Alice can put this signature on a public ledger (a public bulleting board anyone can see)
- **Problems?**
  - Alice can spend more money than she has. She can sign as much as she wants.
- **Ideas how do we solve this?**

Transactions

- **Include only *correct* transactions in the ledger**
  - Assumes a trustworthy ledger owner
- **How would you prevent double spending**
  - Assume all signatures/transactions are sorted in order
  - And include previous transactions

Transactions

- **How does the ledger owner check a transaction?**
  - TX = (sender $\longrightarrow$ receiver ; amount X; prior transactions L)
  - The signature on TX verifies with the PK of the sender
  - Checks sender had X bitcoins: the transactions in L had a total output for sender of Y
  - $Y \geq X$
  - All future transactions using money from any of the transactions in L did not spend more than Y - X

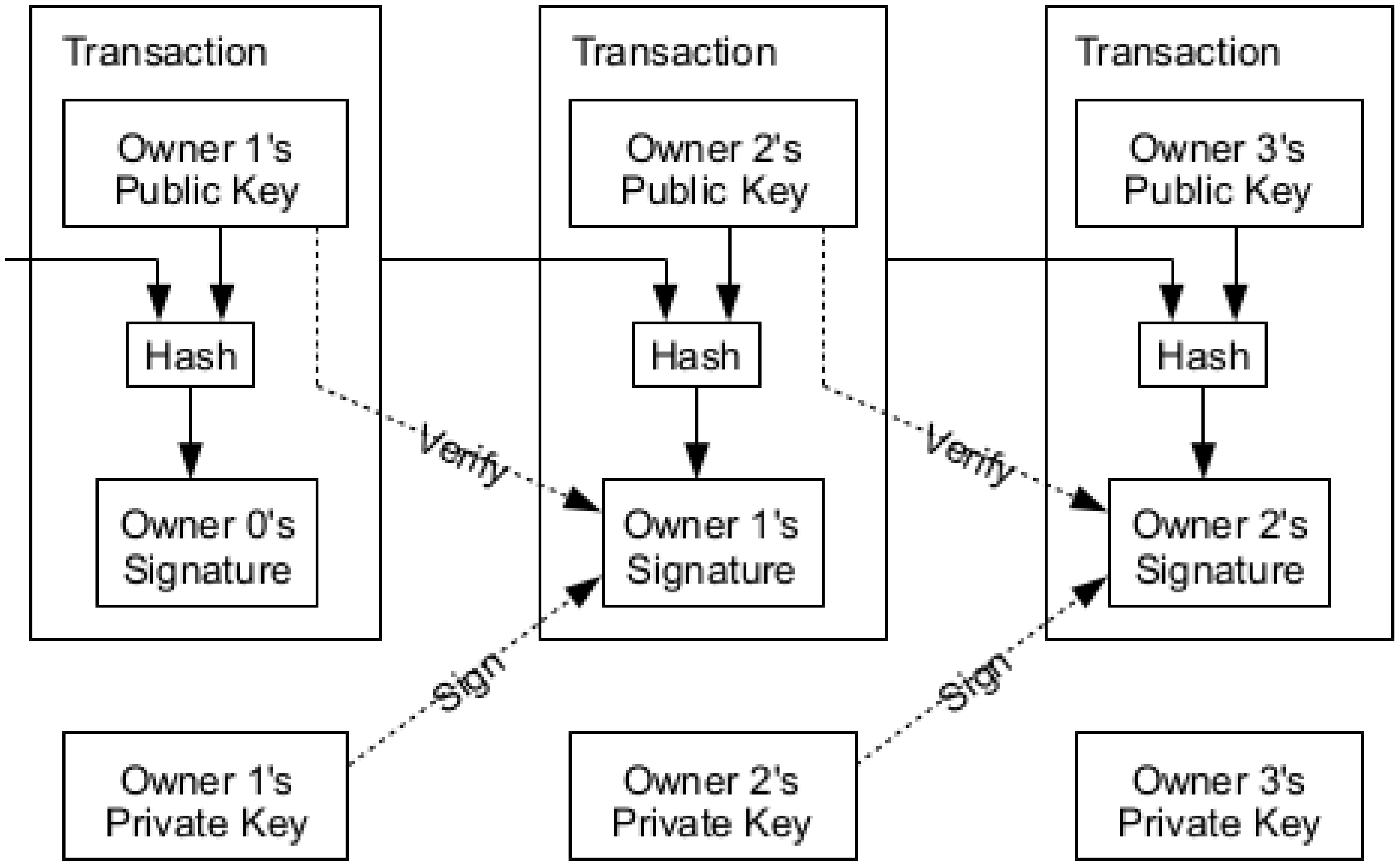
The Ledger

- **But we don’t have a trustworthy public ledger.**
  - Solution: blockchain + proof of work

