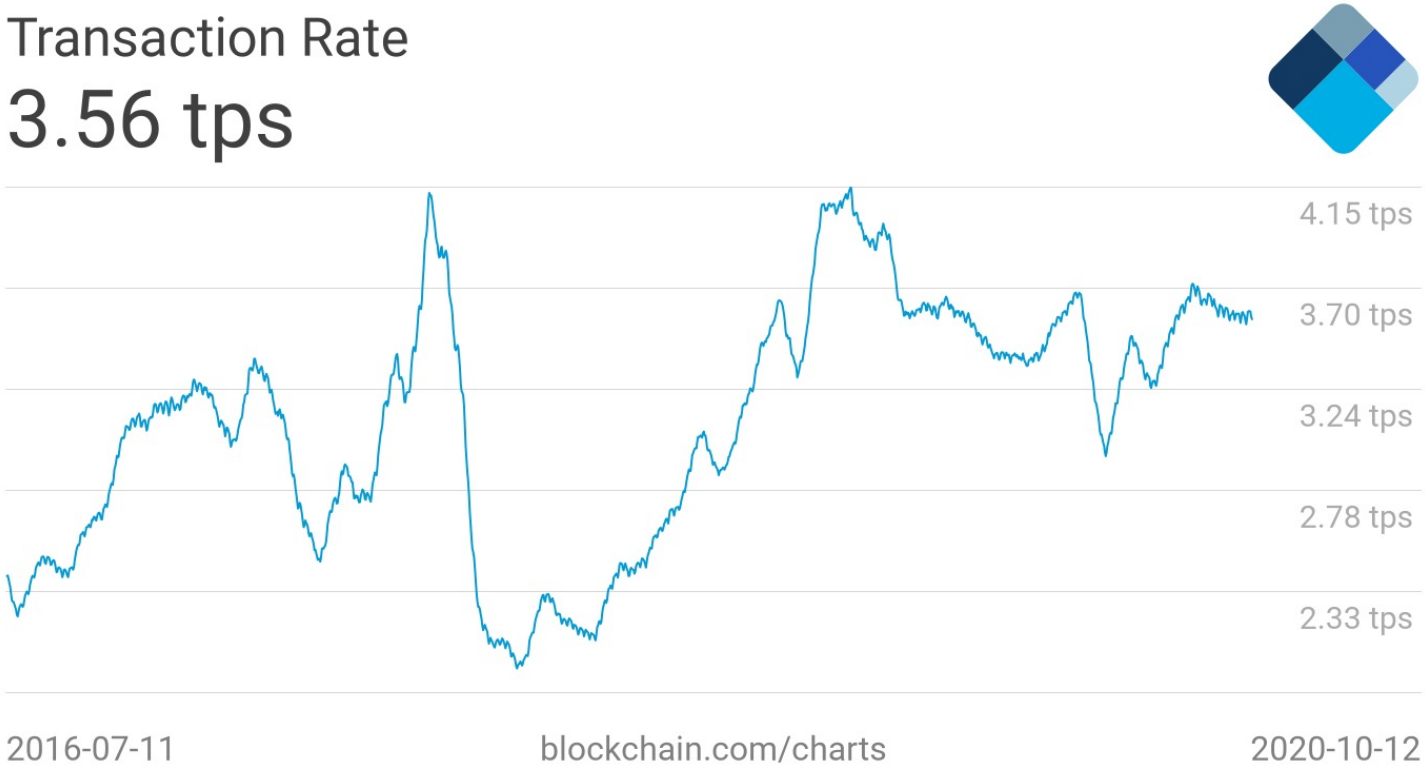
CS251 Fall 2020

(cs251.stanford.edu)

