Storm: layer 2/3 storage & messaging

or «a favorite shitcoins use case is being destroyed with Bitcoin L2»

Maxim Orlovsky, PhD, MD
Chief Engineering Officer,
Pandora Core AG, Swiss

contributors:

Giacomo Zucco,
Federico Tenga, Chainside
Marco Amadori, inbitcoin
Martino Salvetti, inbitcoin
Nicola Busanello, inbitcoin
Sabina Sachtachtinskagia, Pandora Core
Stefan Richter @stefanwouldgo
ZmnSCPxj
...

You Retweeted

Mike Rauchstings - giving away private keys!
@CryptoBacon

Your favorite shitcoins usecase is being built on top of Bitcoin.

#AltSeasonTerminated

> 
>
> **Dr Maxim Orlovsky [LNP/BP]** @dr_orlovsky · Aug 16
>
> A new and shiny piece of Bitcoin technology is out: #Storm: L2/L3 distributed storage & messaging with economic incentives leveraging LNP/BP ecosystem. No ICO and tokens are present! :)
>
> Read more github.com/storm-org/stor... and give your feedback and comments!

7:44 PM · Aug 17, 2019 · Twitter for Android

Problem: storage

- Lightning Network channel state history
- Eltoo channel state
- Scriptless scripts
- Single-use seals off chain data
- … much more

We need economic incentives for all of that!!!

Can it be trestles but guaranteed?

Yes, by utilizing

- Probabilistically checkable proofs
- HTLCs
- PBST

Setting

- **Bob stores data for Alice**

- **Alice must be guaranteed to receive Bobs money** if she stored the data - no matter if Bob is still interested in receiving the data or running with the data away without paying once he has the data

- **Bob must be compensated if Alice fails to keep the data**

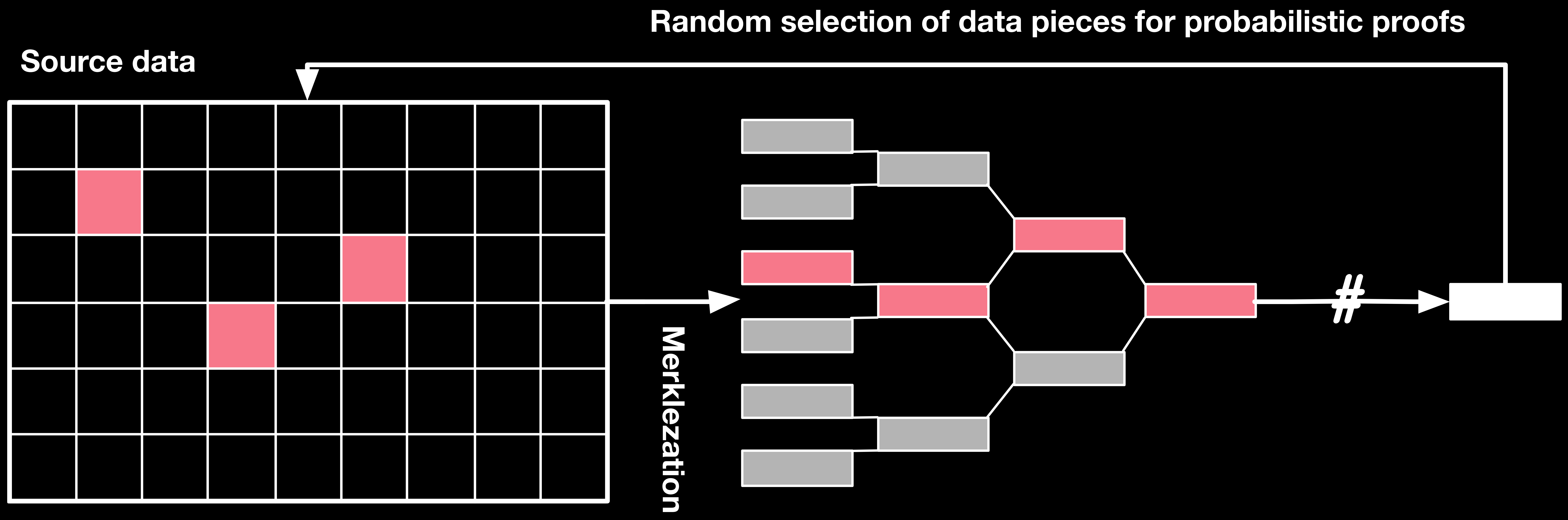
Bob may encrypt the data, split the data across different Alice(s) etc

Intuition for core "tricks"

- Bob can proof the fact that he has the data in a **succinct** way both to Alice and on-chain with the current Bitcoin script by utilizing probabilistically checkable proofs

