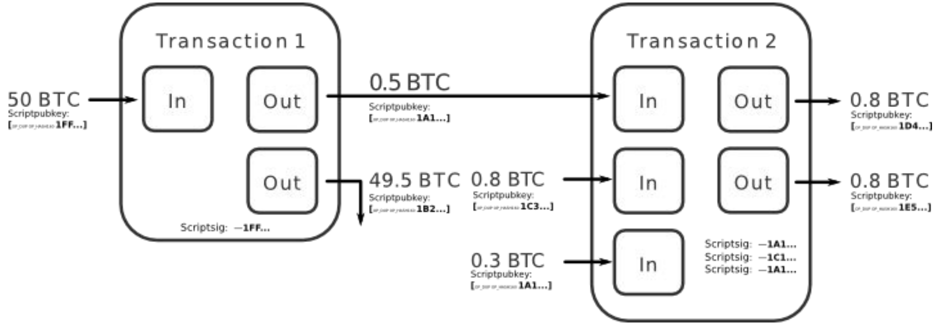
The Lightning Network

LINCS - Paris 2019

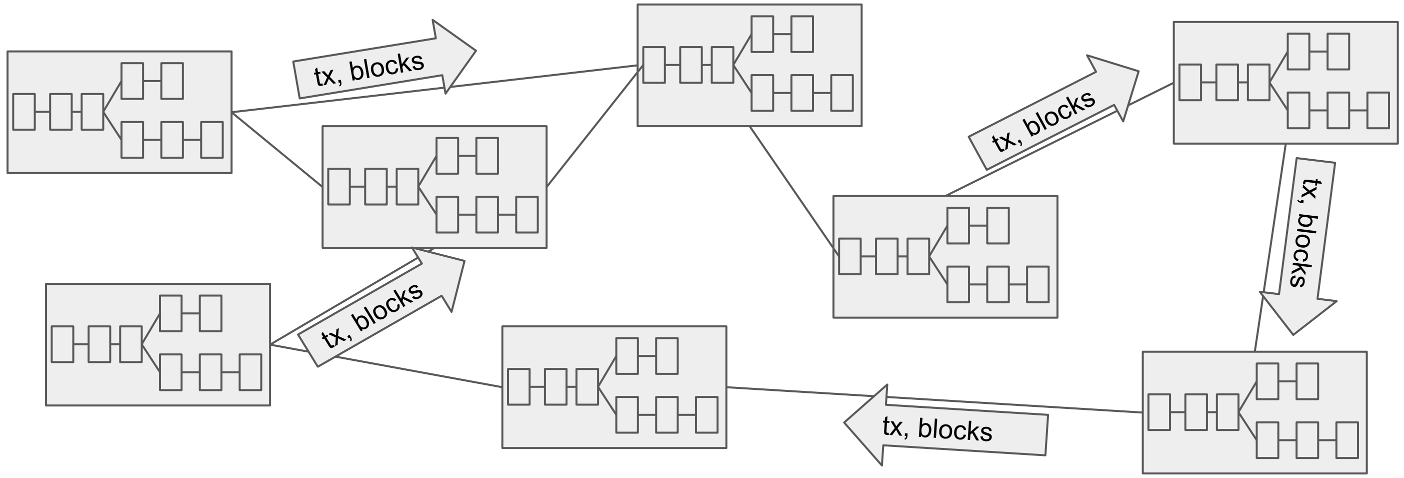
Bastien Teinturier <bastien@acinq.fr> - https://acinq.co/

How does Bitcoin Work?

- Blockchain
  - Transactions grouped into blocks
  - Blocks include the id of the previous block
  - hash(block) < target
- UTXO
  - The “coins” in Bitcoin
  - Transactions spend the outputs of previous transactions
  - UTXO set + block -> new UTXO set
- Scripts
  - Bitcoin has smart contracts too!
  - Set the rules for spending a UTXO



Bitcoin doesn't scale?



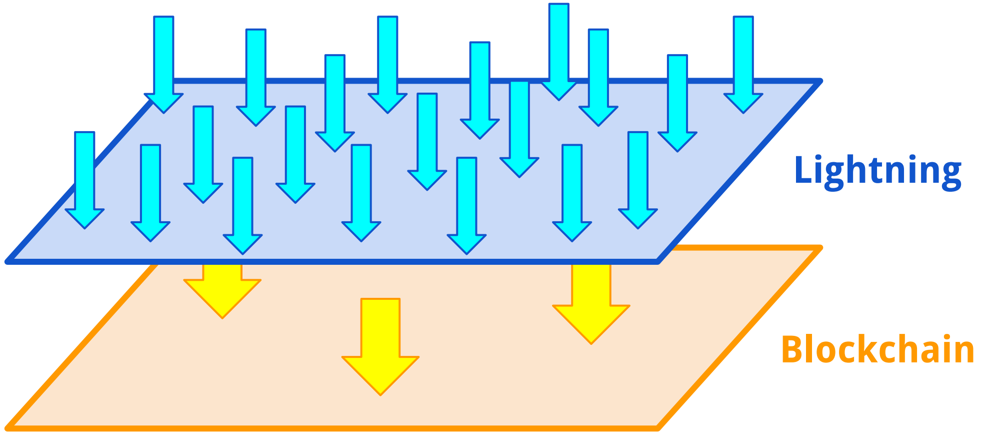
Global Consensus: everyone sees, validates and stores everything, and must agree on the same blockchain.

The Lightning Network

What if we don’t publish everything?

The Lightning Network

- P2P Network of payment channels
- Based on Bitcoin (LN transactions are Bitcoin transactions)
- Enables cheap, instant payments (and micropayments)
- Extremely scalable
- And still trustless