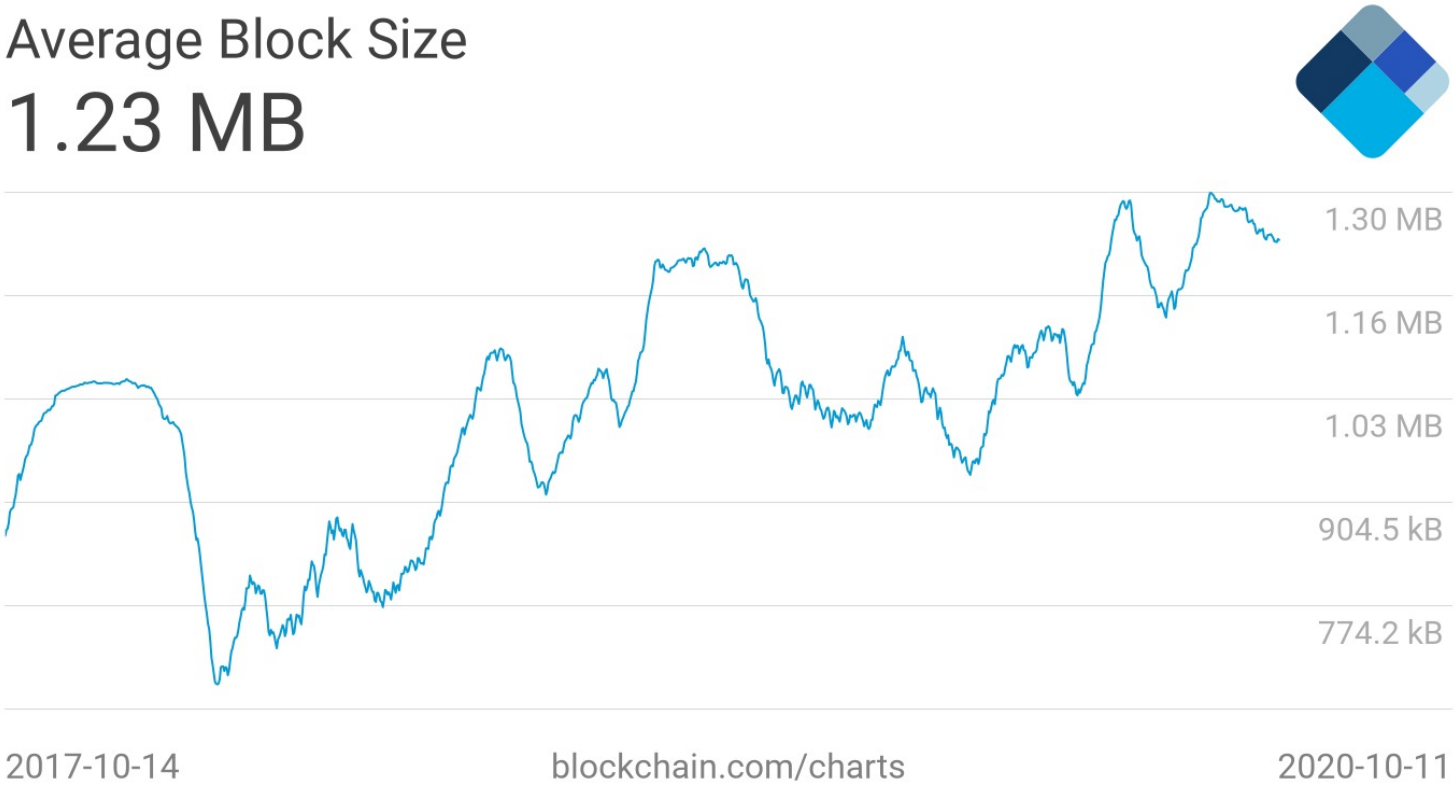
Scaling I: Payment Channels, State Channels

