Time, Bitcoin, and the Lightning Network

Joseph Poon

SF Bitcoin Devs - July 6, 2015

Topics

- Bitcoin, Timestamping, and Ordering
- Financial Systems and Time
- Quick refresher on Lightning Network
- Survey of time-related BIPs
- Enforcing Off-Blockchain ordering of transactions on the Lightning Network

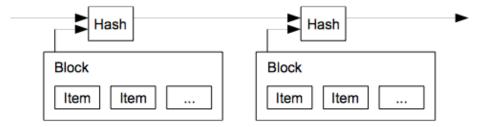
The Blockchain is a Timestamp

- On-chain transactions are ordered because Bitcoin is a timestamping system
- Using the blockchain for unordered data storage is sort-of missing the point, but if you're going to, use a merkle tree
 Ordering events to prevent double-spending is why the entire thing works
 - $\circ~$ It's in the Bitcoin whitepaper

Bitcoin: A Peer-to-Peer Electronic Cash System

3. Timestamp Server

The solution we propose begins with a timestamp server. A timestamp server works by taking a hash of a block of items to be timestamped and widely publishing the hash, such as in a newspaper or Usenet post [2-5]. The timestamp proves that the data must have existed at the time, obviously, in order to get into the hash. Each timestamp includes the previous timestamp in its hash, forming a chain, with each additional timestamp reinforcing the ones before it.



Nakamoto, Satoshi (2008). Bitcoin: A Peer-to-Peer Electronic Cash System. http://www.bitcoin.org/bitcoin.pdf

Quick Review: (Decentralized) Ordering of Events to Prevent Double-Spending

- Alice sends the same 1 BTC to both Bob and Carol
 - Only one of the spends enter into the blockchain
 - Chain is ordered, so one can be sure a spend happened before another
 - Can't assert something happened in the past
 - Double-spends invalidate blocks, so miners cannot maliciously mine a fraudulent block in isolation (without reorgs)

Alice sends to Bob

Green: Entered into the blockchain White: Unconfirmed transaction

0.1 BTC (Alice owns)

0.1 BTC Output: Bob

Alice sends to Bob: Confirmed!

0.1 BTC (Alice owns)

0.1 BTC Output: Bob

Alice attempts double-spend

0.1 BTC

(Alice owns)

Alice is trying to be a jerk! She's trying to double-spend the same output to Carol, but she already sent it to Bob at Block 350,001.

> 0.1 BTC Output: Bob

0.1 BTC Output: Carol

Alice fails double-spend

0.1 BTC

(Alice owns)

The network already accepted Alice's spend in block height 350,001. Any future spends is rejected by miners and the Bitcoin P2P network

> 0.1 BTC Output: Bob

0.1 BTC Output: Caroi

Problem! What if you wanted to conduct transactions off-blockchain securely?

- Blockchain transactions are slow and not that scalable
 - Multi-gigabyte to multi-terabyte blocks if bitcoin really started picking up mass use
 - 10-minute confirmation times too slow
 - Why should the entire world know you bought coffee and process the transaction?
- We need order off-chain and net-settle everything without trust or counterparty risk

Time and Financial Systems

- If one wishes to conduct transactions offchain (for scalability, speed, etc.), why not see how existing systems operate?
- Existing and legacy financial systems are also all about time & ordering of events
 - They've had to solve distributed systems problems with money for a really long time!

Distributed Time Locks

- Bank-to-bank transfers
- Time Locking as a means for atomicity (clearing) is common in financial systems
 - Different obligations to occur within a certain timeframe (overnight, 3 business days, etc.)
 - VISA's 180-day "confirmation" (dispute period)
 - \circ T+3 in equities
 - Flows through to security models in systemic risk analysis, e.g. Regulation T

Clearinghouses for Scalability

- Assuming distributed time-locks, establishing lots of peer-to-peer transactions are a pain, why not net settle across a multi-hop network?
 - Correspondent Banking
 - Clearinghouses