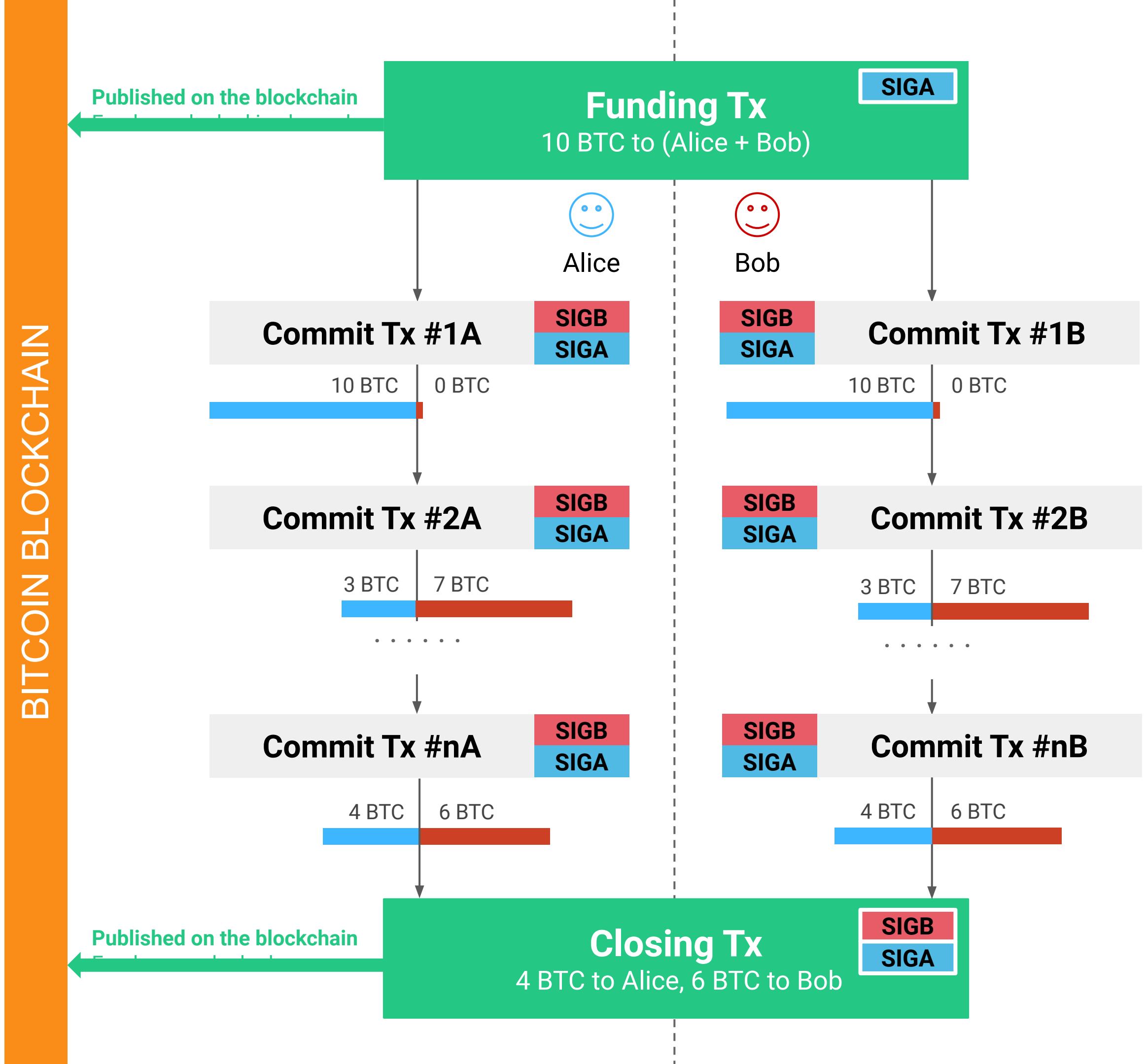


“Layer 2” application based on Bitcoin, with different properties/tradeoffs

The Lightning Network

- Payment Channel
  - Payment model: HTLC
  - Trustless
- Network of Payment Channels
  - Conditional payments: A pays B iff B pays C
  - And still trustless
- Routing
  - Source routing
  - Onion packets

Payment Channel



OPEN

First tx is published on the blockchain.
Funds are “locked” in the channel

UPDATE

Publishable (signed by both parties) but **not published**!

CLOSE

Last tx is published on the blockchain.
Funds are “unlocked”

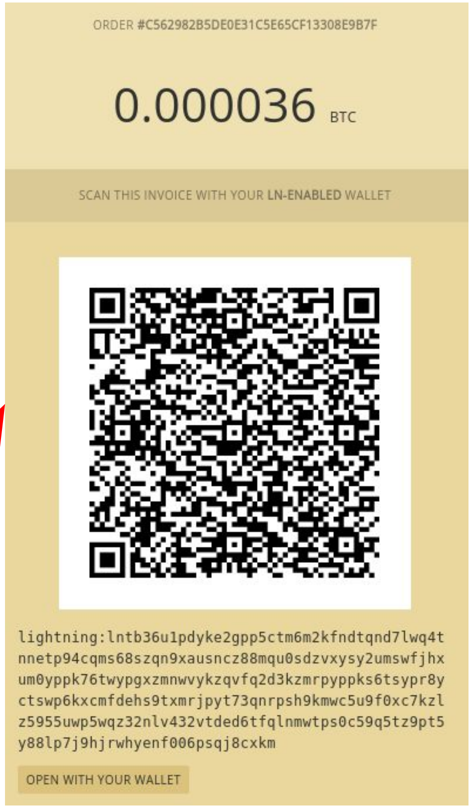
Payment Channel

Payment Model: **Hashed Time Locked Contract (HTLC)**

- I will pay you for the preimage of **hash**
- I you don’t reply, I get my money back **after a delay**

Lighting Payment Request: Amount + Hash + Delay

Description = 1 Espresso Coin Panna, 1 Scala Chip Frappuccino
H = c2f7adaac99b5609b7df702ab9cf2b096b806e1a3c040994dde427811cfb071f
NodeId = 035b55e3e08538afeef6ff9804e3830293eec1c4a6a9570f1e96a478dad1c86fed
Amount = 3600000 MilliSatoshis
Timestamp = 1514890568



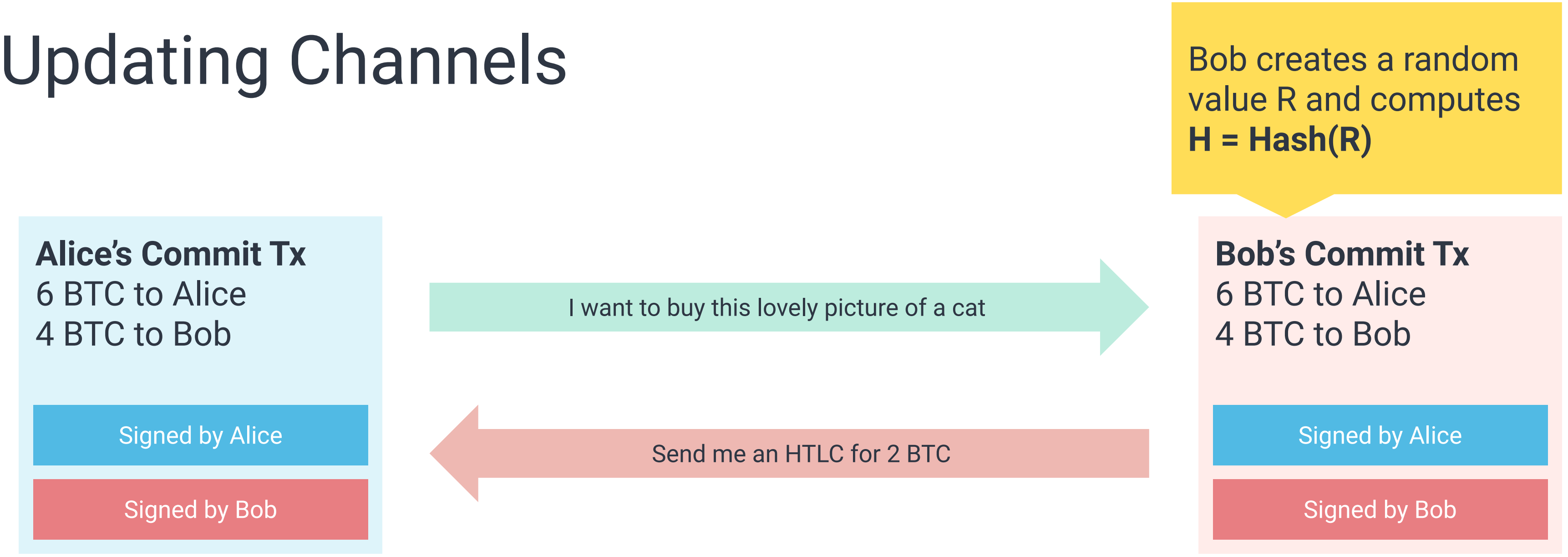
ORDER **#C562982B5DE0E31C5E65CF13308E9B7F**

0.000036 BTC

SCAN THIS INVOICE WITH YOUR **LN-ENABLED** WALLET

```
lightning:lntb36u1pdyke2gpp5ctm6m2kfndtqnd7lwq4t
nnetp94cqms68szqn9xausncz88mqu0sdzvxysy2umswfjhx
um0yppk76twypgxzmnwvykzqvfq2d3kzmrpyppks6tsypr8y
ctswp6kxcmfdehs9txmrjpyt73qnrpsh9kmwc5u9f0xc7kzl
z5955uwp5wqz32nlv432vtded6tfqlnmwtps0c59q5tz9pt5
y88lp7j9hjrwhyenf006psqj8cxkm
```

