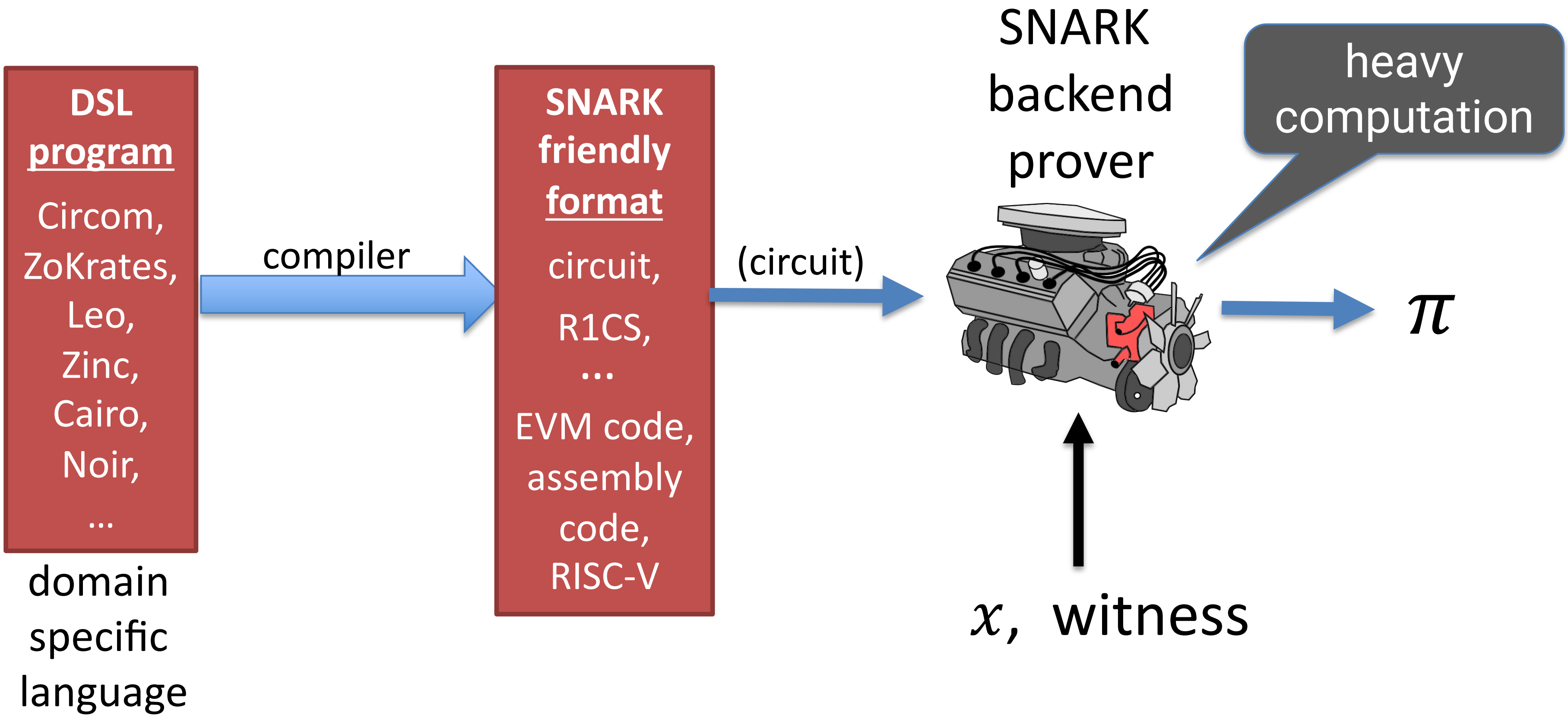
CS251 Fall 2023

(cs251.stanford.edu)

Scaling the blockchain part I: Payment Channels and State Channels

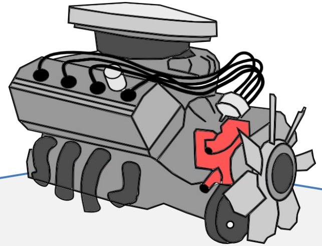
Dan Boneh

… but first, last words on SNARKs (for now)