Blockchain

- **Chain transactions using their hashes**
  - Each transaction contains hash of previous transaction
  - (which contains the hash of its own previous transaction, and so on)
- **Verification**
  - Given a hash function h(·)
  - Fetch blocks 1 … n from an untrusted source
  - Recompute h(1) ... h(n) and confirm no block as mutated, added or dropped

Blockchain Dissection



Blockchain

- **Why can’t an attack work?**
  - Modifying, adding, or dropping a block changes the hash
  - Hashes are propagated forward since they are part of each block

Ledger

- **Building the ledger. Assume:**
  - Every participant in Bitcoin stores a copy of the entire blockchain
- **Process:**
  - Someone creates a new transaction,
  - Broadcast the transaction to everyone
  - Every node checks the transaction
  - If it is correct, create a new block including this transaction

    Add it to its local blockchain
- **Problem?**
  - Node can choose to truncate blockchain or not include certain transactions

Consensus

- **Mallory can fork the hash chain**
  - She submits a new transaction to Bob
  - She finds an older block
  - She starts appending new entries from there.
  - If she gets others to accept this forked chain; she gets her money back.

Mining

- **Miners add to the block chain**
- **All miners try to solve a proof of work**
  - Hash of the new block must start with 33 zero bits
  - Can include a nonce in the block and increment that so the hash changes until the proof of work is solved
- **Once a miner solves a block, it is broadcast**

Consensus

- Consensus: longest correct chain wins
- Everyone checks all blocks and all transactions.
- Incorrect transactions -> the block is ignored
- Assumes most miners are honest

Consensus

- **“Longest chain” wins**
  - What if two different parts of network have different hash chains?
  - Whichever is “longer” wins; the other is discarded

Consensus

- **Can Mallory fork the block chain?**
  - Longest chain wins, and her forked one will be shorter
  - If she has >50% of the computing power in the chain:
  - She can mine new entries faster than aggregate mining power of everyone else in the world
  - And takeover the ledger

How can we convince miners to work?

- **Reward to anyone who successfully appends**
  - Essentially they may include a transaction from no one to their PK
- **This is called the “coinbase”**

Halving

- **Rewards change over time (halving)**
  - After a certain number of successful blocks are added to the blockchain the reward is cut by 50%
  - This is known as halving.
  - Halving occurs in bitcoin after every 210000 mined blocks
  - For bitcoin the rewards for every successful block were 50BTC per block, then 25BTC, then 12.5 BTC, etc.
  - This reward is paid out in the coinbase transaction

Thoughts on Consensus

- **What if Miner A and Miner B solve at the same time?**
  - This would fork the ledger
  - The next miner that appends onto one of these chains, invalidates the other chain.
  - Longest chain wins.
- **What happens if Miner Mallory discards the last few blocks in the block chain and miners from there?**
  - Unless Mallory has >50% of the computation power, she will not be successful
  - The combined power of the other miners will outpace her
- **If a miner included your transaction in the latest block, are you guaranteed that your transaction is on the blockchain?**
  - No, another miner could've appended a different block at the same time
  - That chain might be the new longest
  - Wait for a few blocks, e.g. 5 until your transaction is committed with high probability