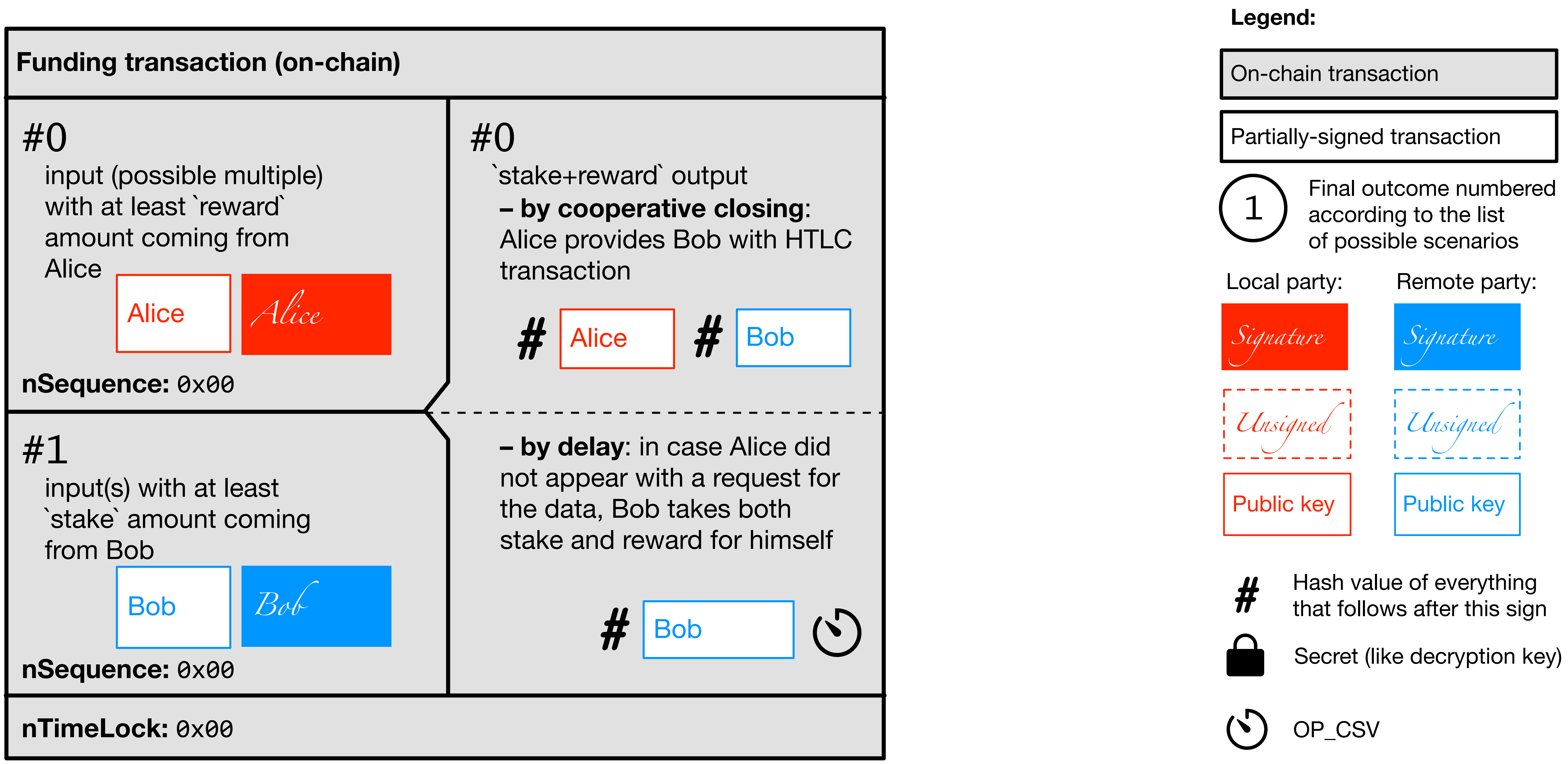
- Alice gets obscured data from Bob **encrypted with his yet unknown public key** and is able to decrypt them only when the Bob takes his payment