… but first, last words on SNARKs (for now)

circuit →
x, witness →



→ π

circuit →
x, witness →

arithmetization →

Polynomial Relations

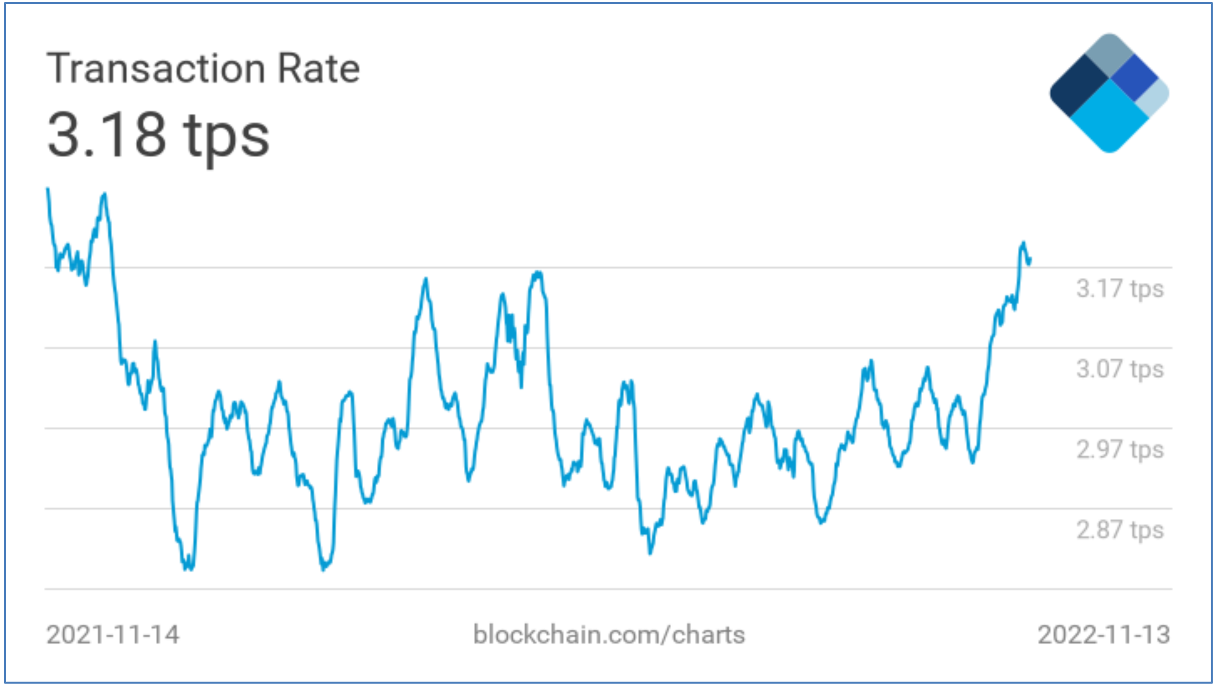
$$\forall y \in H \subseteq \mathbb{F}:$$
$$f_1(y) = 0$$
$$f_2(y) = 0$$

→ π

proves polynomial relations

Scaling the blockchain

Bitcoin Tx per second



≈4200 Tx/block

1 block / 10 mins

⇒ max: 7 Tx/sec

Ethereum Tx per second

Ethereum avg Tx per second:

≈ 15 Tx/sec

Simple Tx: 21k Gas
max 30M Gas per block
$\Rightarrow$ max 1428 tx/block

1 Block/12s
$\Rightarrow$ max 119 tx/s

In comparison …

Visa: up to 24,000 Tx/sec (regularly 2,000 Tx/sec)

PayPal: 200 Tx/sec

Ethereum: 15 Tx/sec

Bitcoin: 7 Tx/sec

Goal: scale up blockchain Tx speed

How to process more Tx per second

Many ideas:

reduced composability

- Use a faster consensus protocol
- Parallelize: split the chain into independent **shards**
- Rollup: move work somewhere else (next lecture)
- Today: payment channels, reduce the need to touch the chain

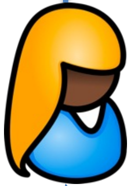
Payment Channels: the basic idea

Tx1: 0.01 ETH

Tx2: 0.01 ETH

Tx3: 0.01 ETH

Tx fee
per purchase!



Instead can do this:

Alice deposits 1 ETH with Bob.
At the end of the month, Bob refunds unused deposit to Alice.

