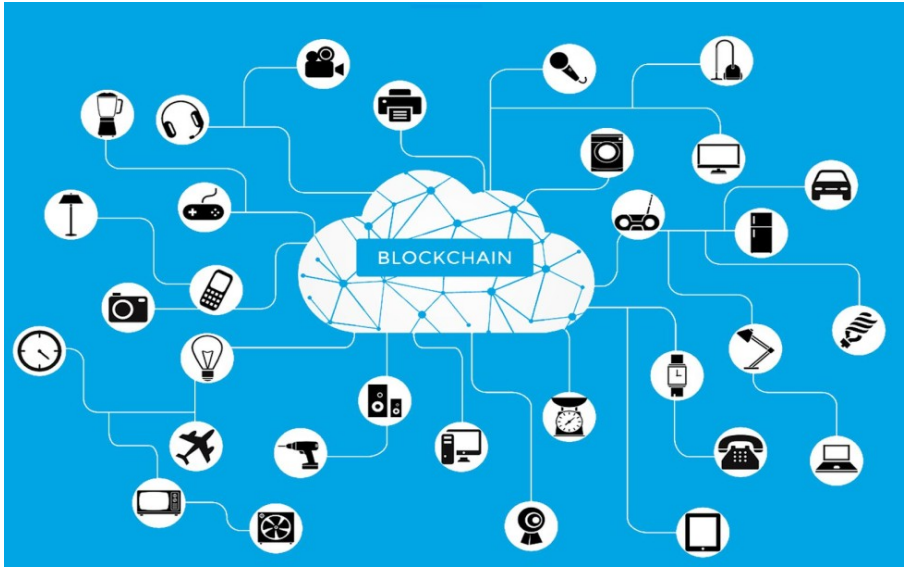
Blockchain as a Platform

- What is revolutionary about Bitcoin and Blockchain **it’s not strictly limited to the idea money**
- A blockchain is a layer that represents **an immutable sequence of events** on which different actors must agree, not by trusting each other, but trusting in in the laws of math.

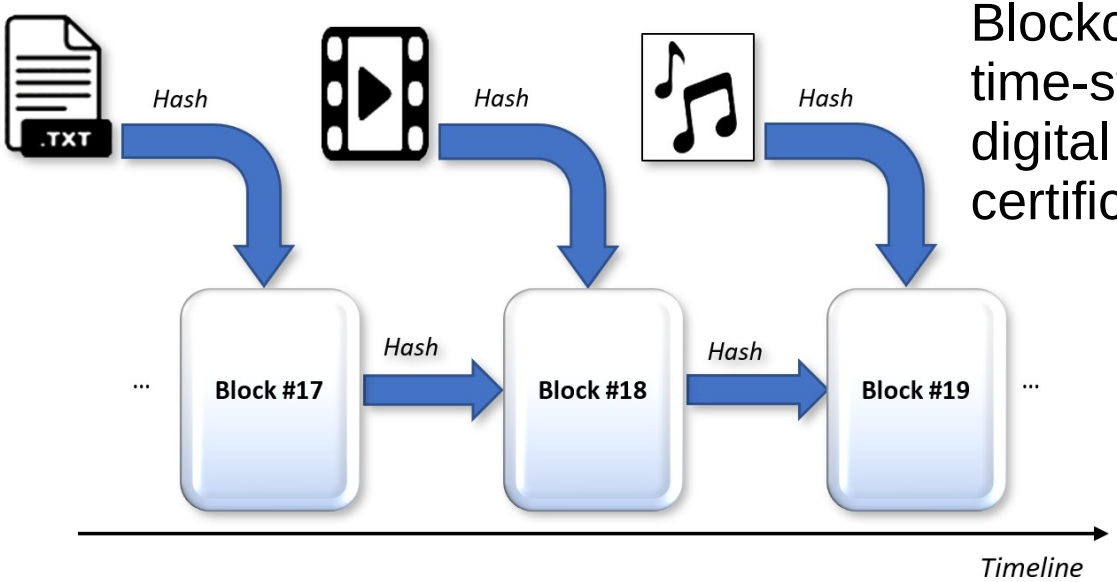


Base Layer: Native Asset BTC

- The simplest kind of transaction that can be imagined is a transfer of value
- 21 Millions of BTC will be asymptotically reached around the year 2140
- The basic unit is the **satoshi**, which is 0.00000001 BTC, that is, 100M **sat** = 1 **BTC**
- At layer 1, the Bitcoin protocol implements the first form ever seen of “digital scarcity”

Proof-of-Existence

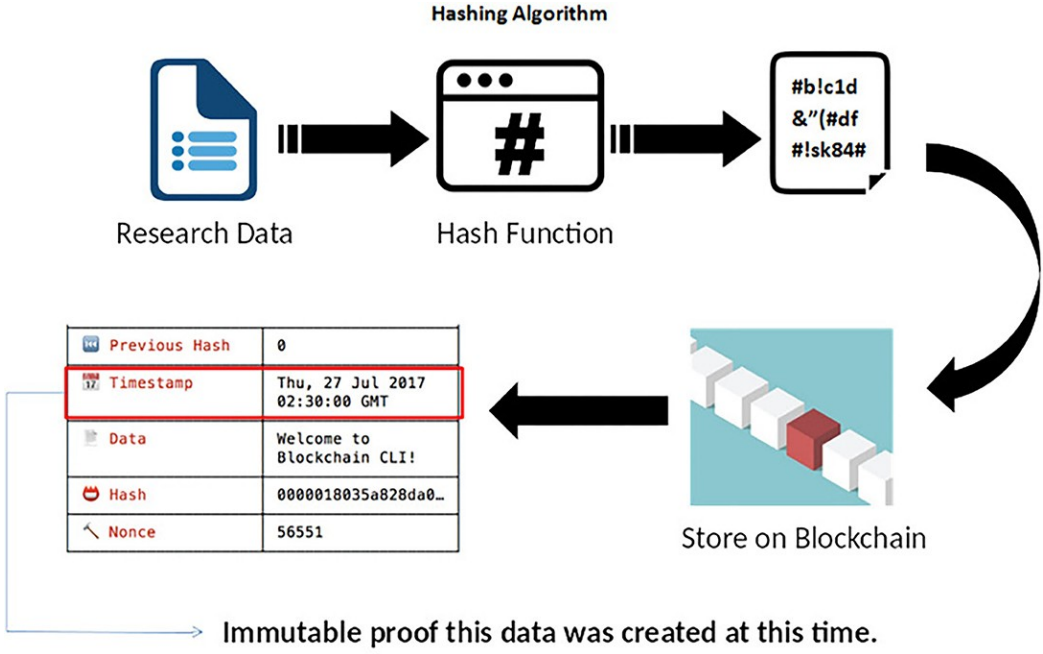
The Blockchain doesn’t know if the 256 bits of a destination are a public key of the SHA256 hash output of some file.



Blockchain as distributed and time-stamped ledger system for digital notary services, stock certificates, smart contracts, etc.

Proof-of-Existence using Unspendable Outputs

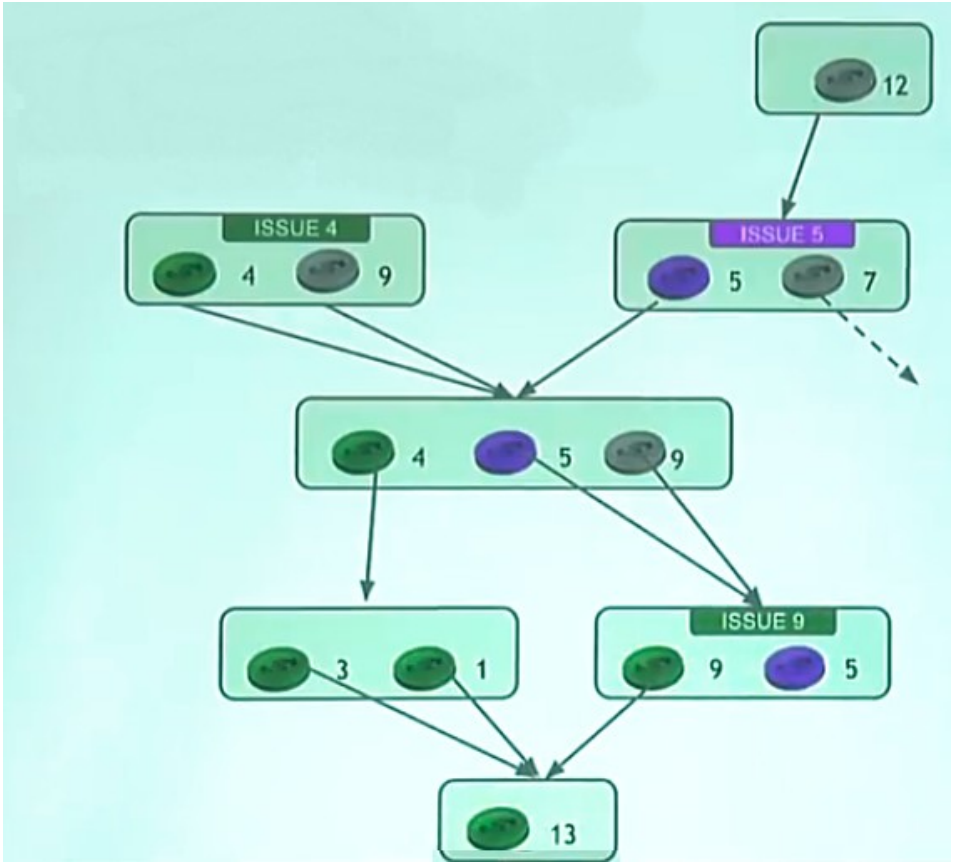
- In place of the public key of the destination, **the hash value of some data** could be used
- Of course, **no one holds the correspoding private key n**, since it hasn't been generated **P=n*G**
- By sending a few satoshis, you stored forever the proof-of-existence of that data at the transaction date.



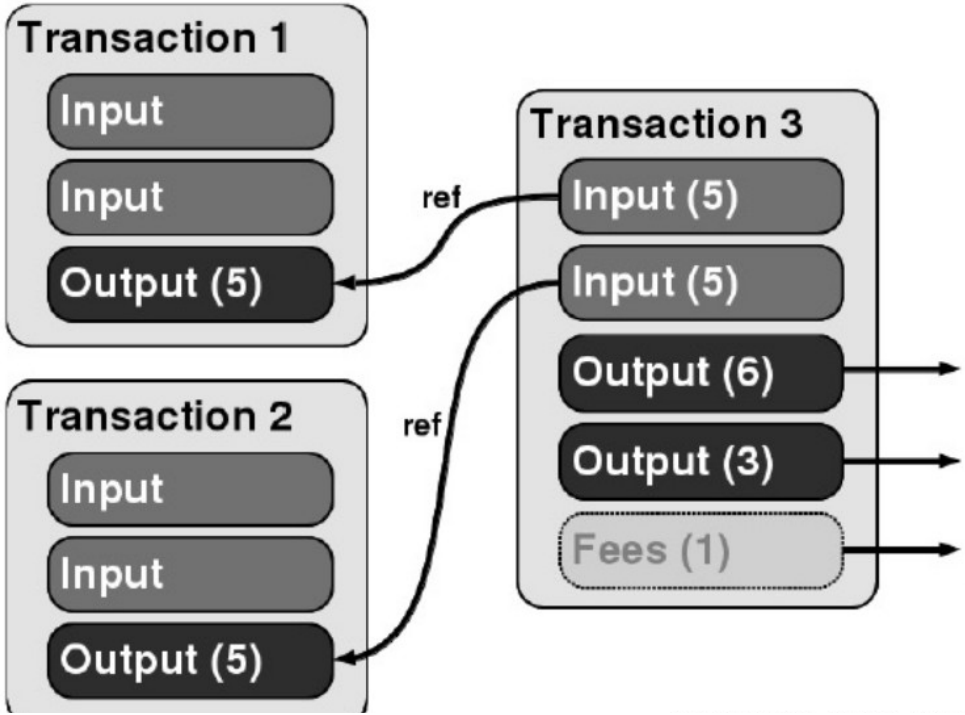
Non-Fungible Tokens (NFT)

- Recall “**fungibility**: every coin is interchangeable with the another
- Applications can be build at layer 2, so that some BTC **represent a different assets other than bitcoin**
- **It’s like virtually “coloring” some coins so that you can follow them**.
- For example: a theather could sell 100 colored tokens, where each represents a ticket.

  Other: voting rights, resource allocation, collectibles, metaverse



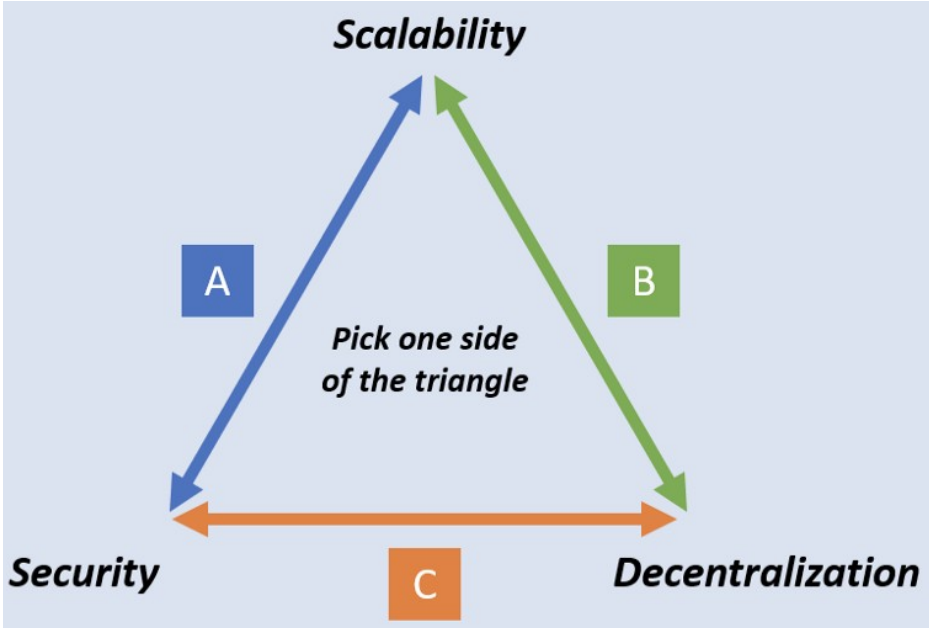
Non fungibility at Layer 2



- Everyone can easily track the movement of colored tokens
- This happens at layer 2
- For the lower layer 1 the tokens are still fungible

The Scalability Trilemma

- **Scalability:**
  - Number of transaction events
  - Number of nodes
- **Security:**
  - malicious attacks
  - chain immutability
- **Decentralisation:**
  - Censorship-resistance
  - No trusted third-party



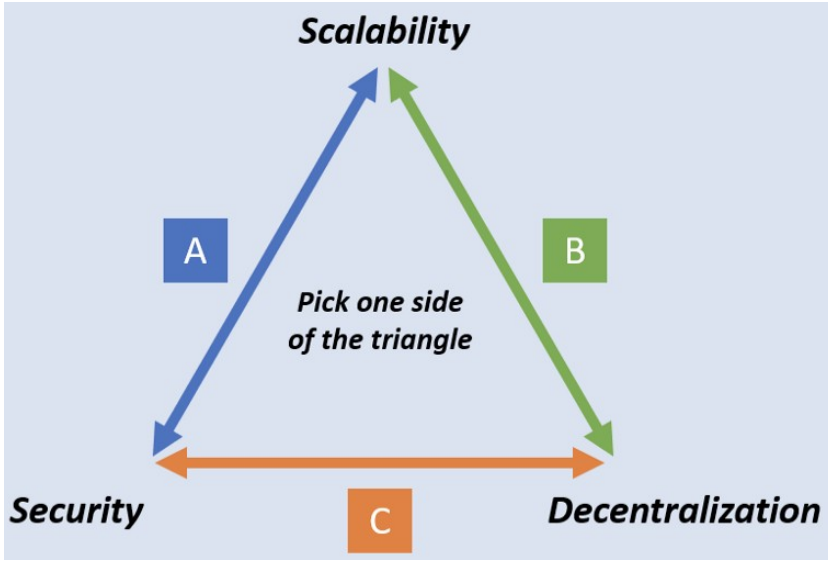
Scaling with Blocksize Increase

- **Bigger Blocks → more hw resources to verify/store blocks → less nodes →** (less decentralisation)

7 billion people doing 2 blockchain transactions per day
- 24 GB blocks, 3.5 TB/day → 1.27 PB/year
- **Bigger blocks = Centralization**
- Very few nodes → De facto inability to validate blockchain