Probabilistically checkable proofs



Steps

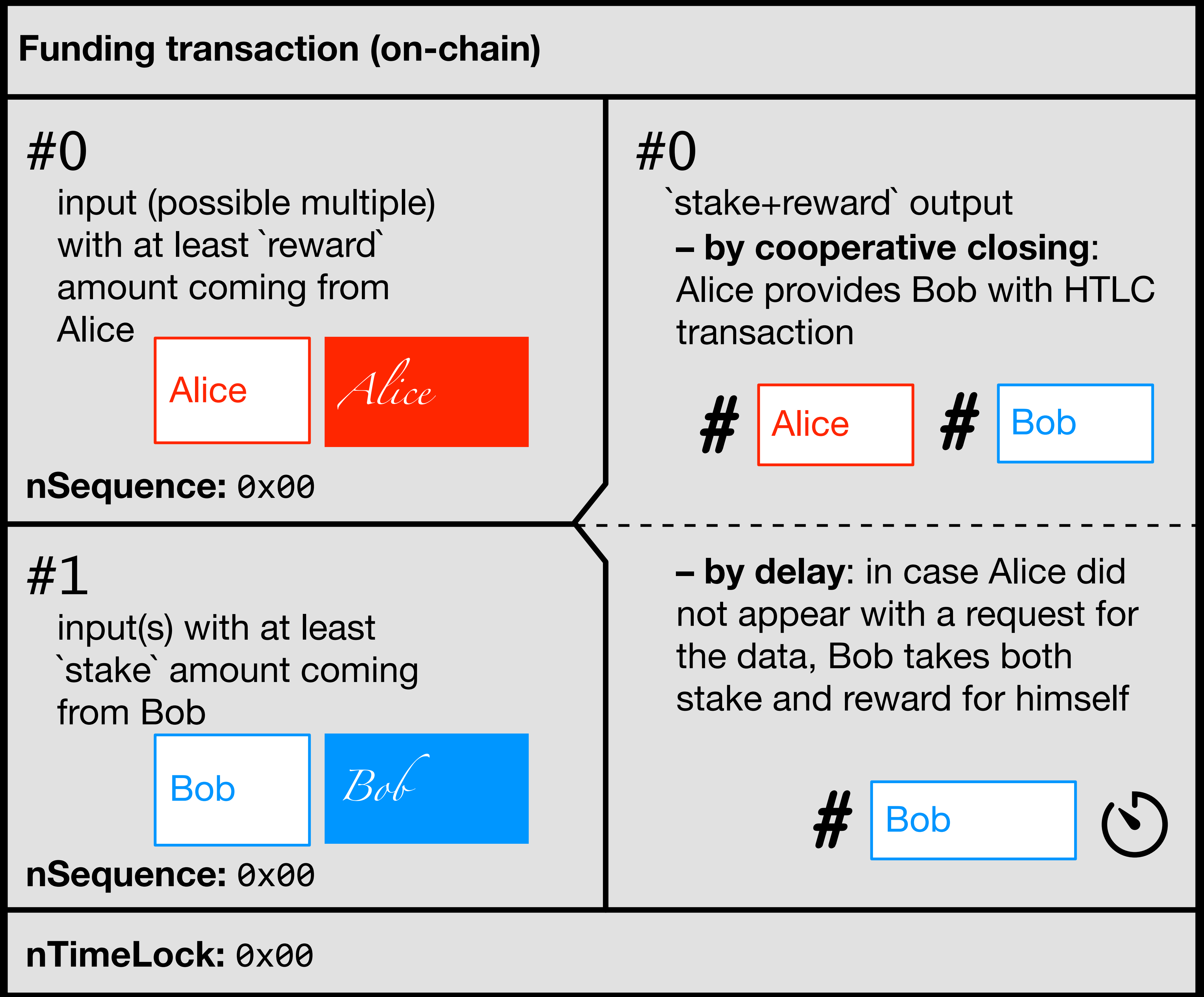
- Bob stores data for Alice
- Alice puts payment and Bob puts stake under escrowed time locked contract


Funding transaction (on-chain)
#0
input (possible multiple) with at least `reward` amount coming from Alice
Alice
Alice
nSequence: 0x00
#0
`stake+reward` output
– by cooperative closing: Alice provides Bob with HTLC transaction
#
Alice
#
Bob
– by delay: in case Alice did not appear with a request for the data, Bob takes both stake and reward for himself
#
Bob
#1
input(s) with at least `stake` amount coming from Bob
Bob
Bob
nSequence: 0x00
nTimeLock: 0x00
Legend:
On-chain transaction
Partially-signed transaction
1
Final outcome numbered according to the list of possible scenarios
Local party:
Remote party:
Signature
Signature
Unsigned
Unsigned
Public key
Public key
#
Hash value of everything that follows after this sign
Secret (like decryption key)
OP_CSV