Updating Channels



Bob creates a random value R and computes **H = Hash(R)**

Alice's Commit Tx
6 BTC to Alice
4 BTC to Bob
Signed by Alice
Signed by Bob

I want to buy this lovely picture of a cat

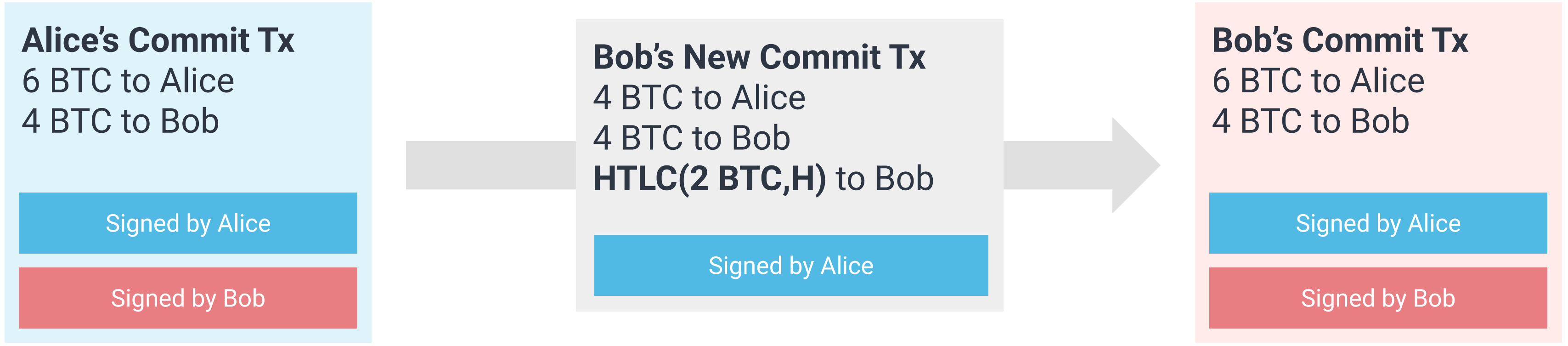
Send me an HTLC for 2 BTC

Bob's Commit Tx
6 BTC to Alice
4 BTC to Bob
Signed by Alice
Signed by Bob

- Alice wants **to buy a picture** of a cat from Bob
- Bob says "**send me an HTLC** for 2 BTC redeemable **with the preimage of H**"
- This dialog happens **off-band** (web pages, QR codes, ...)

Updating Channels

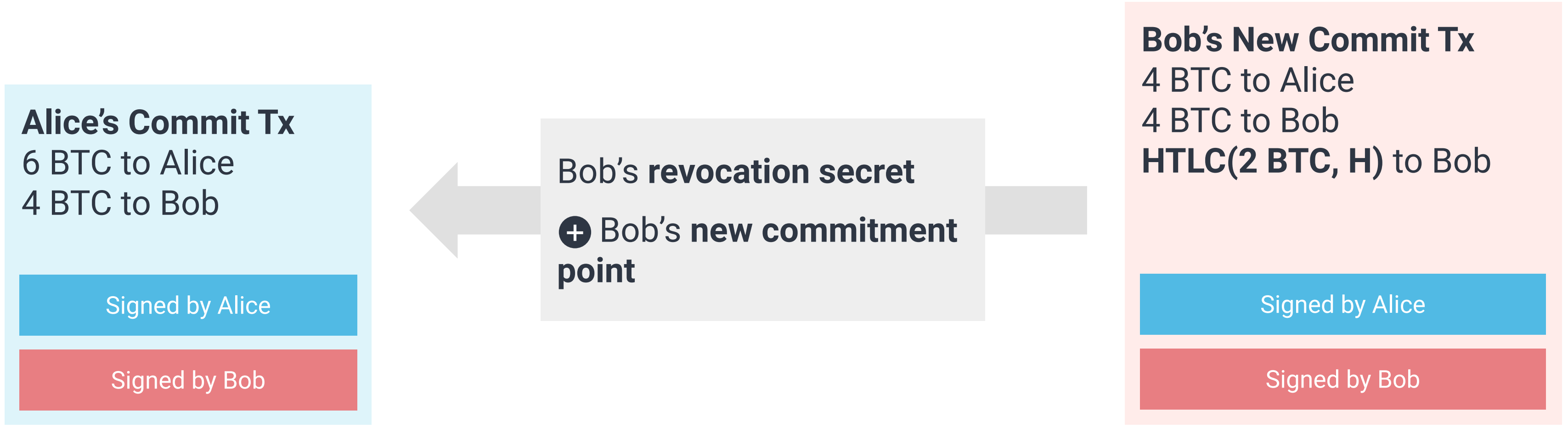
exchanging signatures



- Alice creates a **new Commit Tx for Bob**, which includes the HTLC
- Alice signs Bob's new Commit Tx and send it to Bob

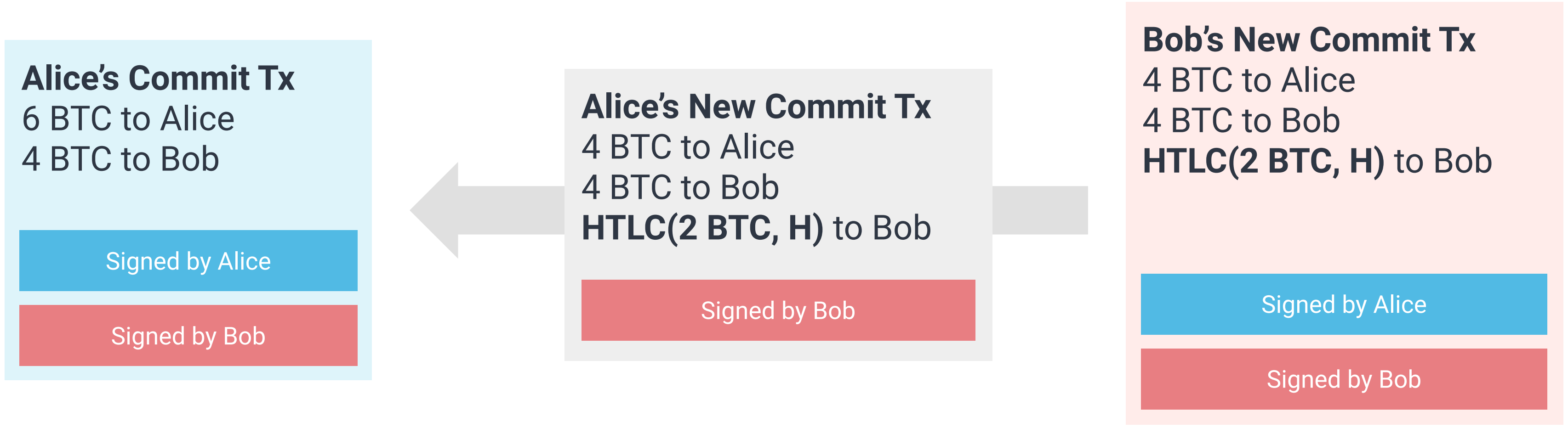
Updating Channels

exchanging signatures



- Bob checks that **Alice's signature is valid**.
- Bob now has a **valid new commit tx** that includes the HTLC
- Bob replies with the **revocation secret for his old commitment tx**
- Alice checks that the **revocation secret is valid**. Bob cannot publish his old tx anymore

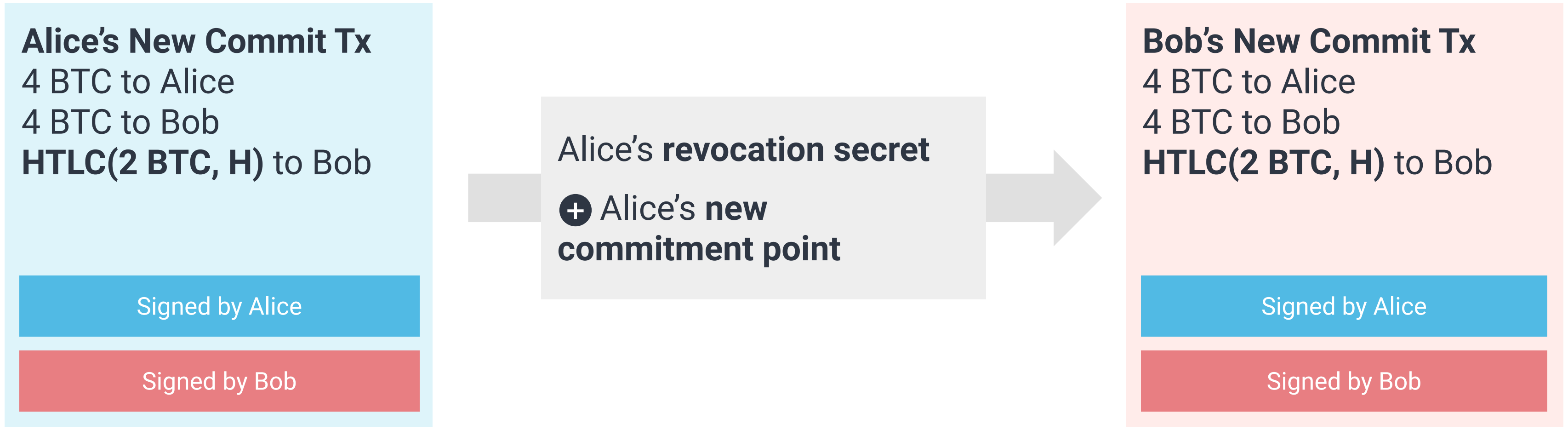
exchanging signatures



- Bob creates a **new Commit Tx for Alice**, which includes the HTLC
- Bob signs Alice's new Commit Tx and send it to Alice