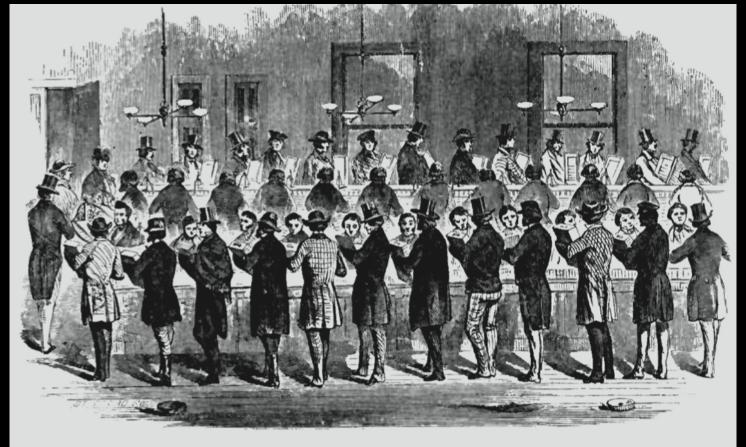
Securing Today. Shaping Tomorrow."







Campbell-Kelly, Martin (2010) Victorian data processing. Communications of the ACM, Vol.53 (No.10). pp. 19-21.



Making the Exchange in Six Minutes, at the Clearing House.

Campbell-Kelly, Martin (2010) Victorian data processing. Communications of the ACM, Vol.53 (No.10). pp. 19-21.

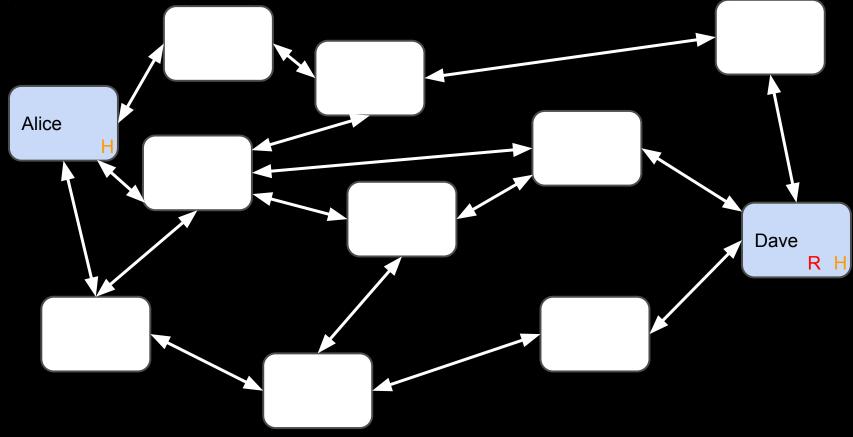
Bitcoin can do better!

- All the prior example systems require trust
 - Failure of trust breaks everything, nobody can trade
 - Multisig & scripting allows for decentralized contracts
- Removing the need for trust and reputation eliminates entry costs
 - Near-zero costs to enter networks reduces fees to near-zero
- The blockchain as court
 - Programmatic Adjudication

