Introduction to Lightning Network

Sergei Tikhomirov (University of Luxembourg)

Luxembourg, 2019-12-18

How many txs per second can Bitcoin process?

7

Theoretical maximum is 27: https://eprint.iacr.org/2019/416

Why so few?

How distributed systems scale (usually)

- Facebook scales by renting more servers
- Users are assigned to different servers (all under Facebook’s control)
- More servers = faster processing

Bitcoin ≠ Facebook

- Users must be able to *independently* verify transaction history
- Load is *replicated*: more nodes ≠ higher throughput
- TPS higher than consumer internet bandwidth threatens decentralization

Blockchain scaling approaches

- Increasing parameters: works only to an extent (Litecoin, Bitcoin Cash)
- Sharding: “traditional” load distribution with a twist (Ethereum 2.0?)
- Zero-knowledge: magical math (Zcash)
- Second layer (L2):
  - In the Bitcoin world: payment channels (**Lightning**), RSK
  - In the Ethereum world: Raiden, Plasma, commit chains, rollups

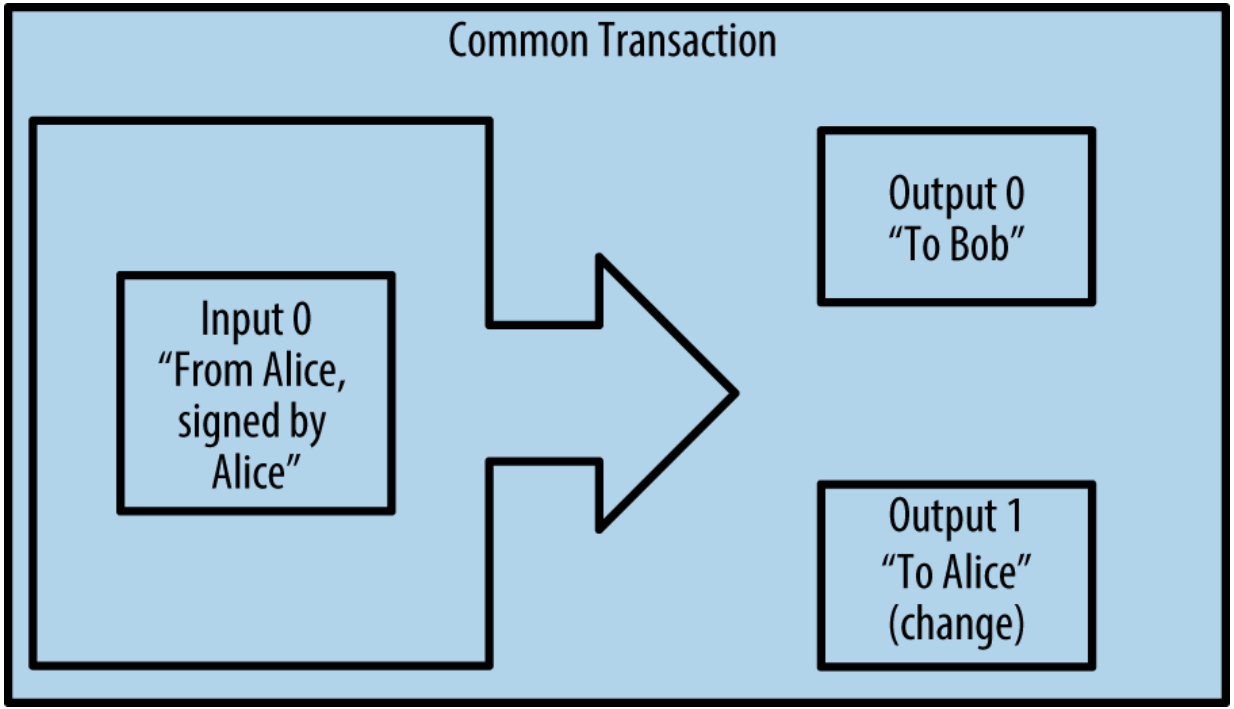
Layer 2: blockchain court

- We can’t put all transactions on the blockchain (L1)
- Let’s do (most of the) transactions *off* the blockchain (L2)
- Use L1 only for dispute resolution

First, let’s discuss the Bitcoin transaction structure...