Benedikt Bünz

Bitcoin Throughput



Block Size



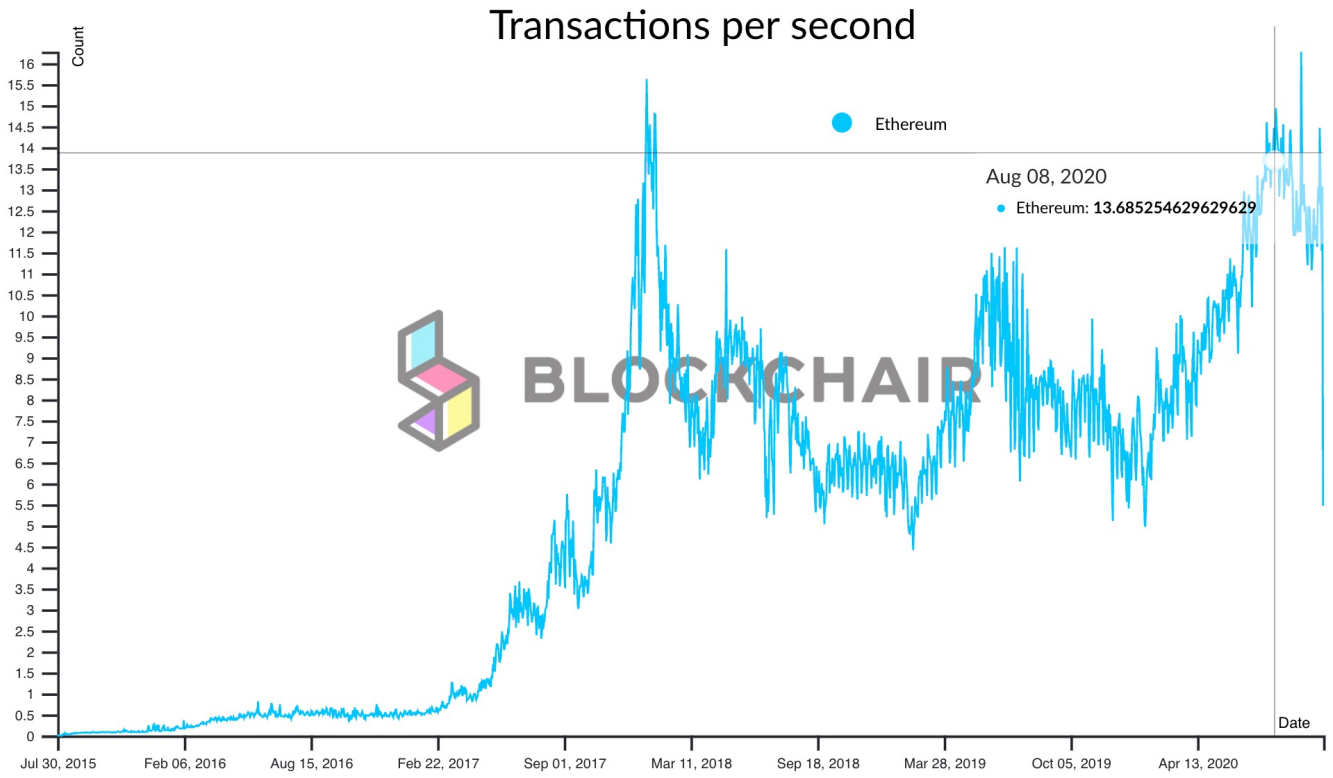
1 MB per Block

250 byte

4000 tx/block

Max: 6.7 tx/s

Ethereum Throughput



TX: 21k Gas

12.5M Gas per block

600tx/block

1 Block/15s

Max 40tx/s

Visa Throughput

Visa ~2000tx/s

Up to 65000tx/s (Christmas shopping season)

Raising Blocksize/Gas limit

TX/s directly dependent on blocksize.

Why not raise it?

Network delay/Consensus security is dependent on block size

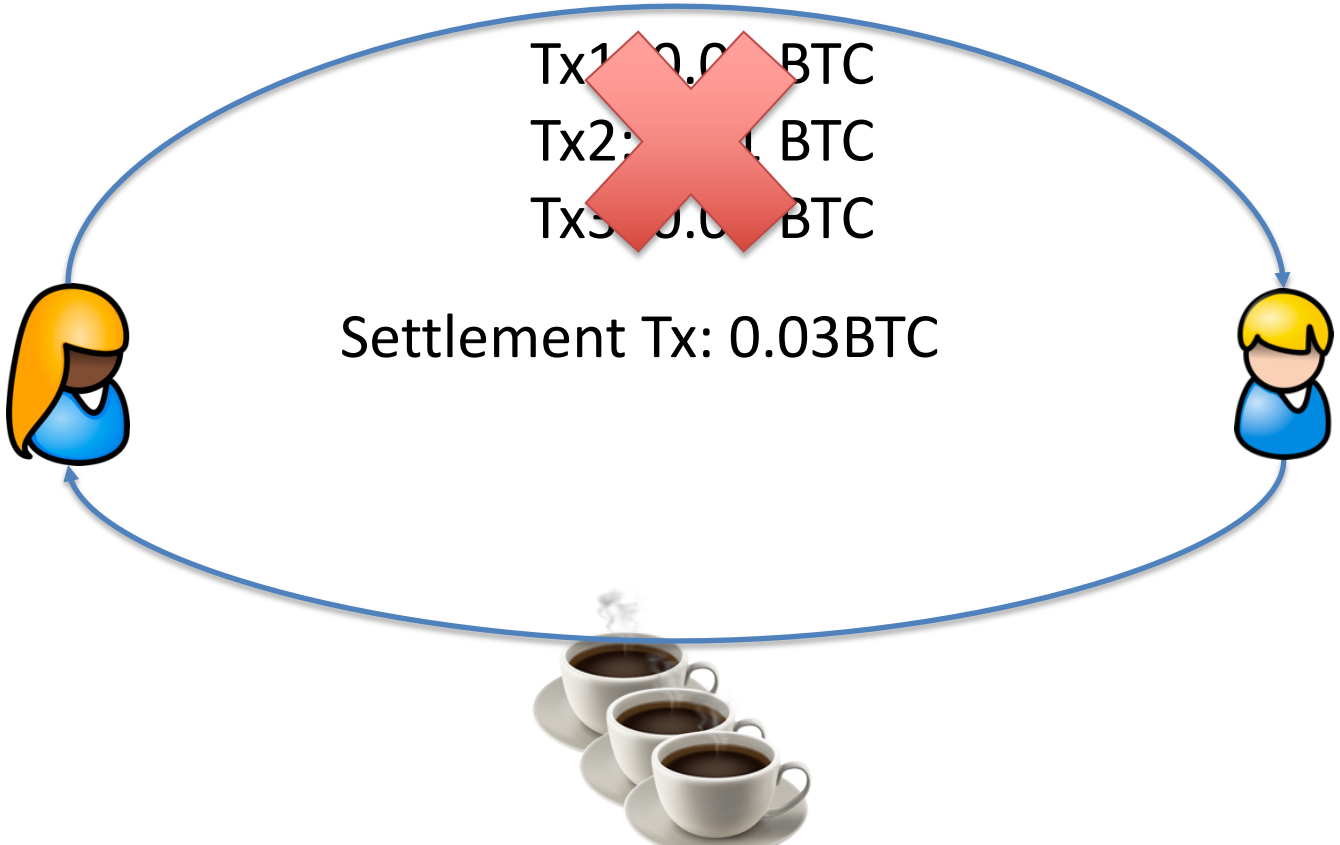
Additional issue: Latency (delay till TX confirmation)