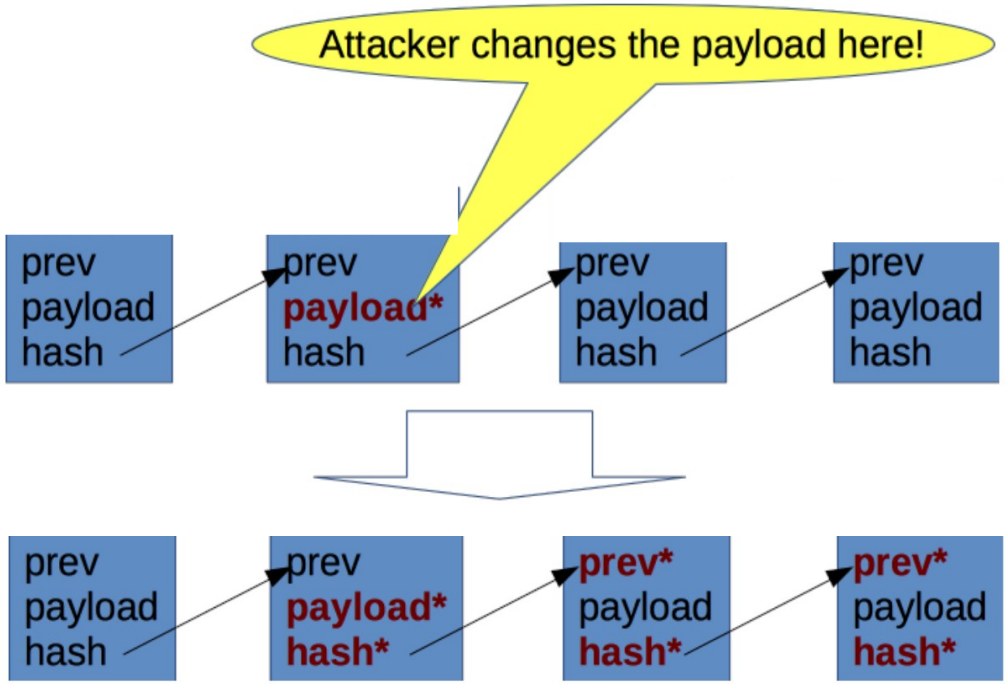
Instrinsic Security: The Trilemma

- **Reduce block time** → **reduce proof-of-work difficulty** → (less security)

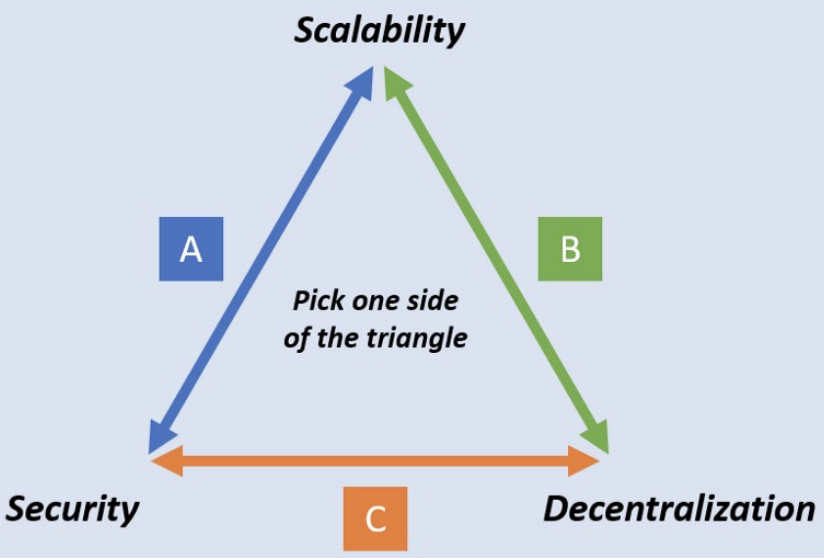


Attacking the Chain

The modification of a block would require the recomputation of the all the hashes for the subsequent blocks...



Instrinsic Security: The Trilemma



As a result, you will have a trade-off between **Security and Decentralisation**

The two Visions

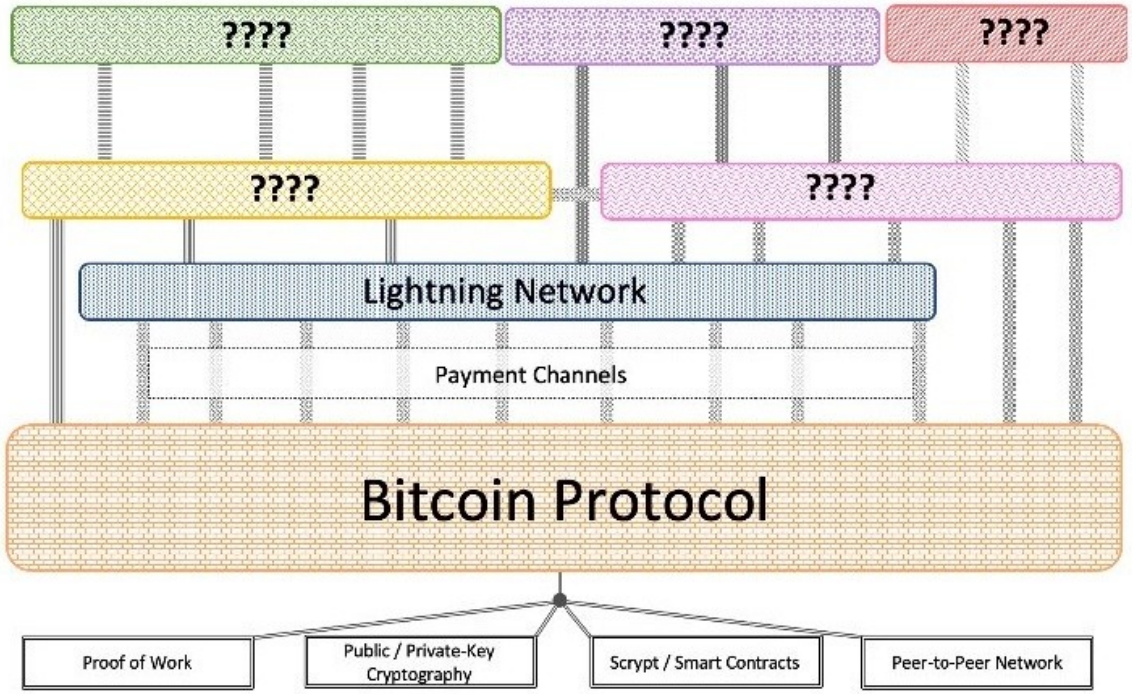
- **Create new blockchains from scratch** → add new features and put everything needed directly at the base layer

- **The decentralisation of the Bitcoin network is the most important thing** → we should scale building on top of this solid layer

Bitcoin emerged under unique and probably irrepetible conditions:

- The lack of a leader
- The slow and silent growth of the Bitcoin network in the first months
- Probably a “one shot” opportunity for computer science and economy

Building on Bitcoin base layer

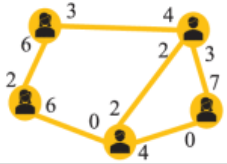
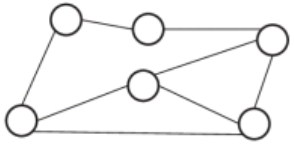


Layer Names	Protocols
Application	HTTP,FTP,POP3, SMTP,SNMP
Transport	TCP,UDP
Networking	IP,ICMP
Datalink	Ethernet, ARP

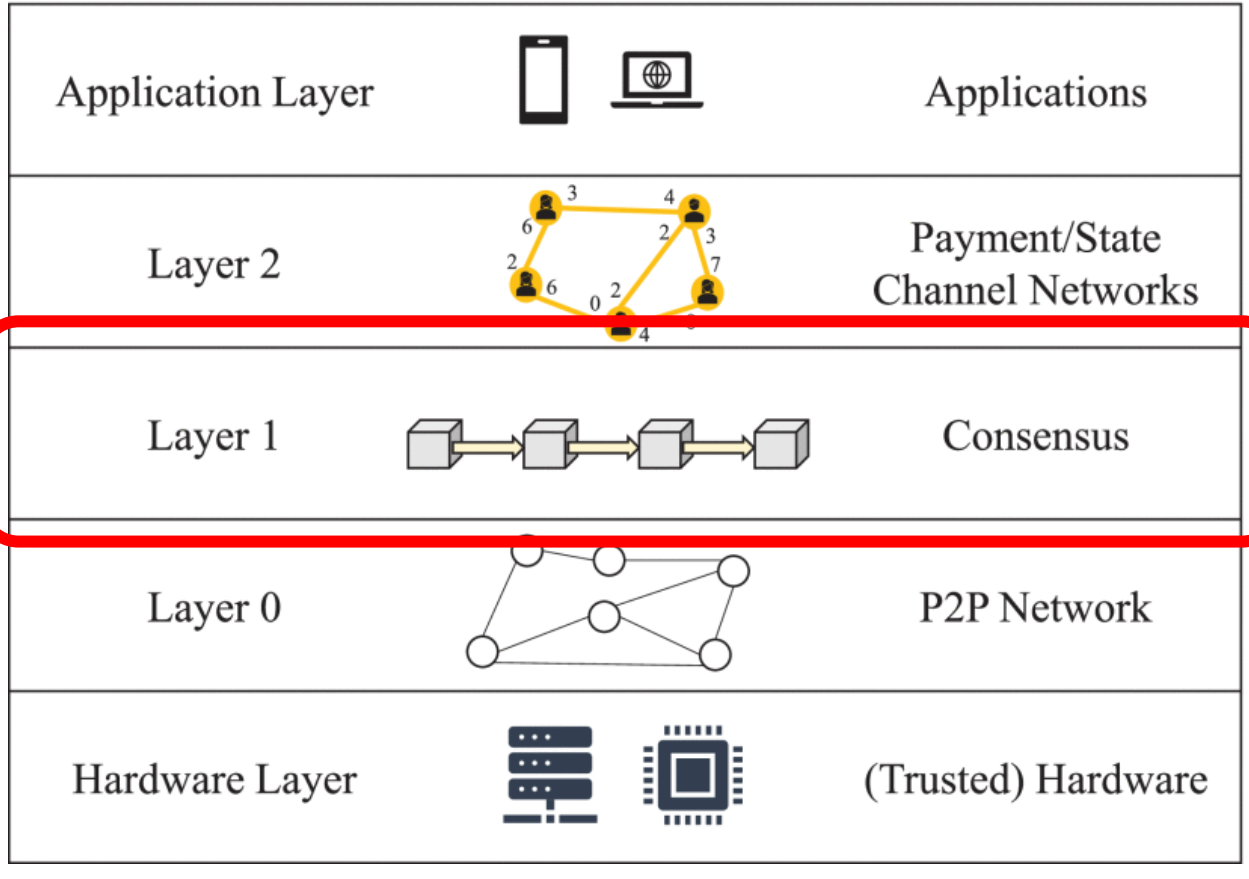
TCP/IP Networking Model

Lower levels change slowly, acting as a solid foundation

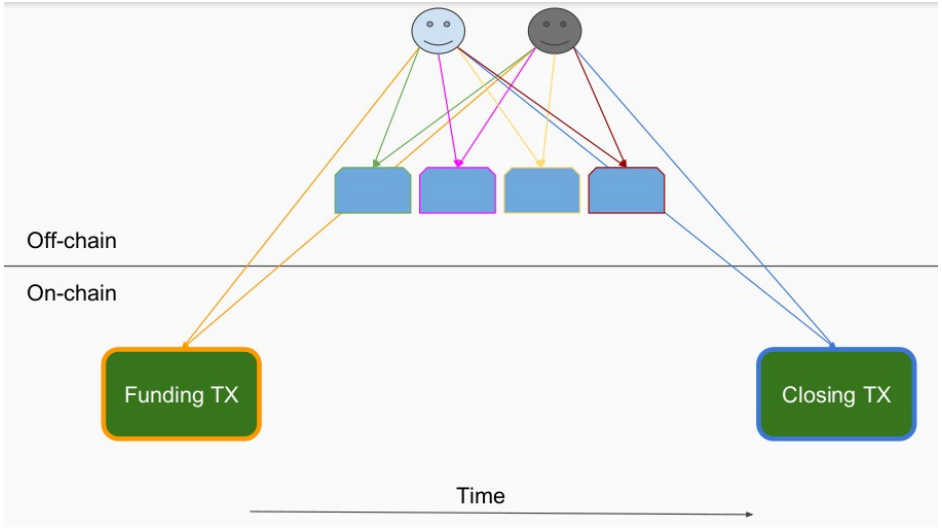
Contexualizing the Blockchain

Application Layer		Applications
Layer 2		Payment/State Channel Networks
Layer 1		Consensus
Layer 0		P2P Network
Hardware Layer		(Trusted) Hardware

