Lecture 5 - Engineering Details and Applications

Rylan Schaeffer and Vincent Yang

April 24, 2016

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

4, 9

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 → Alice; Signed: Miner (usually Alice)
    2. Inputs: 1[0]; Outputs: 17 → Bob, 8 → Alice; Signed: Alice
    3. Inputs: 2[0]; Outputs: 8 → Carol, 9 → Bob; Signed: Bob
    4. Inputs: 2[1]; Outputs: 6 → David; 2.0 → 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.

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.

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 and Bob create transaction for maximum value requiring both Alice and Bob to sign to release the coins. For each microtransaction, Alice signs a new transaction giving more of max value to Bob. Bob waits til Alice is done, then publishes the last transaction.

- 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.

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