Updating Channels

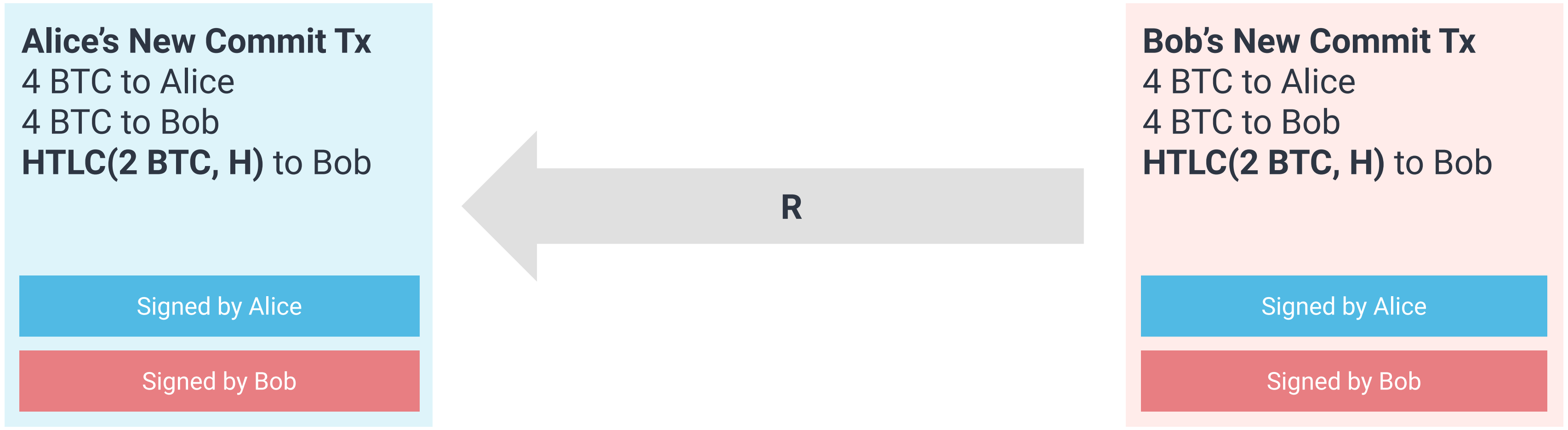
exchanging signatures


Alice’s New Commit Tx
4 BTC to Alice
4 BTC to Bob
HTLC(2 BTC, H) to Bob
Signed by Alice
Signed by Bob
Alice’s **revocation secret**
+ Alice’s **new commitment point**
Bob’s New Commit Tx
4 BTC to Alice
4 BTC to Bob
HTLC(2 BTC, H) to Bob
Signed by Alice
Signed by Bob

- Alice checks that **Bob’s signature is valid**.
- Alice now has a valid new commit tx that **includes the HTLC**
- Alice replies with the **revocation secret for his old commitment tx**
- Bob checks that the **revocation secret is valid**. Alice cannot publish her old tx anymore

Updating Channels

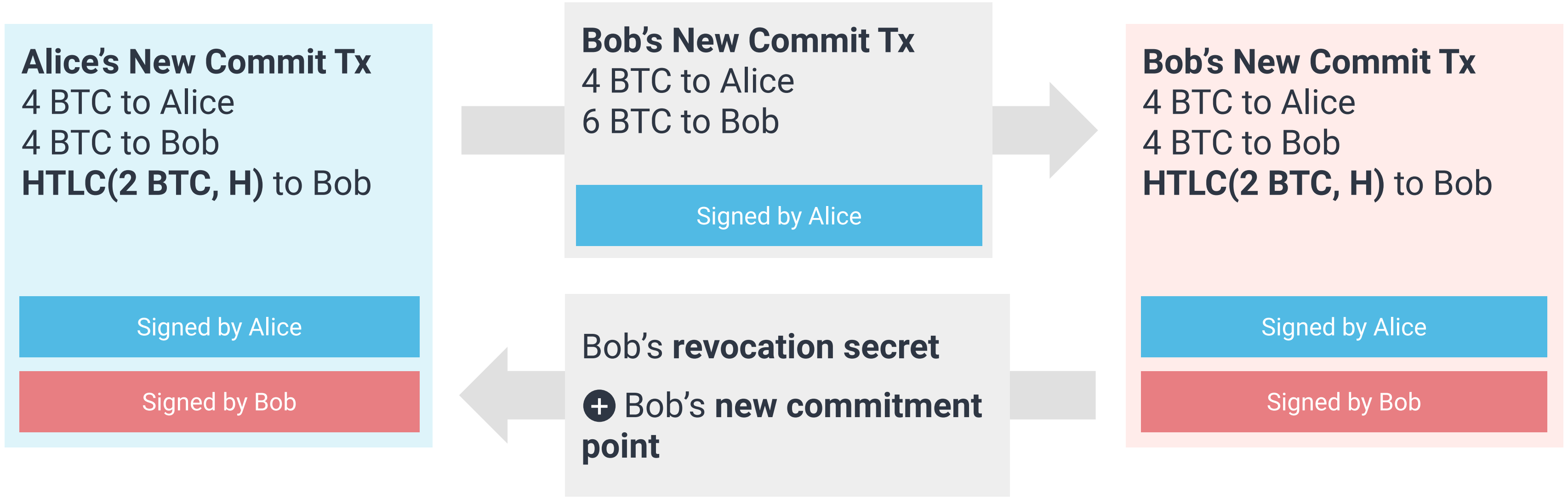
fulfilling HTLCs



- Bob sends **R** to Alice
- Alice checks that **Hash(R) == H**

Updating Channels

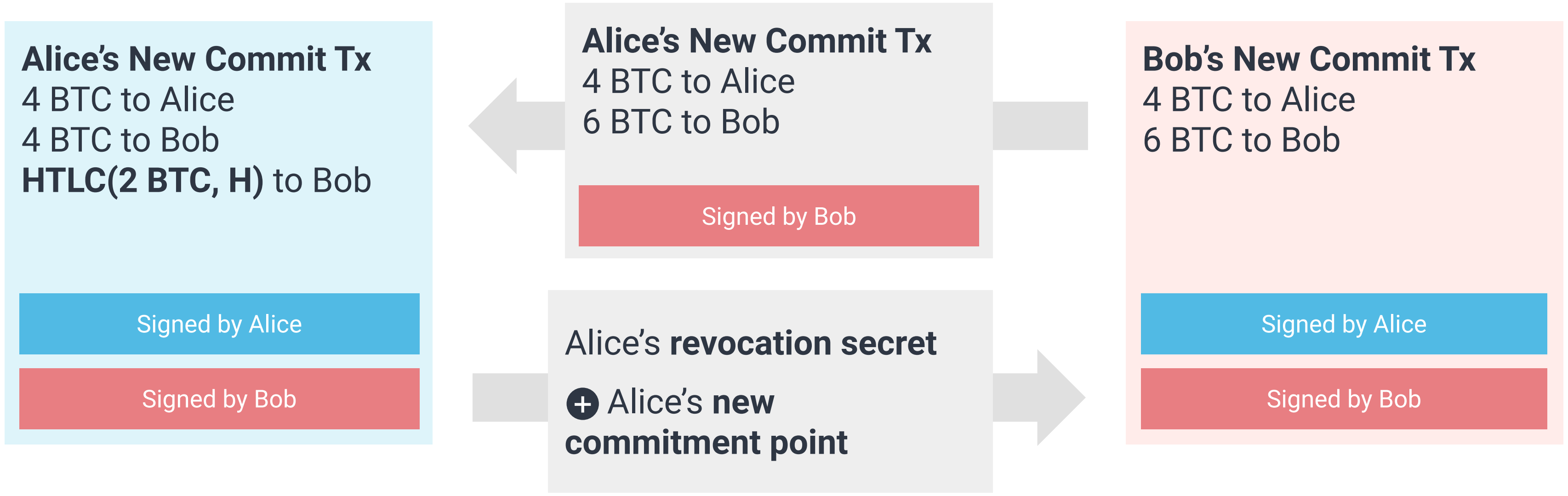
exchanging signatures



- Alice create a new Commit Tx for Bob which **updates his balance and sends her signature to Bob**
- Bob checks the signature and **replies with his revocation secret**