Contexualizing the Blockchain

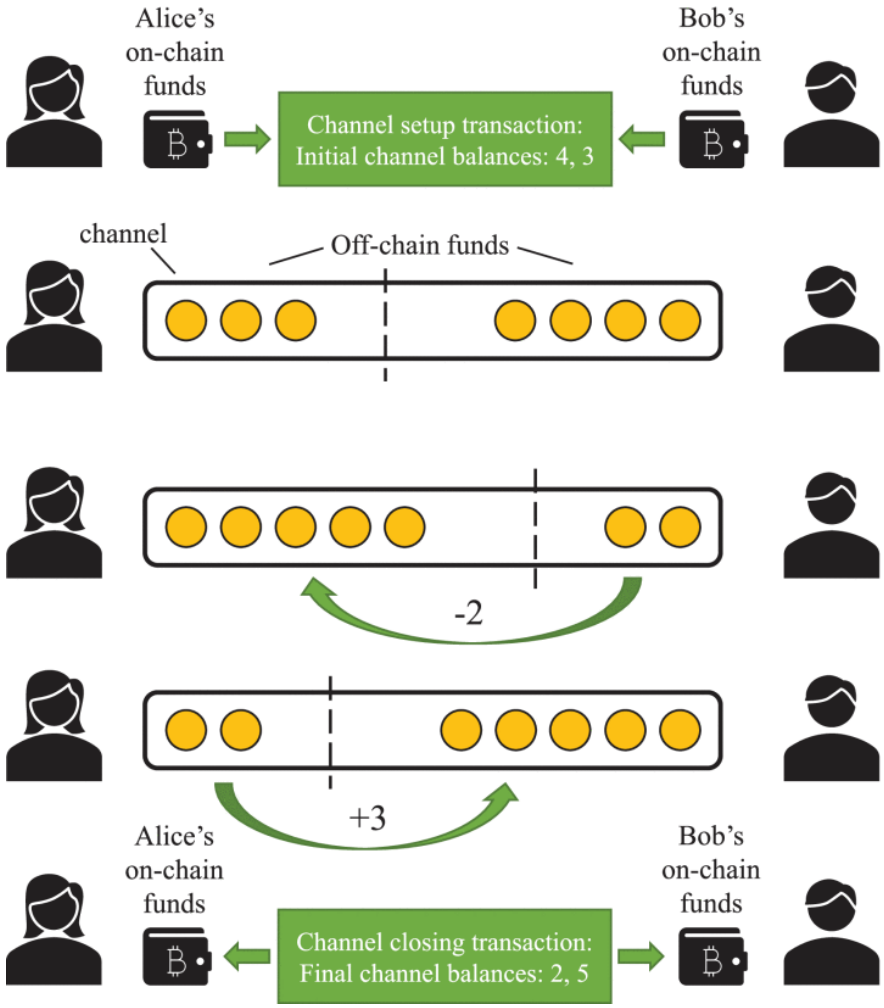


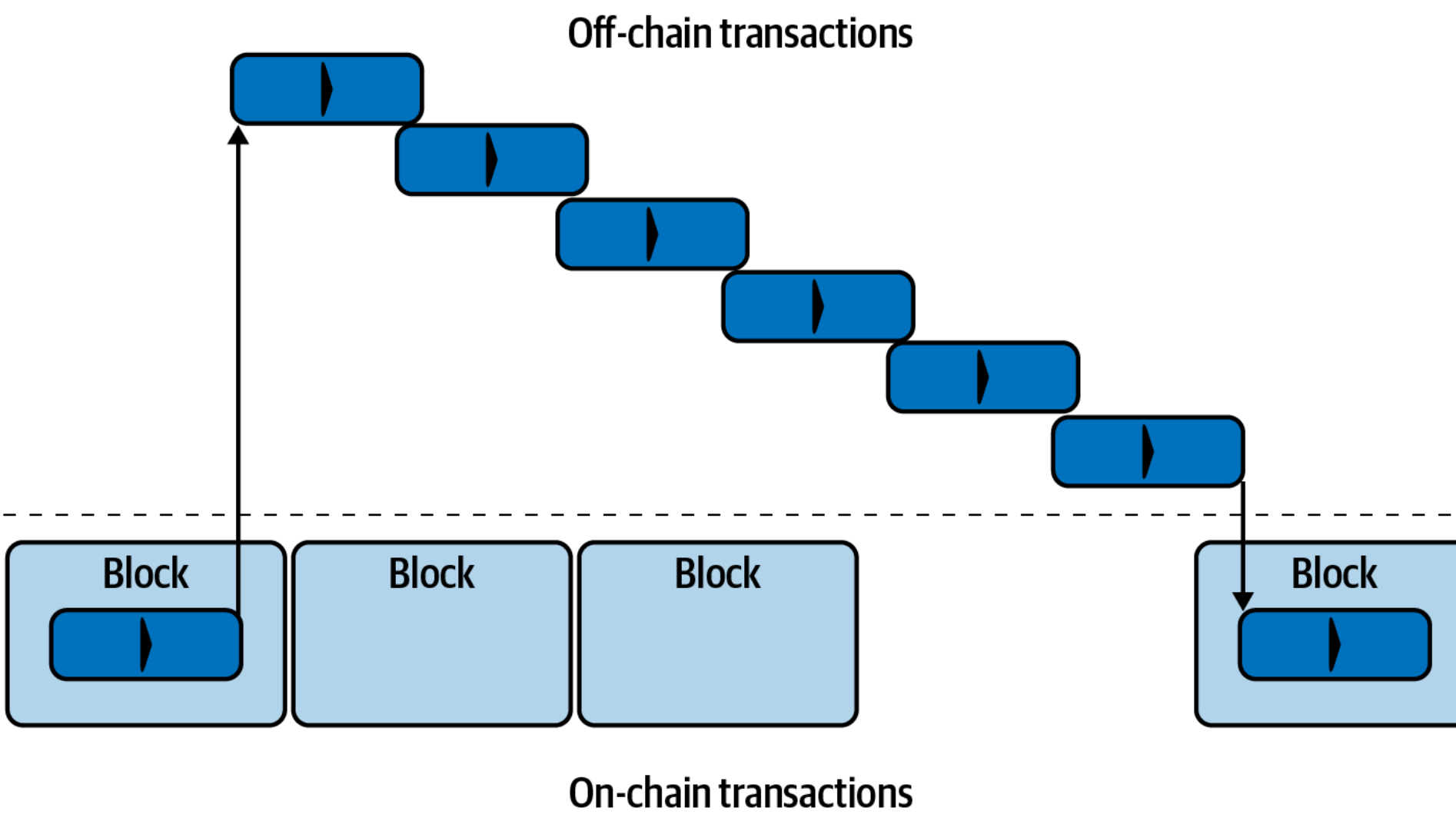
The Idea: Layer 2 Channels



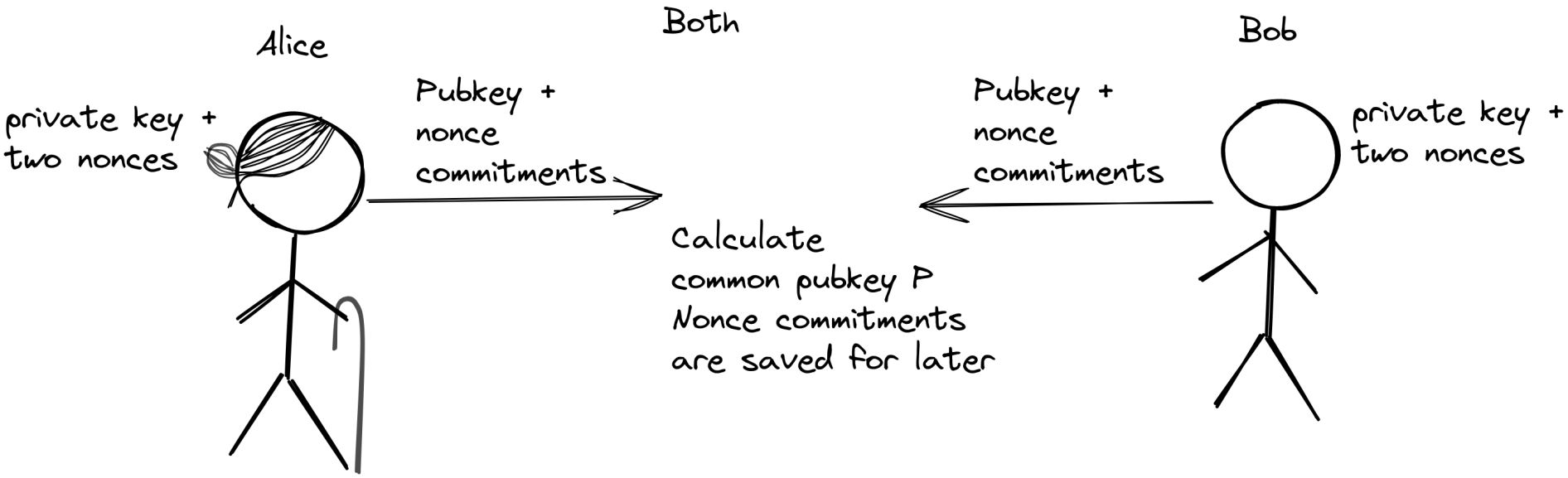
Transaction between two parties are created but not broadcasted on-chain

They are “real” because they “could be made real” without permission from the counterpart


Alice's on-chain funds
Bob's on-chain funds
Channel setup transaction: Initial channel balances: 4, 3
channel
Off-chain funds
-2
+3
Alice's on-chain funds
Bob's on-chain funds
Channel closing transaction: Final channel balances: 2, 5

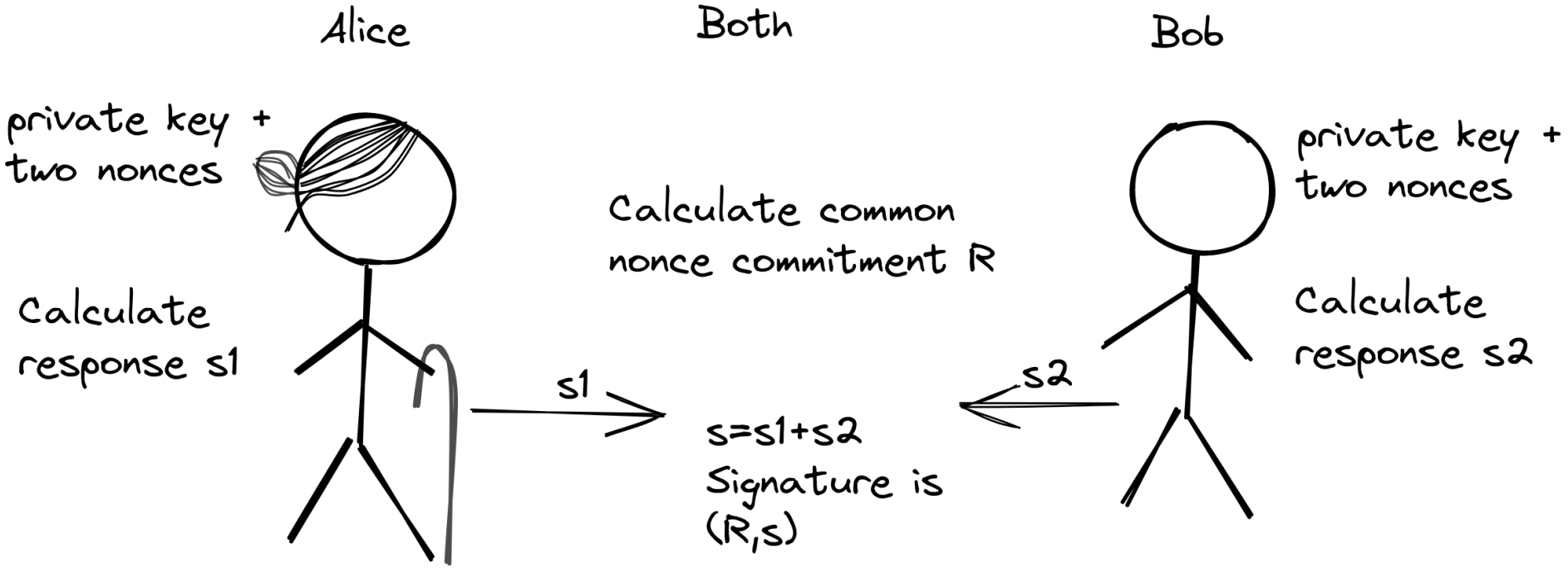

Off-chain transactions
Block
Block
Block
Block
On-chain transactions

2-of-2 Multi Signatures



Let’s consider a public key Pm = Pa+Pb, that is, che combination of Alice and Bob public keys

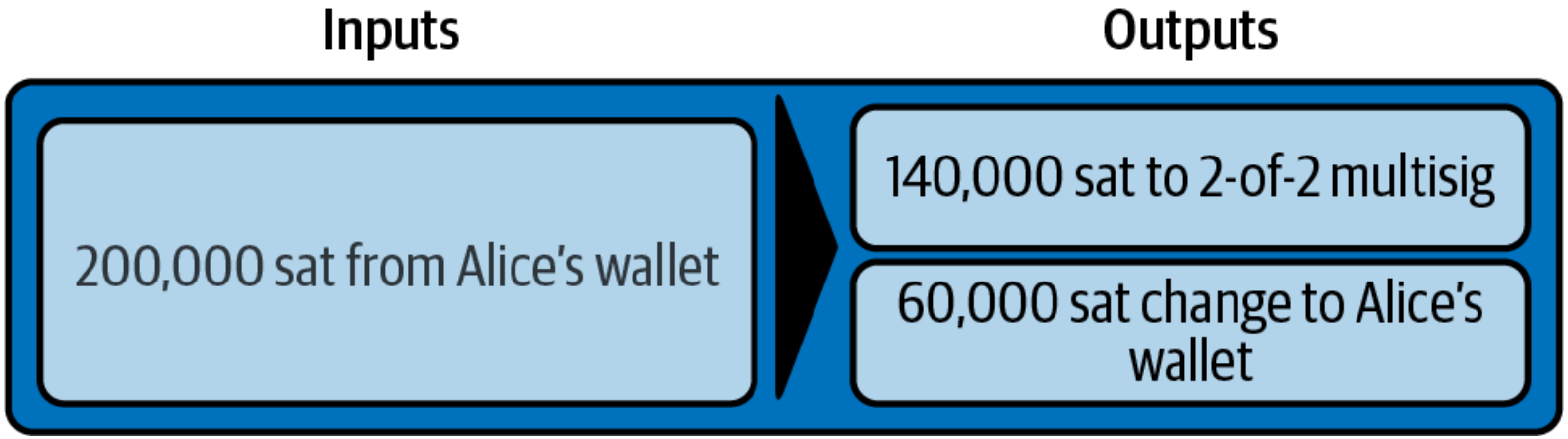
2-of-2 Multi Signatures



The sum of two individual signatures is also valid for the sum of the individual public keys!

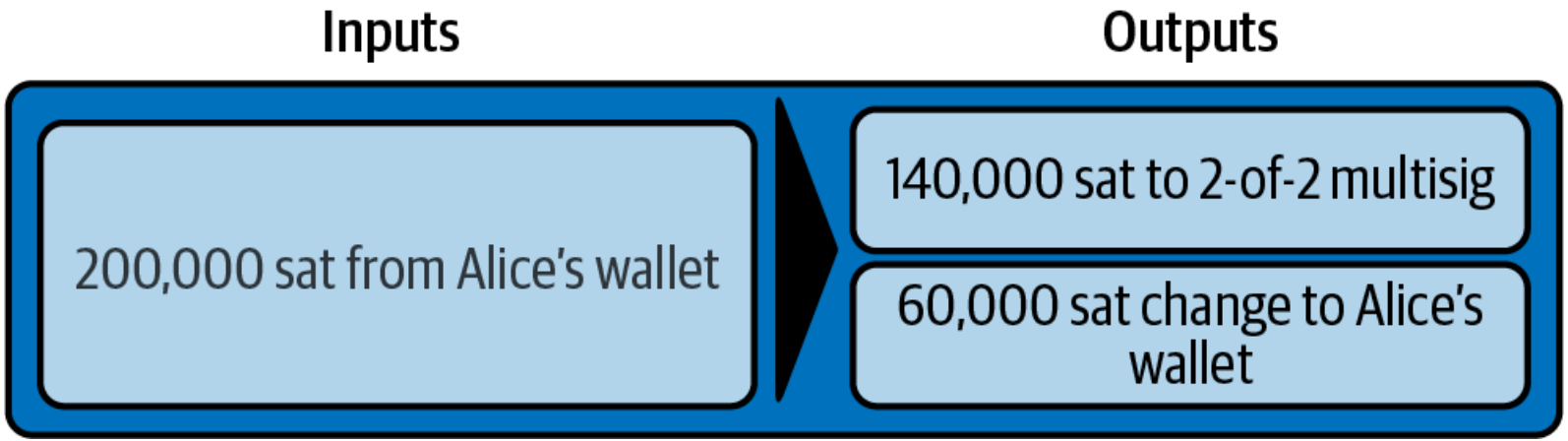
Alice wants to create a channel that starts with 140k sats

Alice creates a “funding transaction” that sends 140k to the multisig public key of Alice+Bob



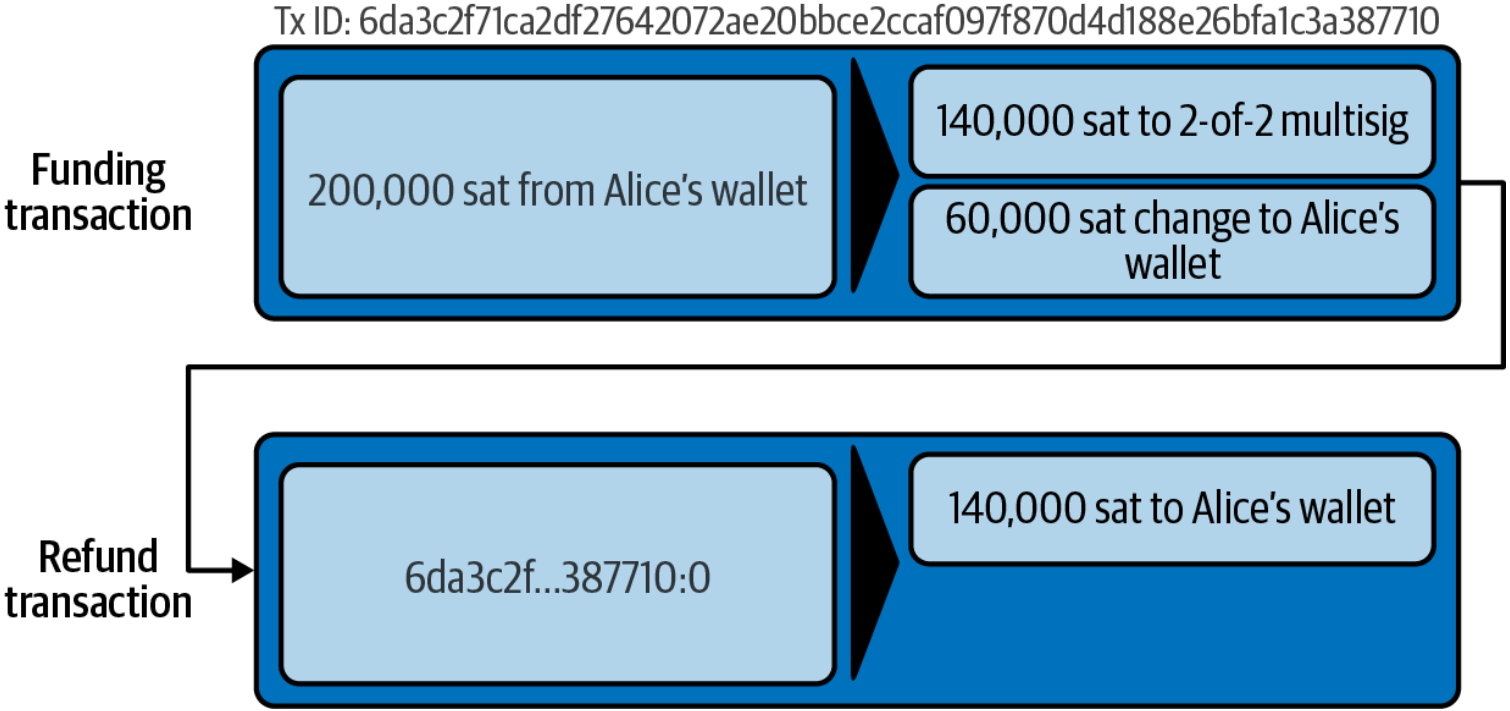
Alice does not put this transaction on blockchain because it would put her 140,000 sats at risk!

Once sent to the 2-of-2 multisig, there is no way for Alice to recover her sats without Bob's signature.

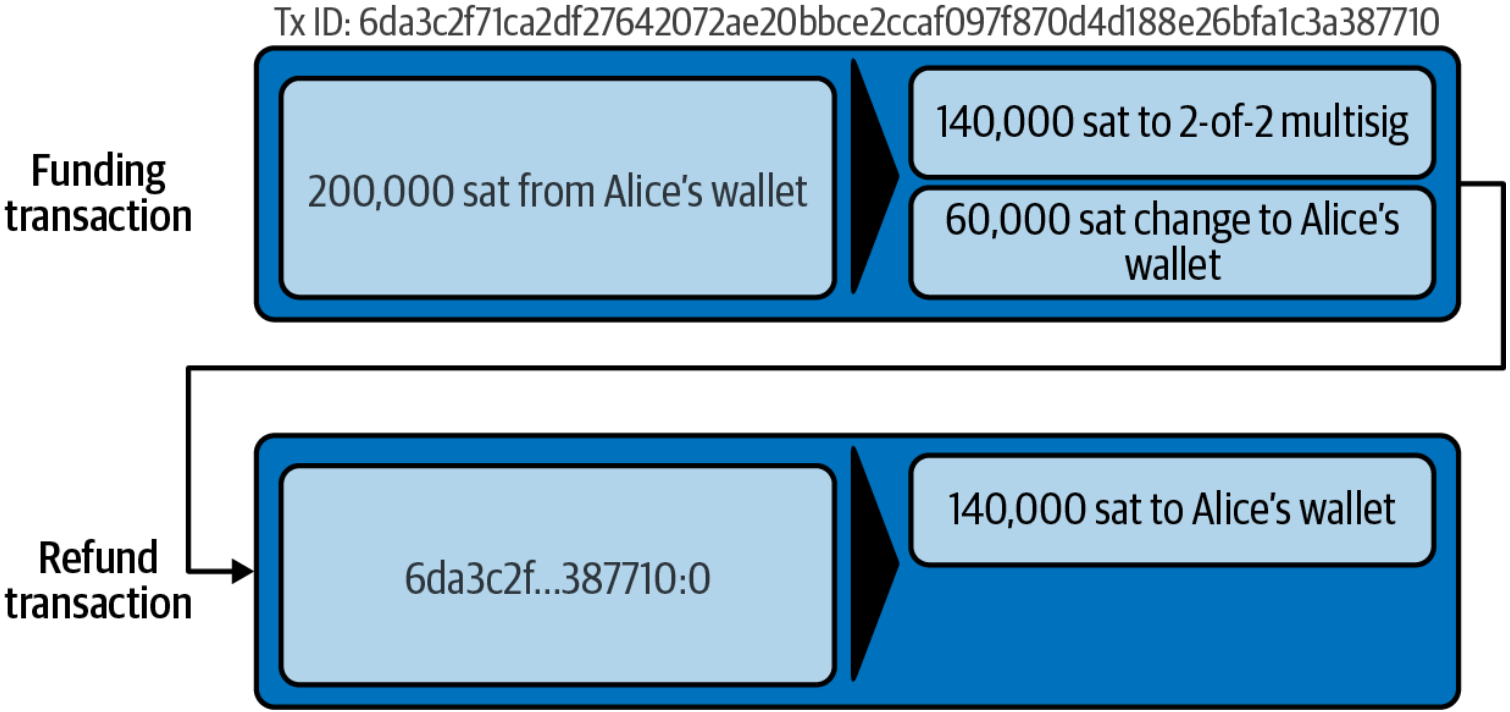


Alice constructs a refund transaction immediately after constructing (but not broadcasting) the funding transaction.

