Blockchains & Cryptocurrencies

Anonymity in Cryptocurrencies III / Scaling I

Instructors: Matthew Green & Abhishek Jain
Johns Hopkins University - Spring 2019

Housekeeping

- 3rd reminder: midterm Weds!
  - You can write a "cheat sheet", handwritten, US letter sized paper, both sides
  - Includes today's lectures
- Assignment 2 due end of day

News?

News?

BITCOIN

Bitcoin [BTC] transaction numbers in Venezuela nosedive as country goes through acute power outage

Published 2 hours ago on March 11, 2019

By **Akash Anand**

Review stuff (from last time)

Pedersen Commitments

- We need a cyclic group. $G = \langle g \rangle$ where it is hard to find *x* given (g, g^x) — AKA the "discrete log problem" (DLP) is hard
  - E.g., G can be a subgroup of a finite field $\{1, \ldots, p-1\}$ where exponentiation/multiplication are modulo *p*
  - We also need two public generators: g, h *such that nobody knows the discrete log of g resp. h (vice versa)*
  - Commitment to message: pick random $r \in \{0, \ldots, groupOrder - 1\}$, compute: $C = g^m \cdot h^r$
  - To open the commitment, simply reveal (m, r)

Pedersen Commitments

- Why is this secure?
  - **Hiding:** If g, h are generators, then h^r is a random element of the group, so. $C = g^m \cdot h^r$ is too
  - **Binding:** Let q be the group order. Let $h = g^x$ for some unknown x. Assume an attacker can find (m, r) != (m', r') such that $g^m h^r = g^{m'} h^{r'}$. Then it holds that:

$$g^m g^{xr} = g^{m'} g^{xr'}$$

and thus,

$$m + xr = m' + xr' \ mod \ q$$

We can solve for x, which means solving the DLP!

Confidential Transactions

- Pedersen commitments are <u>additively homomorphic:</u>

  - Commit to "m1": $C_1 = g^{m_1} h^{r_1}$
    Commit to "m2": $C_2 = g^{m_2} h^{r_2}$

  - Now multiply the two commitments together:

$$\begin{aligned} C_3 &= C_1 \cdot C_2 \\ &= g^{m_1} h^{r_1} \cdot g^{m_2} h^{r_2} \\ &= g^{m_1+m_2} h^{r_1+r_2} \end{aligned}$$

Notice that C3 is a commitment to the <u>sum</u> m1+m2 (under randomness r1+r2)

Confidential Transactions

- Introduced by Maxwell
  - Does not provide privacy for the identity of the input transactions, can be combined with CoinJoin or ringsides
  - Does allow you to hide the value of input transactions
  - Basic idea: use a Pedersen commitment to each transaction value, rather than revealing this in cleartext $C = g^v h^r$
  - Do a CoinJoin, and use additive property of Pedersen commitments to sum the values and then subtract each output commitment (board)

MimbleWimble (Grin)

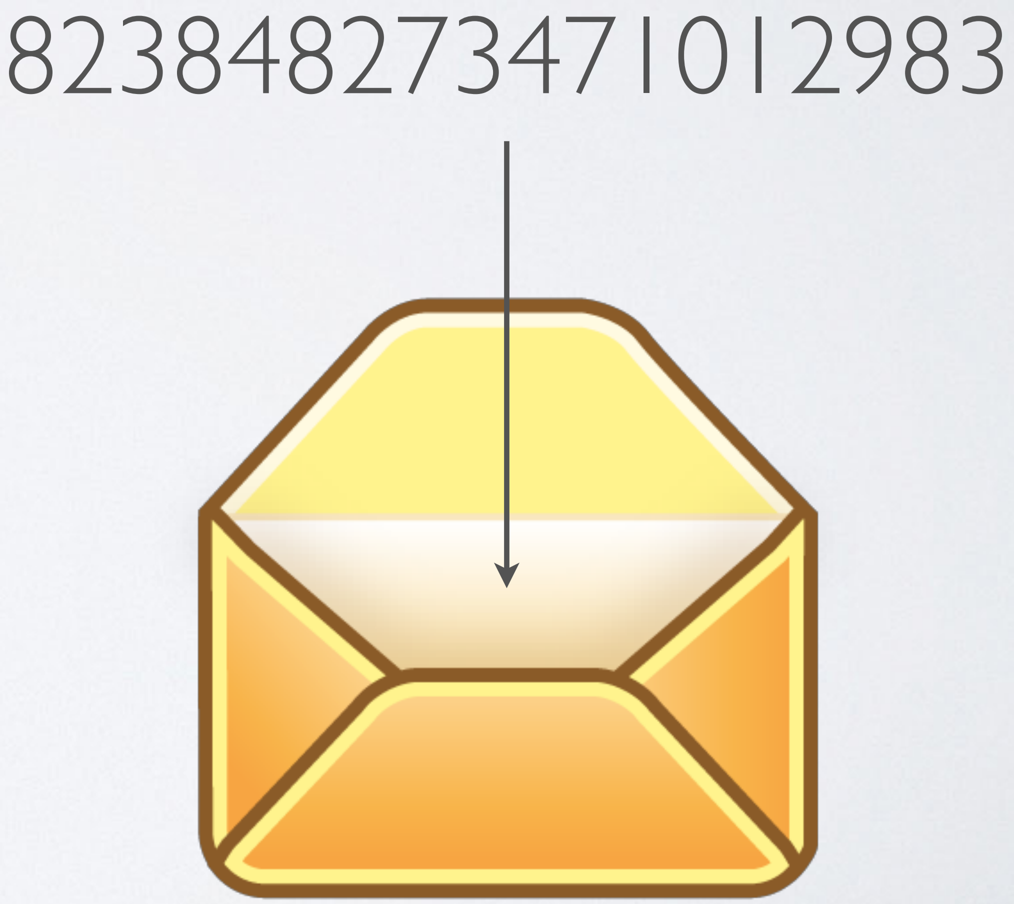
- Combines CoinJoin with Confidential Transactions, provides both services in a single network