Idea: Increase #tx without increasing data

- What if we don’t record every TX on the chain
- Only record settlements
- Use Blockchain to solve disputes
- Potential to scale transactions especially if everything goes well
- Get Blockchain security if things go bad

Blockchain Ledger

Payment Channels

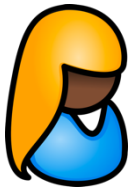


Unidirectional Payment Channel

UTXO A:
1 BTC

Bob does not publish

Publish TX3 on Blockchain



TX1: 0.99 to Alice/0.01 to Bob from UTXO A
Alice

TX2: 0.98 to Alice/0.02 to Bob from UTXO A
Alice

TX3: 0.97 to Alice/0.03 to Bob from UTXO A
Alice

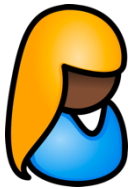
Unidirectional Payment Channel

UTXO A:
1 BTC

Bob does not publish

Attack: Alice double spends UTXO A

Publish TX3 on Blockchain



TX1: 0.99 to Alice/0.01 to Bob from UTXO A
Alice

TX2: 0.98 to Alice/0.02 to Bob from UTXO A
Alice

TX3: 0.97 to Alice/0.03 to Bob from UTXO A
Alice

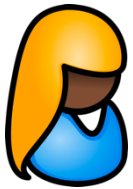
Unidirectional Payment Channel

UTXO A:
1 BTC

→

2-2 Multisig Account AB:
1 BTC

Attack:
Bob never signs



Publish TX3 on Blockchain

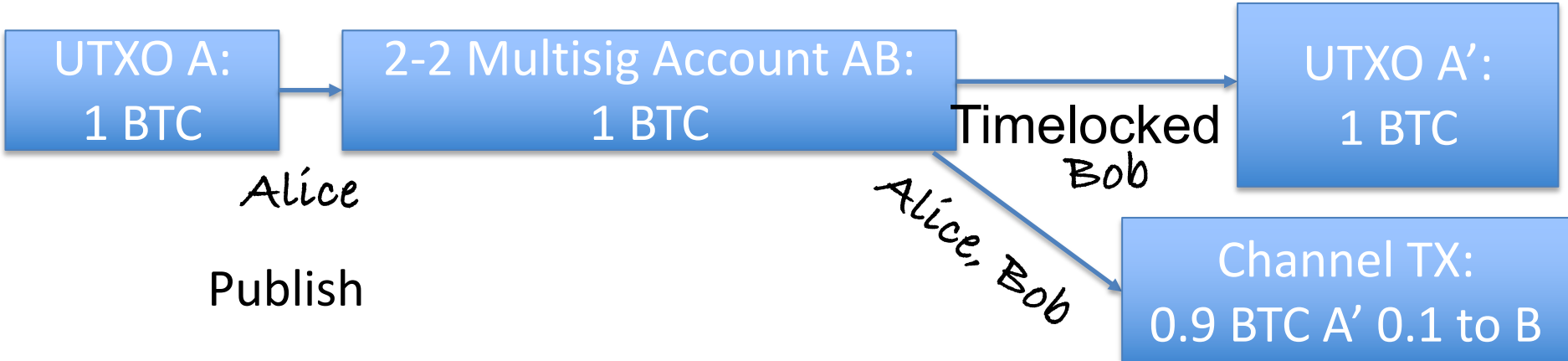
TX1: 0.99 to Alice/0.01 to Bob from AB
Alice

