Lecture 5 - Engineering Details and Applications

Rylan Schaeffer and Vincent Yang

May 3, 2016

Note: This lecture is based on Princeton University's BTC-Tech: Bitcoin and Cryptocurrency Technologies Spring 2015 course.

4, 9

1 Transactions

- Transactions are not account-based because of accounting required i.e. need to track transactions from genesis to current day
- Easier to fragment transaction into inputs and outputs. Refer to specific coins in past transactions e.g.
  1. Inputs: $\emptyset$; Outputs: 25.0 $\rightarrow$ Alice; Signed: Miner (usually Alice)
  2. Inputs: 1[0]; Outputs: 17 $\rightarrow$ Bob, 8 $\rightarrow$ Alice; Signed: Alice
  3. Inputs: 2[0]; Outputs: 8 $\rightarrow$ Carol, 9 $\rightarrow$ Bob; Signed: Bob
  4. Inputs: 2[1]; Outputs: 6 $\rightarrow$ David; 2.0 $\rightarrow$ Alice; Signed: Alice
- Note that a transaction always fully consumes an input. This prevents needing to track the complete history of a transaction.
- Consolidate funds: easy since multiple inputs is permissible
- Joint payments: easy since multiple inputs is permissible. Need multiple signatures.

2 Scripts

- Each transaction input, output actually specifies a script for the Bitcoin scripting language, not a public key or public keys
- Bitcoin scripting language designed specifically for Bitcoin. Imperative (statements used to change program state, as opposed to declarative), stack-based (programming language
- Every instruction is executed exactly once, in linear manner
- Only 256 instructions, 15 currently disabled, 75 reserved
- Proof of Burn: a script that can never be redeemed. Intentionally add an error to the script, invalidating the data in the script.
- Pay-to-Script-Hash: Instead of having sender specify the entire script, sender can specify hash of script that will be needed to redeem those coins. Contrast with normal mode, Pay-To-Public-Key.

3 Smart Contracts

- Applications are frequently called smart contracts because they are contracts enforced through technical means (as opposed to laws)
- Escrow Transactions: Alice wants goods, Bob wants money. Both waiting for other to send theirs first → deadlock. Solution: Alice and Bob find trusted third party Judy. Alice creates transaction, specifies that coins can be spent if 2 of 3 sign. Once accepted in the blockchain, Bob sends the goods to Alice. If Alice and Bob are both honest, Judy has to do nothing.
- Green Addresses: Alice wants to pay Bob, Bob is offline. Solution: Alice and Bob find trusted third party. Alice pays third party. Third party confirms transaction, lets Bob know. The third party's temporary address is called a "green address." Two most prominent previous online services, Instawallet and Mt. Gox, collapsed.
- Efficient Micropayments: Alice wants to continually pay Bob small amounts of money for some service Bob provides e.g. cellular minutes. Can't do transaction per minute because too many, transaction fees add up, takes too long to verify. Solution: Alice sends maximum amount to MULTISIG transaction (think of this as holding account), which requires both Alice and Bob to sign to get money out of. For each minute, Alice signs a transaction giving Bob one more share, and herself the rest. Bob doesn't publish or any of these until Alice is done. Bob discards all but the last. To ensure Bob signs MULTISIG transaction, they start by both signing a transaction which refunds Alice's money back to her after some elapsed time.
- Lock time: Alice and bob both sign a transaction which refunds all of Alice's money, but the refund is locked until some time in the future. Bob signs. New transaction is activated in the future at time t, unless Bob publishes one of the micropayment transactions. Problem solved.

4 Engineering Details

- Types of Nodes

  Fully-validating Nodes: to validate entire blocks, not just transactions, must store entire blockchain. Permanently connected. Requires 10s of GBs of storage. Must maintain entire set of unspent transaction outputs i.e. coins available to be stored in RAM

  Lightweight Nodes (thin clients, Simple Payment Verification): Only store pieces of blockchain necessary to verify specific transactions. Only 10s of MB.
- Storage: storing and managing secret keys

  Tradeoff between availability, convenience and security

  Hot Storage (Wallets): software to track your coins and handle details of spending. Can generate key and share as base58 (set of 58 characters = upper case + lower case + digits - confusing digits e.g. O and 0) or QR code

  Cold Storage: offline, archival, more secure but less convenient. Can send coins to cold storage, since address is known by you.
- Hierarchical Wallets: Want to be able to transfer from hot to cold, using new address each time (perhaps because of privacy). Instead of address and private key, we generate "address generation info" and "key generation info." These are used to generate sequences of addresses and corresponding keys. Bitcoin's Elliptic Curve Digital Signature Algorithm has this capability.
- Brain Wallet: Remember passphrase. Use some algorithm that maps passphrase to public and secret keys.
- Paper Wallet: Print key material to paper, carry it around.

5 Splitting and Sharing Keys

- Idea: split key to prevent single point of failure. Split key up into N pieces. If K pieces are brought together, can reconstruct the original secret, but less than K reveals nothing.
- Suppose K=2. Choose point (0,S) on Y-axis, choose random slope. Draw line with slope through that point. Generate lots of points on the line: (0,S), (1, S+R), (2, S+2R), etc. To make work, we do arithmetic modulo large prime P. Distribute to people. Two points is sufficient.

  Can increase K using higher order polynomials e.g. $(S + R_1X + R_2X^2) \bmod P$.
- Problem: Still requires keys being brought together, a point of vulnerability. Can use threshold signatures.
- Alternative: Multi-signatures. Instead of splitting one key, require that multiple keys be used to sign, sometimes from multiple people.

6 Online Wallets and Exchanges

- Online Wallet: local wallet, but stored in the cloud. Much more convenient. Site needs to store keys for you.
- Exchanges: Places where you can trade one currency for another, including Bitcoin for other cryptocurrencies or real dollars.

  Note that money doesn't actually change - only the Exchange's promise to different customers e.g. "Previously, we owed you $10k. Now we owe you X bitcoins"
- Risks with Exchanges

  Bank Run: Everyone wants their money out, typically because they think market will crash.

  Ponzi Scheme (Pyramid Scheme). Ponzi scheme participants believe they are earning returns from their investment, while pyramid scheme participants are aware that they are earning money by recruiting new participants.

  Hack e.g. Mt. Gox. Mt. Gox was a Bitcoin exchange. Launched 2010, by 2013 was handling 70% of Bitcoin transactions. Filed for bankruptcy in February 2014, after announced that 850,000 Bitcoins worth $450 milion were missing and stolen. Happened because credentials were stolen.
- Most exchanges stabilized by government regulations

  Minimum reserve requirement

  Insure deposits up to a certain amount

  Governments loan exchanges money if exchanges are trapped financially
- Can use cryptography to prove that an exchange has a fractional reserve.

  Proof-Of-Reserve: Publish a list of transactions moving money from yourself to yourself. The challenger issues a random string, and the bank signs it using the secret key corresponding to the public key of the transferred money. This establishes a **minimum** amount held.

  Proof-Of-Liabilities: Don't want to publish everything because of customer privacy. Exchange constructs Merkle Tree of users, publishes root hash. For each node, add an attribute containing how much money is owed to that user (or the users beneath that node). The exchange signs the root of the hash. Every user can verify how much is owed to them. Exchange cannot represent different values to different people. Cannot **underestimate** liabilities