Zerocoin (MGGR14)

- Proposed as an extension to Bitcoin in 2014
  - <u>Requires changes to the Bitcoin consensus protocol!</u>
- I can take Bitcoin from my wallet
  - Turn them into 'Zerocoins'
  - Where they get 'mixed up' with many other users' coins
  - I can redeem them to a new fresh Wallet

Zerocoin

- Zerocoins are just numbers
  - Each is a digital commitment to a random serial number
  - Anyone can make one!



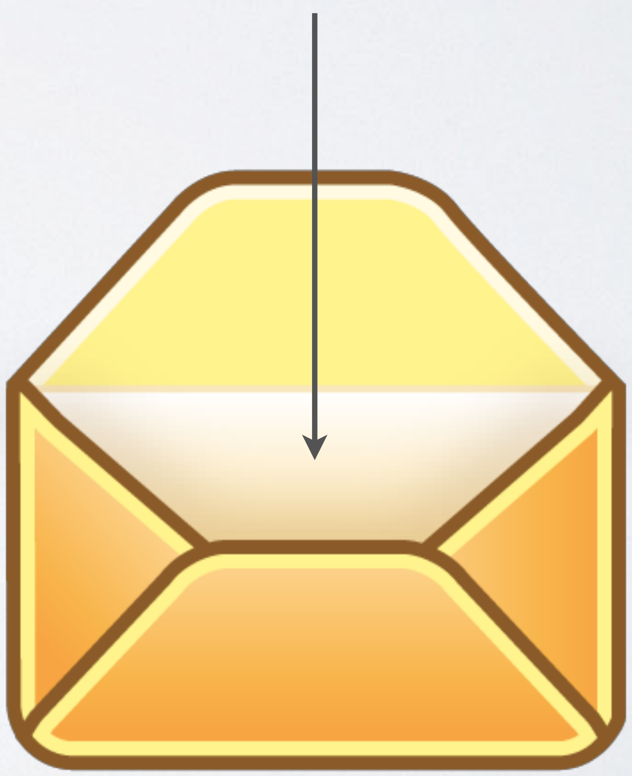
Minting Zerocoin

- Zerocoins are just numbers
  - Each is a digital commitment to a random serial number *SN*
  - Anyone can make one!