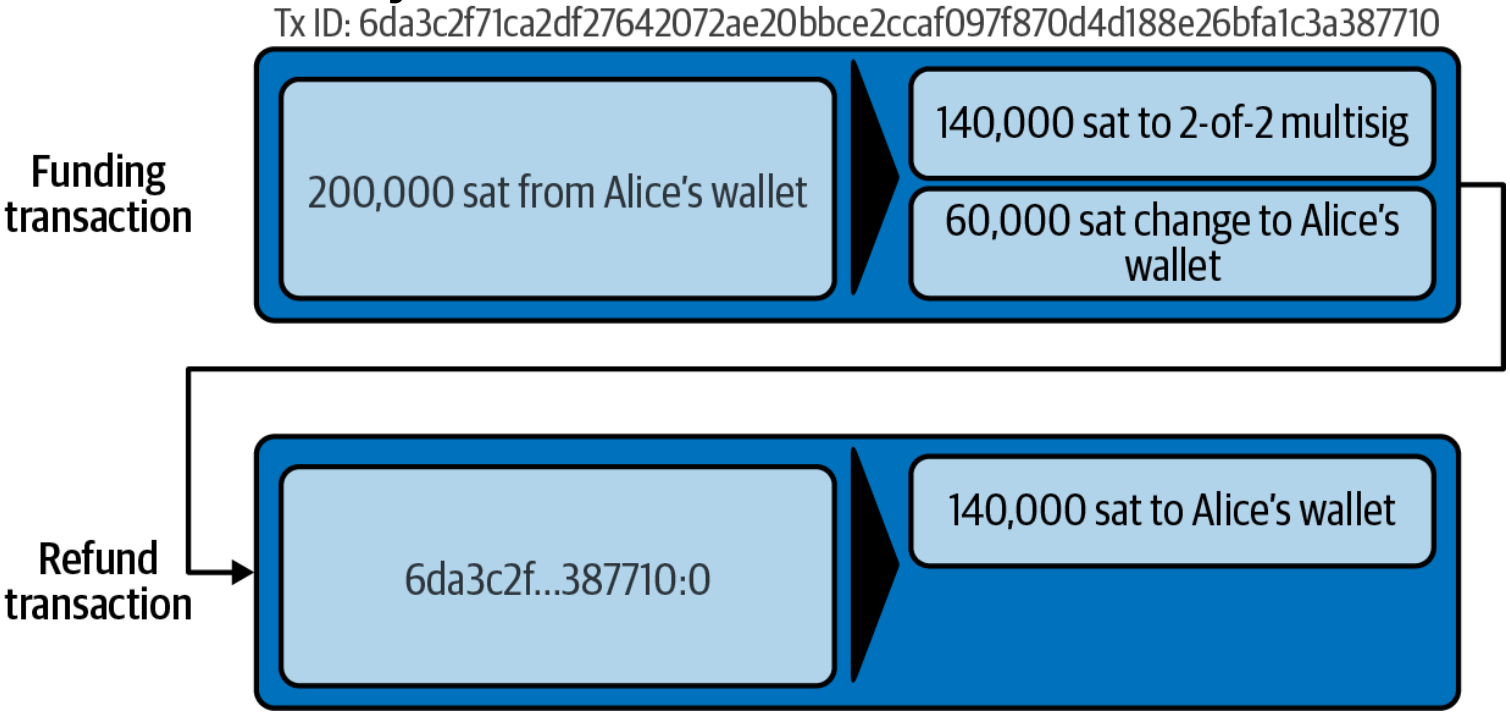
She asks Bob to sign it. Now she can safely put the first on chain



How can you use an output that hasn't been confirmed on the Bitcoin blockchain?!?

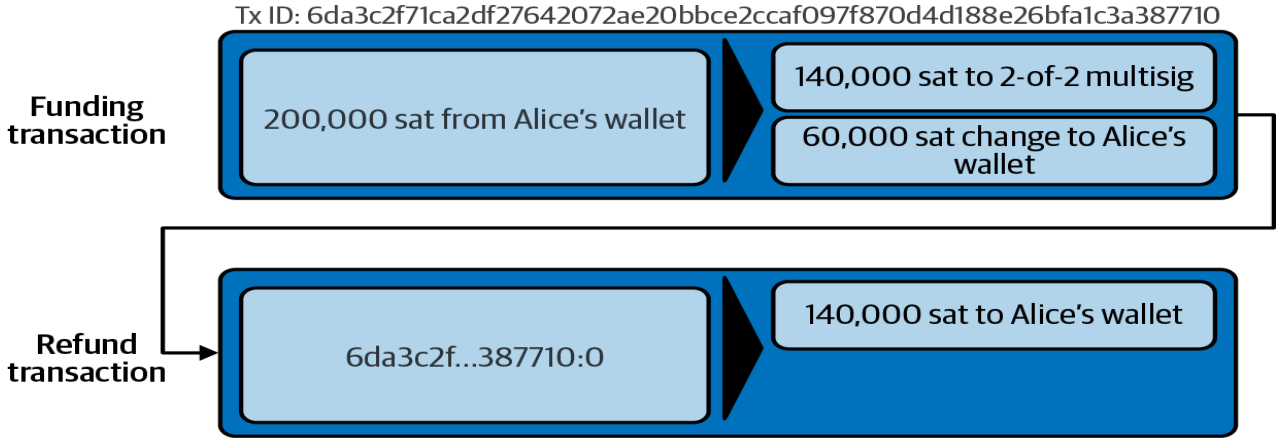


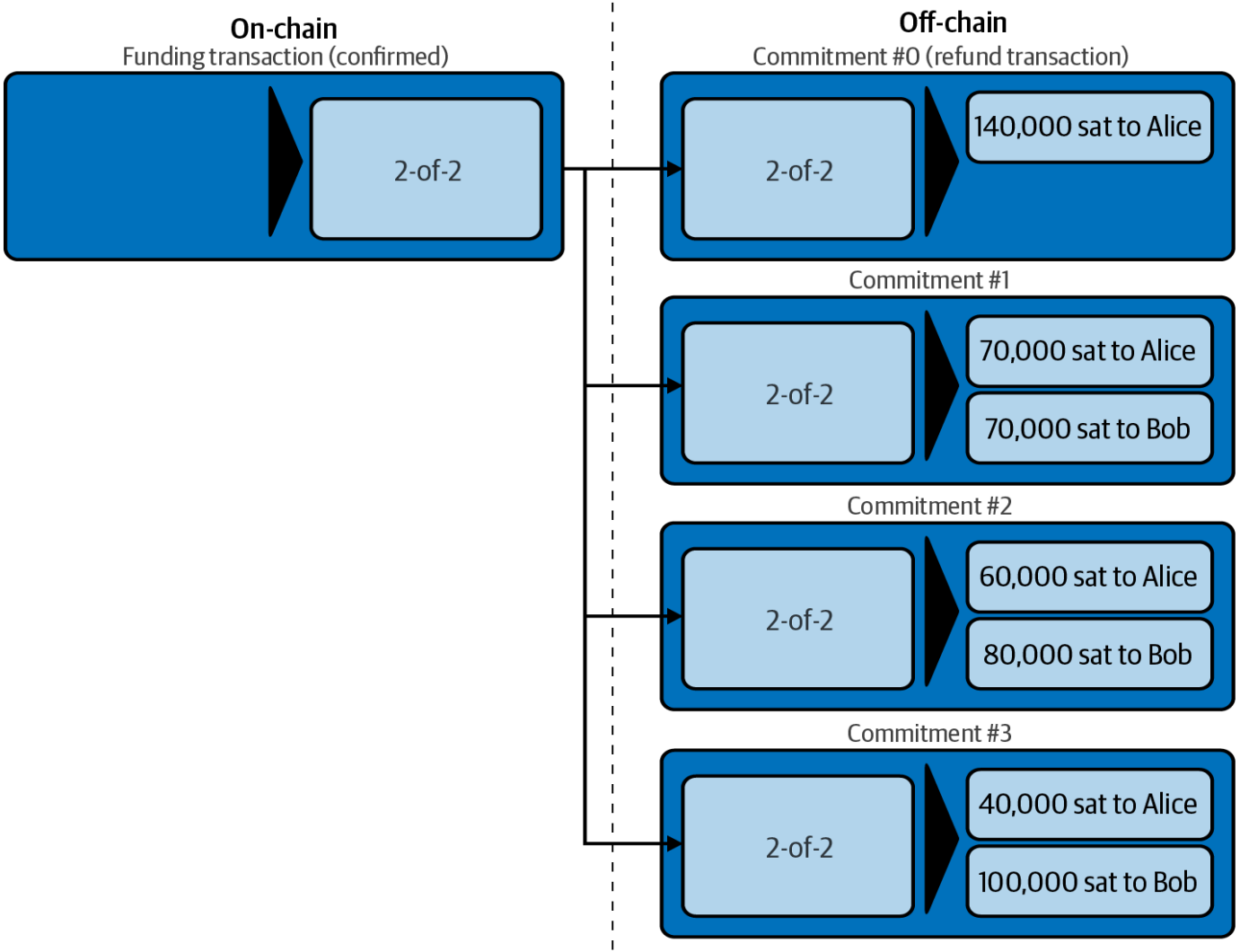
- The content, and thus the ID of a transaction does not depend on being on-chain or not
- So, Alice can just construct a second one



An “exit plan” for Alice

- Alice just want to be sure that funds don’t remain locked in the multisig address if Bob disappears
- She asks to Bob to provide a signature for a tx that restores the initial status, using a timelock that makes it usable only after some blocks




On-chain
Funding transaction (confirmed)
2-of-2
Off-chain
Commitment #0 (refund transaction)
2-of-2
140,000 sat to Alice
Commitment #1
2-of-2
70,000 sat to Alice
70,000 sat to Bob
Commitment #2
2-of-2
60,000 sat to Alice
80,000 sat to Bob
Commitment #3
2-of-2
40,000 sat to Alice
100,000 sat to Bob

It seems We have a Problem...

how is it possible for Alice and Bob to have multiple commitment transactions, all of them attempting to spend the same 2-of-2 output from the funding transaction?

Aren’t these commitment transactions conflicting?

Isn’t this a "double-spend" that the Bitcoin system is meant to prevent?

That’s Not a Real Problem!

This why we have a blockchain:

If Alice or Bob attempts to broadcast more than one transaction, **only one of them will be confirmed and the others will be rejected**

(not deterministically, depending on several factors, e.g., which one arrived first to the miner that some time later found the next block?)

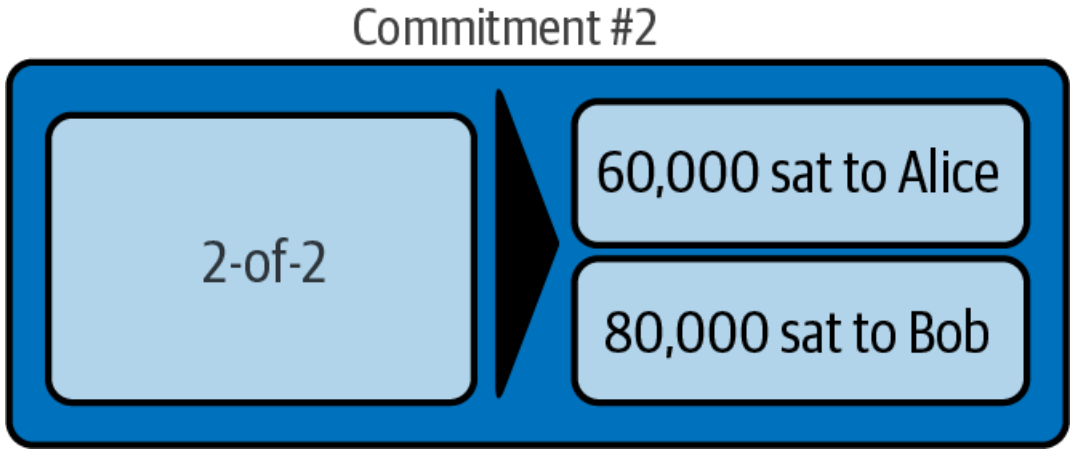
However….

...how can we be sure that ONLY the last commitment will be the one broadcasted?


WOW SO HONEST
MUCH RESPONSIBLE
memegenerator.net

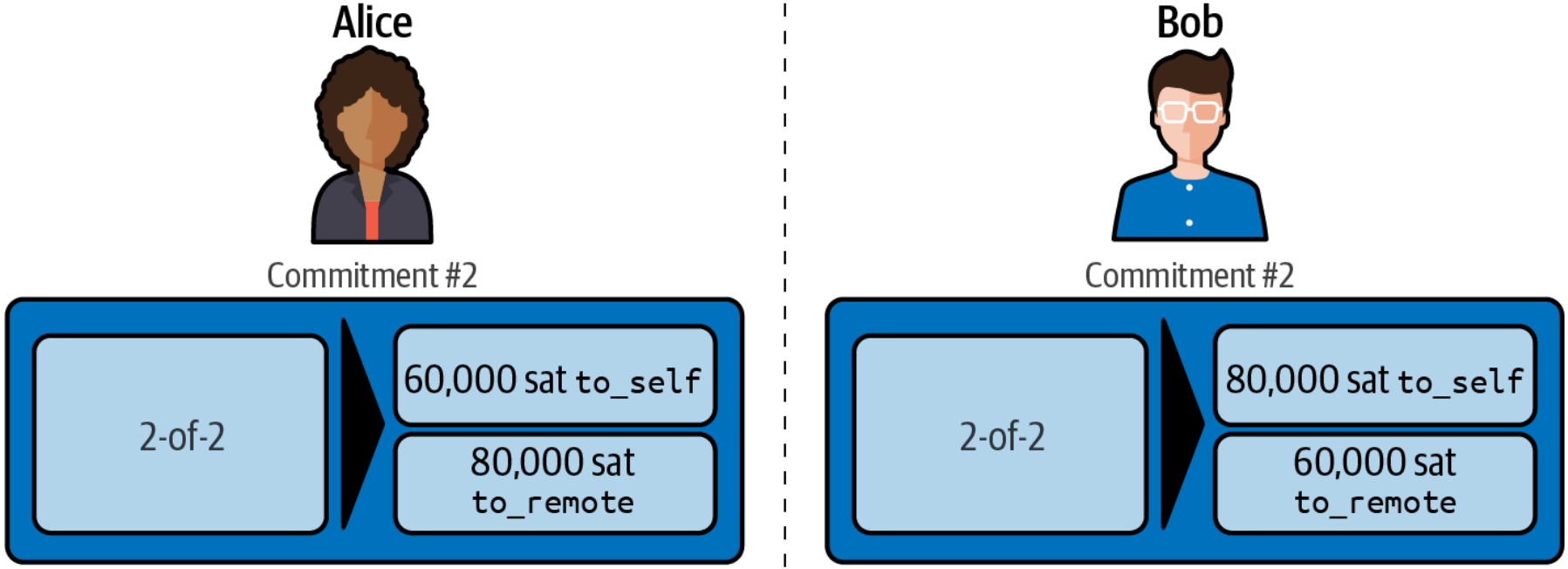
Solution:Asymmetric Commitment

Let’s consider one of the commitments that updates the Alice-Bob channel:

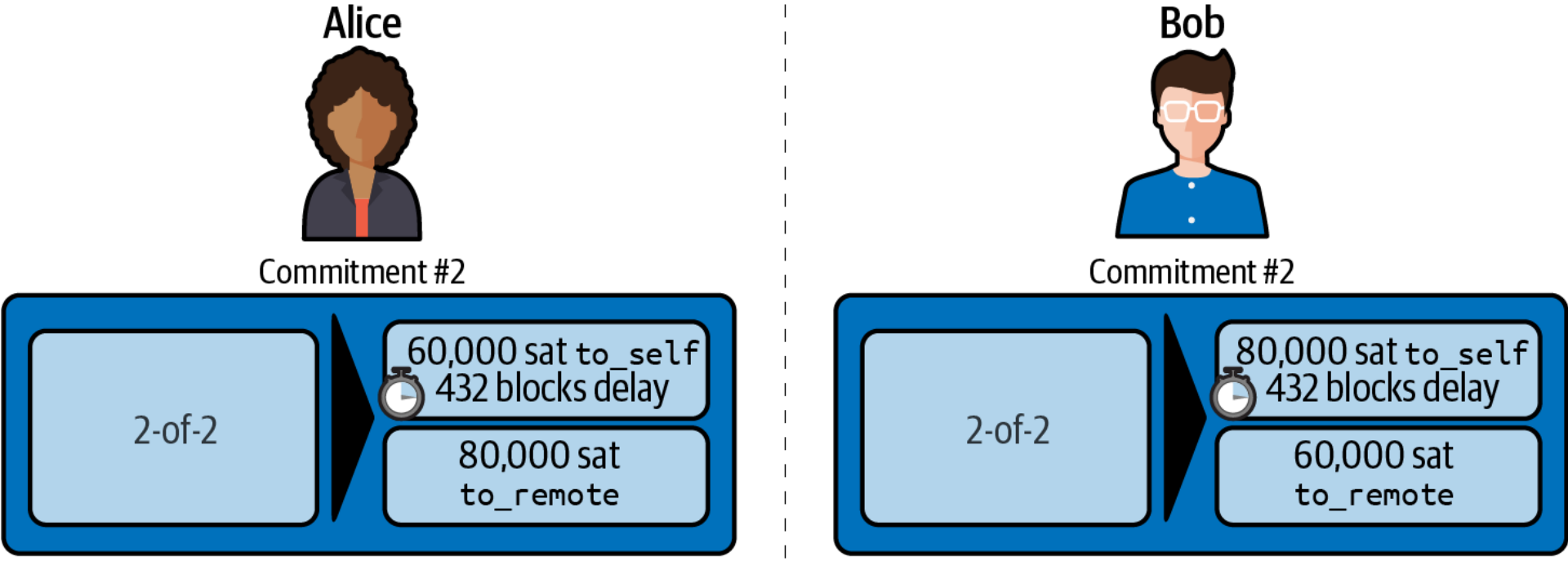


Asymmetric Commitment

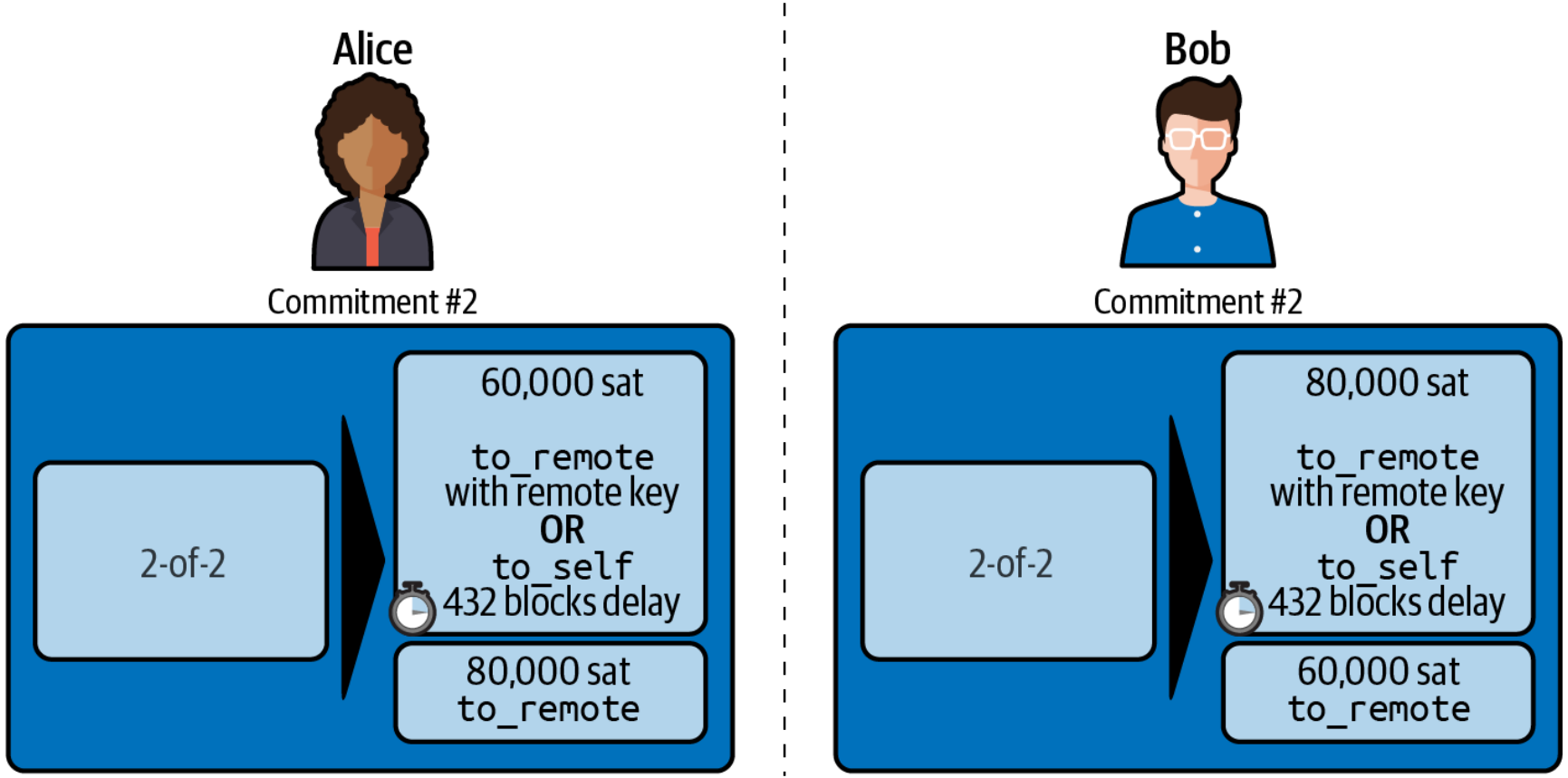
Actually, Alice and Bob will hold two different variations of this transaction



Also, output is always timelocked and can’t be spent immediately, whereas the to_remote output is not timelocked and can be spent immediately.

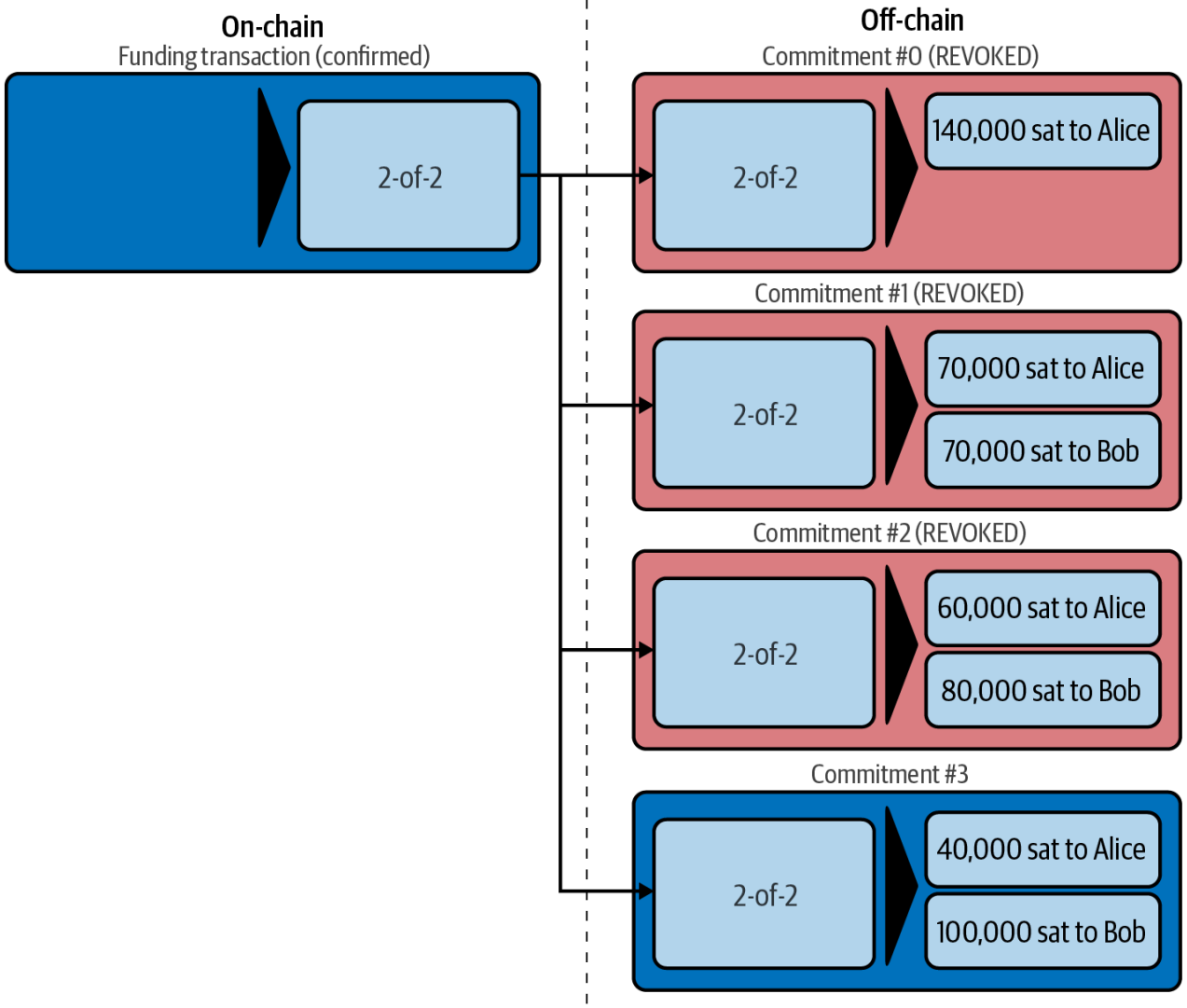


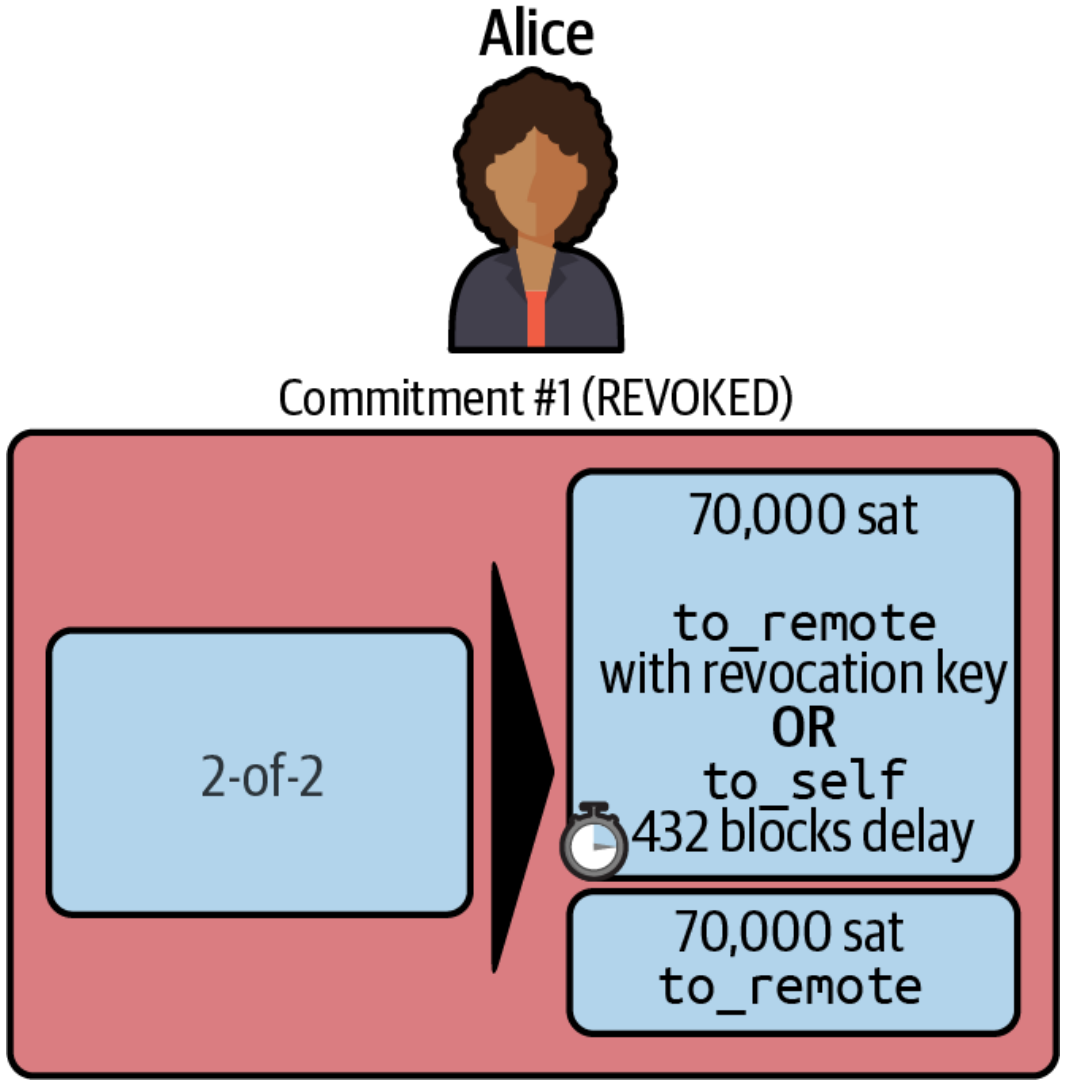
Also, another condition is added, involving a “remote_key” that would nullify the delayed self transaction and just move everything on the other side



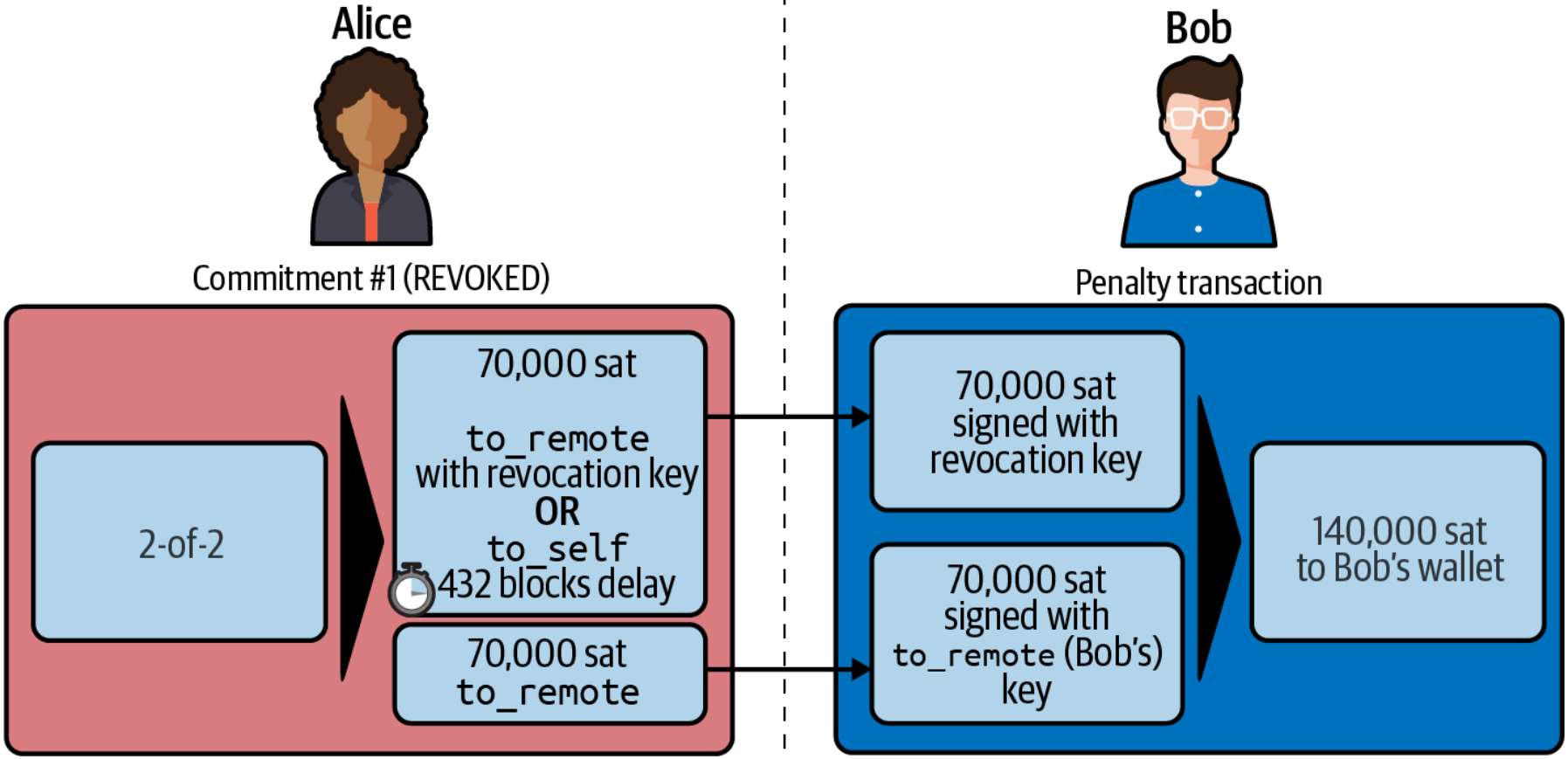
Commitment Steps

- Alice sends her commitment signature to Bob
- Bob sends his commitment signature to Alice
- Once Alice and Bob verified the counterparty signature, they revoke the old commitment by sending her revocation secret (same does Bob)