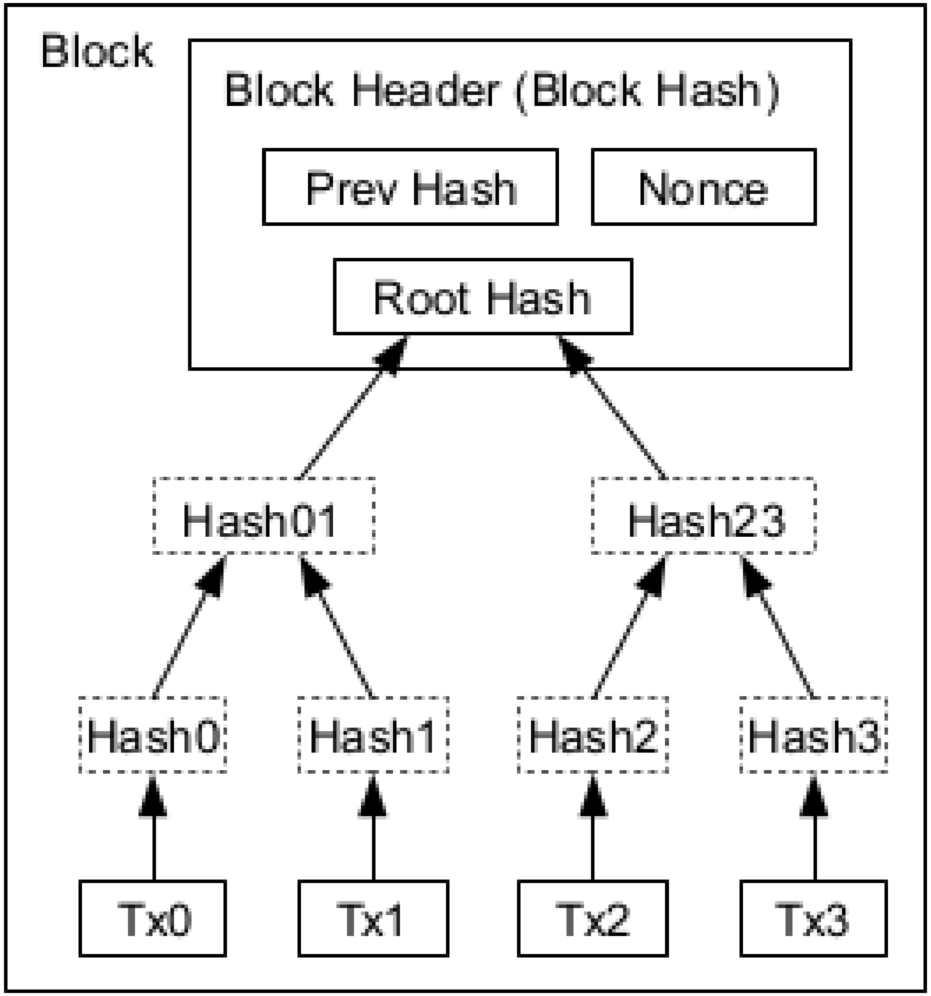
More Thoughts

- **What if a miner refuses to include my transaction?**
  - Hopefully the next miner will not refuse.
  - Each transaction can also include a fee which goes to the miner
  - A miner can pick and choose higher fee transactions first
  - Unincluded transactions live in the “mempool”
  - The size of the pool and transactions fees mirror usage on the network (and provide a health-check)
  - Checkout: https://www.blockchain.com/explorer