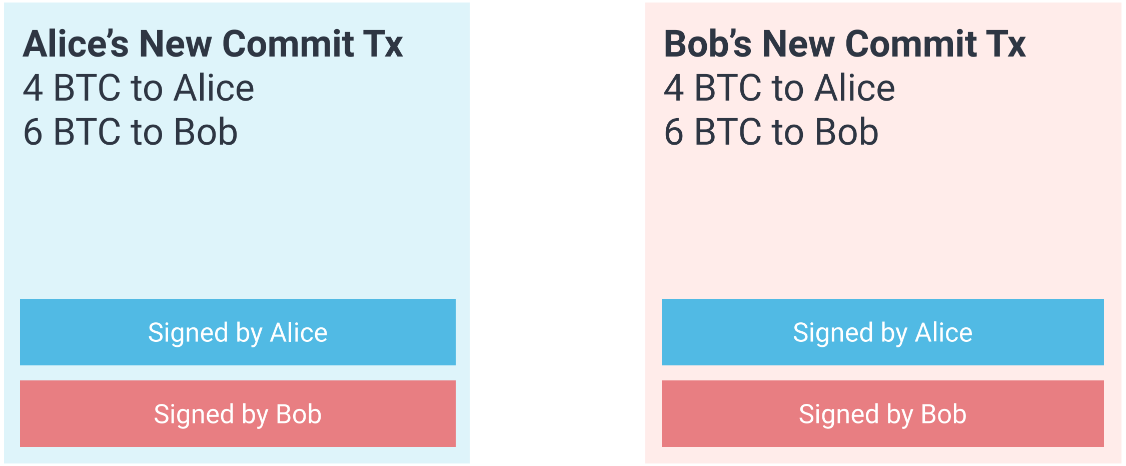
Updating Channels

exchanging signatures



- Bob creates a **new Commit Tx for Alice**, with updated balances
- Bob signs Alice's new Commit Tx and send it to Alice
- Alice checks the signature and **replies with her revocation secret**

fully signed commit tx



Alice and Bob now have **fully signed commit tx with updated channel balances**

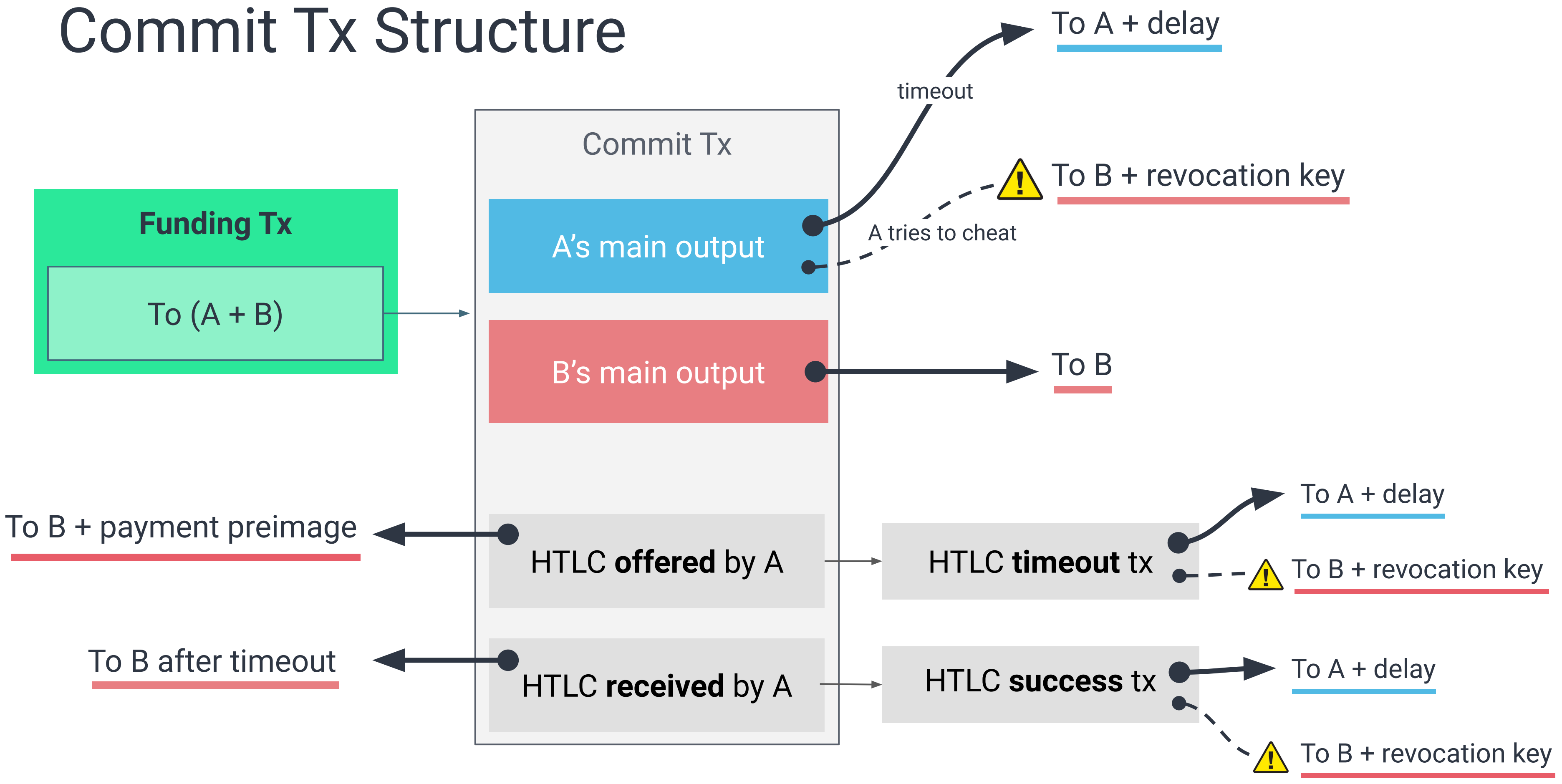
Payment Channel

- Why is this trustless?
- What if Alice or Bob want to publish one of their old commit tx that gives them more money than the current one?
  - After all, each of their commit tx is valid and publishable….

