Bitcoin Transactions and Lightning Network

Kwinten De Backer

August 9, 2018

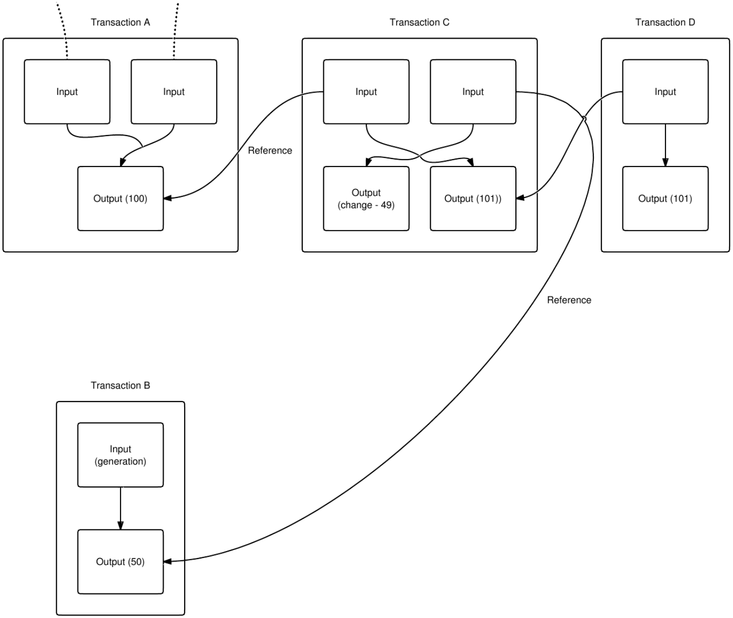
Overview

1. Bitcoin Transactions
2. The double spending problem
3. Problems with scalability
4. The Lightning Network
5. The real mindblowing stuff

General format of a Bitcoin transaction

- Number of inputs
- List of outputs
- Number of outputs
- List of outputs
- LockTime

General format of a Bitcoin transaction



Inputs

Inputs are references to **outputs** of previous transactions.
scriptSig is the first part of the script which makes sure the spending is authorised.

```
Input:
Previous tx: f5d8ee39a430901c91a5917b9f2
dc19d6d1a0e9cea205b009ca73dd04470b9a6
Index: 0
scriptSig: 304502206e21798a42fae0e854281a
bd38bacd1aeed3ee3738d9e1446618c4571d10

90db022100e2ac980643b0b82c0
e88ffdfec6b64e3e6ba35e7ba5fdd7d5d6cc8d25c6b241501
```

Outputs

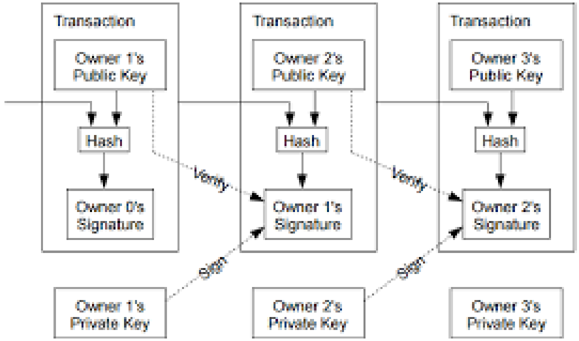
The total value of outputs must be less than the total value of the referenced outputs in the input part.
scriptPubKey is the second part of the script.

```
Output:
Value: 5000000000
scriptPubKey: OP_DUP OP_HASH160
 404371705fa9bd789a2fcd52
d2c580b65d35549d
OP EQUALVERIFY OP CHECKSIG
```

Verification

Bitcoin uses a *Forth*-like scripting language to check if a transaction is authorised. The parts in *scriptSig* and *scriptPubkey* are concatenated, pushed on a stack one by one and if the result is *true*, the transaction is valid.

What does it mean to own Bitcoin?



Pay to public key hash

Stack	Script	Description
sig pubKey	OP DUP OP HASH160 pubKeyHash OP EQUALVERIFY OP CHECKSIG	Constants are added to the stack.
sig pubKey pubKey	OP HASH160 pubKeyHash OP EQUALVERIFY OP CHECKSIG	Top stack item is duplicated.
sig pubKey pubHashA	pubKeyHash OP EQUALVERIFY OP CHECKSIG	Top stack item is hashed.
sig pubKey pubHashA pubKeyHash	OP EQUALVERIFY OP CHECKSIG	Constant added.
sig pubKey	OP CHECKSIG	Equality is checked between the top two stack items.
true	Empty.	Signature is checked for top two stack items.