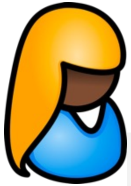
only <u>two</u> Tx,
hundreds of
coffees

Unidirectional Payment Channel

Alice creates:

Contract A:
1 ETH

Bob does not post on chain



Post Tx3 on Blockchain (close channel)

Tx1: send 0.99 to Alice / 0.01 to Bob from Contract A
signed by Alice

Tx2: send 0.98 to Alice / 0.02 to Bob from Contract A
signed by Alice

Tx3: send 0.97 to Alice / 0.03 to Bob from Contract A
signed by Alice

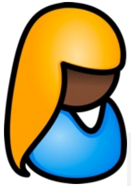
Problem: Alice could post Tx1 before Bob even though she bought three coffees.

A solution?

Alice creates:

Contract A:
1 ETH

Only Bob can close the channel



Post Tx3 on Blockchain
(close channel)

Tx1: send 0.99 to Alice / 0.01 to Bob from Contract A
signed by Alice

Tx2: send 0.98 to Alice / 0.02 to Bob from Contract A
signed by Alice

Tx3: send 0.97 to Alice / 0.03 to Bob from Contract A
signed by Alice

Problem: What if Bob never publishes Tx3?
⇒ Alice never gets her 0.97 ETH back !!

Unidirectional Payment Channel

Alice needs a way to ensure refund if Bob disappears

Solution: create a channel that can be closed in one of two ways

- **Normal close Tx**: Sends 0.97 to Alice / 0.03 to Bob

  … requires signatures by both Alice and Bob.

- **Timelock Tx**: Sends 1 ETH to Alice

  … requires signature by Alice,

  but is accepted 7 days after channel is created

Unidirectional Payment Channel

After 6 days:

- Bob can close channel by signing and posting Tx3.

After 7 days:

- Alice can close channel using timelock Tx, gets back her 1 ETH.

- Timelock period determines the lifespan of channel

- Once Alice sends the full 1 ETH to Bob, the Channel is ”exhausted”

Payment Channel in Solidity

```
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);
    }
}
```

Alice creates contract with funds, specifies timelock and recipient

verify Alice's signature on final amount.
Only Bob can call close() !!

send all funds to sender <u>after</u> timelock

Bidirectional Payment Channel

Alice and Bob want to move funds back and forth

Two Unidirectional Channels?

Not as useful because Channels get exhausted

Bidirectional Payment Channel

On Ethereum: create a shared contract, each contributes 0.5 ETH:

channel state: A: 0.5 ETH, B: 0.5 ETH, Nonce=0

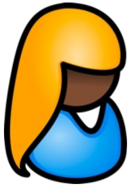
Off chain: Bob sends 0.1 ETH to Alice by both signing new state:

new state: A: 0.6, Bob: 0.4, Nonce=1
Alice sig, Bob sig

Bidirectional Payment Channel

On chain contract does not change:

balance: 1 ETH,   Nonce=0



Off chain: Alice and Bob can move funds back and forth by sending updated state sigs to each other:

A: 0.3, Bob: 0.7, Nonce=7
Alice sig, Bob sig

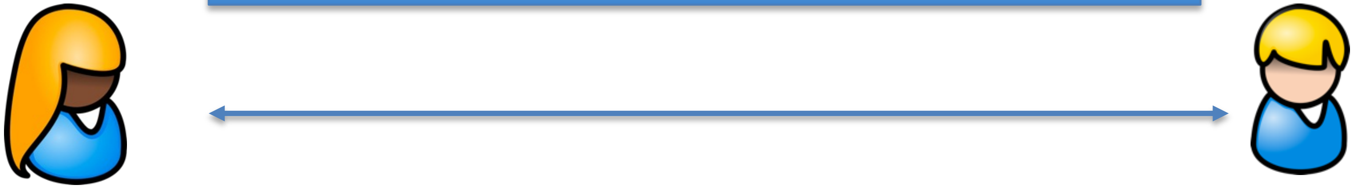
(7th transfer)

Eventually: Alice wants to close payment channel

Alice does: sends latest balances and signatures to contract

⇒ starts challenge period (say, 3 days)

on chain: A: 0.3 ETH, B: 0.7 ETH, Nonce=7 (pending close)



if Bob does nothing for 3 days:

⇒ funds disbursed according to Alice's submitted state

if Bob submits signed state with a higher nonce (e.g., nonce=9)

⇒ funds disbursed according to Bob's submitted state

Watchtowers

Bidirectional channel requires Bob to constantly check that Alice did not try to close the channel with an old stale state

$\Rightarrow$ post latest state if she did

Watchtowers outsource this task

Trusted for availability

Bob sends latest state to watchtower.