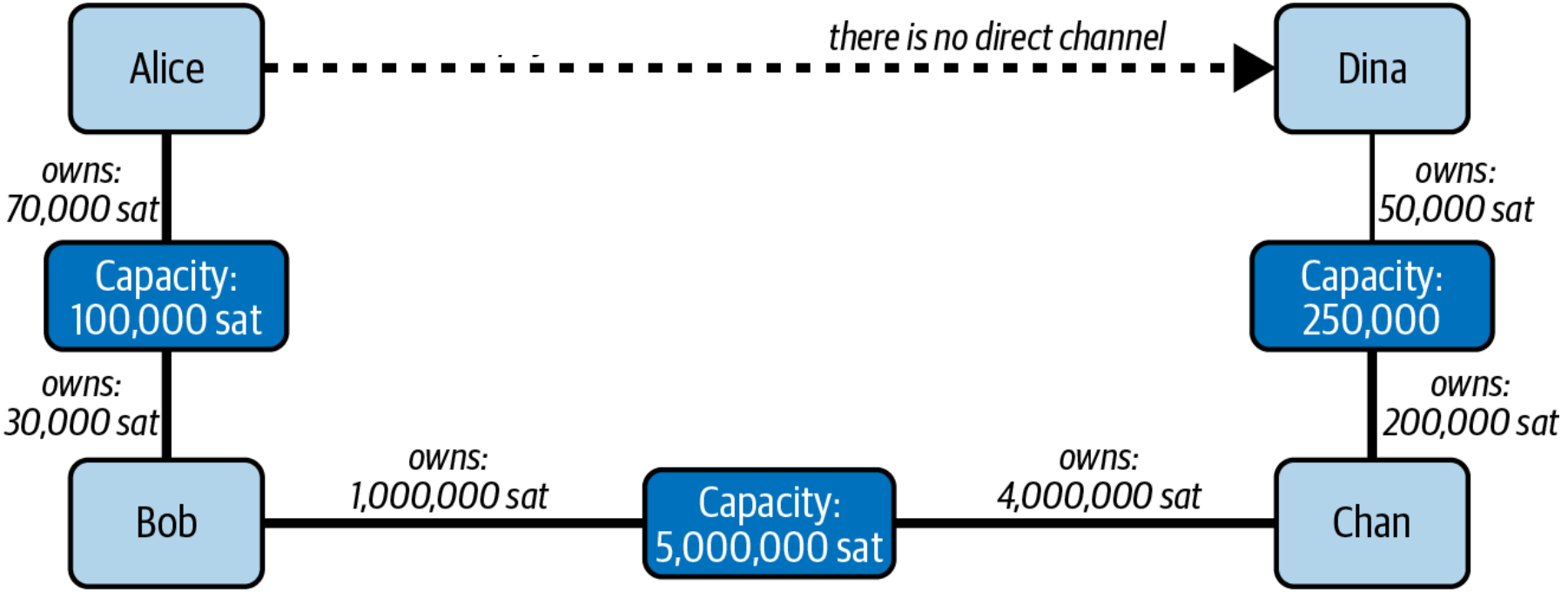

On-chain
Funding transaction (confirmed)
2-of-2
Off-chain
Commitment #0 (REVOKED)
2-of-2
140,000 sat to Alice
Commitment #1 (REVOKED)
2-of-2
70,000 sat to Alice
70,000 sat to Bob
Commitment #2 (REVOKED)
2-of-2
60,000 sat to Alice
80,000 sat to Bob
Commitment #3
2-of-2
40,000 sat to Alice
100,000 sat to Bob

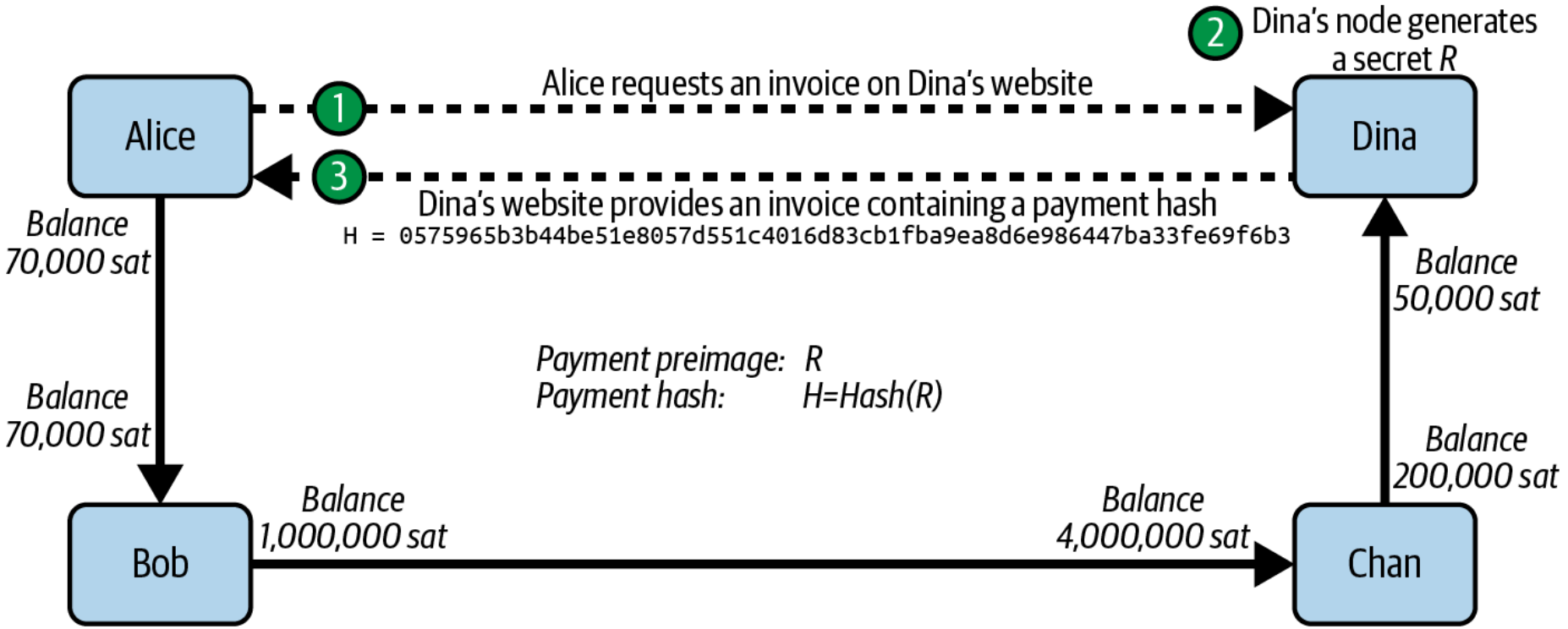

Alice
Commitment #1 (REVOKED)
2-of-2
70,000 sat
to_remote
with revocation key
OR
to_self
432 blocks delay
70,000 sat
to_remote

If Alice cheats putting old tx into blockchain, Bob has 432 blocks of time to use the revocation key and take also the other 70k sats



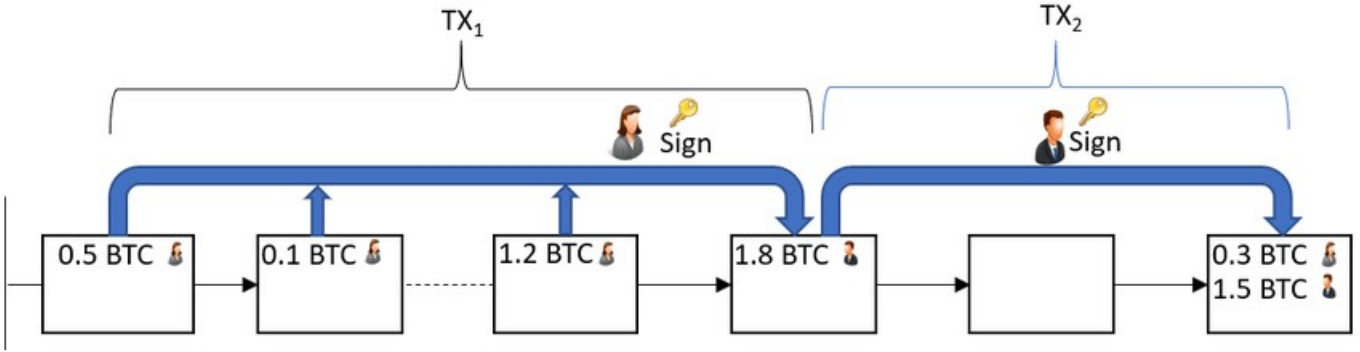
In a multi-node scenario, we can’t expect a channel of each couple of nodes

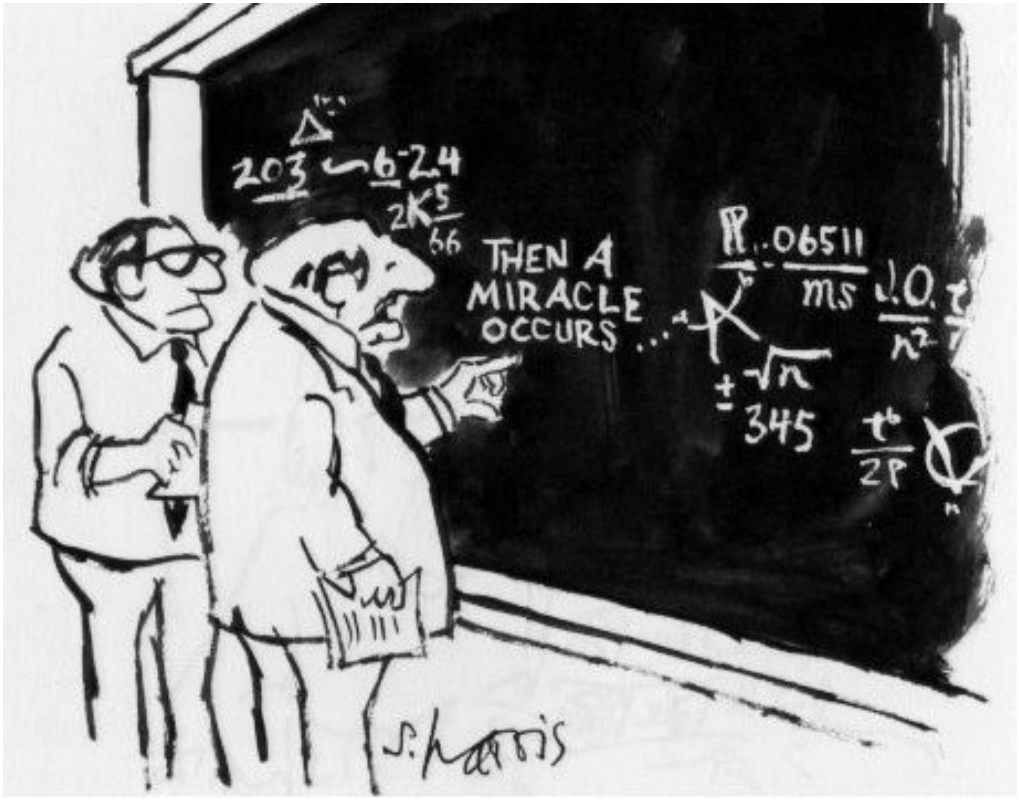



2
Dina's node generates a secret R
Alice
1
Alice requests an invoice on Dina's website
Dina
3
Dina's website provides an invoice containing a payment hash
H = 0575965b3b44be51e8057d551c4016d83cb1fba9ea8d6e986447ba33fe69f6b3
Balance 70,000 sat
Balance 50,000 sat
Payment preimage: R
Payment hash: H=Hash(R)
Balance 70,000 sat
Balance 200,000 sat
Bob
Balance 1,000,000 sat
Balance 4,000,000 sat
Chan

Let’s revisit Transactions...

- Alice sends to Bob means:
  Alice sign one or more of her previous received tx and use them as input for a new output to the public key of Bob





"I think you should be more explicit here in step two."

from *What's so Funny about Science?* by Sidney Harris (1977)

Alice send “to Bob”

“To Bob” = “to Bob public key” **P_bob**
Bob will be able to unlock showing the signature that demonstrate he “knows” the private key **n_bob**
Looking inside the transaction, you will find a “locking script”
<Pubkey of Bob here> **OP_CHECKSIG**

How Bob “unlocks”?

To unlock, Bob just complete the locking script with the missing piece. Every node can execute the script that will return true if Bob provided a valid signature

<Signature> **<Pubkey of Bob here>**
OP_CHECKSIG

Bitcoin Script Language

- Can do more than the simple:
  “show me the signature that demostrate you have the private key”
- Multiple logic conditions can be embbeded in transaction
- It’s a simple scripting language, Turing-incomplete by design to avoid loops

Locking Logic can be Everything

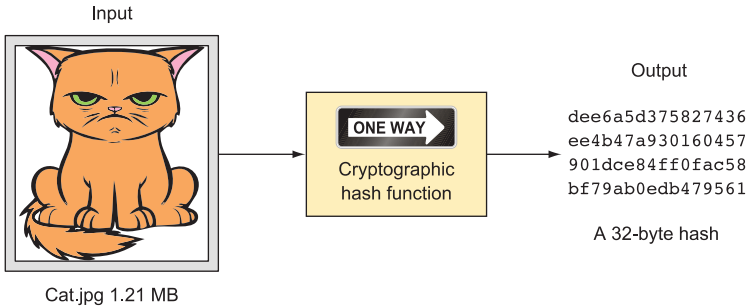
- ….But some scripts would be really stupid :)
- Example: Imagine an output sent to the solver of

  “1 2 OP_ADD OP_EQUAL”
- Instead of needing to provide signature corresponding to a public key, it can be unlocked by simply…

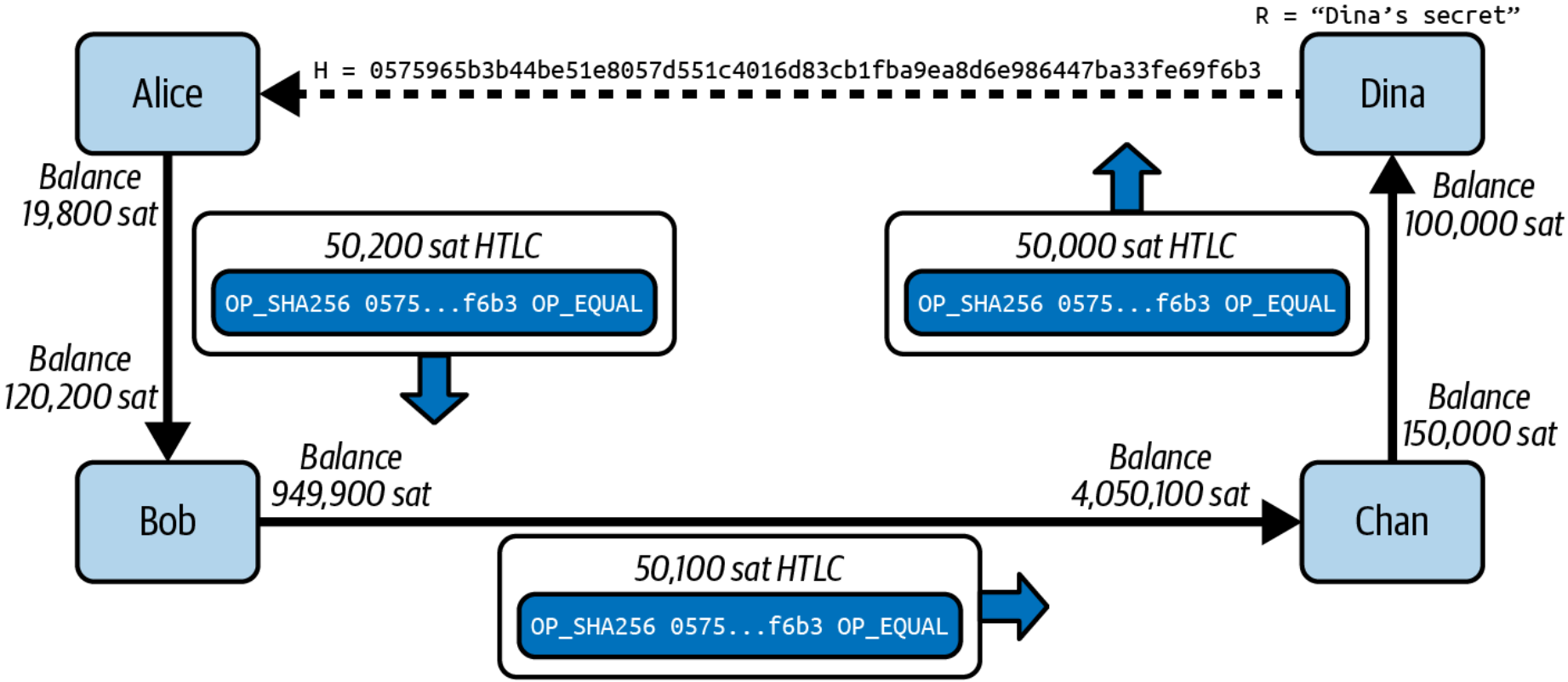
https://siminchen.github.io/bitcoinIDE/build/editor.html