$$C = Commit(SN; r)$$

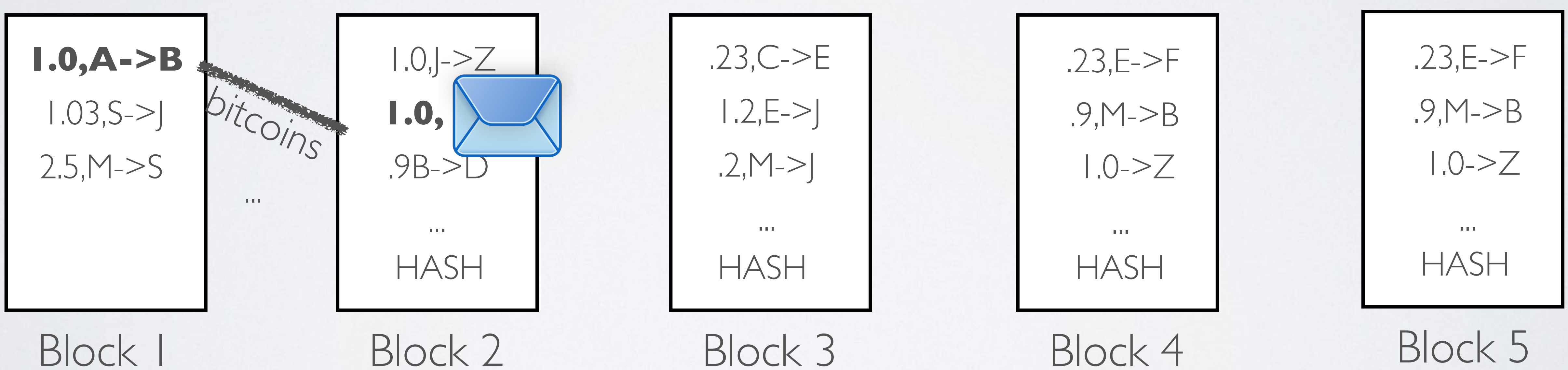
$$C = g^{SN} h^{r} \ mod \ p$$

823848273471012983



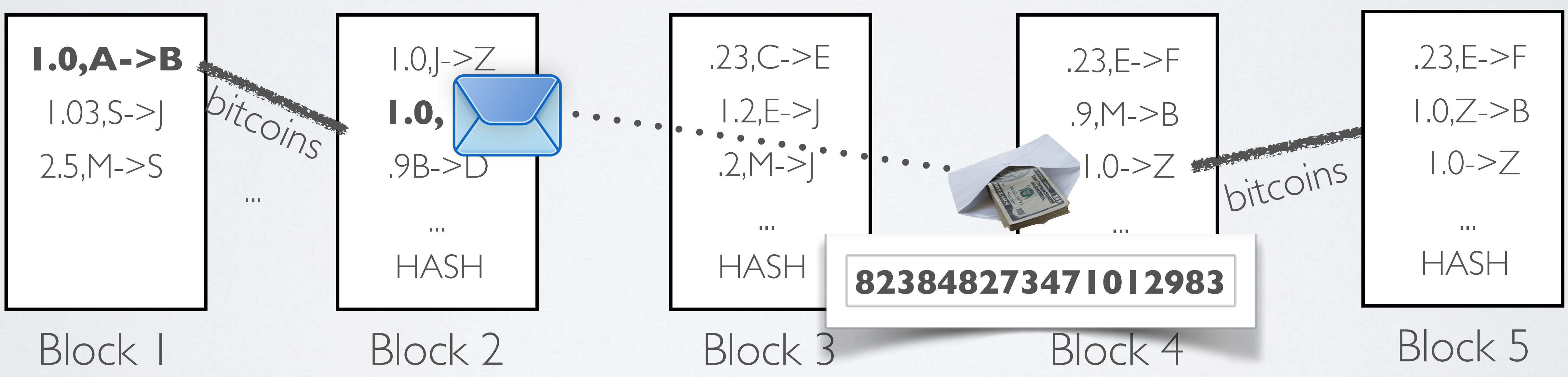
Minting Zerocoin

- Zerocoins are just numbers
  - They have value once you write them into a valid transaction on the blockchain
  - Valid: has inputs totaling some value e.g., 1 bitcoin



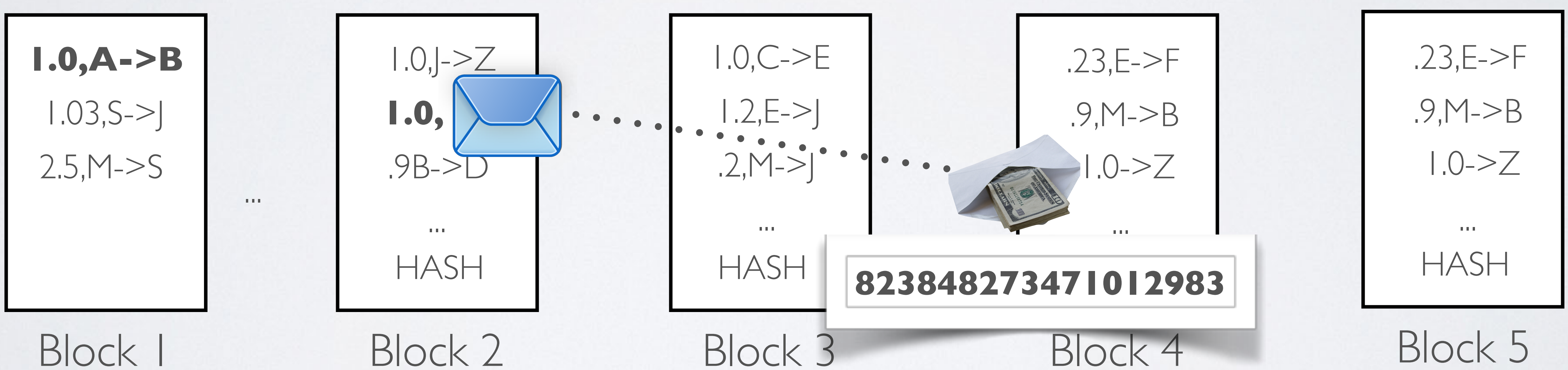
Redeeming Zerocoin

- You can redeem zerocoins back into bitcoins
  - Reveal the serial number &
    Prove that it corresponds to some Zerocoin on the chain
  - In exchange you get one bitcoin (if SN is not already used)



Spending Zerocoin

- Why is spending anonymous?
  - It's all in the way we 'prove' we have a Zerocoin
  - This is done using a <u>zero knowledge proof</u>



Spending Zerocoin

- Zero knowledge [Goldwasser, Micali 1980s, and beyond]
  - Prove a statement without revealing <u>any other information</u>
  - Here we prove that:
    (a) there exists a Zerocoin in the block chain
    (b) we just revealed the actual serial number inside of it
  - Revealing the serial number prevents double spending
  - The trick is doing this efficiently!