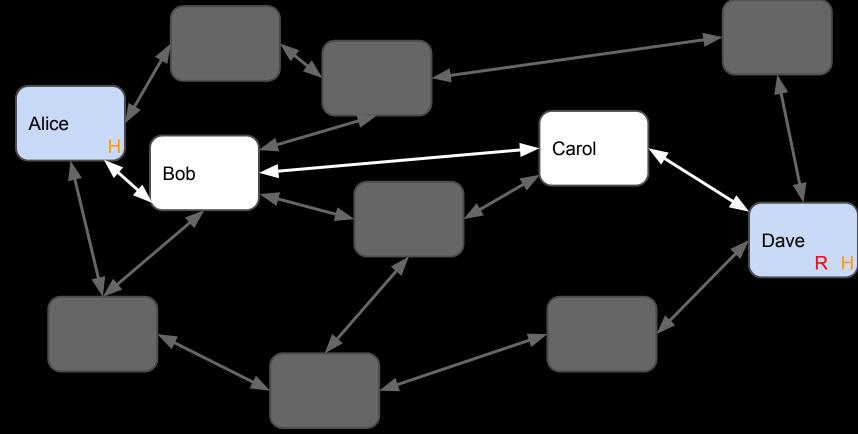
Lightning Network Overview

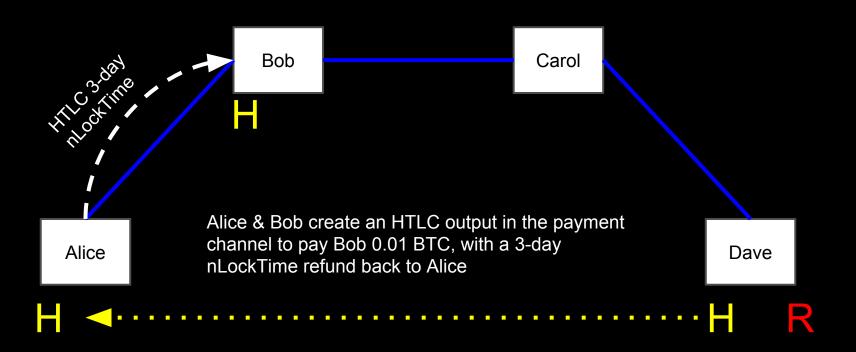
- A network of Bitcoin payment channels
 - Real bitcoin transactions which can be net-settled on-chain at any time
- Hashed Time Lock Contract: Payment conditional upon knowledge of secret preimage R which produces known cryptographic hash H within n blocks

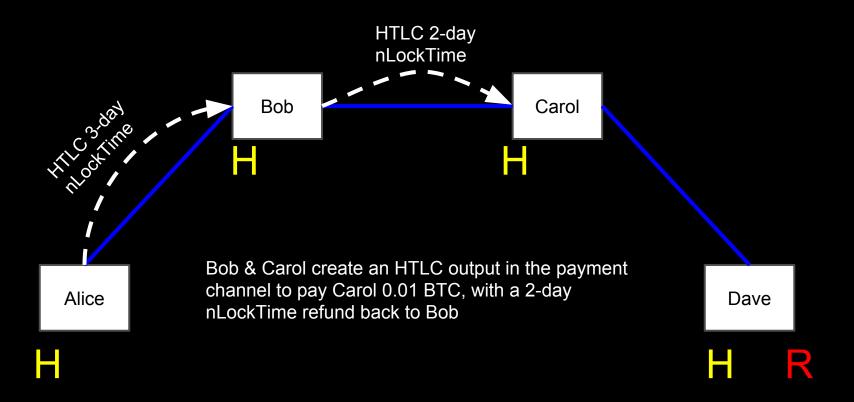
Alice wants to pay Dave 0.01 BTC. Dave tells Alice, "Here' s H, if you know R, consider your payment fulfilled"

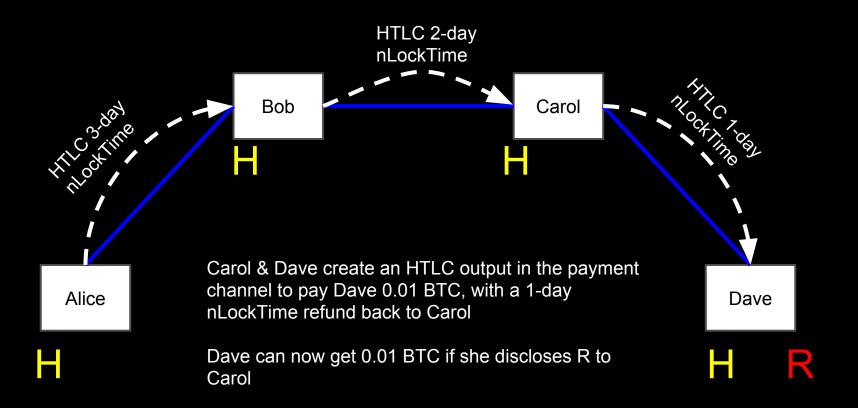


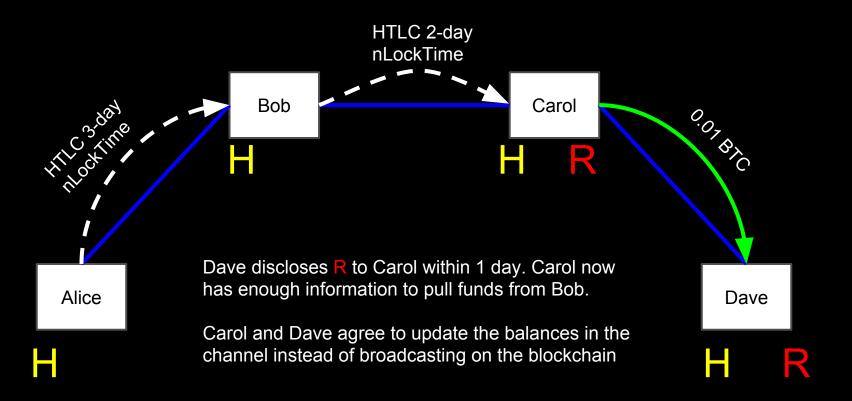
Alice doesn't have a direct channel open with Dave, so she finds a route.