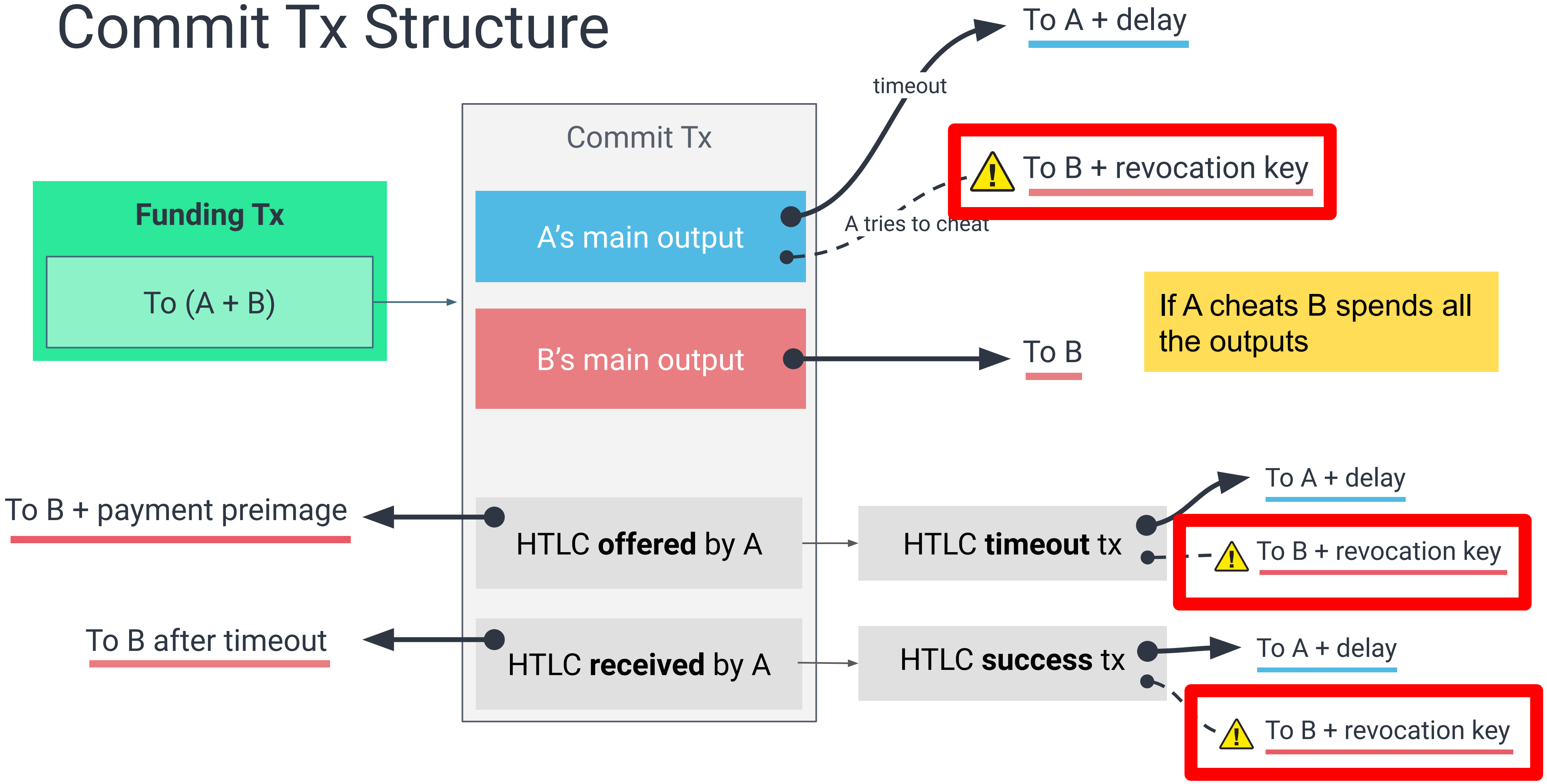
Payment Channel

- Why is this trustless?
- What if Alice or Bob want to publish one of their old commit tx that gives them more money than the current one?
  - After all, each of their commit tx is valid and publishable…
- Penalty Transaction!
  - In Alice’s commit tx, each output that sends money to her includes a “circuit breaker”
  - This “circuit breaker” says: send everything to Bob if he knows a **revocation secret**
  - When Bob signs Alice’s new commit tx, she sends back the revocation secret for her current commit tx
  - **You can only publish your latest commit transaction!**
    - Or you risk losing everything