Bitcoin transactions

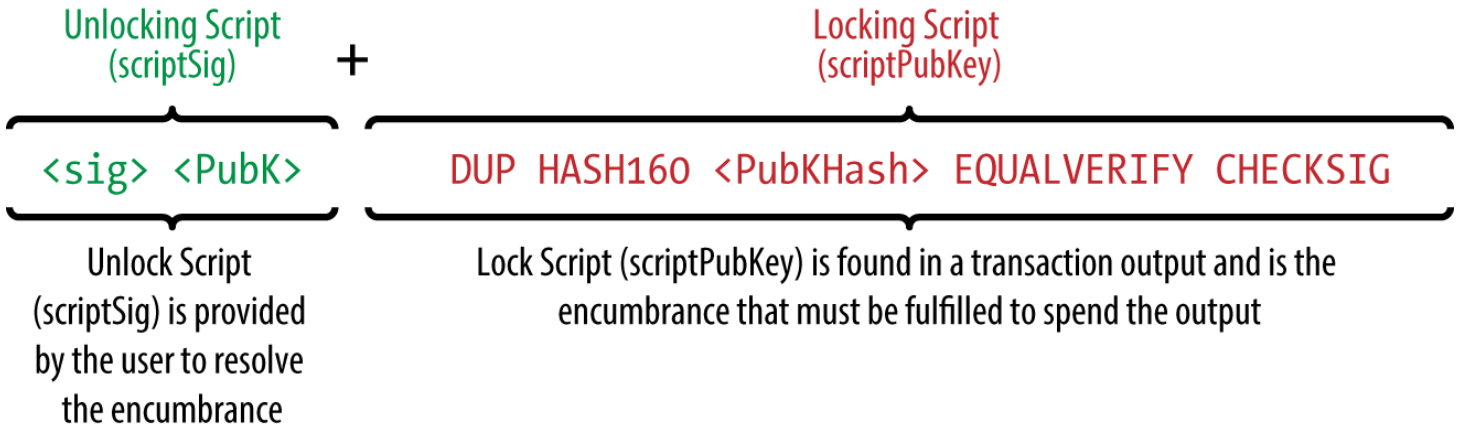
A tx *consumes* UTXOs (unspent tx outputs) as inputs and *creates* new UTXOs.



Transaction outputs

Each UTXO contains a script that defines the spending conditions.

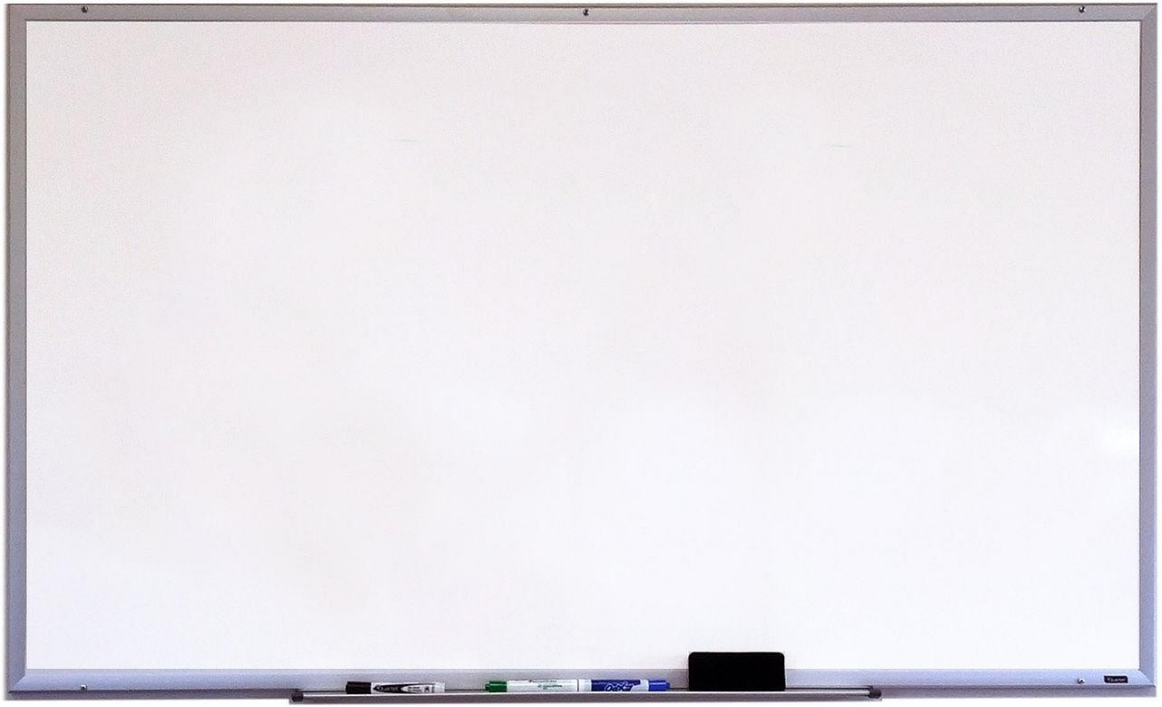
Most common condition: signature for a given (hash of a) public key.