TX2: 0.98 to Alice/0.02 to Bob from AB
Alice

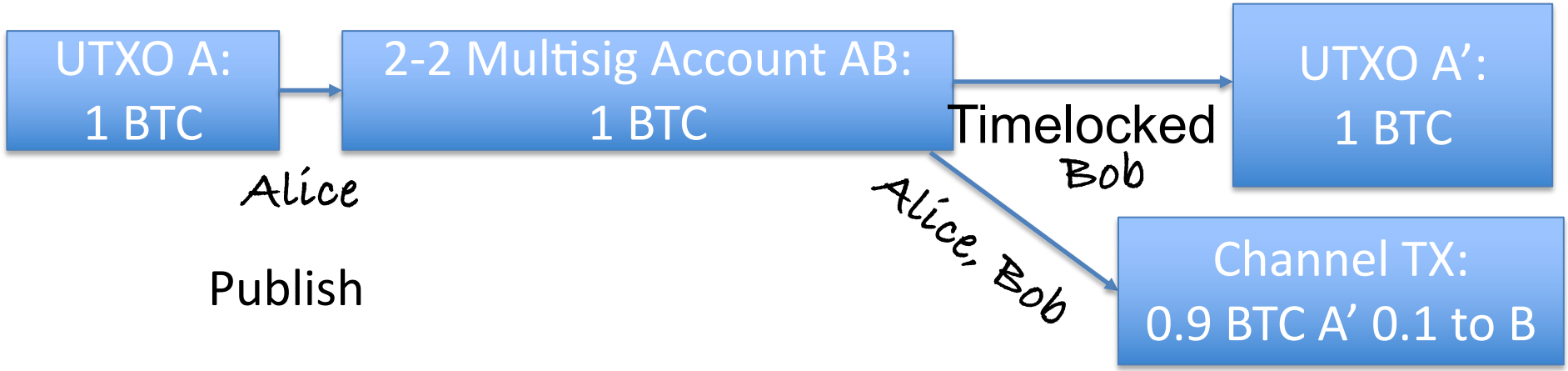
TX3: 0.97 to Alice/0.03 to Bob from AB
Alice

Unidirectional Payment Channel

- Alice needs a way to ensure refund of funds
- Basic idea: If Bob doesn't publish after some time Alice gets 1 BTC refunded
- Refund transaction signed before funding Account AB



Unidirectional Payment Channel



- If Alice and Bob cooperate use multisig, otherwise timelock
- In Ethereum implemented as smart contract
- Non expiring: Refund TX starts claim period for Bob
- Once Alice sent 1 BTC to Bob Channel is ”exhausted”

Payment Channel in Solidity

Home | UniChannel.sol

```
pragma solidity >=0.4.24 <0.6.0;

contract SimplePaymentChannel {
    address payable public sender;      // The account sending payments.
    address payable public recipient;   // The account receiving the payments.
    uint256 public expiration;  // Timeout in case the recipient never closes.

    constructor (address payable _recipient, uint256 duration)
        public
        payable
    {
        sender = msg.sender;
        recipient = _recipient;
        expiration = now + duration;
    }


    /// the recipient can close the channel at any time by presenting a
    /// signed amount from the sender. the recipient will be sent that amount,
    /// and the remainder will go back to the sender
    function close(uint256 amount, bytes memory signature) public {
        require(msg.sender == recipient);
        require(isValidSignature(amount, signature));

        recipient.transfer(amount);
        selfdestruct(sender);
    }

    /// if the timeout is reached without the recipient closing the channel,
    /// then the Ether is released back to the sender.
    function claimTimeout() public {
        require(now >= expiration);
        selfdestruct(sender);
    }
}
```

Bidirectional Payment Channel

Alice and Bob want to move funds back and forth

Two Unidirectional Channels?

Not as useful, Channels get exhausted

Bidirectional Payment Channel

Shared Account:
A: 0.5 ETH, B: 0.5 ETH Nonce 0

A: 0.6, Bob: 0.4 Nonce 1

Alice Bob

Bidirectional Payment Channel

Alice and Bob want to move funds back and forth

Shared Account:
A: 0.6 ETH, B: 0.4 ETH Nonce 1

A: 0.3, Bob: 0.7 Nonce 2

Alice Bob

Closing Payment Channel

Shared Account:
A: 0.3 ETH, B: 0.7 ETH Nonce 2

Before funding Alice and Bob get sign initial state

Alice submits balances and signatures to contract.