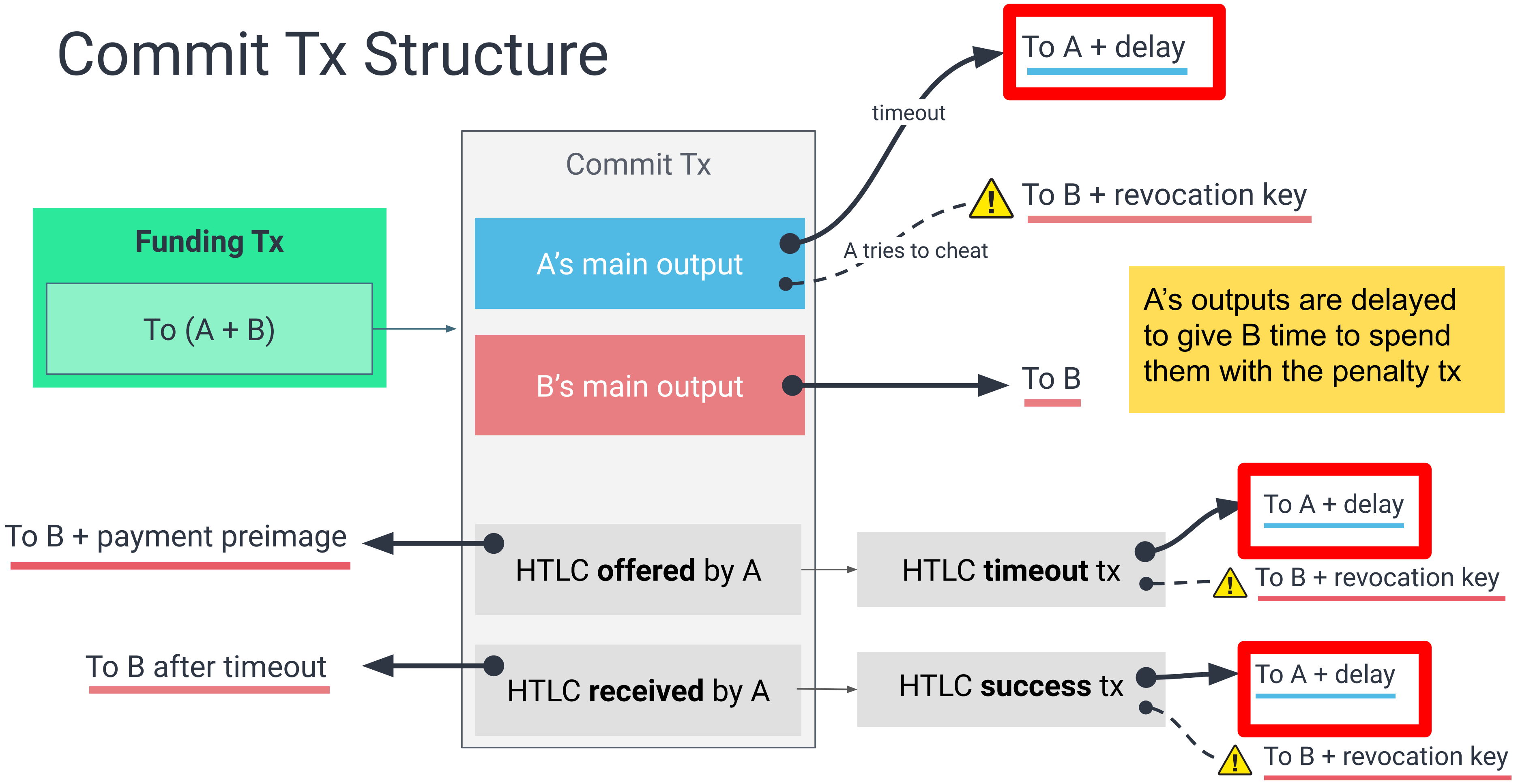
Commit Tx Structure



Commit Tx Structure

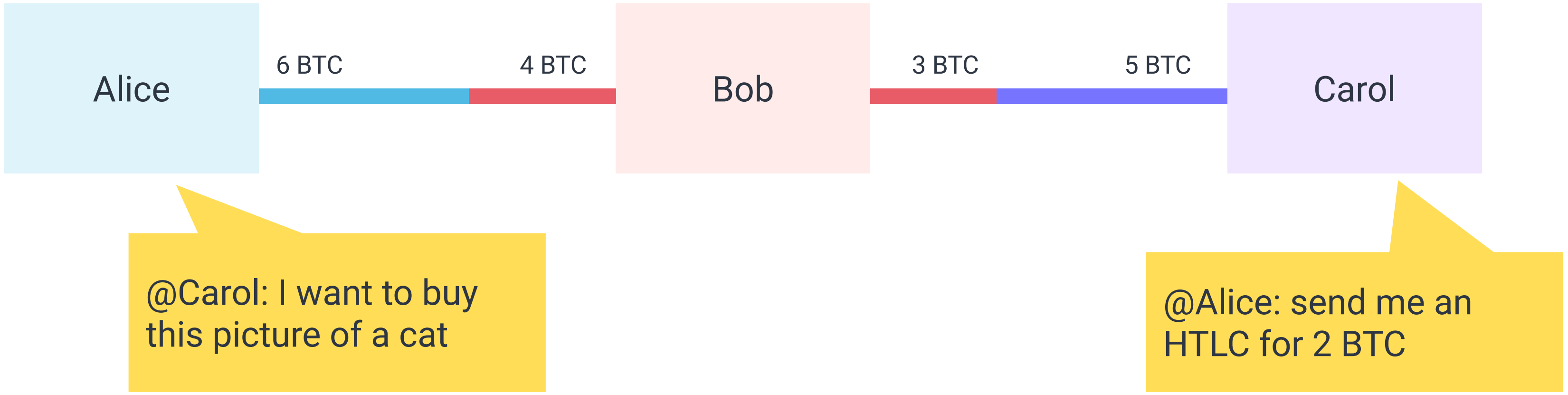


Commit Tx Structure



Network of Payment Channels

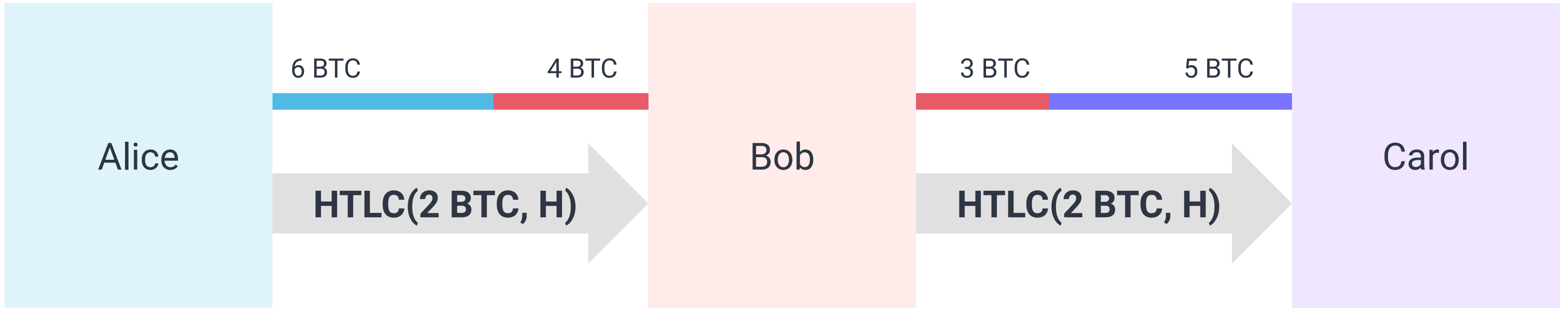
Multi-Hop Payments


Alice
6 BTC
4 BTC
Bob
3 BTC
5 BTC
Carol
@Carol: I want to buy this picture of a cat
@Alice: send me an HTLC for 2 BTC

Carol tells Alice to send her an HTLC for 2 BTC redeemable for the preimage of H

Multi-Hop Payments

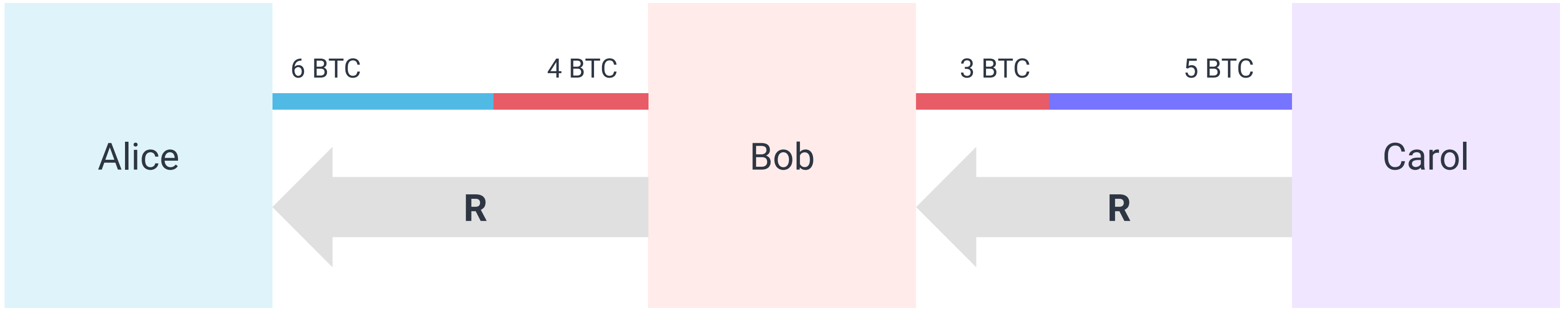
forward HTLC



Alice sends an HTLC to Bob and asks him to **forward the same HTLC** to Carol

Multi-Hop Payments

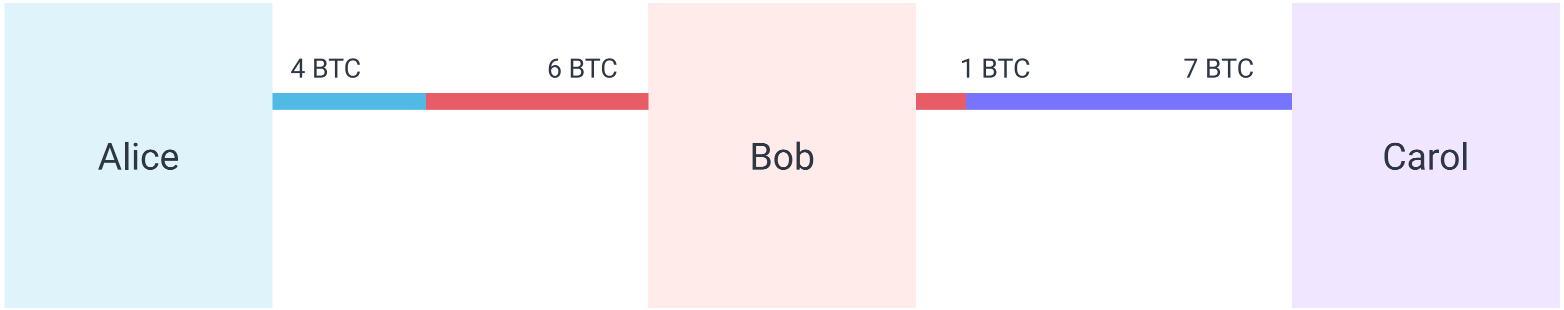
forward Preimage



- Carol sends the Payment Preimage to Bob
- Bob forwards the Payment Preimage to Alice