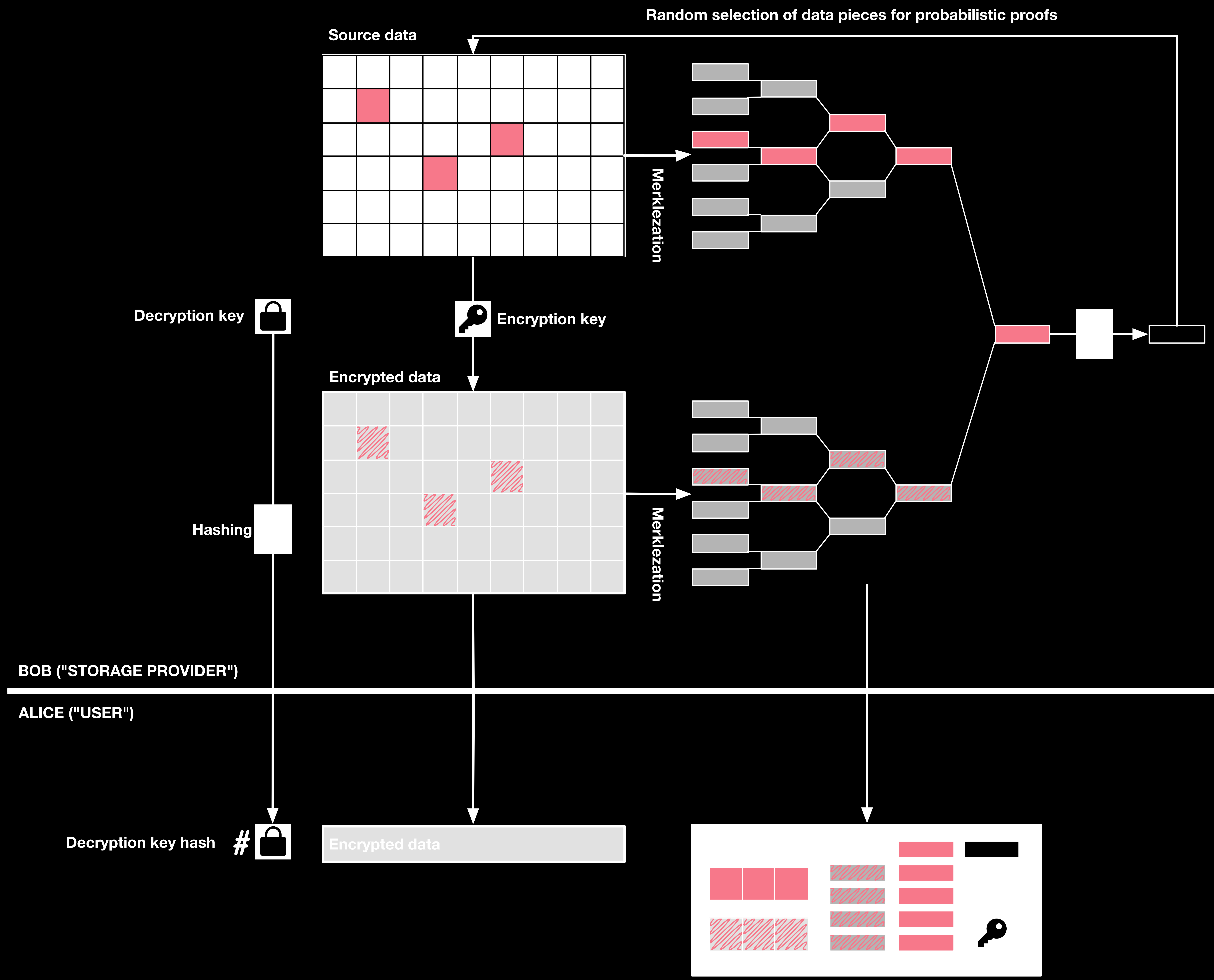
Steps

- Bob stores data for Alice
- Alice puts payment and Bob puts stake under escrowed time locked contract
- They pre-sign partial transactions for different scenarios