Spending Zerocoin

- Zero knowledge [Goldwasser, Micali 1980s, and beyond]
  - Prove a statement without revealing <u>any other information</u> (other than that a statement is true)

Spending Zerocoin

- Zero knowledge [Goldwasser, Micali 1980s, and beyond]
  - Prove a statement without revealing <u>any other information</u>
  - Here we prove that:
    (a) there exists a Zerocoin (commitment) in the block chain
    (b) the thing we revealed is the opening to that commitment
  - Revealing the serial number prevents double spending
  - The trick is doing this efficiently!

Spending Zerocoin

- Possible proof statement (not efficient, see CryptoNote):
  - Public values: list of Zerocoin commitments $C_1, C_2, \ldots, C_N$
    Revealed serial number *SN*
    - Prove you know a coin *C* and randomness *r* such that:

$$C = C_1 \ \vee \ C = C_2 \ \vee \ \ldots \ \vee \ C = C_N$$
$$\wedge \ C = Commit(SN; r)$$

    - Problem: using standard techniques, this ZK proof has cost/size O(*N*)

Spending Zerocoin

- Zerocoin (actual protocol)
  - Use an efficient RSA one-way accumulator
  - Accumulate $C_1, C_2, \ldots, C_N$ to produce a short value A
  - Then prove knowledge of a short witness s.t. $C \in inputs(A)$
  - And prove knowledge that C opens to the serial number

Requires a DDL proof (**~25kb**)
for each spend. In the block chain.

Spending Zerocoin

- Zerocoin (actual protocol)
  - Use an efficient RSA one-way accumulator
  - Accumulate $C_1, C_2, \ldots, C_N$ to produce a short value A
  - Then prove knowledge of a short witness s.t. $C \in inputs(A)$
  - And prove knowledge that C opens to the serial number

Requires a DDL proof (**~25kb**) for each spend. In the block chain.

Anonymity set comparison

- Anonymity set in CoinJoin:
  - **M**: where **M** is number of inputs in the transaction (bounded by TX size)
- Anonymity set in ByteCoin/RingCT:
  - **N**: where **N** is the number of inputs allowed in a transaction (bounded by TX size, 7-11 historically)
- Anonymity set in Zerocoin:
  - **P**: where **P** is number of total Zerocoins minted on the blockchain thus far* (independent of TX size)

Scaling

The problem

- Bitcoin transaction rate: 5-7 tx/sec
  - Bounded by block size (Segwit helps), TX size
  - All transactions must be globally verified, stored
- Ethereum: 15 transactions per second <u>if they're small</u>
- Visa: 24,000/sec peak (150M/day globally)
- WeChat 256,000/sec peak

Faster computers?

- Why not just build faster computers?

Faster computers?

- Why not just build faster computers?
  - Loss of decentralization
  - Eventually we saturate links, due to broadcast network
  - Replicated global state falls apart
  - Scaling is possible (see Visa, WePay etc.) but it requires dedicated, centralized servers

Can we do better?

Can we do better?

- Current ideas:
  - “Off chain”
  - “Sharding”
  - New consensus algorithms

Off-chain transactions (channels)

- In current Bitcoin-style networks, every transaction appears on the blockchain
  - This allows the whole network to verify financial integrity
  - I.e., we can’t go off an do transactions elsewhere, accidentally/deliberately inflate the money supply
- But why does the network need to see <u>every</u> transaction?