Pay to script hash

Created to move the responsability for supplying the conditions to redeem a transaction from the sender to the receiver. Makes funding a scripted transaction identical to funding a regular one.

```
scriptPubKey: OP_HASH160 <scriptHash> OP_EQUAL
scriptSig: ..signatures... <serialized script>
```

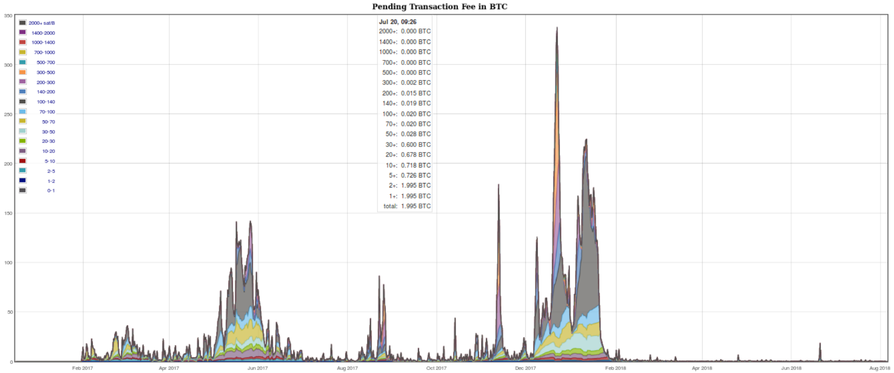
Multisignature transactions

```
m-of-n multi-signature transaction:
scriptSig: 0 <sig1> ... <script>
script: OP_m <pubKey1> ... OP_n OP_CHECKMULTISIG
```

Mining (short)

Transactions need to be included in a block by a miner to be considered valid, to protect against double spendings. The more blocks are added on top of the block with the transaction, the more unlikely it is to be reversed. After 6 confirmations (1 hour), the transaction can safely be regarded as final.

Transaction fees



`jochen-hoenicke.de/queue`

Blockspace and decentralization

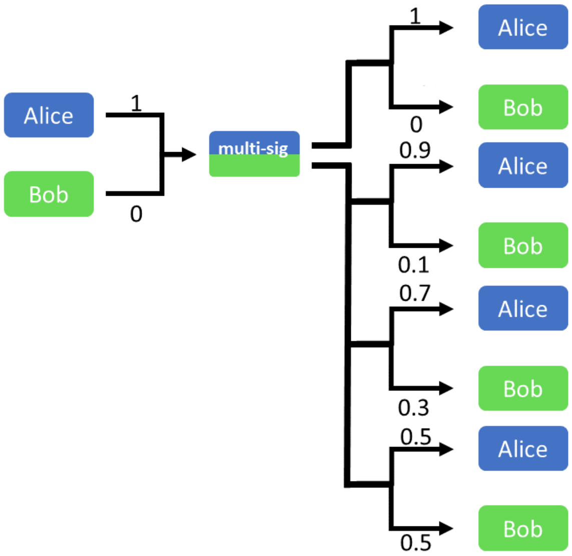
All of the properties Bitcoin has, it has because of it's decentralization. It is absolutely vital that the cost of running a node which validates all rules independently is kept as low as possible.

- Size of the Blockchain is already around 200 GB
- Upload is around 200 GB/month, unmetered connection necessary.
- Need to keep up with a new block every 10 minutes.

Lightning Network

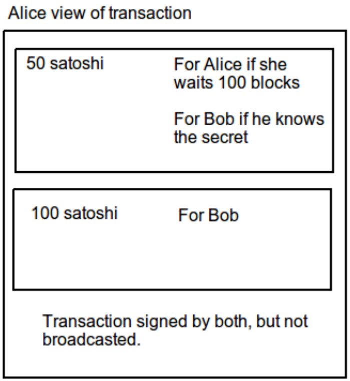
The Lightning Network is a *second-layer* payment protocol built on the Bitcoin blockchain. Using the blockchain for small value payments is completely overkill. Lightning uses the blockchain as a completely neutral and fair 'judge', and moves almost all transactions off the chain, only settling on-chain in the case of disputes.