Closing scenarios: Alice timeout

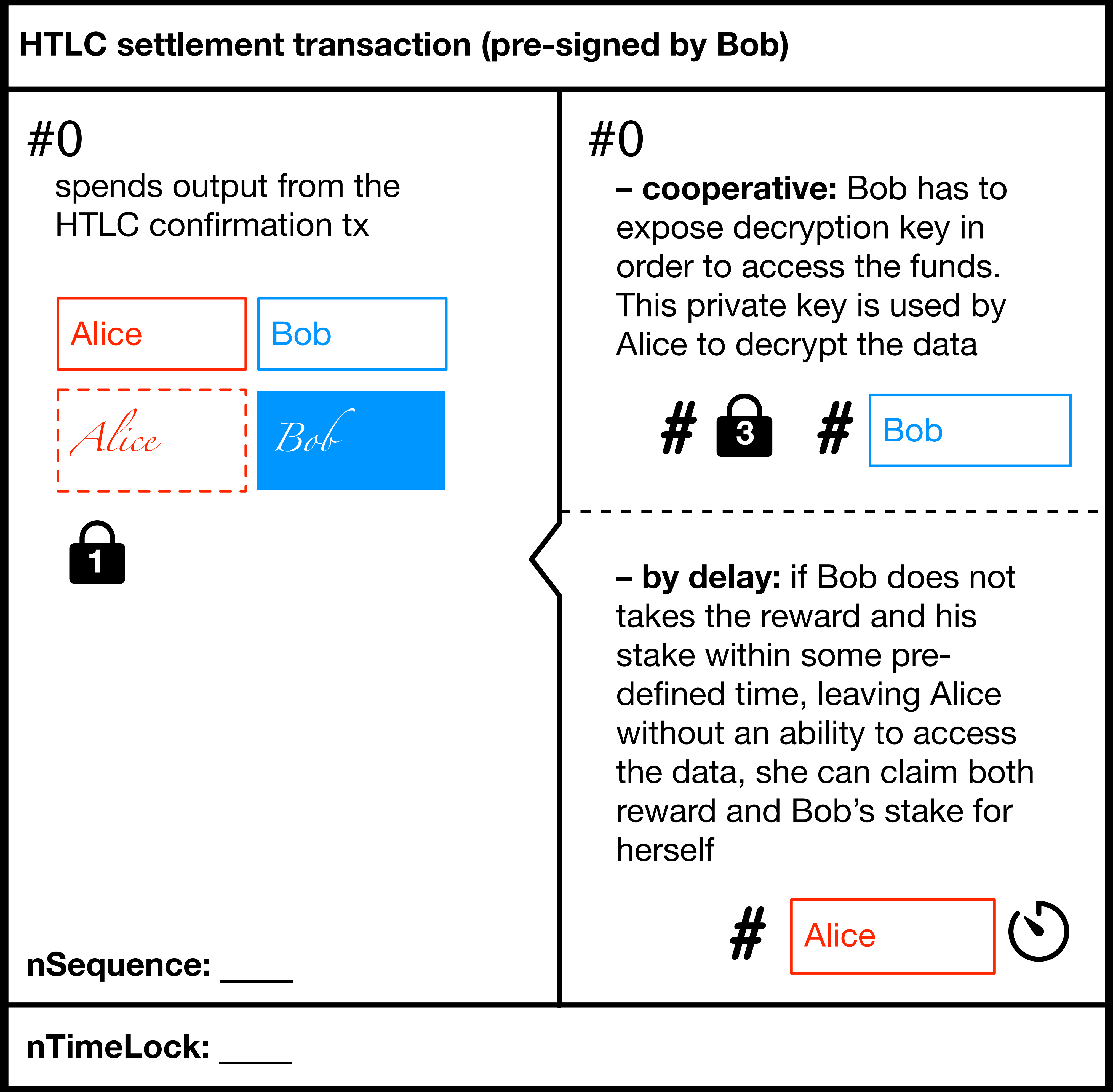


- If Alice forgets about her data, Bob still takes the payment for storage and his stake back



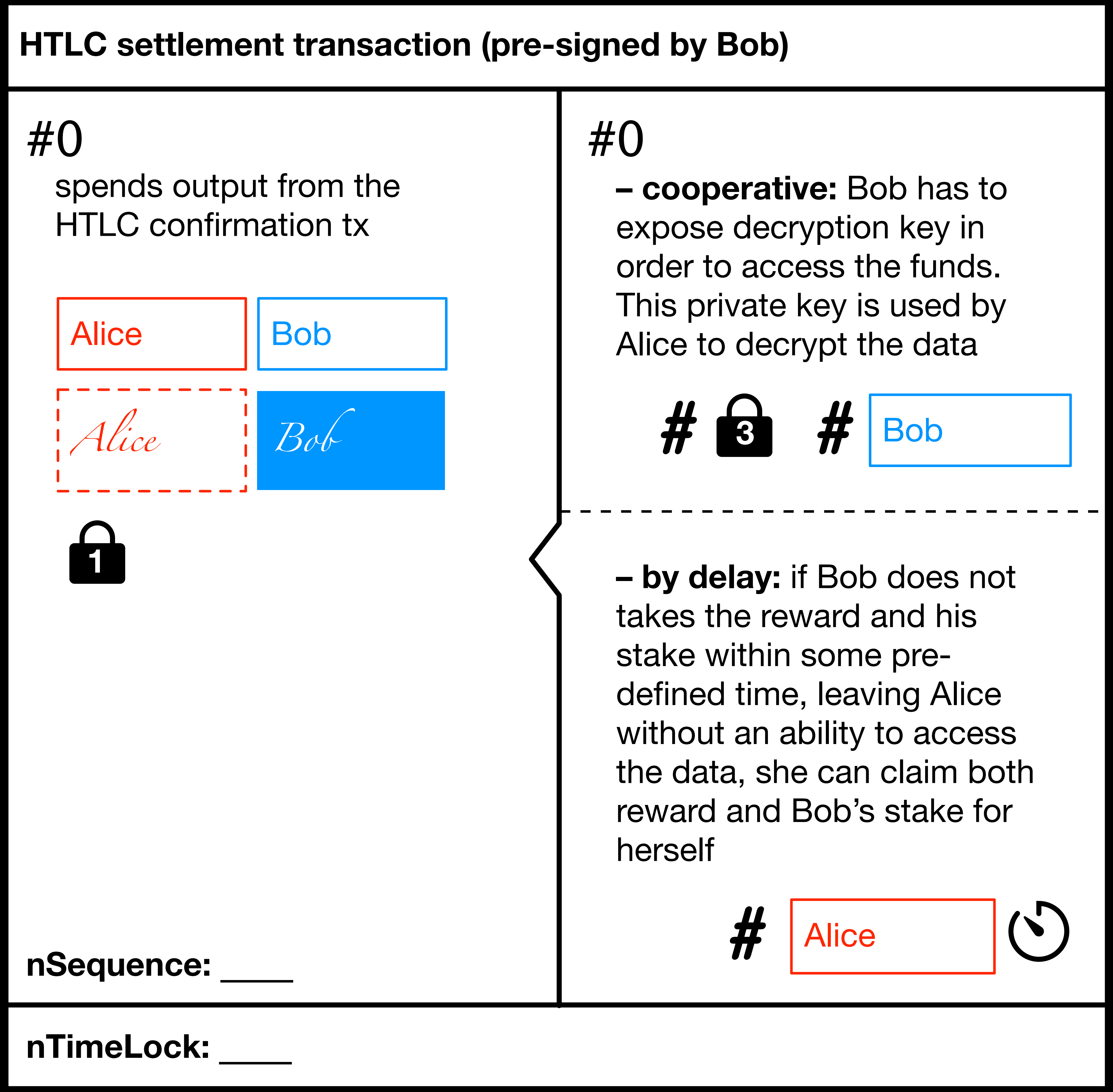
- Bob encrypts Alice data with some public and private key pair
- Bob constructs special PCP proof showing Alice that he has really encrypted the original data