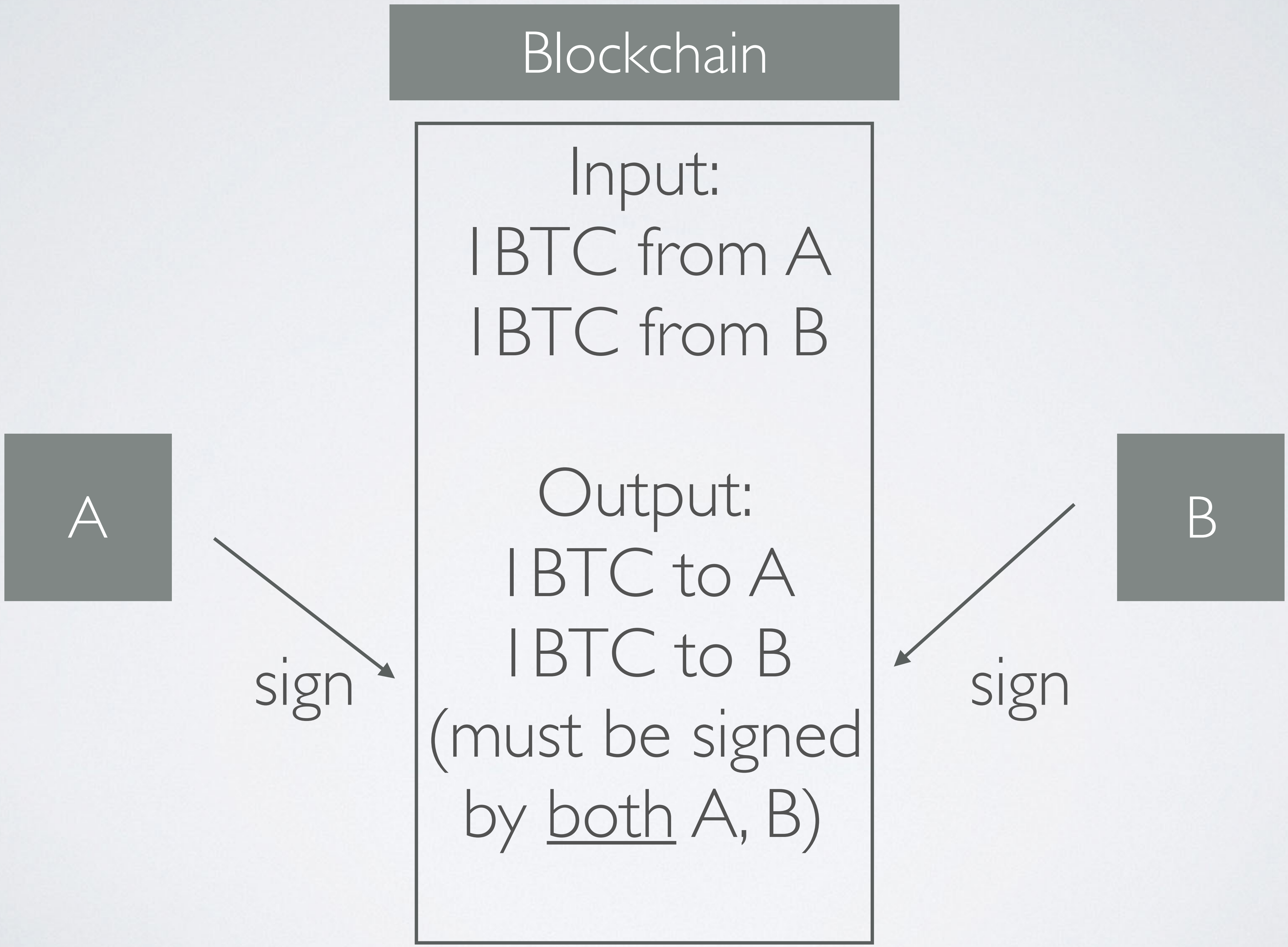
Off-chain transactions (channels)

- Overarching idea:
  - If a transaction doesn’t affect anyone else (except for the parties willing to risk money), chain doesn’t need to see it
  - Simplest example (but centralized):
    - Multiple parties deposit money into an exchange
    - Exchange is just a centralized bank, so everyone can quickly transmit money by adjusting balances
    - Only withdrawals need on-chain transactions

Off-chain transactions (channels)