-> Starts challenge period

If Bob can submit tx with greater nonce: New state is valid.

Instant closing?

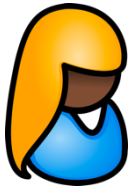
State Channels

Smart contract that implements a game between Alice and Bob

Game has a state

State Channels

Shared Contract:
State: Board state Nonce i



Can be used to move arbitrary 2 party contracts off chain

Payment Chanels with UTXOs

Problem: No state -> Can’t store nonce

Solution:

When updating the channel to Alices benefit,

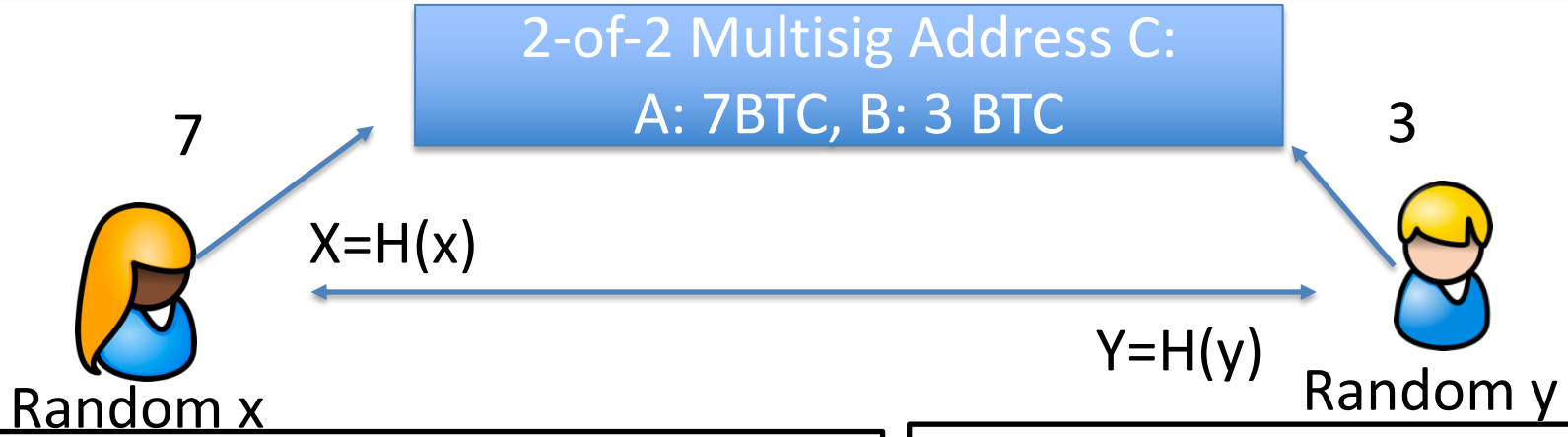
Alice gets TX that invalidates Bob’s old state

UTXO payment channel concepts

- **Relative time-lock**: output can be claimed t timesteps (i.e., blocks) from the time the TX is accepted to the blockchain
- **Hash lock**: Claiming output is pre-conditioned on providing the preimage of a cryptographic hash

Intuition: Both A and B hold TXs they can submit to settle the current split balance. Balance is updated by exchanging new TXs and "invalidating" old. Unilateral settlement is time-locked for one party, allows the other to challenge by providing hash-lock preimage. TXs invalidated by exchanging hash-lock preimages.

UTXO Payment Channel



TX1 from C:

Out1: Pay 7 -> A

Out2: Either 3 -> B (7 Day timelock)

Or 3 -> A y s.t. H(y)=Y

Alice

TX2 from C:

Pay 3 -> B

Either 7 -> A (7 Day timelock)

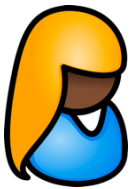
Or 7 -> B given x s.t. H(x)=X

Bob

UTXO Payment Channel Update

2-of-2 Multisig Address C:
A: 6 BTC, B: 4 BTC

x



Random x'

X'=H(x')

TX3 from C:

Out1: Pay 6 -> A

Out2: Either 4 -> B (7 Day timelock)

Or 4 -> A y s.t. H(y)=Y

Alice

TX4 from C:

Pay 4 -> B

Either 6 -> A (7 Day timelock)

Or 6 -> B given x s.t. H(x')=X'

Bob

Security

Alice has TX2,TX4

TX2 from C:

Pay 3 -> B

Either 7 -> A (7 Day timelock)

Or 7 -> B given x s.t. H(x)=X

Bob

TX4 from C:

Pay 4 -> B

Either 6 -> A (7 Day timelock)

Or 6 -> B given x' s.t. H(x')=X'

Bob

Bob has TX1,TX3, x

TX1 from C:

Pay 7 -> A

Either 3 -> B (7 Day timelock)

Or 3 -> A y s.t. H(y)=Y

Alice

TX3 from C:

Pay 6 -> A

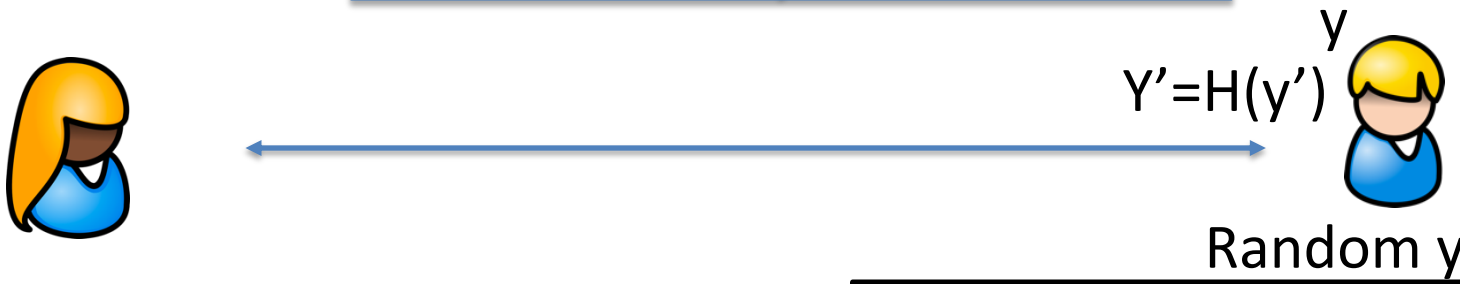
Either 4 -> B (7 Day timelock)

Or 4 -> A y s.t. H(y)=Y

Alice

UTXO Payment Channel Update

2-of-2 Multisig Address C:
A: 8 BTC, B: 2 BTC



TX5 from C:
Pay 8 -> A
Either 2 -> B (7 Day timelock)
Or 2 -> A y s.t. H(y')=Y'
Alice

TX6 from C:
Pay 2 -> B
Either 8 -> A (7 Day timelock)
Or 8 -> B given x s.t. H(x')=X'
Bob

Security

Alice has TX2,TX6, y

TX2 from C:

Pay 3 -> B

Either 7 -> A (7 Day timelock)

Or 7 -> B given x s.t. H(x)=X

Bob

TX6 from C:

Pay 2 -> B

Either 8 -> A (7 Day timelock)

Or 8 -> B given x s.t. H(x')=X'

Bob

Bob has TX3,TX5, x

TX3 from C:

Pay 6 -> A

Either 4 -> B (7 Day timelock)

Or 4 -> A y s.t. H(y)=Y

Alice

TX5 from C:

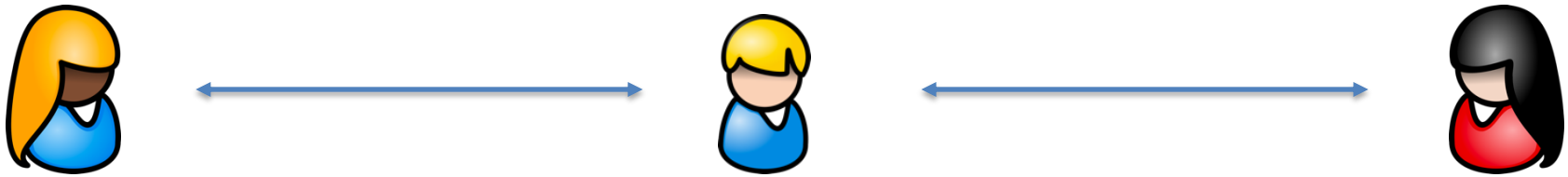
Pay 8 -> A

Either 2 -> B (7 Day timelock)

Or 2 -> A y s.t. H(y')=Y'

Alice

Multi-hop payments



Pay through *untrusted* intermediary

Multi-hop payments

R=H(r)