Restricting the outputs

- Multisig: m out of n possible keys are presented
- Hashlock: value hashing to X is presented
- Timelock: current time is after T (absolute of relative)

Whiteboard time!



There have been multiple attempts at creating a payment channel protocol...

Attempt #1: replace by incentive

- The idea of updating an unconfirmed tx dates back to Satoshi
- Uni-directional channel: Alice pays Bob, not vice versa
- Bob publishes the last tx, because it gives him more coins

Can we generalize it to bi-directional payments?

Key challenge: invalidating old states

- L2 protocols maintain a *shared state* between parties
- The goal: provide security close to L1
- Only the *last* state must be enforceable on L1
- On each state update, the parties *invalidate* the previous state

How do we prove to L1 which state is the latest?

Attempt #2: replace by timelock

- All off-chain txs are timelocked (valid after given time)
- The next timelock is closer to the present than the previous one
- If Bob broadcasts an *old* state, Alice can confirm the *latest* state earlier
- Bi-directional!
- Total channel lifetime is limited by the first tx’s timelock
- Total number of updates is limited by the safety margin between timelocks

Attempt #3: replace by revocation (Lightning)

Key idea: previous states are invalidated economically.

The victim can punish the cheater.

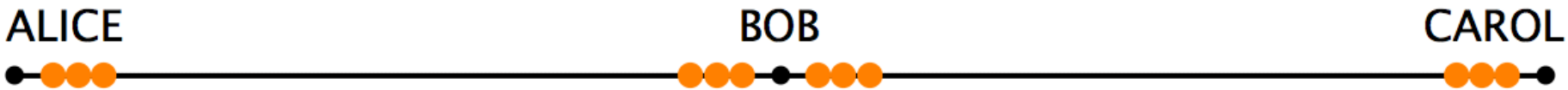
- Alice to Bob: here is key_A what lets you take all coins **if and only if** I cheat
- Bob to Alice: here is key_B what lets you take all coins **if and only if** I cheat
- Victim must raise dispute not later than T hours after cheating attempt

Payment channel lifecycle

- Alice and Bob **open** a channel
  - Lock bitcoins in a 2-of-2 multisig with some initial distribution of funds
- The parties **update** the funds distribution in two steps:
  - Invalidate the previous state
  - Agree on the new state
- The parties **close** the channel in one of three ways:
  - The good: cooperatively sign, no delay
  - The bad: Bob is offline, Alice gets her coins after a delay
  - The ugly: Alice tries to cheat, Bob takes all chanel funds

Connecting payment channels

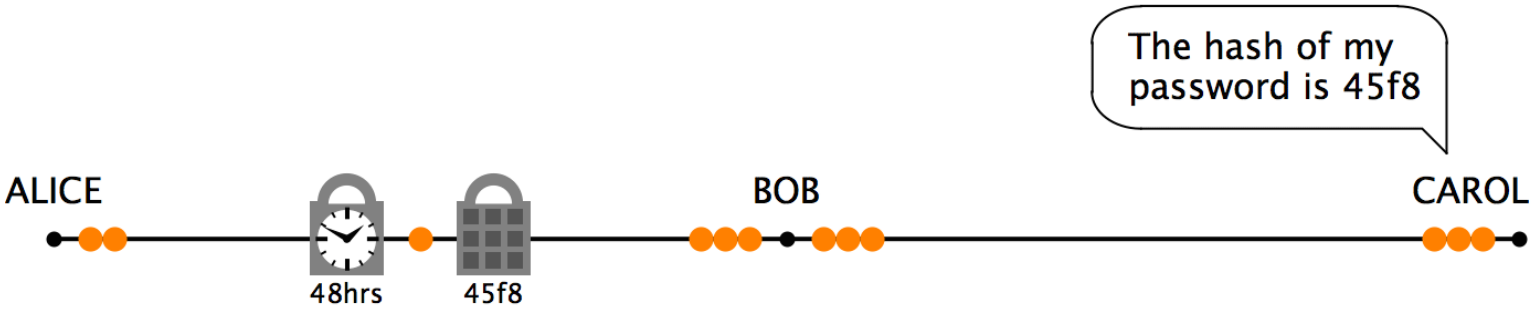
- All users can’t open channels to all other users
- How can Alice pay to Carol, if they both have a channel to Bob?
- Ensure atomicity with a *common* secret