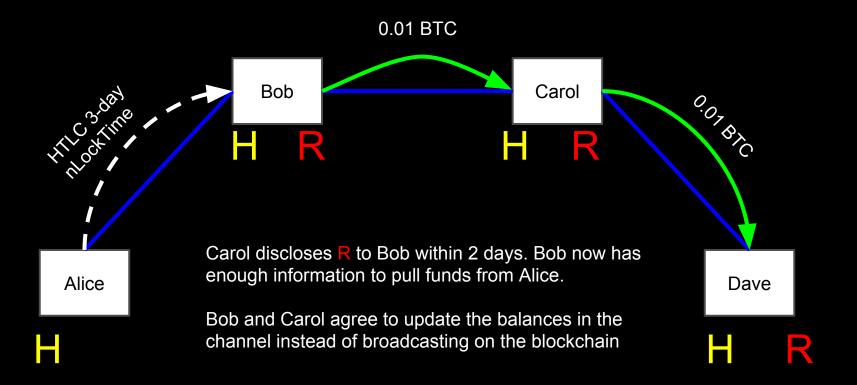


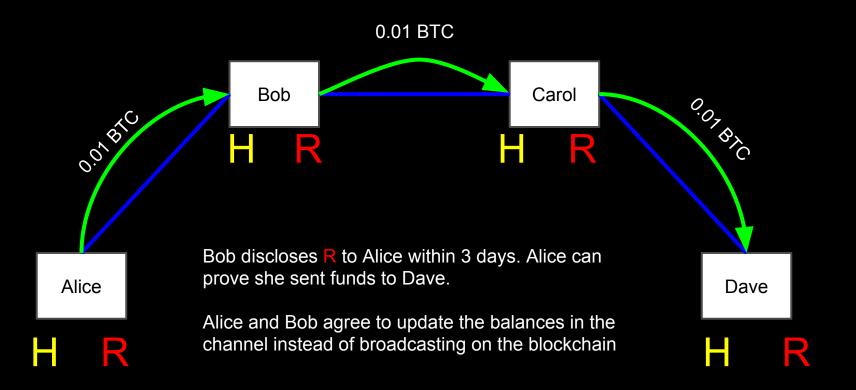












But how exactly can you order the transactions off-chain -- without hitting the blockchain every time a payment is made; how do you enforce how much the counterparties own in each channel?

Time-related BIPs

- BIP65: OP_CHECKLOCKTIMEVERIFY
 - Peter Todd
- BIP68: Consensus-enforced transaction replacement signalled via sequence numbers
 - Mark Friedenbach

Example Bitcoin Transaction

```
"hash":"f4184fc596403b9d638783cf57adfe4c75c605f6356fbc91338530e9831e9e16",
 "ver":1.
 "vin sz":1,
 "vout sz":2,
 lock time":0.
 "size":275.
 "in":[
   "prev out":{
    "hash":"0437cd7f8525ceed2324359c2d0ba26006d92d856a9c20fa0241106ee5a597c9",
    "n":0
    "sequence":UINT MAX
   "scriptSig":"
304402204e45e16932b8af514961a1d3a1a25fdf3f4f7732e9d624c6c61548ab5fb8cd410220181522ec8eca07de4860a4
acdd12909d831cc56cbbac4622082221a8768d1d0901"
 "out":[.....
```

BIP65: CHECKLOCKTIMEVERIFY

- New Bitcoin opcode lets you script time
 - The most expensive timestamping system in the world can't even tell time in its own scripts?
- Script opcode evaluates as true (lets you spend) after a particular block height
- Useful for committing to the world some time dependency OR for *limited* malleability fixes

BIP65 Example Payment Channel

ΙF

<Bob pubkey> CHECKSIGVERIFY
ELSE
 <expiry time> CHECKLOCKTIMEVERIFY DROP
ENDIF
 <Alice pubkey> CHECKSIG

BIP65 Example Payment Channel

• Good:

• Money will eventually return /w mutated transaction

• Limitations:

- Single-funder, unidirectional, limited time duration
- Lightning Network not compatible (requires a different soft-fork)
- Very useful for proving commitment to a particular date, non-revocability when compared to nLockTime