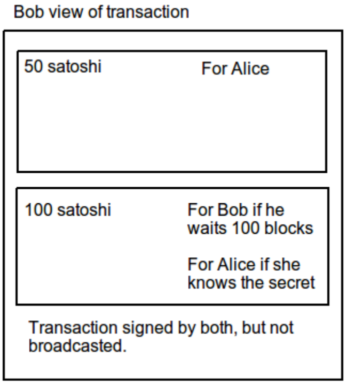
Payment channels (naive)



Payment channels(better)

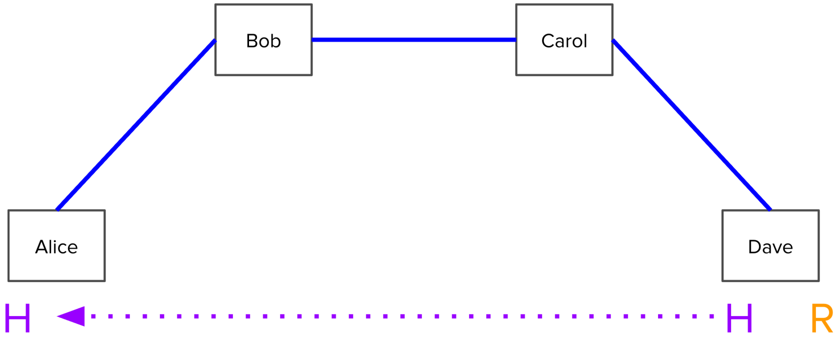
Constructed like this, revealing the secret effectively means revoking your claim on the old states. Channel updates can thus happen trustlessly.


Alice view of transaction
50 satoshi
For Alice if she waits 100 blocks
For Bob if he knows the secret
100 satoshi
For Bob
Transaction signed by both, but not broadcasted.


Bob view of transaction
50 satoshi
For Alice
100 satoshi
For Bob if he waits 100 blocks
For Alice if she knows the secret
Transaction signed by both, but not broadcasted.

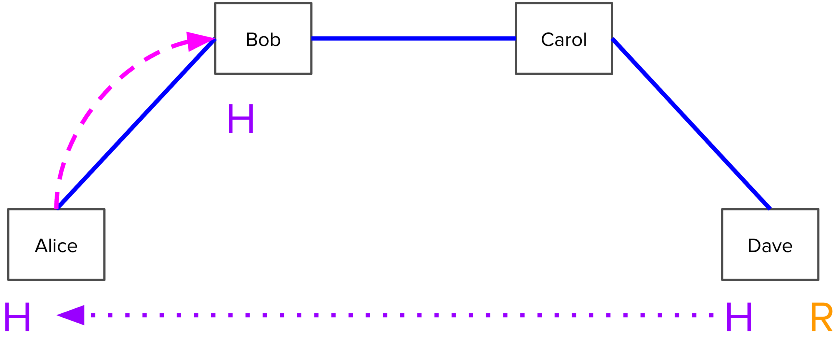
Payment channel network

Alice wants to pay Dave. In order for Dave to accept this payment, he must generate a random number R. He keeps R secret, but hashes it and gives the hash H to Alice.



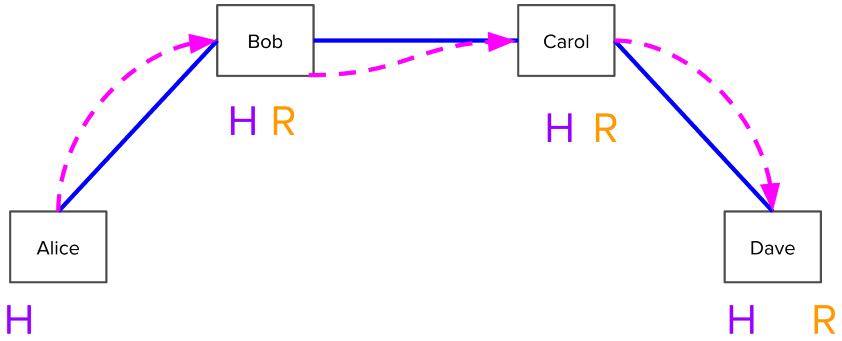
Payment channel network

Alice tells Bob: I will pay you if you can produce the preimage of H within 3 days. In particular, she signs a transaction and shares it with Bob where for the first three days after it is broadcast, only Bob can redeem it with knowledge of R, and afterwards it is redeemable only by Alice.