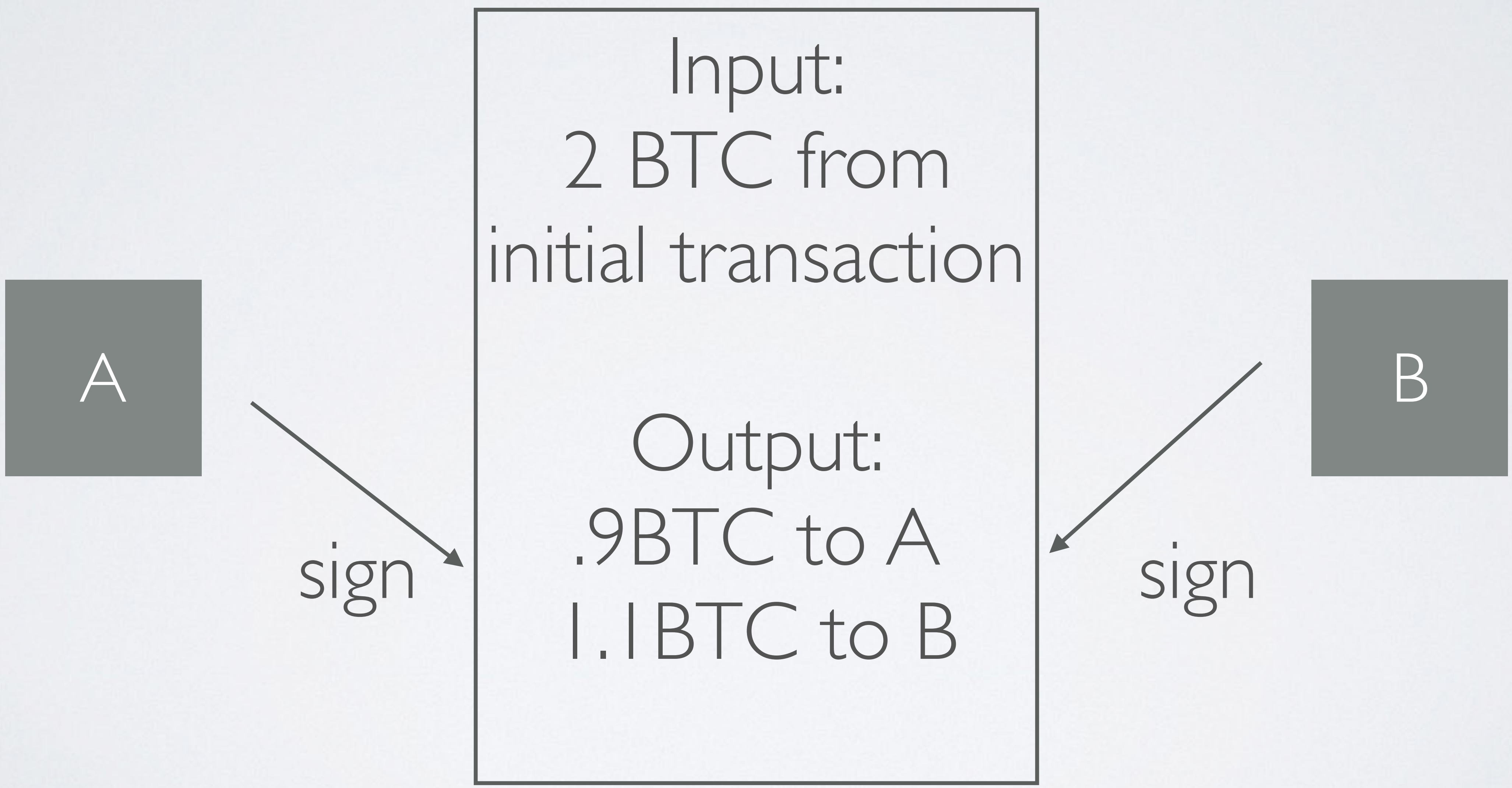
- Off-chain exchange example still risks loss of funds
  - If the exchange disappears, your money goes with it
  - See e.g., QuadrigaCX
  - The only benefit here is that the <u>rest</u> of the network can't lose money, e.g., due to inflation