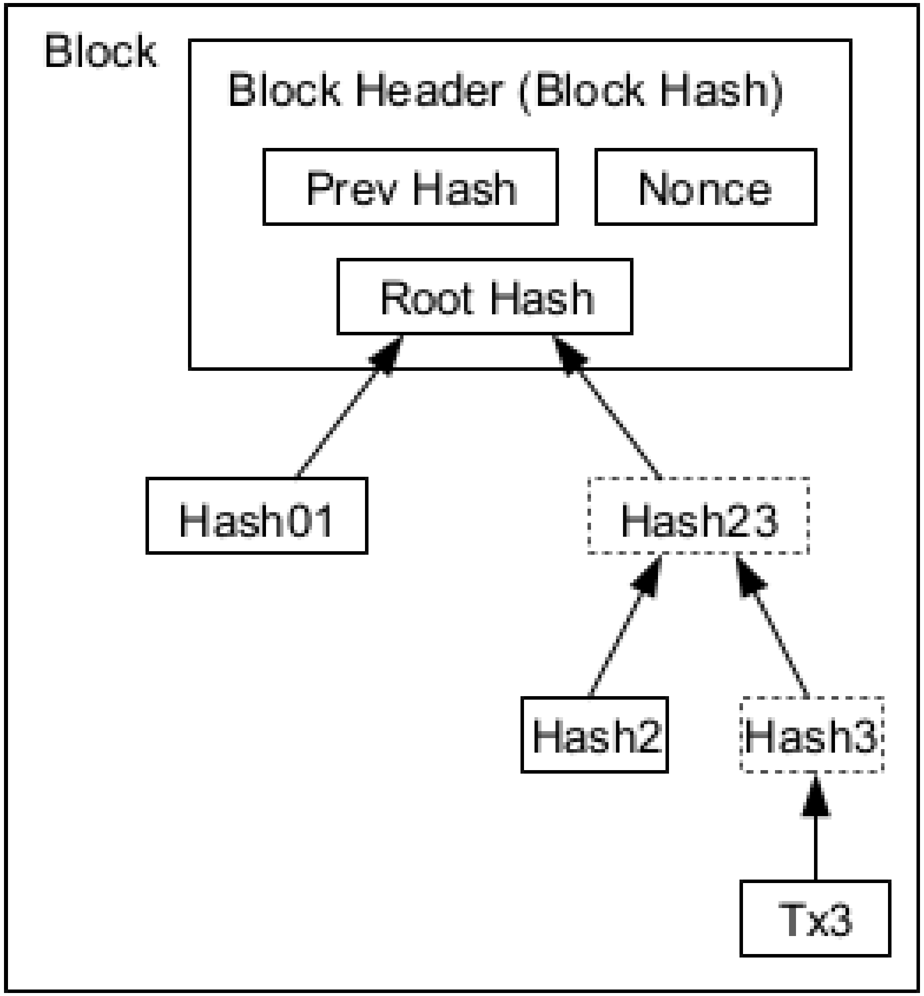
Mining Pools

- **Mining was easy in early days (CPU/GPU)**
- **Nowadays you need too much compute (ASIC)**
- **Pool resources**
  - Contribute cycles to a pool: a group of many machines
  - Receive a predictable income based on the combined mining power of the group
  - Remember evening with a optimized hardward mining is probablistic (you are searching for the right nonce)

Saving Disk Space



Transactions Hashed in a Merkle Tree



After Pruning Tx0-2 from the Block

Is Bitcoin anonymous?

- **No.**
  - All transacations are public.
  - All transactions linked to a public key.
  - Only one step is needed to identify any PK.

Points for Discussion

- **Why is Bitcoin popular?**
  - First mover? Post-crisis fear of banks?
- **How can Alice turn dollars into bitcoins, or vice versa?**
  - Exchanges, and fiat bridges. Regulation and control thereof.
- **Is it ethical to build a system that relies upon massive energy consumption?**
  - Proof-of-stake. Pre-mined coins. Research into alternatives for consensus and security.

Hardness scales

- **Mining frequency (aka blocktime) is ~10 mins**
  - If it takes too long to mine on average, make the proof of work easier (less zeros), else make it harder (more zeros)
- **What is the economic incentive?**
  - Mining is slow, give more incentives to join the network

But How Does It Actually Work?

- **We’ve discussed**
  - Blockchain
  - Ledger and storage
  - Mining
- **We’ve left out**
  - Broadcasting
  - Node operations (how to run a node)
  - Network operations (joining/leaving)
  - Wallets (end user applications)

Network Protocol

- **Broadcast network to propagate transactions and blocks**
- **Communication over TCP (port 8333)**
  - Able to use ports other than 8333 via the -port parameter.
  - IPv6 is supported with Bitcoind/Bitcoin-Qt v0.7.
- **Bitcoin over tor is also supported.**
- **What problems do you see here?**

https://en.bitcoin.it/wiki/Network

Connecting

- **Handshake**
  - Send (version, block_count, current_time)
  - Receive *verack* if version is supported by peer (contains peer’s version)
  - Send *verack* if you support peer’s version
- **Fetch timestamps from peers**
  - The median time amoung peers is used for all purposes expect *version* messages (to connect)
- **Exchange addresses**
  - Send *addr* and *getaddr* to update your list of known addresses

Relaying

- **A new transaction is sent in an *inv* message to all peers**
- **The peers will then *getdata* to request the full transaction**
  - This is verified by the peers
  - If valid, each peer will further broadcast to their peers
  - Peers do not rebroadcast transactions they already know
- **Uncommited transactions live in the "mempool"**
  - This is eventually cleared, so the sender must rebroadcast

Relaying (cont.)

- **Miners will collect received transactions and work on including them in a block**
- **When a new block is found, the miner sends an *inv* containing it to all their peers**

Heartbeat

- **Everyone broadcasts an addr containing their own IP address every 24 hours.**
  - Nodes relay these messages to their peers and store the address if it's new to them.
  - After connecting, you get added to everyone's address database because of your initial addr.
- **Network alerts are broadcast with alert messages.**
  - No inv-like system is used; these contain the entire alert.
  - If a received alert is valid (signed by one of the people with the private key), it is relayed to all peers.
  - For as long as an alert is still in effect, it is rebroadcast at the start of every new connection.