BIP68: Sequence Numbers

- Original intention for Sequence Numbers looks to be related to off-chain transaction replacement
- Redefines Sequence Numbers to actually work
 - Current behavior is non-functional
 - Prior behavior doesn't work; impossible to guarantee miners will pick the highest and block propagation, requires actively watching the bitcoin P2P network

BIP68: Sequence Numbers

- New Behavior:
 - If the sequence number field is filled in, require the output being spent to have a relative minimum number of confirms
 - E.g. Child has sequence number of 500. There must be 500 blocks between the parent transaction being spent and the child transaction
 - This number is relative to block inclusion of the parent

Alice and Bob have Transaction A&B

Green: Entered into the blockchain White: Unconfirmed transaction

Presume Alice and Bob have both Transaction A and B1.

Alice and Bob don't know yet when Transaction A will enter into the blockchain. Transaction A

0.1 BTC Multisig Alice & Bob

Transaction B1

0.1 BTC Output: Bob Sequence No. 200

Block 349,999

Alice Broadcasts a Transaction

Green: Entered into the blockchain White: Unconfirmed transaction

Transaction A enters into the blockchain at block height 350,000

Transaction B1 is not yet valid. Transaction B1 will only be valid at 350,200 Transaction A

0.1 BTC Multisig Alice & Bob

Transaction B1

0.1 BTC Output: Bob Sequence No. 200

Alice Broadcasts a Transaction

Green: Entered into the blockchain White: Unconfirmed transaction

Transaction A enters into the blockchain at block height 350,000

Transaction B1 is not yet valid. Transaction B1 will only be valid after 200 blocks have elapsed after Transaction A has entered into the blockchain. Transaction A

0.1 BTC Multisig Alice & Bob

Transaction B1

0.1 BTC Output: Bob Sequence No. 200

Alice Broadcasts a Transaction

Green: Entered into the blockchain White: Unconfirmed transaction

Transaction B1 can enter into the blockchain now that 200 blocks have elapsed. The money is now Bob's.

Sequence numbers are useful compared to nLockTime when you don't know when Transaction A will enter into the blockchain!

Transaction A

Transaction B1

0.1 BTC

Output: Bob

Sequence No. 200

0.1 BTC Multisig Alice & Bob

Why This is Useful

- Relative to parent's block height is very important, you can establish conditional rules which are valid after a transaction enters into the blockchain
 - nLockTime does not permit this since it relates to hard dates
- You can create outputs that are revocable by spending with different sequence numbers

Alice&Bob have Transaction A, B1 & B2

Green: Entered into the blockchain White: Unconfirmed transaction

Let's go back in time. Presume Tx A has never entered into the blockchain. However, there is a new transaction, B2! Let's see what happens... Transaction A

0.1 BTC Multisig Alice & Bob

Transaction B1

0.1 BTC Output: Bob Sequence No. 200 Transaction B2

0.1 BTC Output: Alice Sequence No. 0

Block 349,999

Alice Broadcasts a Transaction

0.1 BTC

Transaction A

Multisig Alice & Bob

Green: Entered into the blockchain White: Unconfirmed transaction

Bob when signing B2, has functionally revoked transaction B1, as if it doesn't exist! (Provided that Alice watches the blockchain).

If either party can broadcast Tx A...

Transaction B1

0.1 BTC Output: Bob Sequence No. 200 \land

Transaction B2

0.1 BTC Output: Alice Sequence No. 0

Alice Broadcasts a Transaction

0.1 BTC

Transaction A

Green: Entered into the blockchain White: Unconfirmed transaction

If either party can broadcast Tx A...

Alice can immediately take the money by broadcasting B2, since there is no sequence number limitation!

Bob has revoked transaction B1!

Transaction B1

0.1 BTC Output: Bob Sequence No. 200

Multisig Alice & Bob Transaction B2

> 0.1 BTC Output: Alice Sequence No. 0

Revoking Transactions

- By creating unbroadcasted spends with lower (or no) sequence number requirement, it's possible for two parties to conduct transactions off-chain
- Spends from transactions create a "dispute resolution period" whereby either party may dispute it with transactions which do not require a sequence number

Application for Lightning Network

- This isn't a requirement for the Lightning Network (it can use nLockTime), but the benefit is channels can remain open indefinitely!
- With malicious counterparties, they can lock up funds for only a very limited number of time