Closing scenarios: cooperative



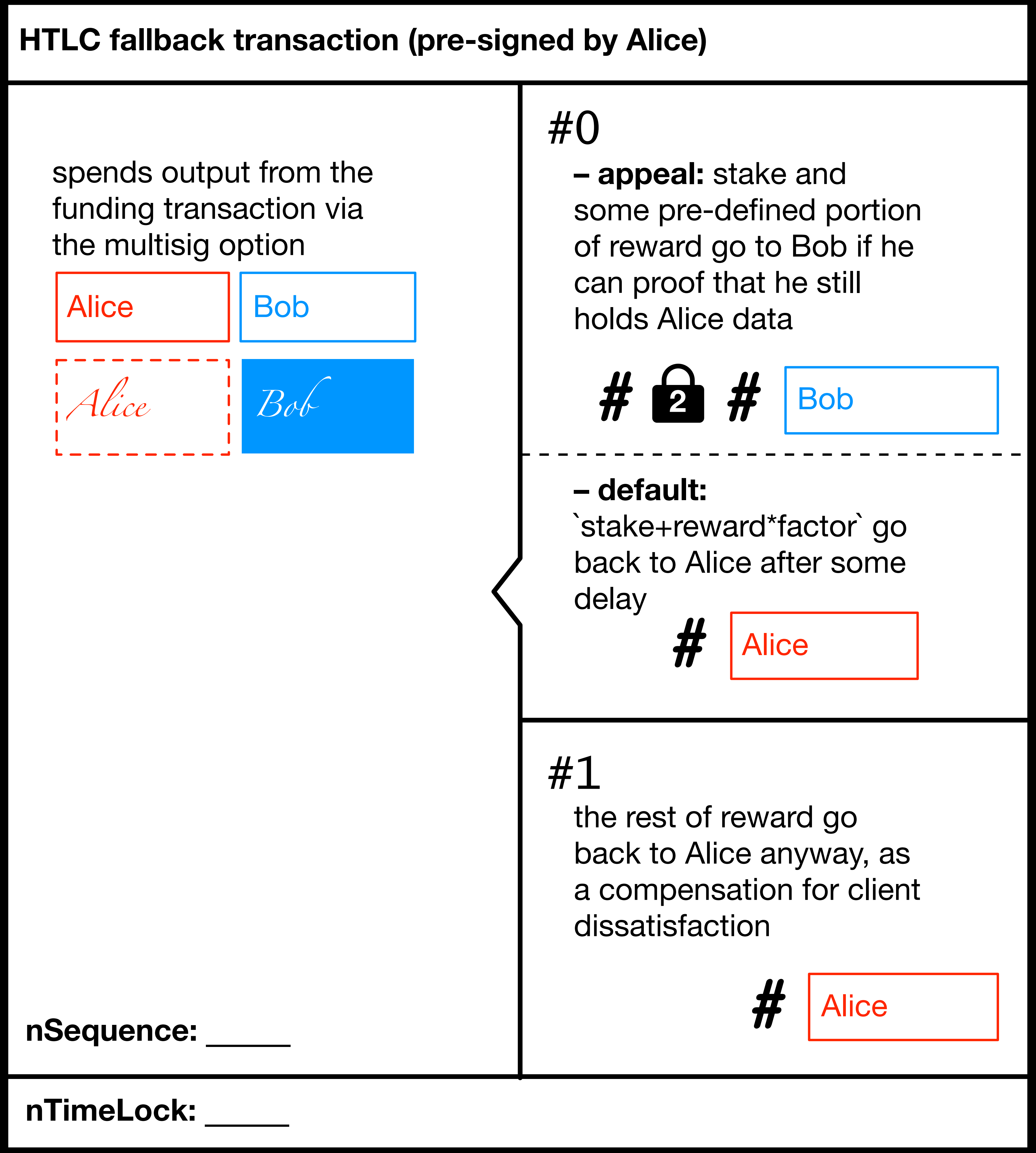
- If Alice is happy with Bob's proof, she signs pre-signed Bob's transaction.
- When Bob claims funds from #0 output, he reveals encryption key, so Alice is able to decrypt her data