Main points: summary

Payment channel between Alice and Bob:

- <u>**One**</u> **on-chain** Tx to create channel (deposit funds);
- Alice & Bob can send funds to each other **off-chain**
  … as <u>many</u> Tx as they want;
- <u>**One**</u> **on-chain** Tx to close channel and disburse funds

$\Rightarrow$ only two on-chain Tx

A more general concept: State Channels

Smart contract that implements a game between Alice and Bob.

Begin game & end game: on chain. **All moves are done off-chain.**

State Channels

Can be used to implement any 2-party contract off chain!

two Tx on-chain: contract creation and termination

Bidirectional channels on Bitcoin

The Lightning Network

Bidirectional payment channels on Bitcoin

Problem: no updatable state in UTXOs $\Rightarrow$

much harder to implement a bidirectional channel

Solution:

- When updating the channel to Alice's benefit,
  Alice gets TX that invalidates Bob's old state

UTXO payment channel concepts

Will create UTXO that can be spent in one of two ways: (using IF opcode)

- **Relative time-lock**: UTXO contains a positive number t.
  A properly signed Tx can spend this UTXO
  t blocks (or more) after it was created (CLTV opcode)

- **Hash lock**: UTXO contains a hash image X.
  A properly signed Tx can spend this UTXO immediately
  by presenting x s.t. X = SHA256(x).

(x is called a hash preimage of X)

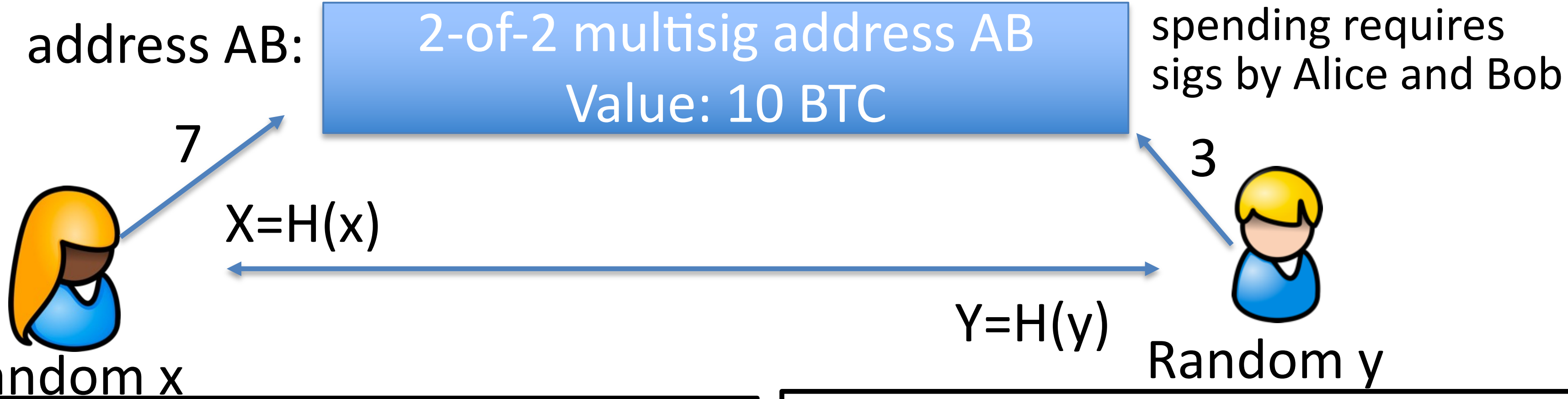
Example script

Example locktime redeem script: two ways to redeem UTXO

```
OP_IF       // Alice can redeem UTXO any time using a preimage
    OP_HASH256  <digest>  OP_EQUALVERIFY              // redeem by providing <digest> preimage,
    DUP  HASH256  <AlicePKhash>  EQVERIFY  CHECKSIG   // and Alice's signature
OP_ELSE     // Bob can redeem UTXO only after timelock
    <num>  OP_CLTV  OP_DROP                           // redeem <num> blocks after UTXO created,
    DUP  HASH256  <BobPKhash>  EQVERIFY  CHECKSIG     // and Bob's signature
OP_ENDIF
```

This is called a **hash-timelock contract** (HTLC).