A multi-channel payment: step 1/4

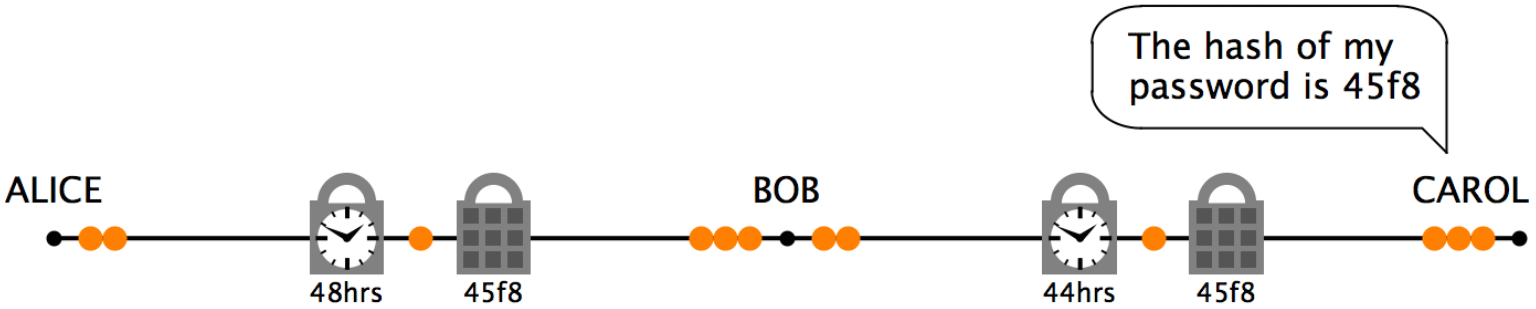
Carol → Alice: sent coins to this hash (I know the secret)

Alice → Bob: you get coins if you know the secret (Hint: Carol may know it!)