How Lightning Network Channels Work

- Each "Commitment" has two versions with the same outputs, one for Alice and one for Bob
 - Only Alice can broadcast Alice's version, only Bob can broadcast Bob's version
 - The final payout is locked via Sequence Number
 - To revoke the commitment, make the final payout to pay 100% of the funds to the counterparty as a penalty

Alice and Bob Have a Channel Open

Channel 1.0 BTC

output

Multisig Alice and Bob

Green: Entered into the blockchain White: Unconfirmed transaction

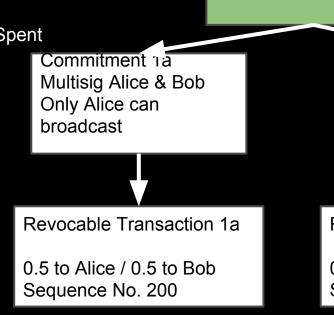
Channel open with 0.1 BTC. Spent using Commitment 1, current balance: 0.5 Alice / 0.5 Bob

Alice can broadcast Commitment 1a

Bob can broadcast Commitment 1b

Either can broadcast all Revocable Transactions

**Simplified model



Commitment 1b Multisig Alice & Bob Only Bob can broadcast

Revocable Transaction 1b

0.5 to Alice / 0.5 to Bob Sequence No. 200

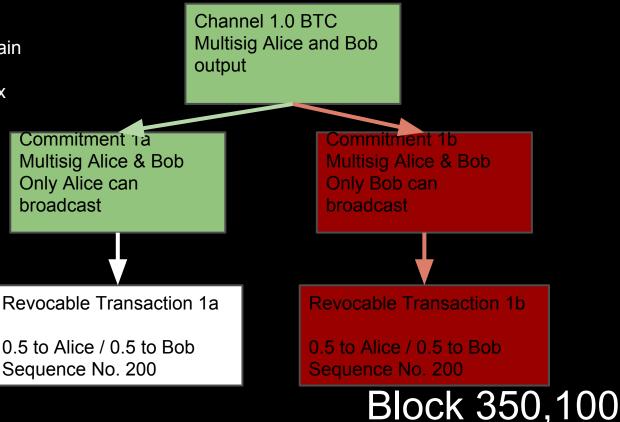
Alice Broadcasts Commitment 1a

Green: Entered into the blockchain White: Unconfirmed transaction Red: Already spent by another tx

Let's say Alice wants to close out the channel. No trust is needed, they can't steal money!

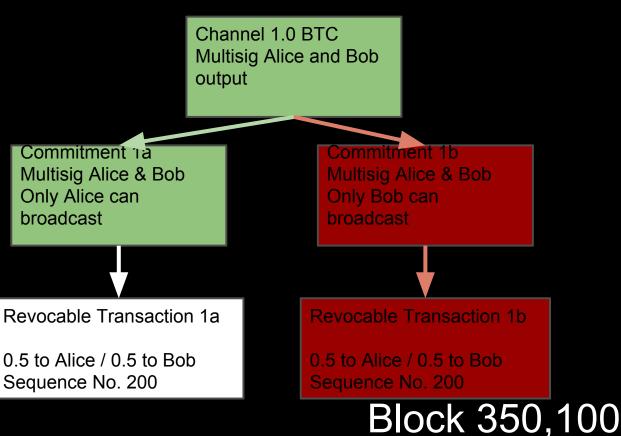
Alice broadcasts Commitment 1a. She cannot broadcast Commitment 1b, only Bob can do that.

The righthand path is now invalidated, since the output has been redeemed.

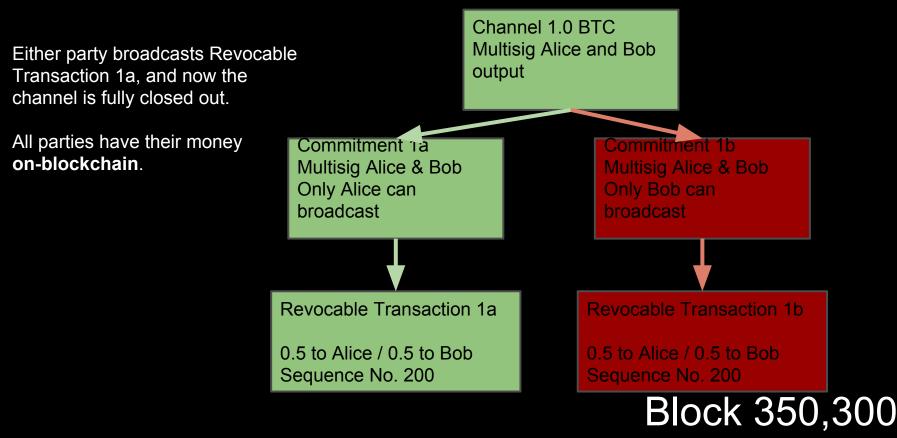


Alice has to wait!