Closing scenarios: cooperative



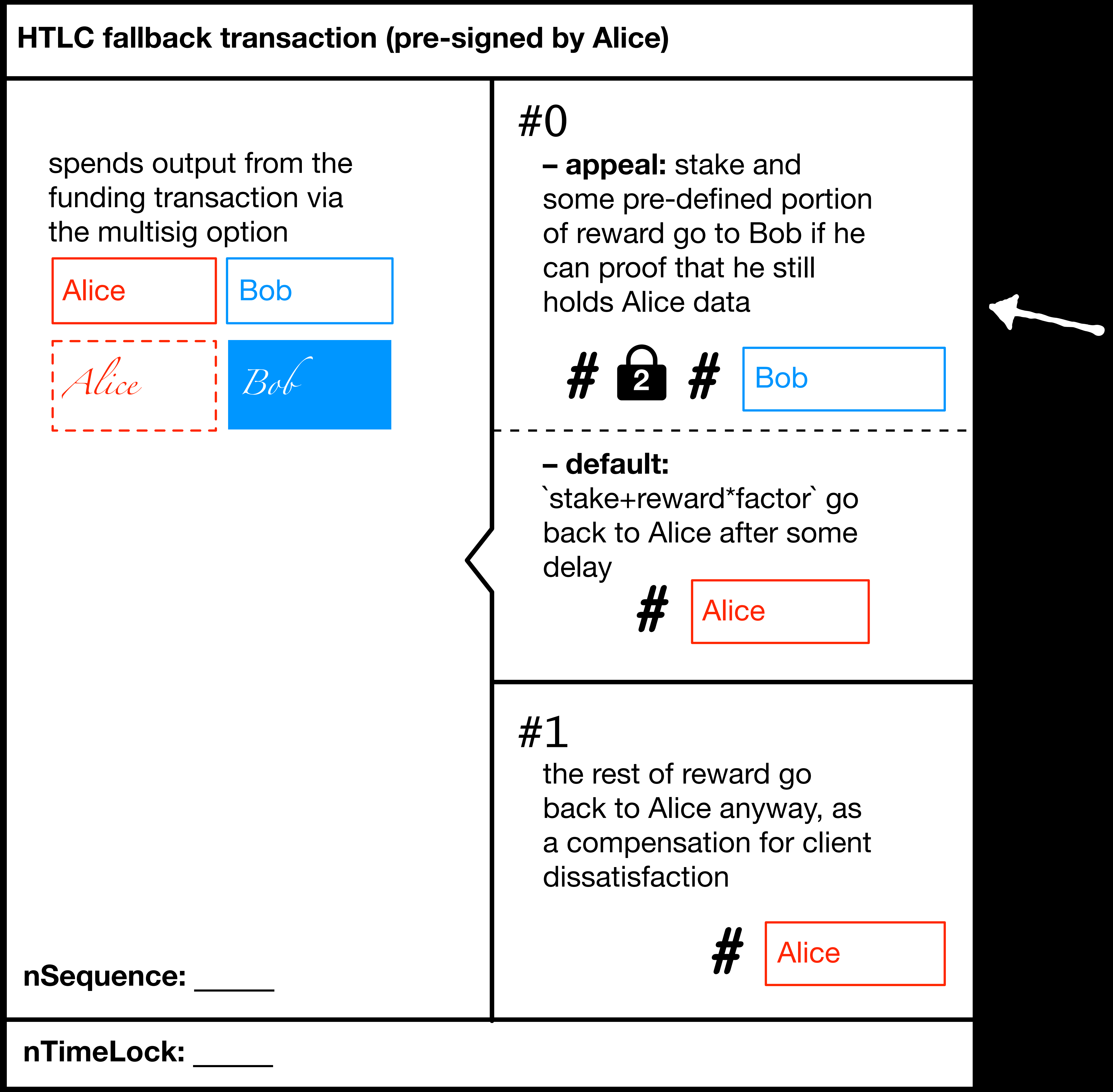
- If Alice is happy with Bob's proof, she signs pre-signed Bob's transaction
- If Bob disappears after that, Alice will be able to get her money back plus Bob's stake to compensate the loss of the data

Closing scenarios: non-cooperative



- If Alice is **not** happy with Bob's proof, she signs another pre-signed Bob's transaction
- with it, after some delay she will get both her money and Bob's stake to compensate the loss of the data

Closing scenarios: non-cooperative



- If Alice is **not** happy with Bob's proof, she signs another pre-signed Bob's transaction
- **Bob can appeal** to that and prove that he has actually kept the data. He has to provide a pre-image composed of parts of the data selected according to the Alice public key exposed to Bob by this closing transaction
- In this case **Bob still gets his stake back plus the reward** (or part of the reward, since Alice as a client is unhappy)