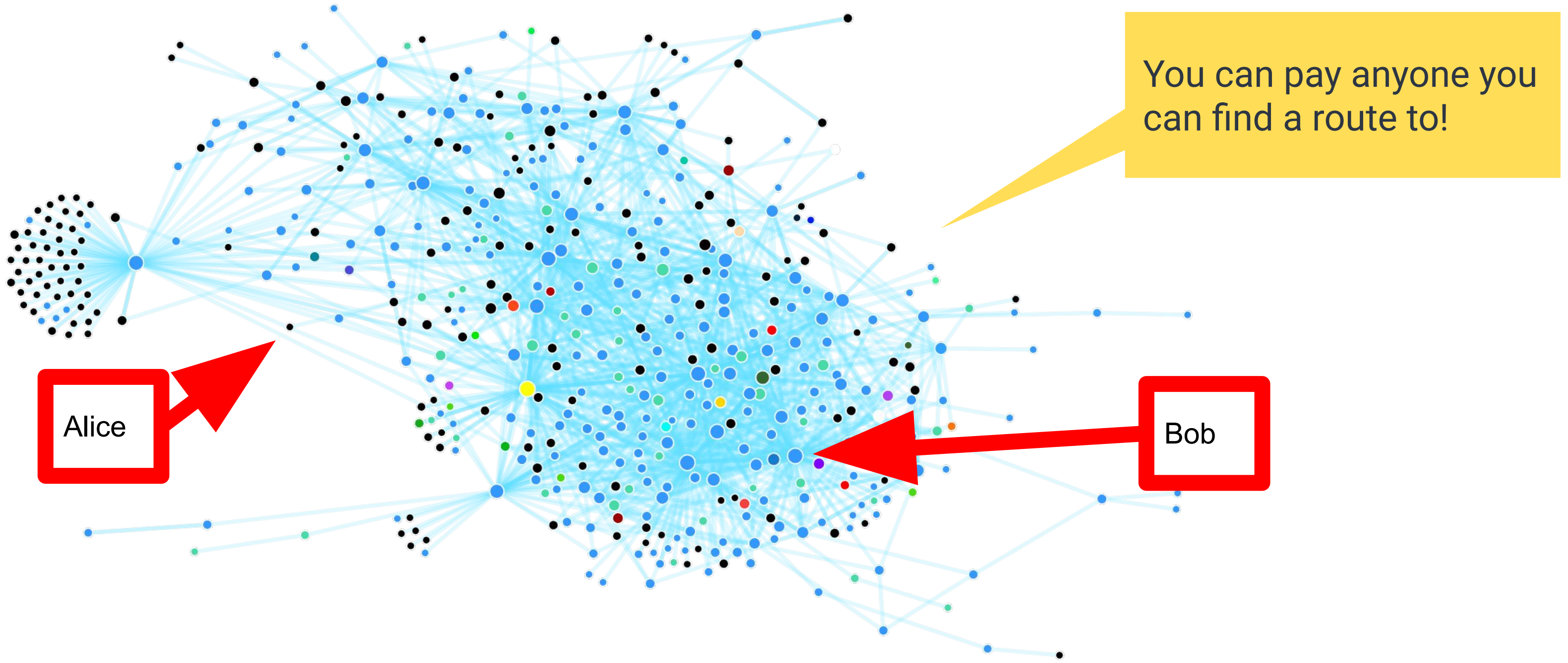
Multi-Hop Payments

update balance



- Alice, Bob and Carol have **updated their balances**
- Bob **still has 7 BTC** (but Bob might ask for a small fee to relay payments)

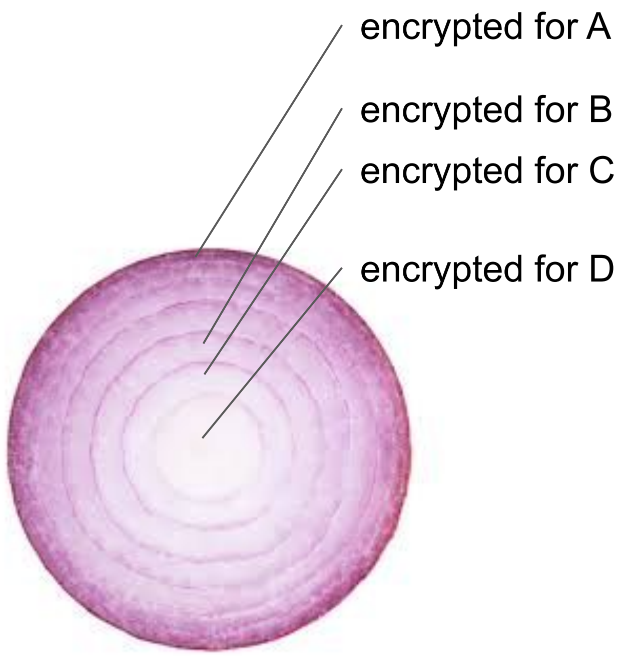
Network of Payment Channels



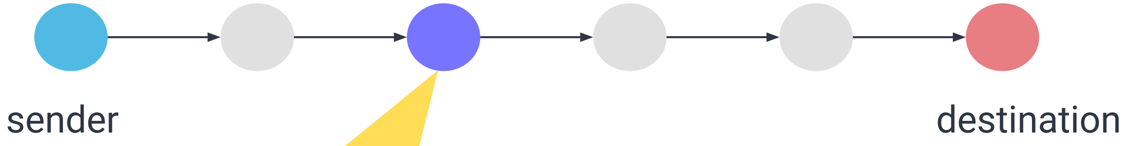
Routing

Routing in LN mixes 2 different concepts:

- How to find a route?
  - This is not really part of the LN protocol. You can compute routes locally, ask someone to do it for you…
- How to send and relay payments once you have a route?
  - This is part of LN
  - Sender creates an “onion” packet that tells each node what the next hop is
  - Nodes “peel off” their own decryption layer, and forward the decrypted packet



Routing

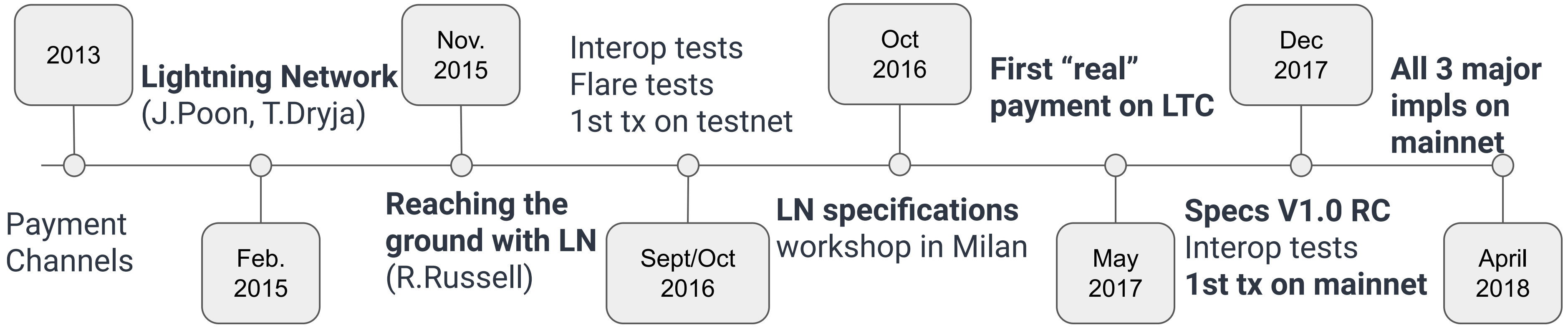


Does not know its position in the route .i.e

- Does not know who the sender is
- Does not know who the final recipient is
- But knows the amount and preimage (*)

Lightning TL;DR

The Lightning Network



Open Source, RFC-like specifications

- BOLT #1: Base Protocol
- BOLT #2: Peer Protocol for Channel Management
- BOLT #3: Bitcoin Transaction and Script Formats
- BOLT #4: Onion Routing Protocol
- BOLT #5: Recommendations for On-chain Transaction Handling
- BOLT #7: P2P Node and Channel Discovery
- BOLT #8: Encrypted and Authenticated Transport
- BOLT #9: Assigned Feature Flags
- BOLT #10: DNS Bootstrap and Assisted Node Location
- BOLT #11: Invoice Protocol for Lightning Payments

See https://github.com/lightningnetwork/lightning-rfc

The Lightning Network

Several independent, open source implementations

- C-Lightning (C)
- Eclair (Scala)
- Lnd (Go)
- And Ptarmigan (C++), Rust-Lighting (Rust), LIT (Python), Electrum (Python)...

The Lightning Network

Public Network

- 8500 Nodes
- 34000 Channels
- 940 BTC ($7.3M)
- Average Node Age: 5 months
- Average Channel Age: 3 months

The Lightning Network

Public Network

- 8500 Nodes
- 34000 Channels
- 940 BTC ($7.3M)
- Average Node Age: 5 months
- Average Channel Age: 3 months

This is just the public LN network!

Nodes that don’t relay payments don’t need to be announced (mobile nodes are not for example)

LN would work with millions of terminal nodes and thousands of relaying nodes