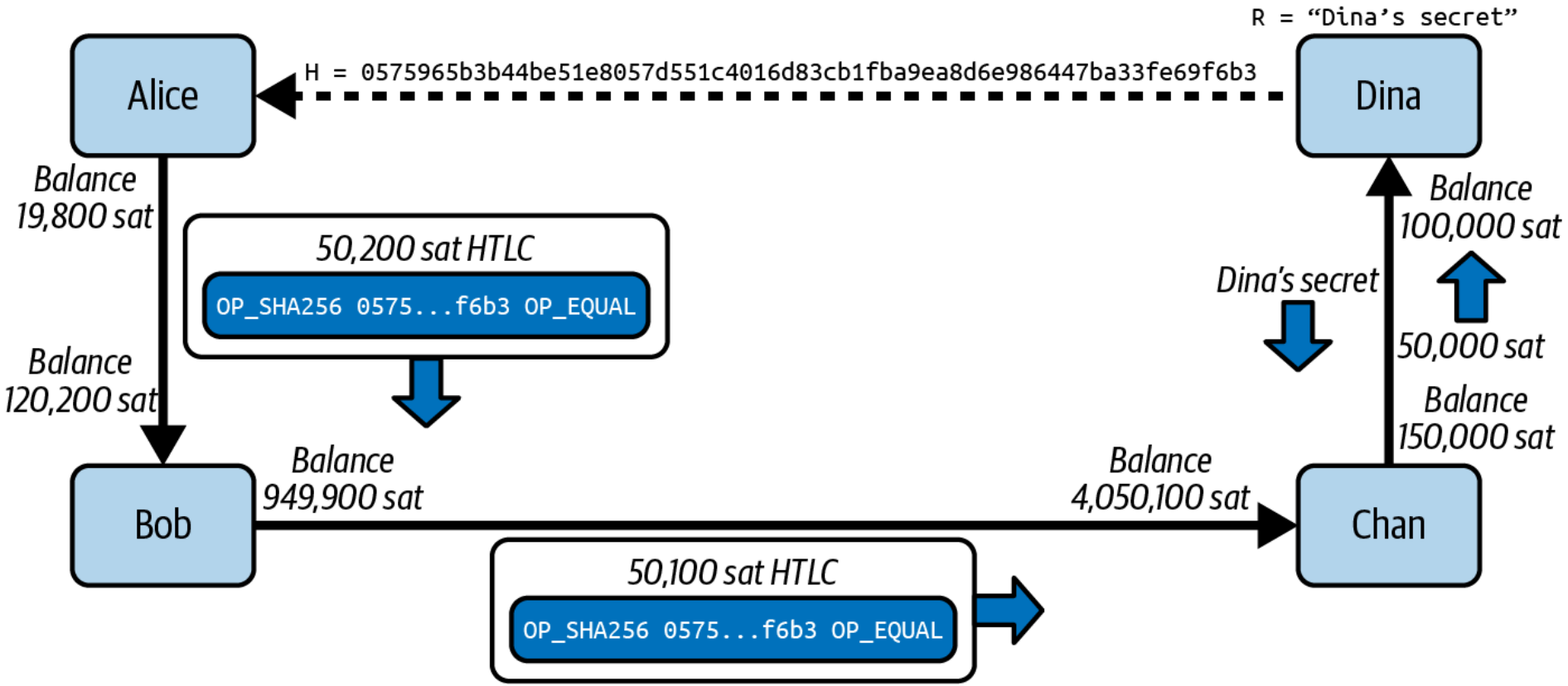
Hash Time-Locked Contracts

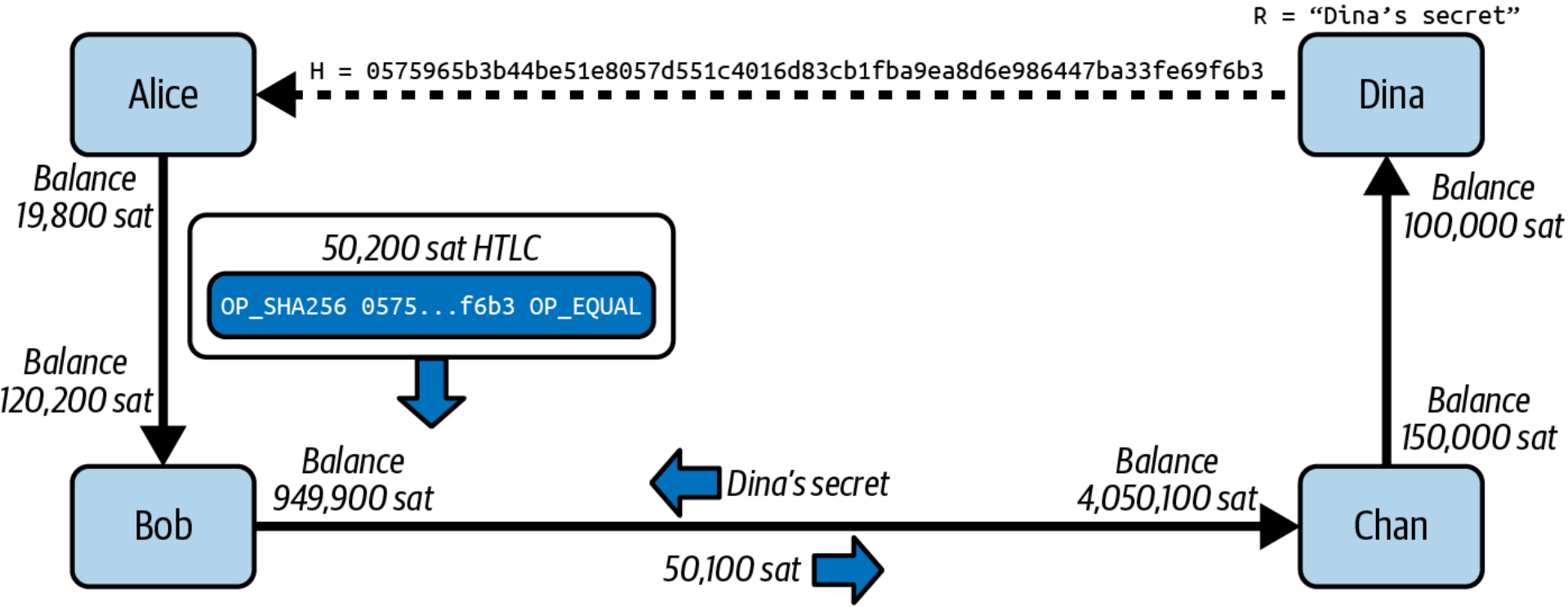
- As usual, you must demostrate of owning the private key corresponding to your public key
- **You must also** demostrate of knowing the secrent number that was used to produce some hash H

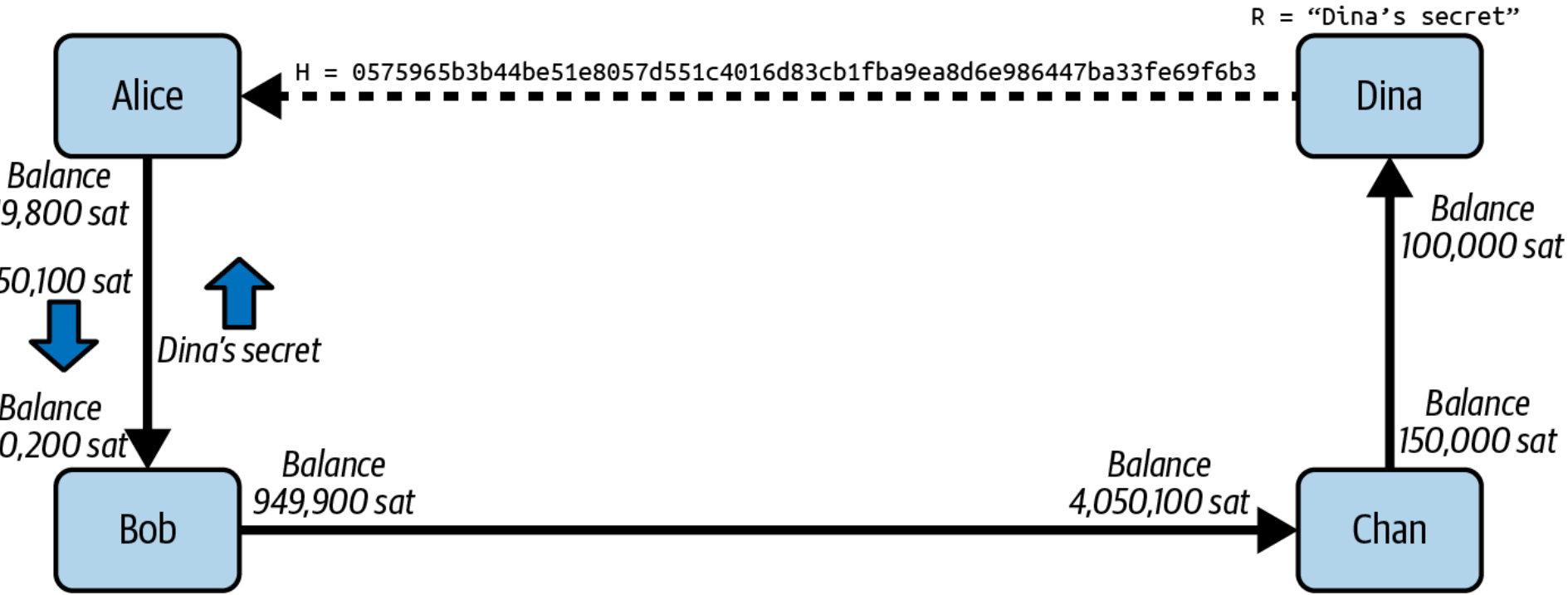


When receiving the tx, Dina must reveal the secret to Chan to get the 50k sat and update the channel balance
So Chan will be able to get its 50,100, gaining 100 sats as routing “cost”

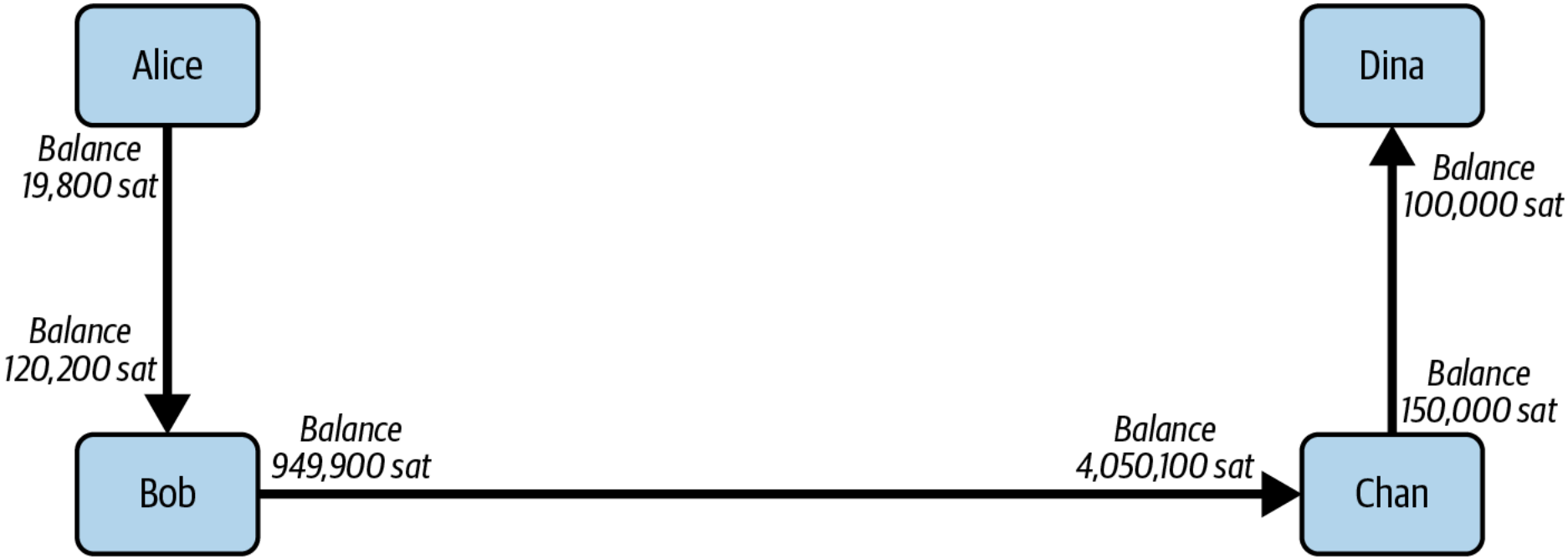



R = "Dina's secret"
Alice
H = 0575965b3b44be51e8057d551c4016d83cb1fba9ea8d6e986447ba33fe69f6b3
Dina
Balance
19,800 sat
50,200 sat HTLC
OP_SHA256 0575...f6b3 OP_EQUAL
Dina's secret
Balance
100,000 sat
50,000 sat
Balance
120,200 sat
Balance
150,000 sat
Balance
949,900 sat
Balance
4,050,100 sat
Bob
Chan
50,100 sat HTLC
OP_SHA256 0575...f6b3 OP_EQUAL


R = “Dina’s secret”
Alice
H = 0575965b3b44be51e8057d551c4016d83cb1fba9ea8d6e986447ba33fe69f6b3
Dina
Balance
19,800 sat
50,200 sat HTLC
OP_SHA256 0575...f6b3 OP_EQUAL
Balance
100,000 sat
Balance
120,200 sat
Balance
150,000 sat
Balance
949,900 sat
Dina’s secret
Balance
4,050,100 sat
Bob
Chan
50,100 sat


R = “Dina’s secret”
H = 0575965b3b44be51e8057d551c4016d83cb1fba9ea8d6e986447ba33fe69f6b3
Alice
Dina
Balance
9,800 sat
50,100 sat
Dina’s secret
Balance
100,000 sat
Balance
0,200 sat
Balance
949,900 sat
Balance
4,050,100 sat
Balance
150,000 sat
Bob
Chan

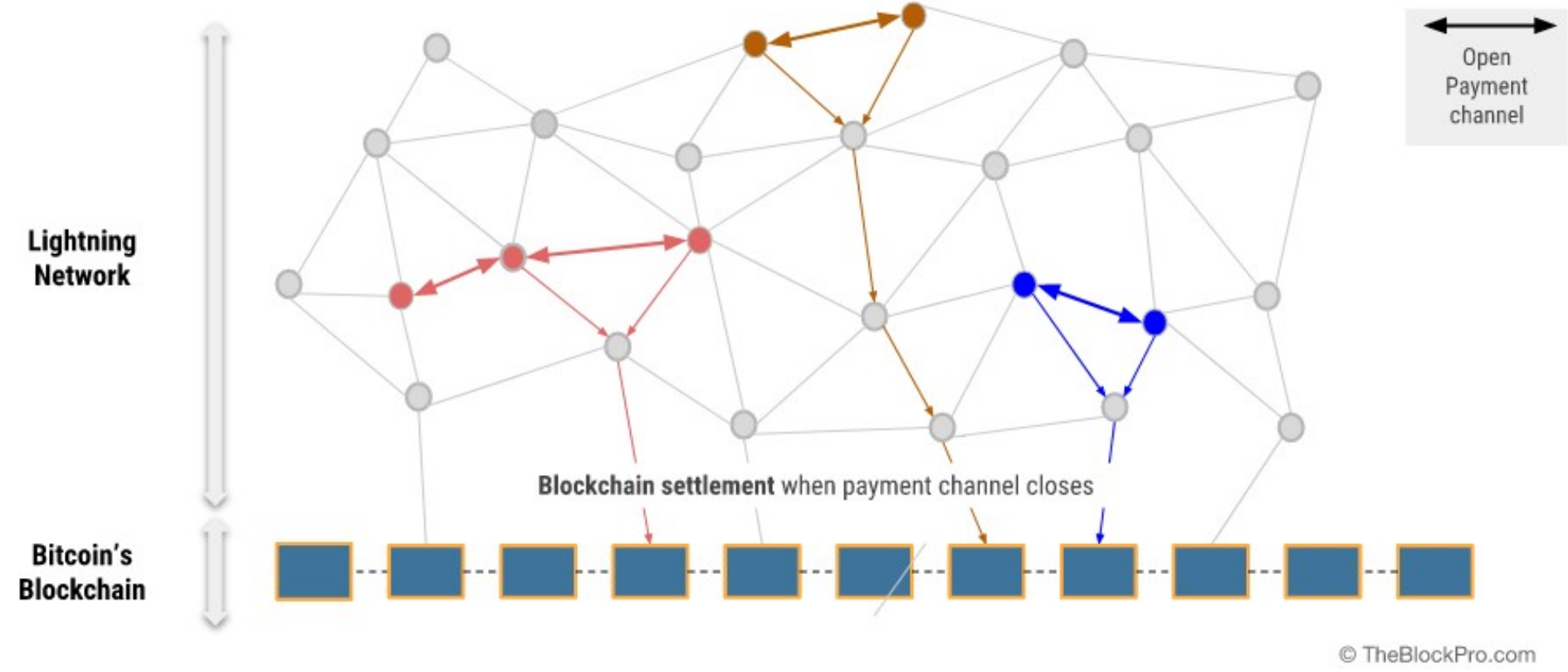
Final Status: 50k sats moved



Streaming satoshis

- Each Bitcoin or BTC can be split into 100,000,000 pieces. A Satoshi is one-hundred millionth of a Bitcoin (0.00000001).
- The LN allows sats to be further subdivided into millisatoshis. This allows users to make payments as small as $0.0000001 or one hundred thousandth of a dollar cent per transaction.
- This design feature will make it possible for satoshis to be streamed on a per second base.