Pay 1.01 BTC to B
Hashlocked with R
Timelock to refund

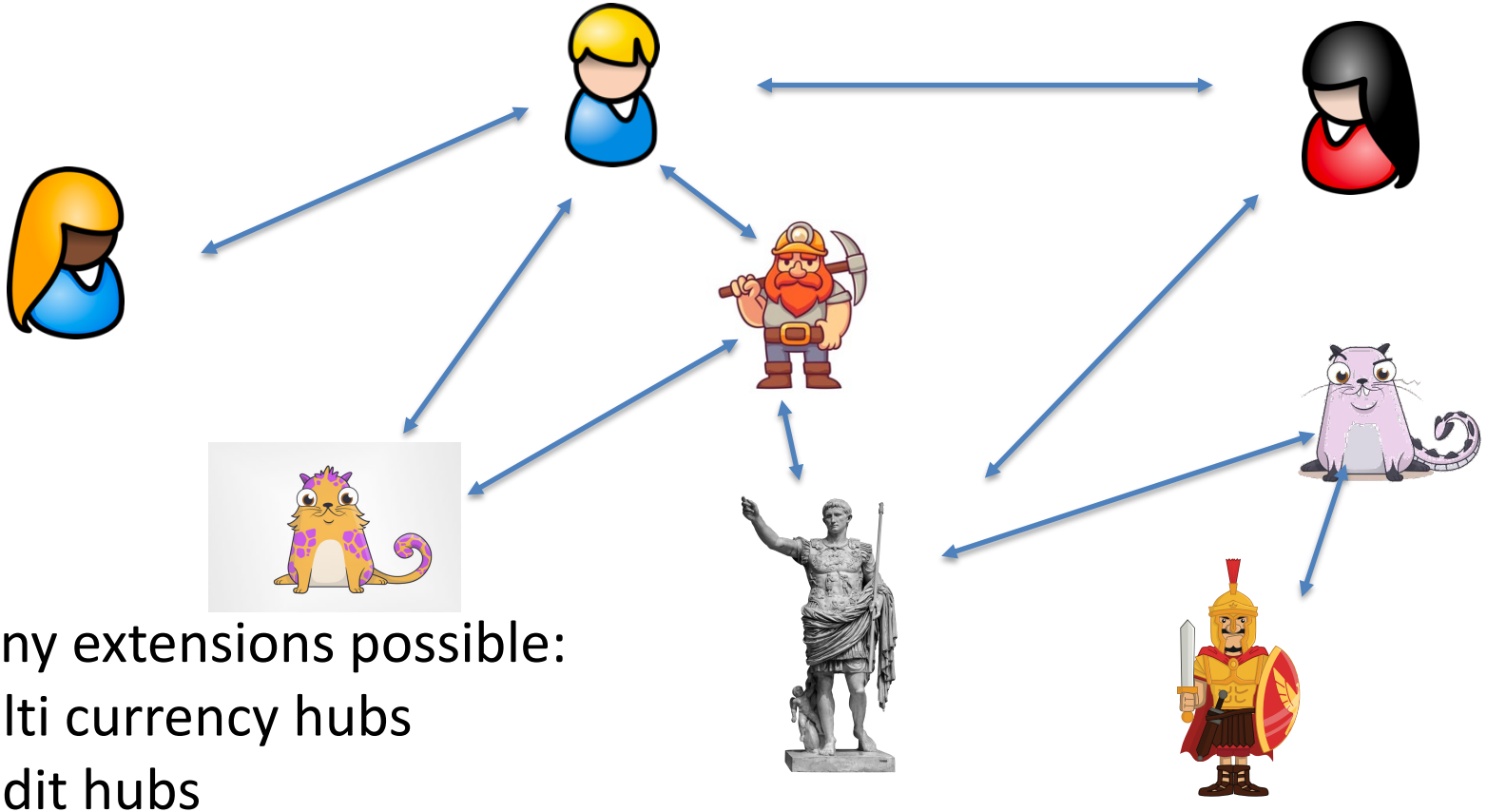
Pay 1 BTC to C
Hashlocked with R
Timelock to refund

Random r

B claims 1.01 BTC with r

C claims 1 BTC with r

Lightning network



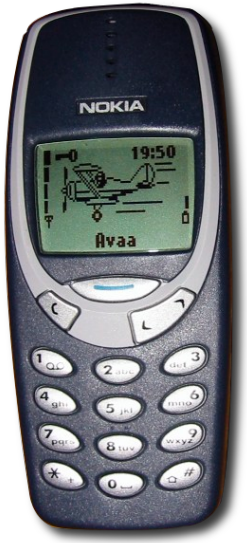
Many extensions possible:
Multi currency hubs
Credit hubs

Watchtowers

Lightning requires nodes to be periodically online to check for claim TX

Watchtowers outsource this task

User gives latest state to watchtower.



Trusted for availability
not custodian of funds
Risk of bribing

END OF LECTURE

Next lecture:

Scaling II: Accumulators and Rollup