Channels: Step 1



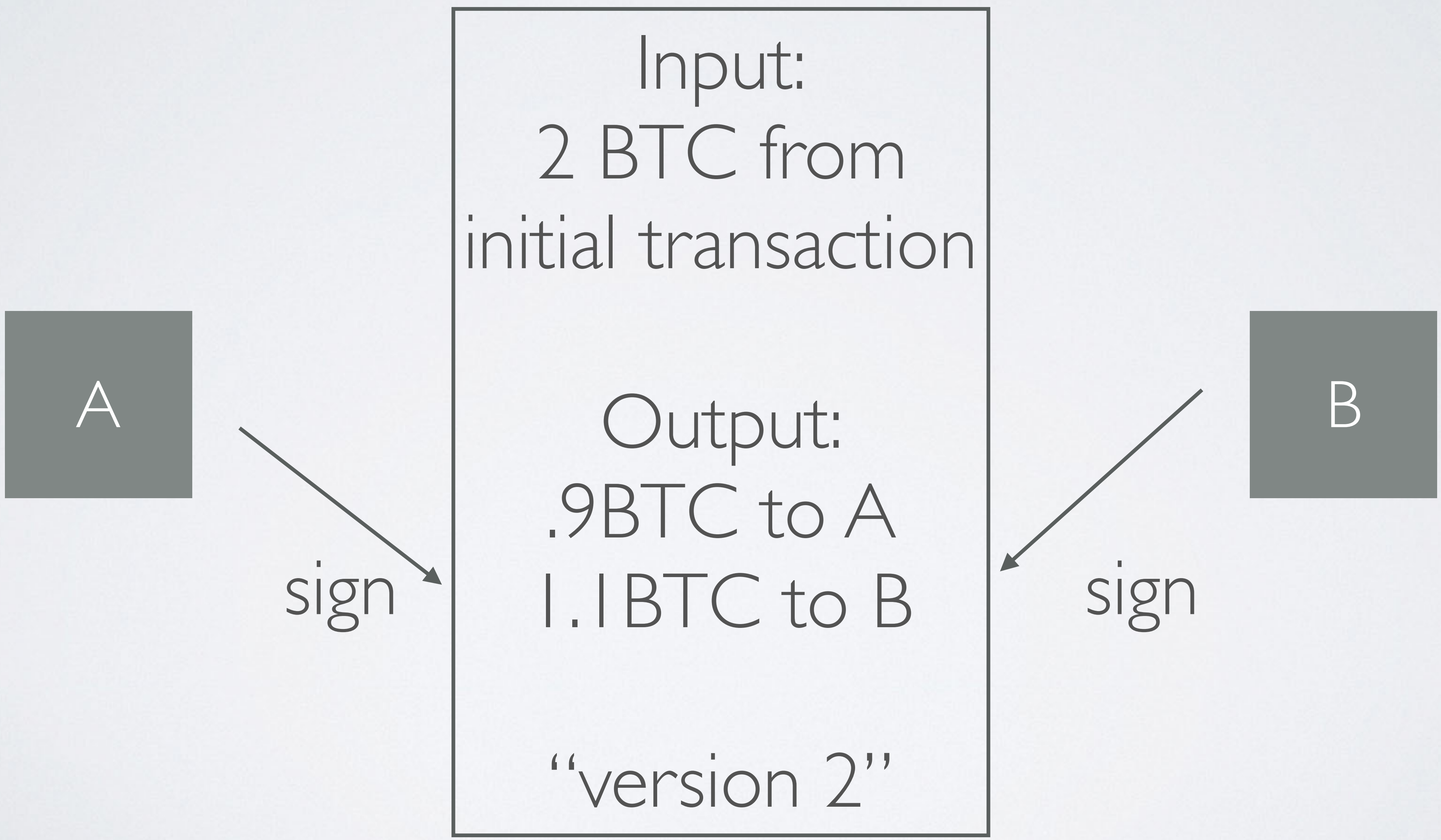
Channels: Step 2

* A pays B 0.2BTC



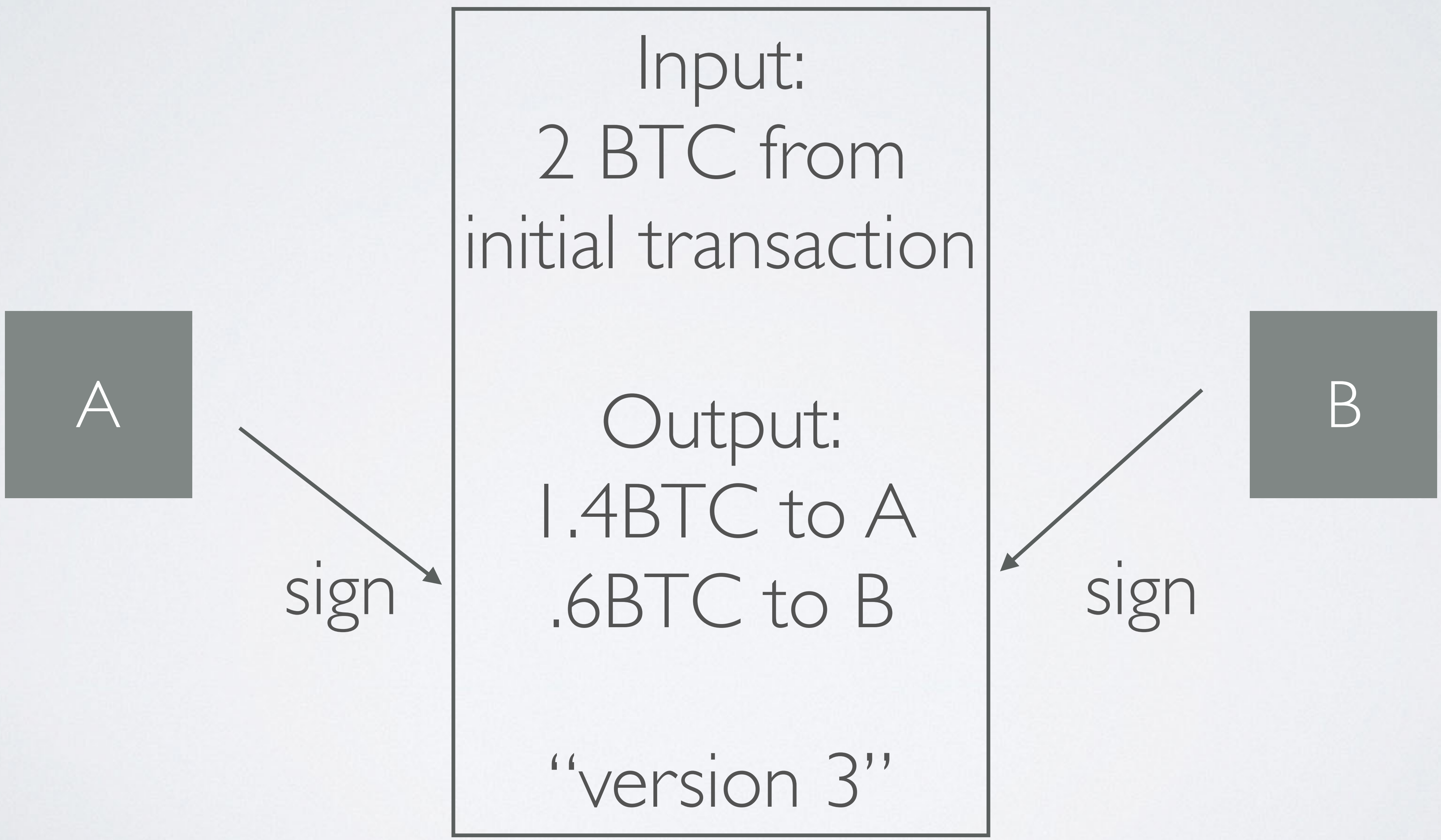
Channels: Step 2

* A pays B 0.2BTC



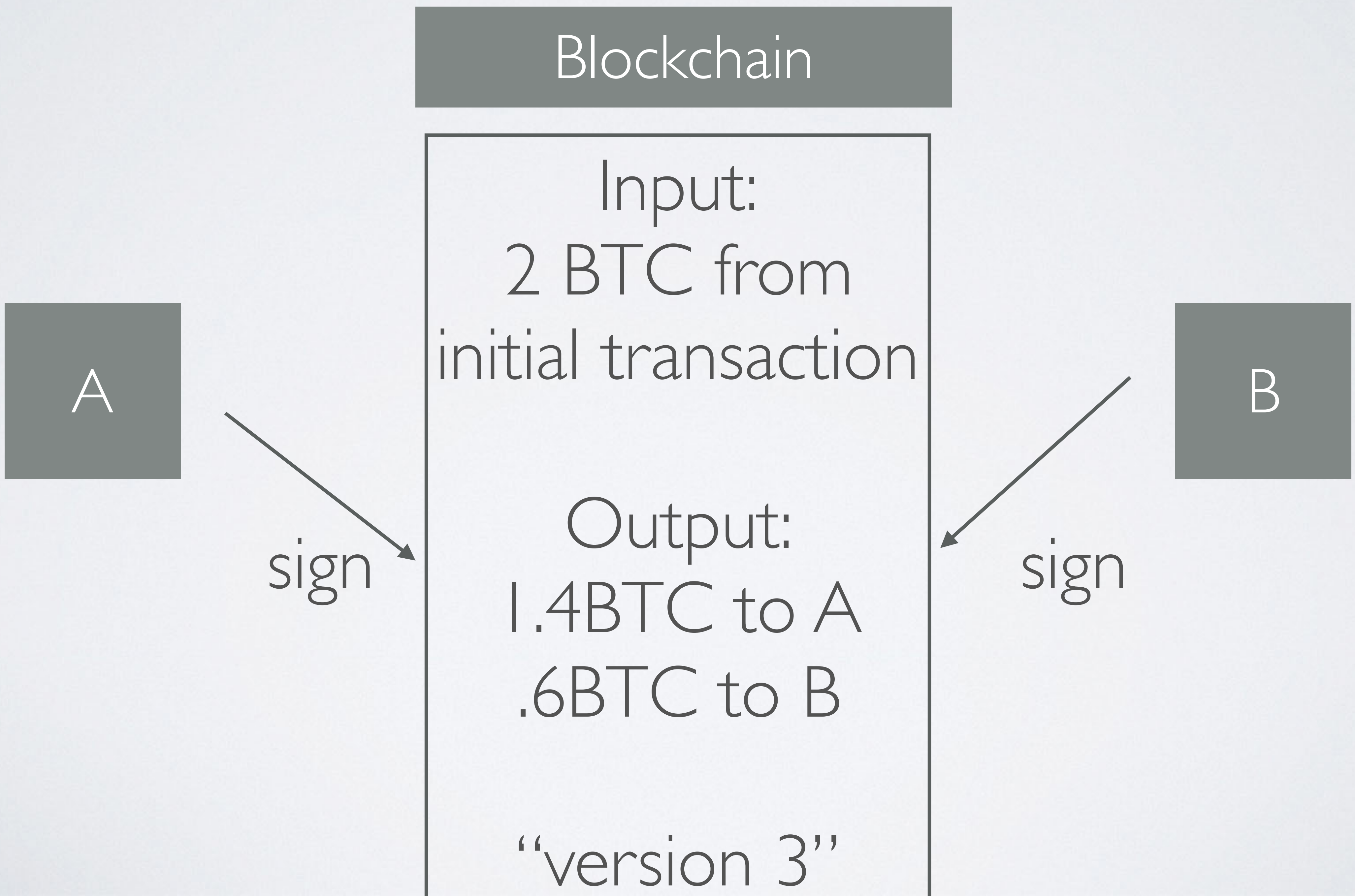
Channels: Step 3……

* B pays A .5 BTC



Channels: Closure

* Either party posts the most recent version of the transaction to the blockchain (all older versions get ignored)



Dispute resolution

* What if someone posts an older, out-of-date version?
* What if nobody signs the first "closure" transaction? How do you escape a payment channel in the worst case?