Alice or Bob can broadcast Revocable Transaction 1a, but only after 200 confirmations.



After 200 confirmations, finished!



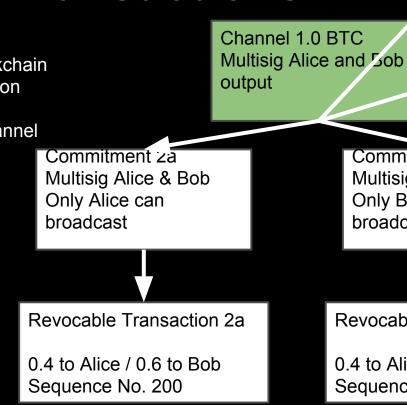
Revoking Transactions

Green: Entered into the blockchain White: Unconfirmed transaction

Instead of closing out the channel they want to update their balances to be: 0.4 Alice / 0.6 Bob in Commitment 2

Cool.... BUT WAIT! What about Commitment 1! Alice can still broadcast Commitment 1a and she has more money.

**Simplified model



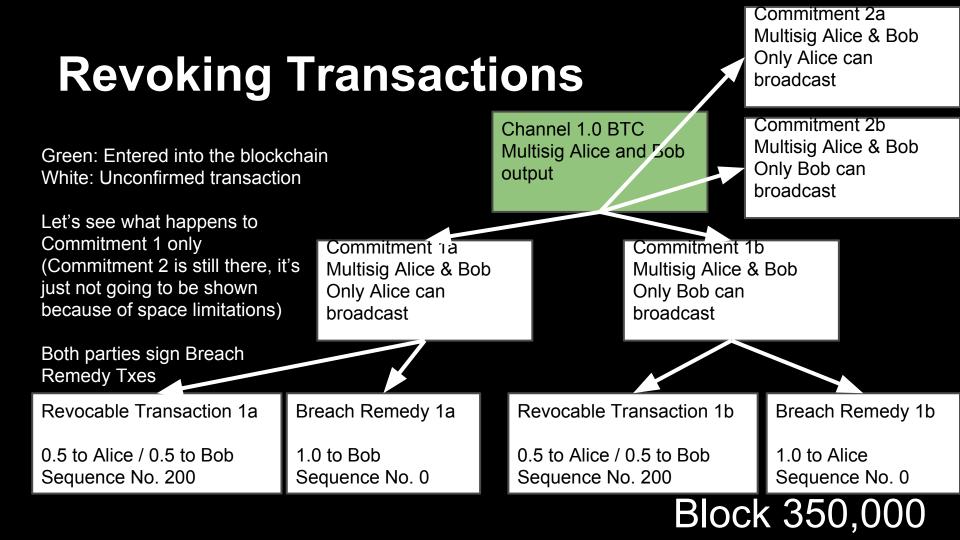
Commitment 1a Multisig Alice & Bob Only Alice can broadcast

Commitment 1b Multisig Alice & Bob Only Bob can broadcast

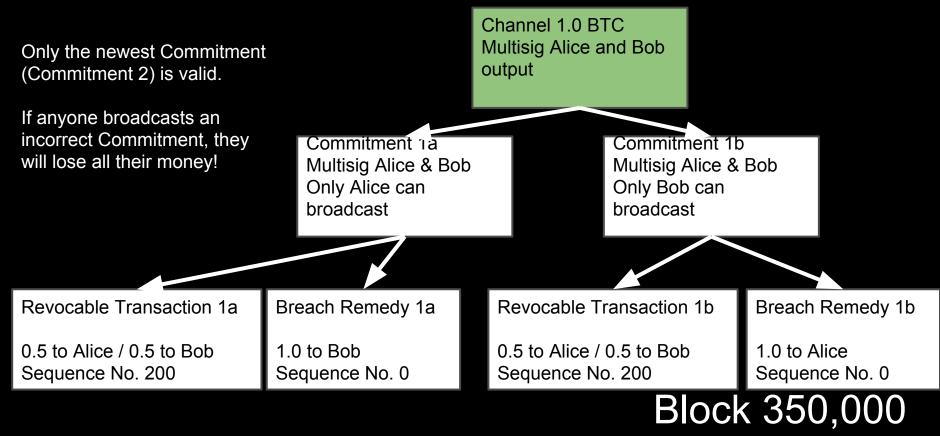
Commitment 2b Multisig Alice & Bob Only Bob can broadcast