UTXO Payment Channel



TX1: input: UTXO for address AB

Out1: pay 7 → A

Out2: either 3 → B, 7 day timelock

or 3 → A now, given y s.t. H(y)=Y

Alice sig

TX2: input UTXO for address AB

pay 3 -> B

either 7 → A, 7 day timelock

or 7 → B now, given x s.t. H(x)=X

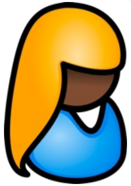
Bob sig

Alice can sign and post Tx2, wait 7 days, and get her funds back

Payment Channel Update: Alice pays Bob

2-of-2 Multisig UTXO AB
Value: 10 BTC

x 



X’=H(x’)

Alice sends 1BTC to Bob (off chain)

Random x’

TX3 input: UTXO for address AB
Out1: pay 6 → A
Out2: either 4 → B, 7 day timelock
or 4 → A now, given y s.t. H(y)=Y
Alice sig

TX4 input: UTXO for address AB
pay 4 → B
either 6 → A, 7 day timelock
or 6 → B now, given x’ s.t. H(x’)=X’
Bob sig

Security: ways to close the channel?

Alice has TX2,TX4, x, x’

Bob has TX1,TX3, y, x

TX2: (stale state) pay 3 → B either 7 → A, 7 day timelock or 7 → B now, given x s.t. H(x)=X Bob sig	TX1: (stale stale) pay 7 → A either 3 → B, 7 day timelock or 3 → A now, given y s.t. H(y)=Y Alice sig
TX4: (current state) pay 4 → B either 6 → A, 7 day timelock or 6 → B now, given x’ s.t. H(x’)=X’ Bob sig	TX3: (current state) pay 6 → A either 4 → B, 7 day timelock or 4 → A now, given y s.t. H(y)=Y Alice sig

Security: ways to close the channel?

Alice has TX2,TX4, x, x'

Bob has TX1,TX3, y, x

TX2:

pay 3 →

either 7

or 7 → B

Bob sig

ock

H(y)=Y

The good case:

Alice can post Tx4 or Bob can post Tx3 to chain and close channel after 7 days

A gets 6, B gets 4

TX4: (current state)

pay 4 → B

either 6 → A, 7 day timelock

or 6 → B now, given x' s.t. H(x')=X'

Bob sig

TX3: (current state)

pay 6 → A

either 4 → B, 7 day timelock

or 4 → A now, given y s.t. H(y)=Y

Alice sig

Security: ways to close the channel?

Alice has TX2,TX4, x, x’

TX2: (stale state)

pay 3 → B

either 7 → A, 7 day timelock

or 7 → B now, given x s.t. H(x)=X

Bob sig

Bob has TX1,TX3, y, x

TX1: (stale state)

pay 7 → A

either 3 → B, 7 day timelock

or 3 → A now, given y s.t. H(y)=Y

Alice sig

TX

pa

eit

or

Bob sig

Alice sig

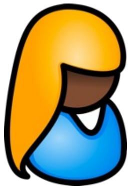
The bad case (Alice cheats):

if Alice posts the stale Tx2 then Bob will use x to take all 10 BTC

⇒ sending x to Bob revokes the stale Tx2 held by Alice

Payment Channel Update: Bob pays Alice

2-of-2 Multisig Address C:
Value: 10 BTC



y

Y'=H(y')

Bob sends 2BTC to Alice (off chain)

Random y'

TX5 input: UTXO for address AB
pay 8 → A
either 2 → B,   7 day timelock
or 2 → A now, given y s.t. H(y')=Y'
Alice sig

TX6 input: UTXO for address AB
pay 2 → B
either 8 → A,   7 day timelock
or 8 → B now, given x s.t. H(x')=X'
Bob sig

Security: ways to close the channel?