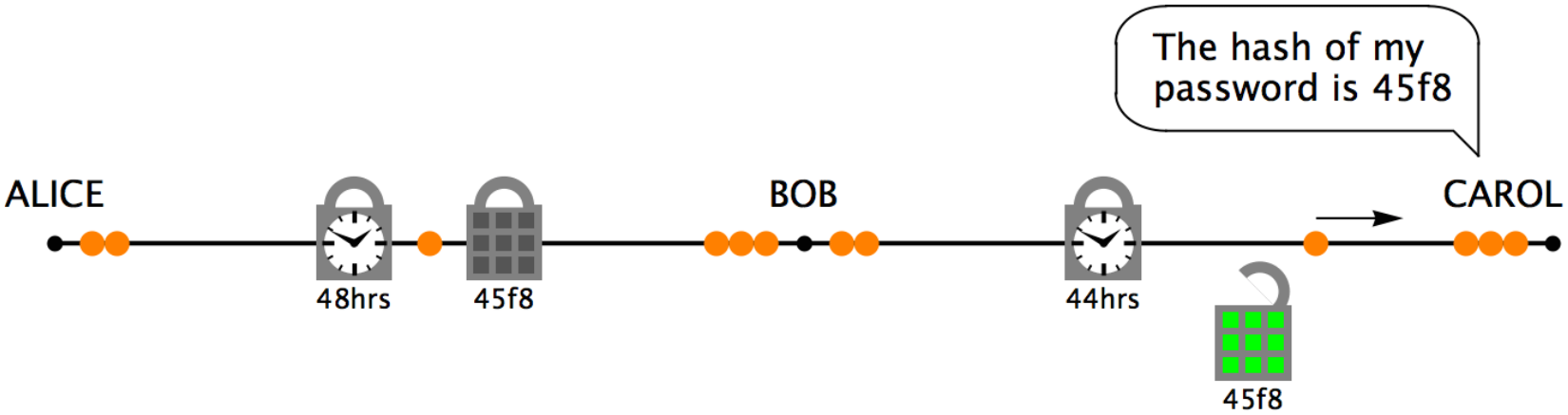
A multi-channel payment: step 2/4

Bob → Carol: you get coins if you know the secret



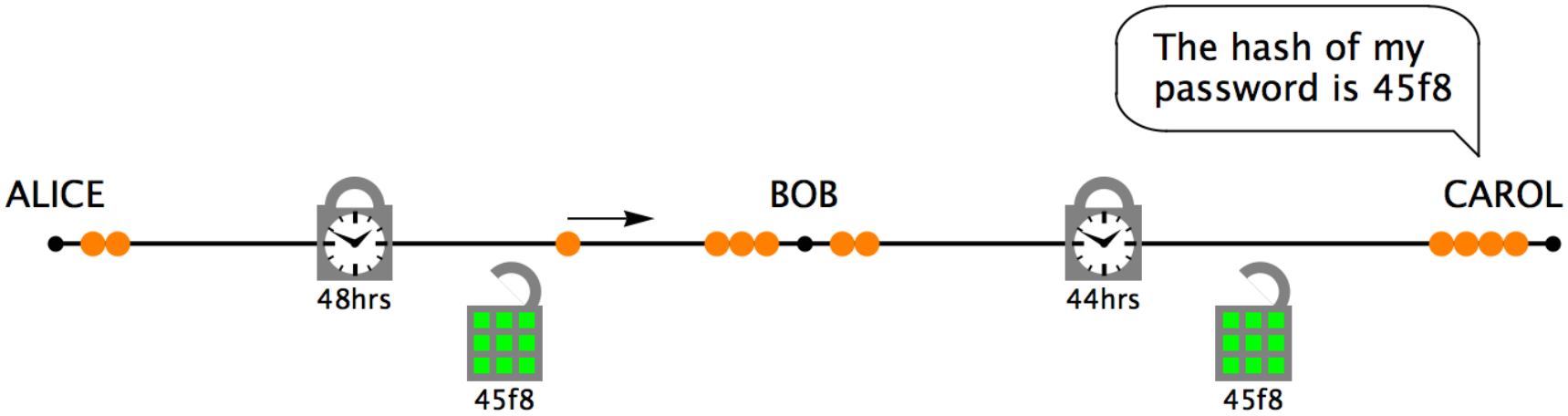
A multi-channel payment: step 3/4

Carol → Bob: here is the secret! (takes money)



A multi-channel payment: step 4/4

Bob → Alice: here is the secret! (takes money)




4,532
NODES
30,565
CHANNELS

Lightning: pros & cons (aka open problems)

Pros: instant payments **with real bitcoins**.

Cons:

- Liquidity: can’t combine coins from channels, no multi-path payments (yet)
- Online requirement: users must watch their channels
- Complex UX: many balances, must be online to receive
- Security guarantees violated for payments < L1 fee
  - Like sueing is not economical if compensation < lawyer fees
  - Note also that “dust” can’t be spent on L1 either
- Attacks! Trade-off between lack of identity and DoS vectors

Will Lightning save Bitcoin?

- LN is cool but has many trade-offs
- Best thought of as an *alternative way* to move bitcoins
- Is LN economical? Cost of locked-up capital may be too high
- A trade-off between good UX and being trustless is hard
- LN for small txs? Much simpler to be custodial
- LN for large txs? Liquidity problems

Lightning is not a silver bullet (but nothing is).

Q&A

Follow @serg_tikhomirov on Twitter

Further reading

- Aaron van Wirdum’s “Understanding the LN” series
  - https://bitcoinmagazine.com/articles/understanding-the-lightning-network-part-building-a-bidirectional-payment-channel-1464710791/
- J. Lopp’s Lightning resources
  - https://www.lopp.net/lightning-information.html
- Awesome LN list
  - https://github.com/bcongdon/awesome-lightning-network

Image credits:

- https://explorer.acinq.co/
- https://medium.com/@peter_r/visualizing-htlcs-and-the-lightning-networks-dirty-little-secret-cb9b5773a0
- https://github.com/bitcoinbook/bitcoinbook
- https://bitcoinvisuals.com/lightning
- https://twitter.com/michaelbatnick/status/1019680856837849090
- https://graph.lndexplorer.com/
- https://mg.wikipedia.org/wiki/Sary:Lightning_NOAA.jpg