Revocable Transaction 2b

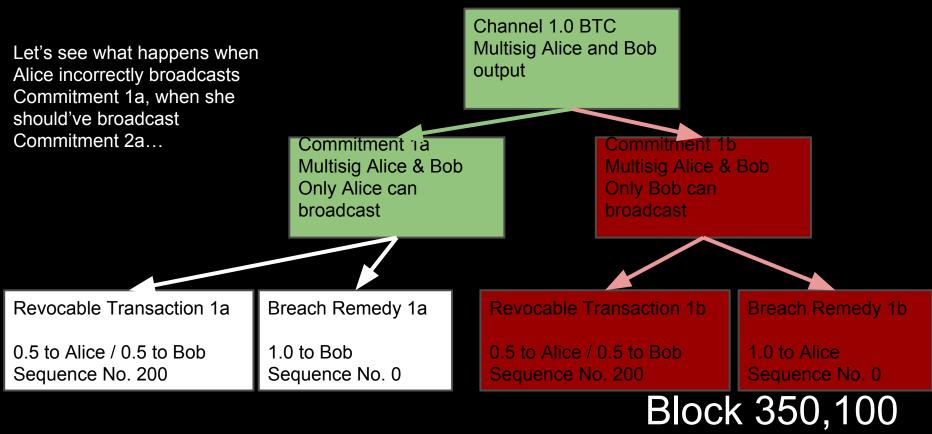
0.4 to Alice / 0.6 to Bob Sequence No. 200



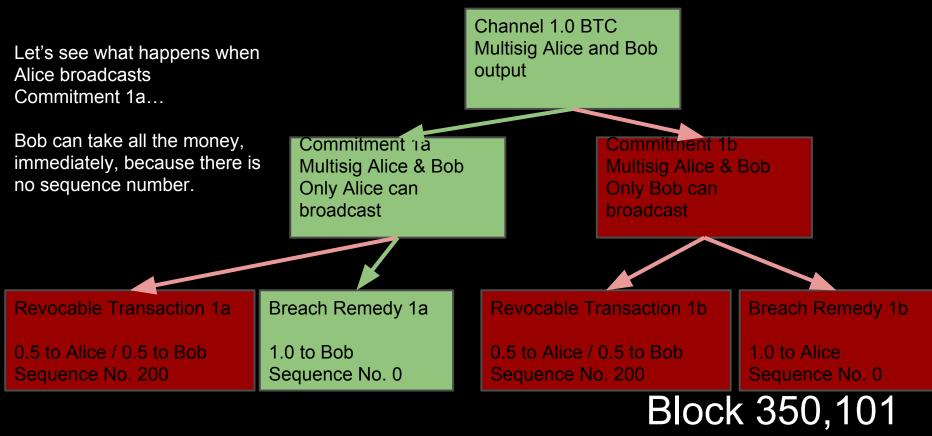
Revoking Transactions



Enforcing Order Via Penalties



Enforcing Order Via Penalties



Result...

- Both parties should only broadcast the current Commitment Transaction
 - If they don't they will lose ALL their money, because they have given the other party power to take all their money immediately
 - The other party has to watch the blockchain periodically, higher sequence numbers mean less frequent checks (e.g. monthly). Can be delegated to a 3rd party without custodial risk.

What this means

- It's possible to order transactions off-chain
 - These are **real bitcoin transactions**, this isn't some altcoin or disconnected trusted system
- Consensus is achieved through the threat of on-chain enforcement
 - Everything stays off-chain if everyone is cooperative
 - Take all the money as a penalty with hostile/invalid state commitments to the blockchain.

Lightning Network

- It's possible to create channels that stay open an effectively unlimited amount of time
 Just keep revoking old transactions!
- With these channels as building blocks, payments can route over an untrusted network!
 - You can embed HTLCs in these channels
 - Payments can be made completely off-chain, only settling on-chain when non-cooperative