Alice has TX2,TX6, x, x', y

Bob has TX3,TX5, y, y', x

TX2:
pay 3 → B
either 7 → A, 7 day timelock
or 7 → B now, given x s.t. H(x)=X
Bob sig

TX6:
pay 2 → B
either 8 → A, 7 day timelock
or 8 → B now, given x s.t. H(x')=X'
Bob sig

TX3:
pay 6 → A
either 4 → B, 7 day timelock
or 4 → A now, given y s.t. H(y)=Y
Alice sig

TX5:
pay 8 → A
either 2 → B, 7 day timelock
or 2 → A now, given y s.t. H(y')=Y'
Alice sig

Security: ways to close the channel?

Alice has TX2,TX6, x, x’, y

TX2:

pay 3 → B

either 7 → A, 7 day timelock

or 7 → B now, given x s.t. H(x)=X

Bob sig

TX6:

pay 2 → B

eith

or 8

Bob sig

Bob has TX3,TX5, y, y’, x

TX3:

pay 6 → A

either 4 → B, 7 day timelock

or 4 → A now, given y s.t. H(y)=Y

Alice sig

TX5:

pay 8 → A

=Y’

Alice sig

The bad case (Bob cheats):

Bob posts the stale Tx3 ⇒ Alice will use y to take all 10 BTC

Watchtowers again

Bidirectional channel requires Bob to constantly check that Alice did not try to close the channel with an old stale state

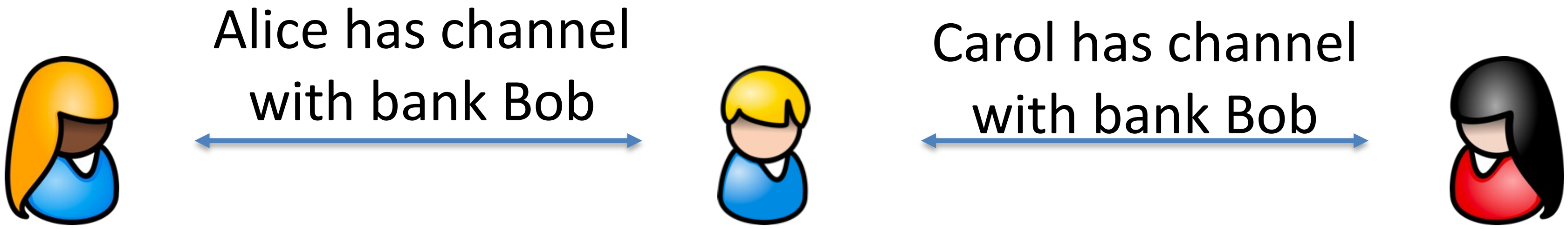
$\Rightarrow$ use hashlock value if she did

Trusted for availability

Bob needs to always send latest hashlock value to watchtower.

Multihop payments

Multi-hop payments



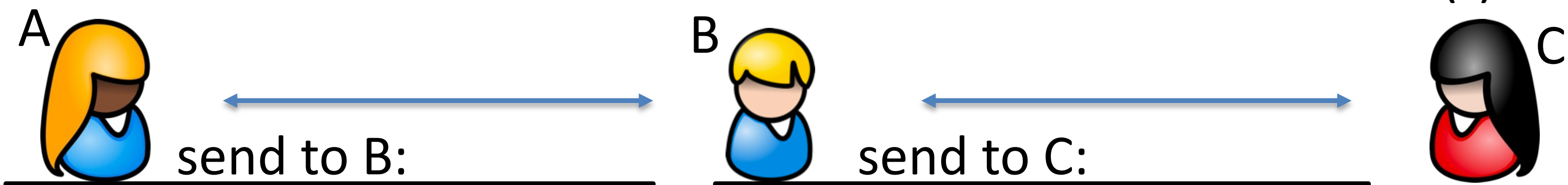
Alice wants to pay Carol 1 BTC through *untrusted* intermediary Bob

How: (i) Alice pays Bob 1.01 BTC, (ii) Bob pays Carol 1 BTC

The challenge: steps (i) and (ii) need to be atomic

Multi-hop payments (briefly)

R=H(r)



Pay 1.01 BTC to B:	Pay 1 BTC to C:
Hashlocked to B with R,	Hashlocked to C with R,
Timelock to A for refund	Timelock to B for refund
Alice sig	*Bob sig*