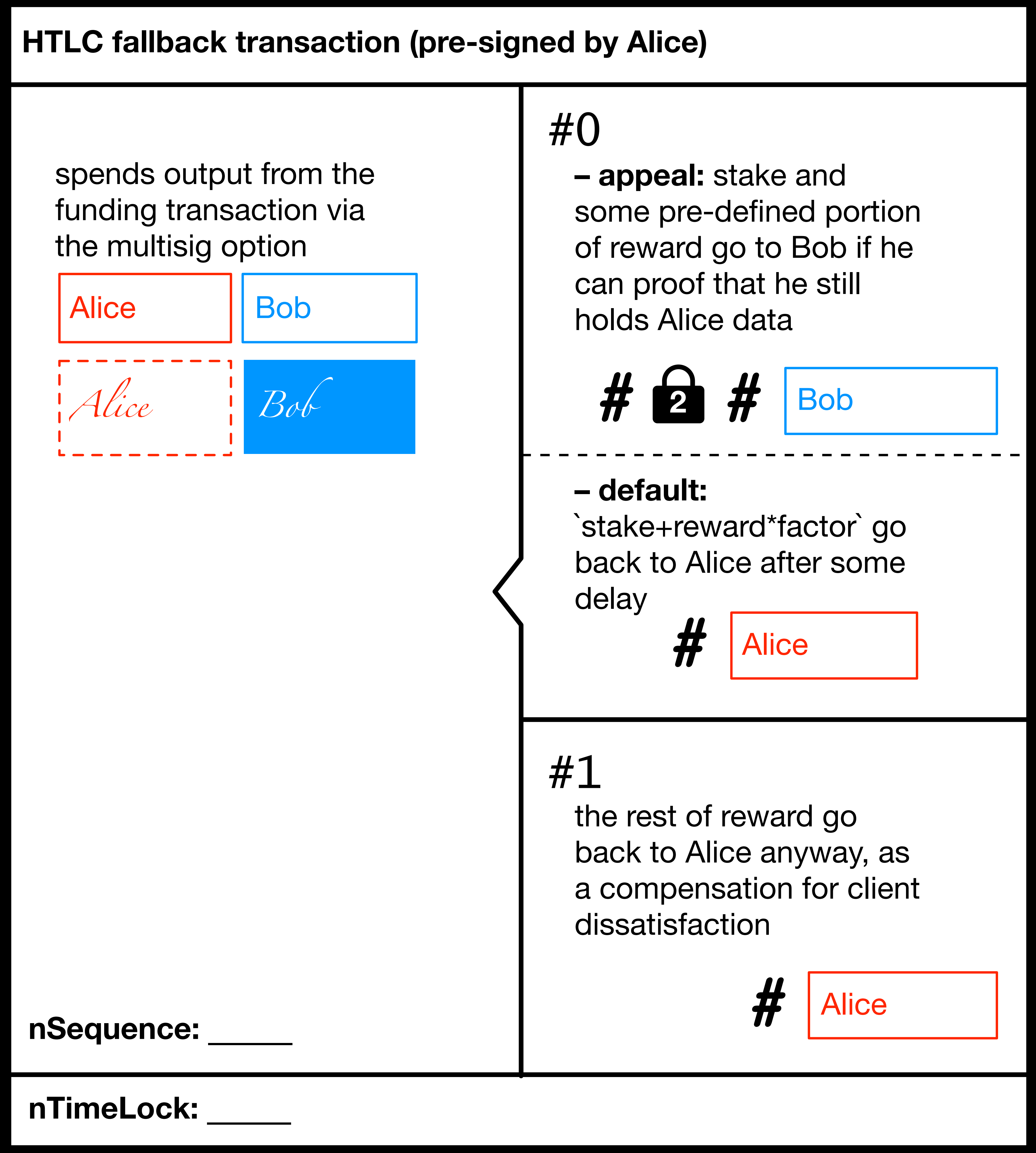
Bob’s proof of data storage

- At setup time **Alice** uses her newly-derived **public key** for both funding transaction output and deterministic definition of some small portion of the source data.

- This portion of the data is double-hashed to 160-bit hash and included into HTLC fallback tx by Alice as a hash lock.

- When Bob wants to prove that he still has the data available, he see the published HTLC transaction, **extracts Alice public key and uses it to get the same deterministic piece of the source data as Alice**. Bob computes a single hash on the data, which gives him a preimage to unlock the hash lock from the HTLC transaction output before Alice will spend it (Alice's branch is timelocked).

Closing scenarios: non-cooperative



- If Alice is **not** happy with Bob's proof, she signs another pre-signed Bob's transaction

- **Bob can appeal** to that and prove that he has actually kept the data. He has to provide a pre-image composed of parts of the data selected according to the Alice public key exposed to Bob by this closing transaction

- In this case **Bob still gets his stake back plus the reward** (or part of the reward, since Alice as a client is unhappy)

Why important?

- Alice needs to store only seed phrase to keep all here L2/L3 data
- Incentivization for watchtowers and other schemes
- Potentially can be done on top of Lightning Network: zero transactions will reach blockchain

Limitations

- The same security assumptions as for ZP: proofs are probabilistic
- Bob can cheat with hash of the decryption key. ZK can be used to avoid that, but this will be very computationally-expensive.
- Tradeoff between protecting data storage providers from DDoS attacks and protecting clients from being wrong-treated by data storage providers
  - adjustable parameter in each case
  - reputation system for storage providers may help
  - storage redundancy for critical data for anonymous providers is required

What’s next?

Potentially can be done on top of Lightning Network: zero transactions will reach blockchain

To find out more

- https://github.com/storm-org/storm-spec
- https://lists.linuxfoundation.org/pipermail/bitcoin-dev/2019-August/017269.html
- https://bitcoinmagazine.com/articles/dr-maxim-orlovsky-on-storm-and-bitcoin-l2-l3-file-storage

Ways to contact me

- https://twitter.com/dr_orlovsky
- https://github.com/dr-orlovsky
- @dr_orlovsky on Telegram
- orlovsky@pandoracore.com

https://tippin.me/@dr_orlovsky

bc1qdzyxmdjqq4cl2k2u4kp8vash3t2qhtfvnswhnu