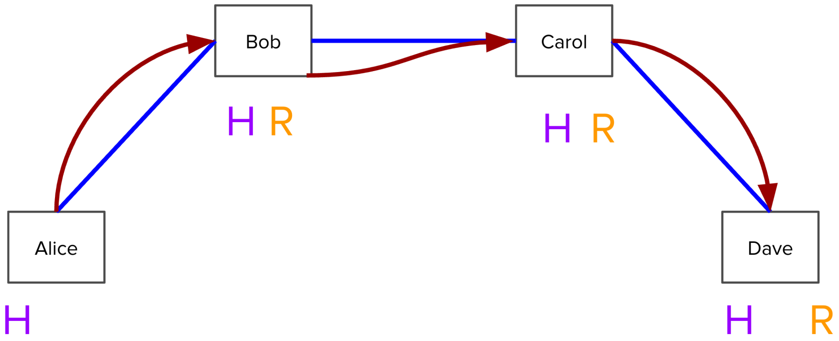
Payment channel network

Bob, knowing that he can pull funds from Alice if he knows R, now has no issue telling Carol: I will pay you if you can produce the preimage of H within 2 days.



Payment channel network

Now, everyone can clear out, because they are guaranteed their money if they broadcast. They would prefer not to do that though, and instead settle each of these hops off chain. Alice knows that Bob can pull funds from her since he has R, so she tells Bob: Ill pay you, regardless of R, and in doing so well terminate the HTLC so we can forget about R.



Payment channel network

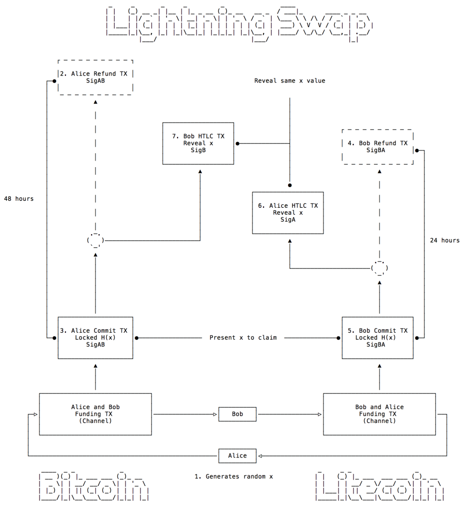
Now, what if Dave is uncooperative and refuses to give R to Bob and Carol? Note that Dave must broadcast the transaction from Carol within 1 day, and in doing so must reveal R in order to redeem the funds. Bob and Carol can simply look at the blockchain to determine what R is and settle off-chain as well.

Lightning problems and solutions

- On-boarding new users still requires on-chain transaction(s)
- Channel depletion
- Nodes need to be online permanently or frequently

- 3rd party funding of channels, channel factories
- Submarine swaps, atomic multipath payments
- Watchtowers

Cross chain atomic swaps



Submarine swaps

It is possible to trustlessly trade on-chain-coins for off-chain coins, using a process known as submarine swaps. This can be used to replenish channels, "cash-out" large channel balances, and generally to manage channel values. This sort of swap can also be done between blockchains, one of which does not even need a second layer.

Channel Factories

Just as payment channels allow for trustless off-chain updates between two participants, Channel Factories would allow for trustless off-chain creation of channels between a (large) group of participants. This would allow a group of people/organizations to come together and would extremely cut the potential costs of opening channels, especially if Schnorr signatures would be used.

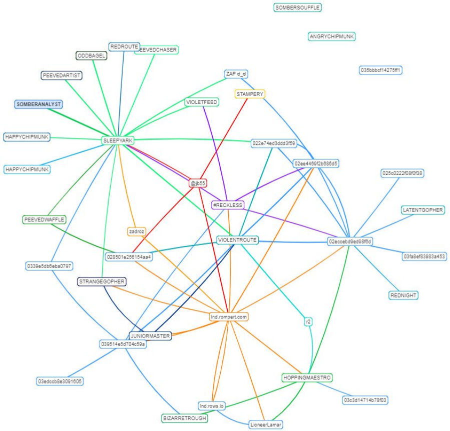
Implementations

- LND, Lightning Labs (Golang): `github.com/lightningnetwork/lnd`
- C-Lightning, Blockstream (C): `github.com/ElementsProject/lightning`
- Eclair, ACINQ (Scala): `github.com/acinq/eclair`

LND by far the most mature implementation by now. Mobile option (Android): Bitcoin-Lightning-Wallet.

Growth of the network

Network topology in January 2018



Growth of the network

Network topology in as of this moment:
`https://rompert.com/recksplorer`

Applications

- Micropayments: API pay-per-call
- `satoshis.place`
- `yalls.org`
- `lightningspin.com`
- Various merchants accept lightning payments today: `lightningnetworkstores.com`

References

- dev.lightning.community
- en.bitcoin.it/wiki/Transaction
- bitcoin.org/bitcoin.pdf