Random r

Then B can claim 1.01 BTC with r

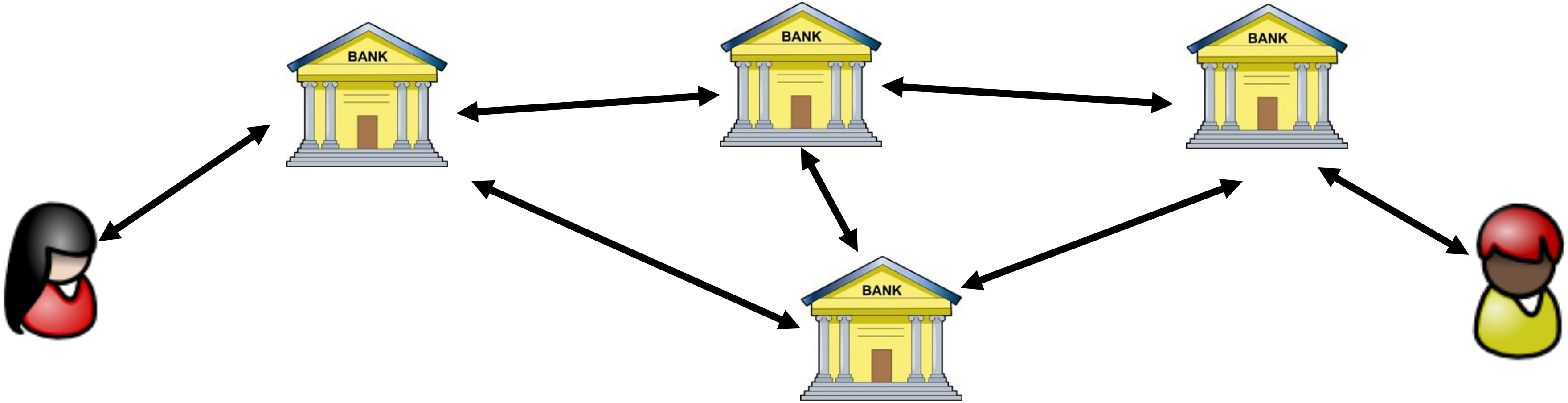
C can claim 1 BTC on-chain with r
$\Rightarrow$ r is publicly known

if Carol never claims, Alice & Bob get funds back after timelock

The lightning network

The network: lots of open bi-directional payment channels.

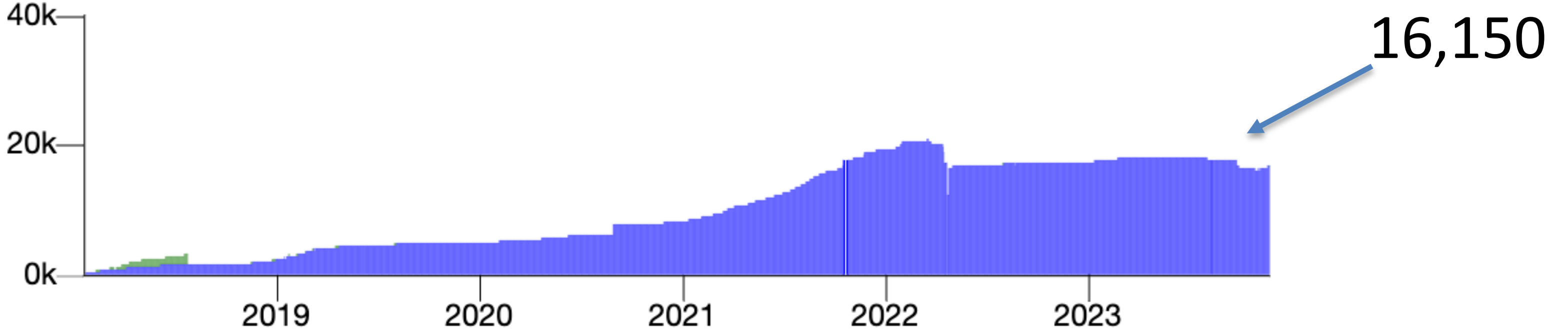
Alice wants to pay Bob: she finds a route to Bob through the graph



Many extensions possible: multi currency hubs, credit hubs, …

Stats

nodes in lightning network (Nov. 2023)



Number of channels: 63K

Network capacity: ≈$205M

END OF LECTURE

Next lecture: scaling via Rollups