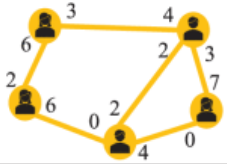
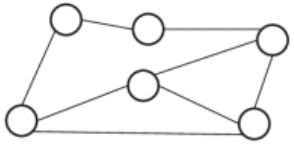
Lightning Network



Exploring The Lightning Network

https://explorer.acinq.co/

Research and Challenges

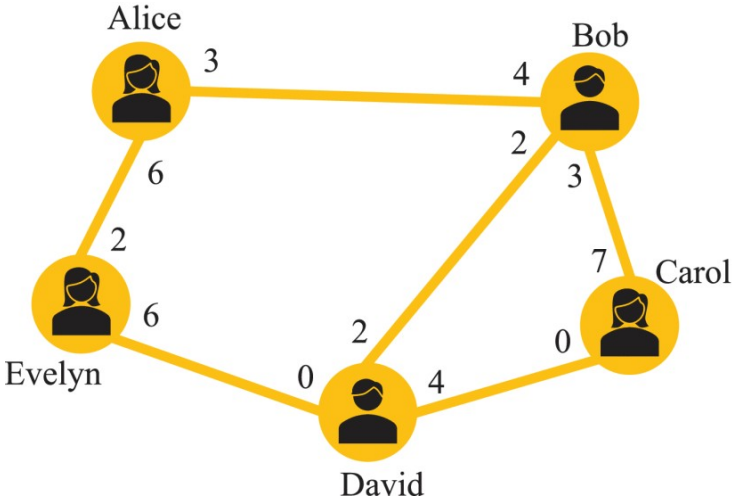
Application Layer		Applications
Layer 2		Payment/State Channel Networks
Layer 1		Consensus
Layer 0		P2P Network
Hardware Layer		(Trusted) Hardware

Research and Challenges

- The base layer represent a solid and well estabilished foundation on top of which a flourishing interest from Academia and Industry is growing
- **The Lightning Network research is a mix of Computer Science, Economy and Game-Theory**

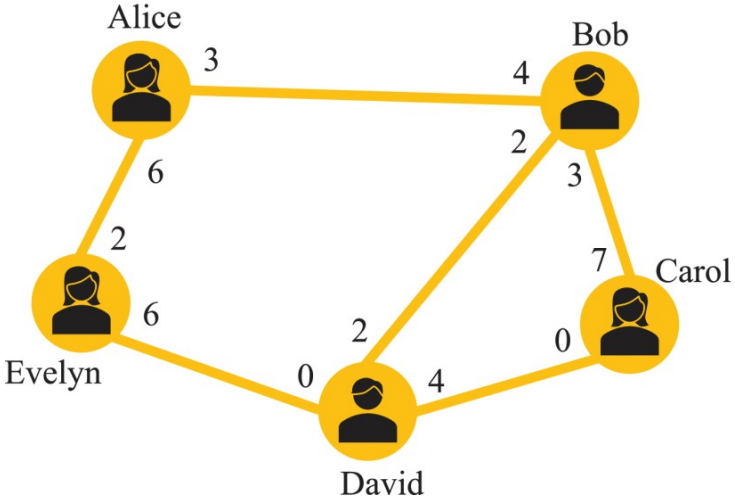
LN Challenges: Routing

- Routing, a classic computer science problem, to be addressed from new perspectives
- **Total Channels Capacity is known, but the current distribution in the two sides is hidden**
- This can lead to failed attempts
- Alice move 3 sats to Carol
  - What changes if the amount is 4?



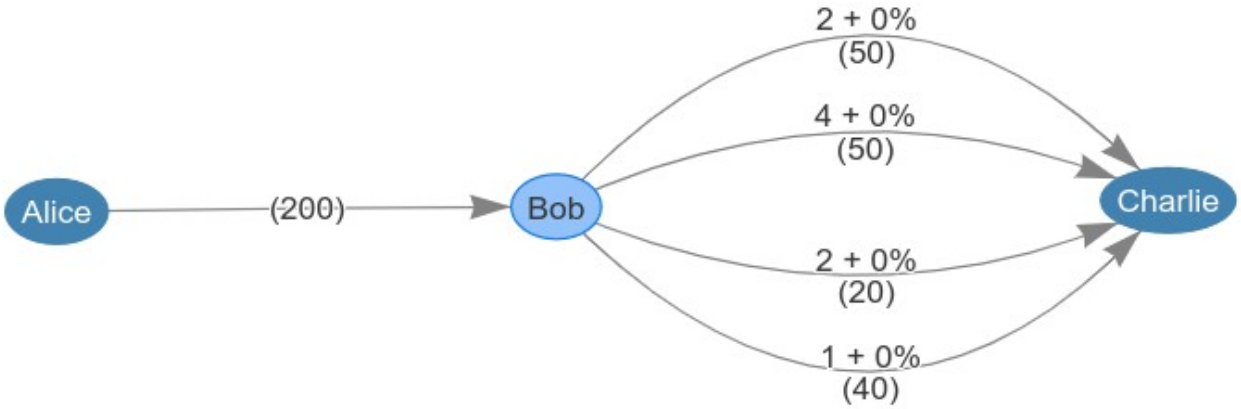
LN Challenges: Routing

- Not only each channel balance is unkown, but it will ALSO change after each routing
- For example, if Alice move 3 sat to Carol via Bob
- The updated balances will be (0, 7) in the Alice-Bob channel and (0, 10) in the Bob-Carol channel.



Atomic Multi-Path Routing

Idea: A single big transfer can be split into multiple small ones, each of them possibly following a different path and joining the others at the final destination



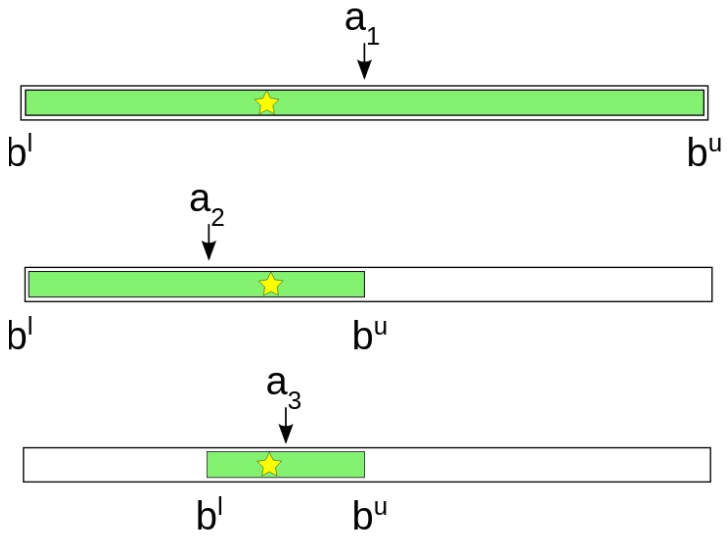
https://www.robtex.com/lnemulator.html

Lightning Network Routing

- Channel balances are not “consumed”, but moved between sides
- Currently, source routing is used: Alice tries the paths according to some metrics
- **How to compute the probability of failure?**
- **What information you get when you fail a some point?**

Channel Probing

- Failures can be used to perform binary search
- **privacy-utility tradeoffs**: revealing more information (e.g., the exact channel balances) would lead to higher success rates, but less privacy

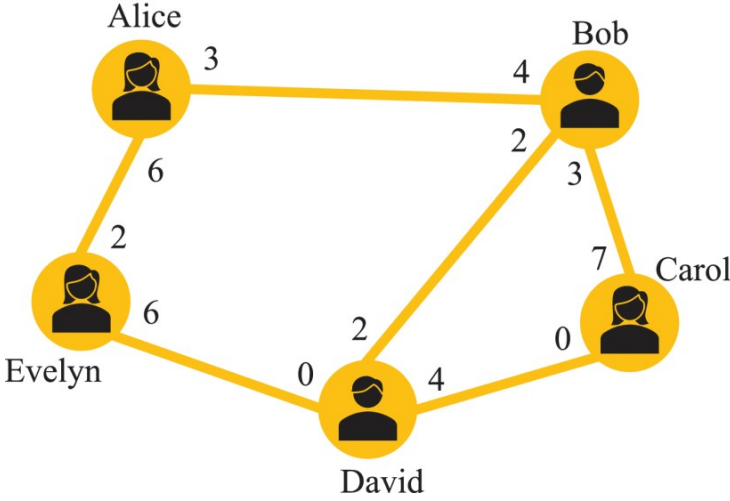


W. Tang, W. Wang, G. Fanti and S. Oh, "Privacy-utility tradeoffs in routing cryptocurrency over payment channel networks", Proc. ACM Meas. Anal. Comput. Syst., vol. 4, no. 2, pp. 81-82, Jun. 2020

Routing Fees

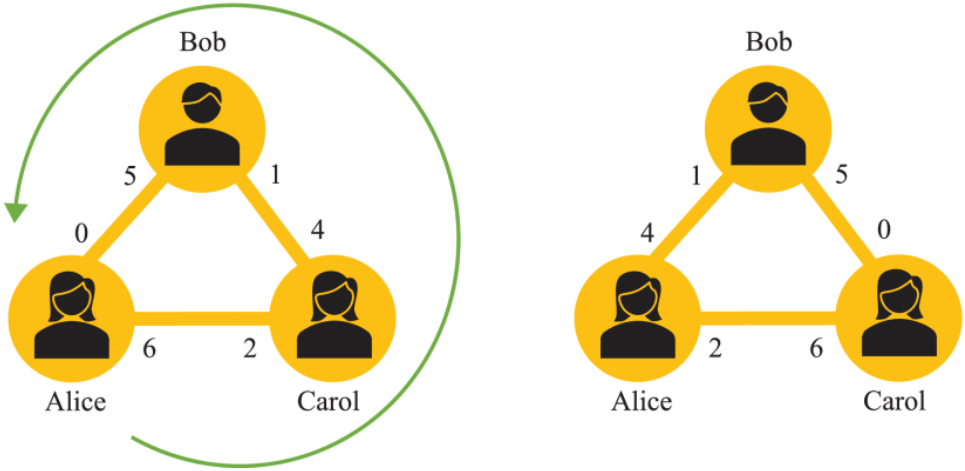
- While a node (e.g. Bob) is partecipating to the transfer to 3 sats from Alice to Carol, 3 sats of his channel remains “locked” until the transfer fails or succeed
- Each node announce in public “how much” it will ask for this forwarding service

Which logic should Bob use for fees?
Hight fees of Bob toward Carol, Alice will not choose him for its paths
Low fees, Bob-Carol channel will be often unbalanced toward Carol



Rebalancing

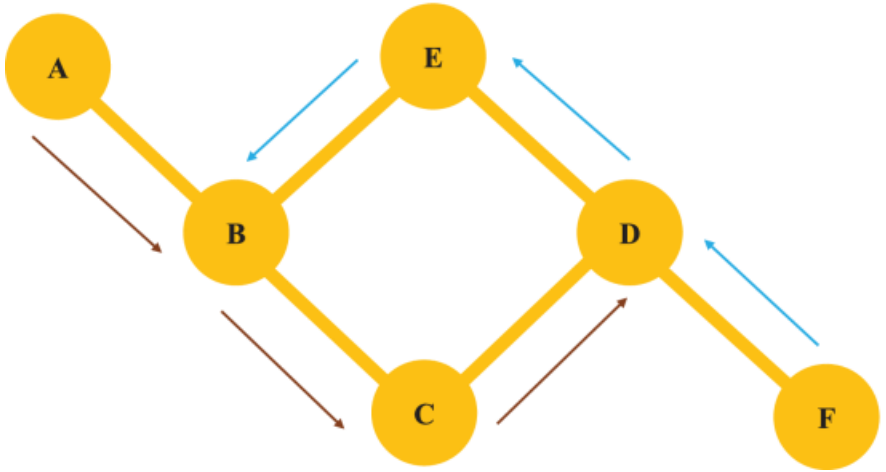
Think about Alice sending to herself just to balance her channel to Bob... What are your ideas about it?



Deadlock Example

A deadlock occurs because X sats of the channel D – E needed to complete the first payment are reserved by the second, and at the same time, Y sats of the channel B – C needed to complete the second payment are reserved by the first.

Notice: it’s not necessary that X=Y, just that remaining balance on the side in not sufficient



Experimenting with Containers

```
$ git clone https://github.com/lnbook/lnbook.git
$ cd lnbook


$ docker pull lnbook/bitcoind


$ cd code/docker
$ docker run -it --name bitcoind lnbook/bitcoind


$ docker exec -it bitcoind /bin/bash
```

Experimenting with LN Testnet

https://faucet.lightning.community/
https://htlc.me/

*“If you can replace the word “**blockchain**” with “**distributed database**” and the meaning doesn’t change... that’s not a blockchain”*

Andreas M. Antonopoulos

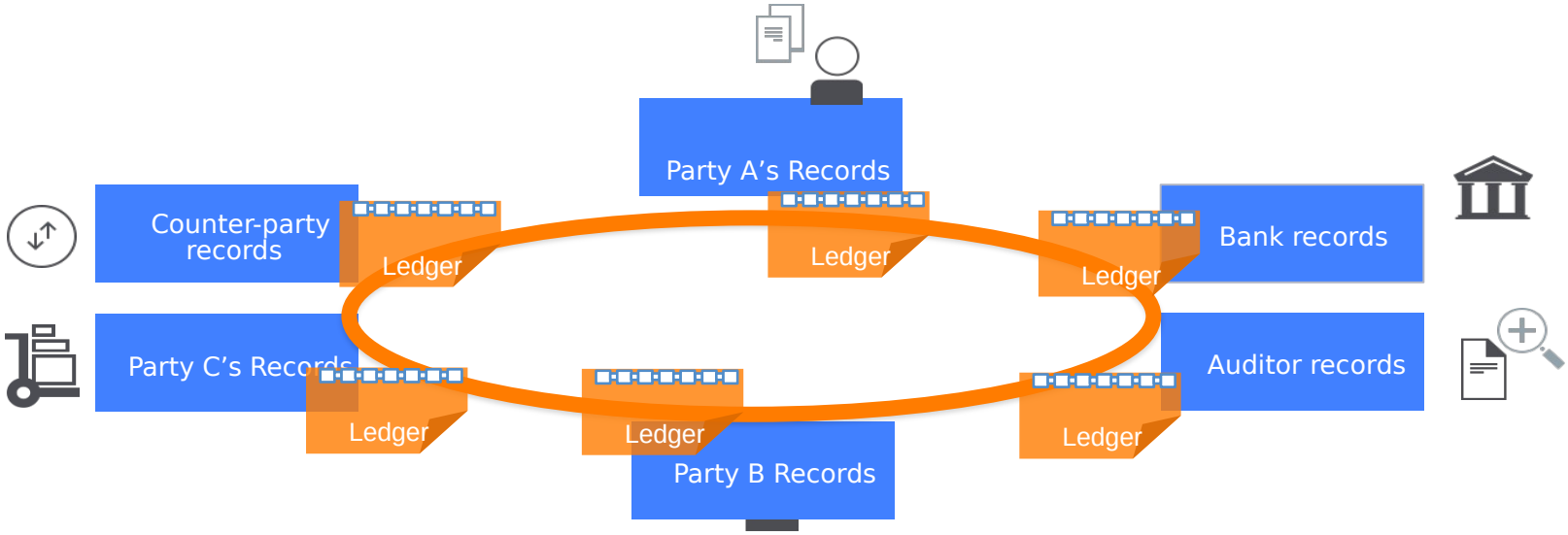
https://www.youtube.com/watch?v=SMEOKDVXlUo&t=0s&ab_channel=aantonop

Other Research Topics

- Chain Analysis (data mining etc...)
- Application Level (mobile, services)
- Hardware (IoT devices, embedded blockchain clients, pervasive mining)

Distributed Ledgers (DLT) are NOT blockchains:

- A consortium of entitities that **still need to trust each other**,
- They need and want control over the permission in partecipation
- **Why they should write events in a sequence of chained blocks and don’t just use private key to access a centralized database?**
- 6 of them could reverse a 12 partecipants blockchain history!



Blockchain checklist

- **Decentralized**: no need of trusted third-parties or centralized authorities
- **Immutable**: Can’t change or reverse events
- **Open:** anyone can partecipate, no permission
- **Trustless**: everyone can verify the entire chain history on its own

Final goal:

Realize a “digital scarcity” whose “atoms” can be exchanged between entities

References: Lightning Network

Poon, Joseph, and Thaddeus Dryja. "The bitcoin lightning network: Scalable off-chain instant payments." (2016).

N. Papadis and L. Tassiulas, "Blockchain-Based Payment Channel Networks: Challenges and Recent Advances," in IEEE Access, vol. 8, pp. 227596-227609, 2020, doi: 10.1109/ACCESS.2020.3046020.

Croman, K. et al. (2016). On Scaling Decentralized Blockchains. Financial Cryptography and Data Security. FC 2016. Lecture Notes in Computer Science(), vol 9604. Springer, Berlin, Heidelberg.

Pickhardt, Rene, and Mariusz Nowostawski. "Imbalance measure and proactive channel rebalancing algorithm for the Lightning Network." 2020 IEEE International Conference on Blockchain and Cryptocurrency (ICBC). IEEE, 2020.

Resources

General:

- https://nakamotoinstitute.org/static/docs/bitcoin.pdf
- https://www.youtube.com/watch?v=UlKZ83REIkA&ab_channel=aantonop
- https://www.lopp.net/bitcoin-information/history.html

A complete Simulator and Tutorial:

- https://www.bitcoinsimulator.tk/

Security-related Collection:

- https://www.lopp.net/bitcoin-information/security.html

Technical Links Collection:

- https://www.lopp.net/bitcoin-information/technical-resources.html

Philosophical:

- https://dergigi.com/2021/01/14/bitcoin-is-time/
- https://www.sfbitcoindevs.org/

Thank You