Segwit, Lightning, Sidechains

Segwit

- Encountered in 2 presentations

Segwit

- Core idea:
  - Transaction data structure:
    - <Inputs, Outputs, Scripts>
- Scripts?
  - <Signature, Pubkeys>
- TXID = H(<Inputs, Outputs, Scripts>)

Segwit

- Bitcoin has a 1MB block limit
- Script is large
  - takes up good chunk of this limit
- Increase # of transactions by making this leaner

Segwit

- Script:
  - Alice -> Bob
    - Alice’s public key
    - Signature with Alice’s private key
    - Bob’s public key

Segwit

- Script:
  - Alice -> Bob
    - Alice’s public key
    - Signature with Alice’s private key
    - Bob’s public key

Segwit

- Move Alice’s public key and signature (witness) out of the transaction
- Witness data in a new field
- New merkle tree root from witnesses
  - Placed into input of coinbase

Segwit

- If you are running a pre-segwit node:
  - You see similar data (i.e you ignore witness)
  - And get blocks with a lot more transactions
- Post segwit:
  - Lot more transactions and a new witness field
  - Remember, no protocol level change

Issues

- Pre-segwit nodes see a lot of txns
  - That seemingly are spent by anyone
- Blocks from pre-segwit nodes
  - Won’t be accepted by post-segwit nodes
- Blocks from post-segwit nodes
  - Won’t be accepted by pre-segwit nodes

Pros

- Transaction:
  - Alice -> Bob
  - <Inputs, Outputs, Bob’s public key>
  - Bob can compute TXID

Segwit

- How does Segwit help?

Lightning Network

Lightning Network

- Transactions as fast as your communication allows
  - Not encumbered by protocol
- Payment Channels
- Records just 2 transactions - start and finish of economic activity

Multisig

- Multiple parties sign off
- 2/3, 3/3, 2/2 etc.

Payment Channels

- Exchange Bitcoin Transactions
- On a communication channel
- Only broadcast the final

Payment Channels

- Transactions between a pair of individuals
- Primitives:
  - Open
  - Send/Recv
  - Close

Payment Channels

- Open:
  - Between Alice, Bob
  - Record on the blockchain
  - Use a transaction
  - Output conditions?

Open

- Alice, Bob
- Transaction
  - Inputs: Alice puts up 1 BTC
  - Output: 1 BTC, multisig(Alice, Bob) || (timelock(X), Alice)

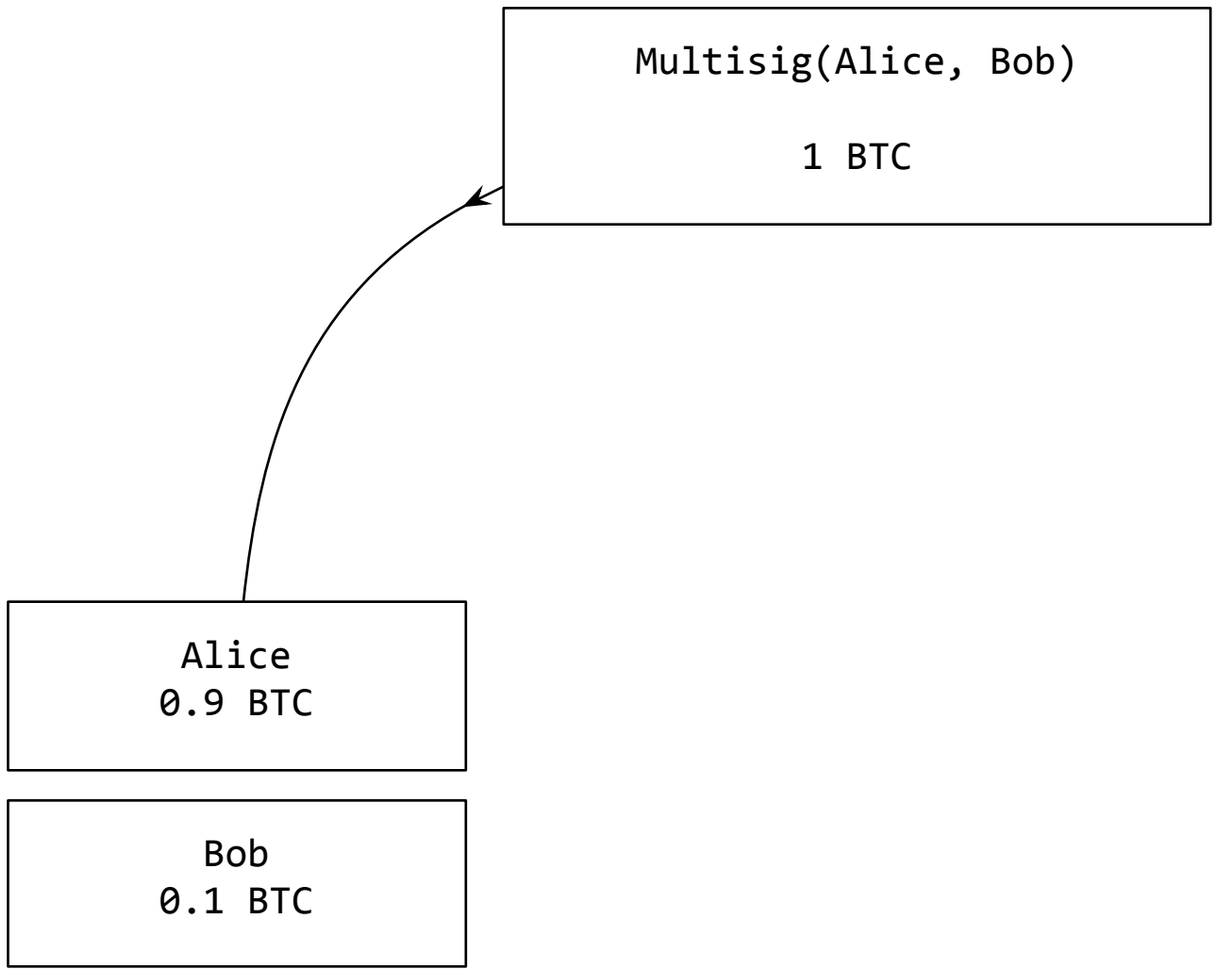
Output Cond.

- multisig(Alice, Bob) || (timelock(X), Alice)
  - Both Alice & Bob have to agree to spend this
  - OR
    - When X blocks pass
    - Alice can spend it

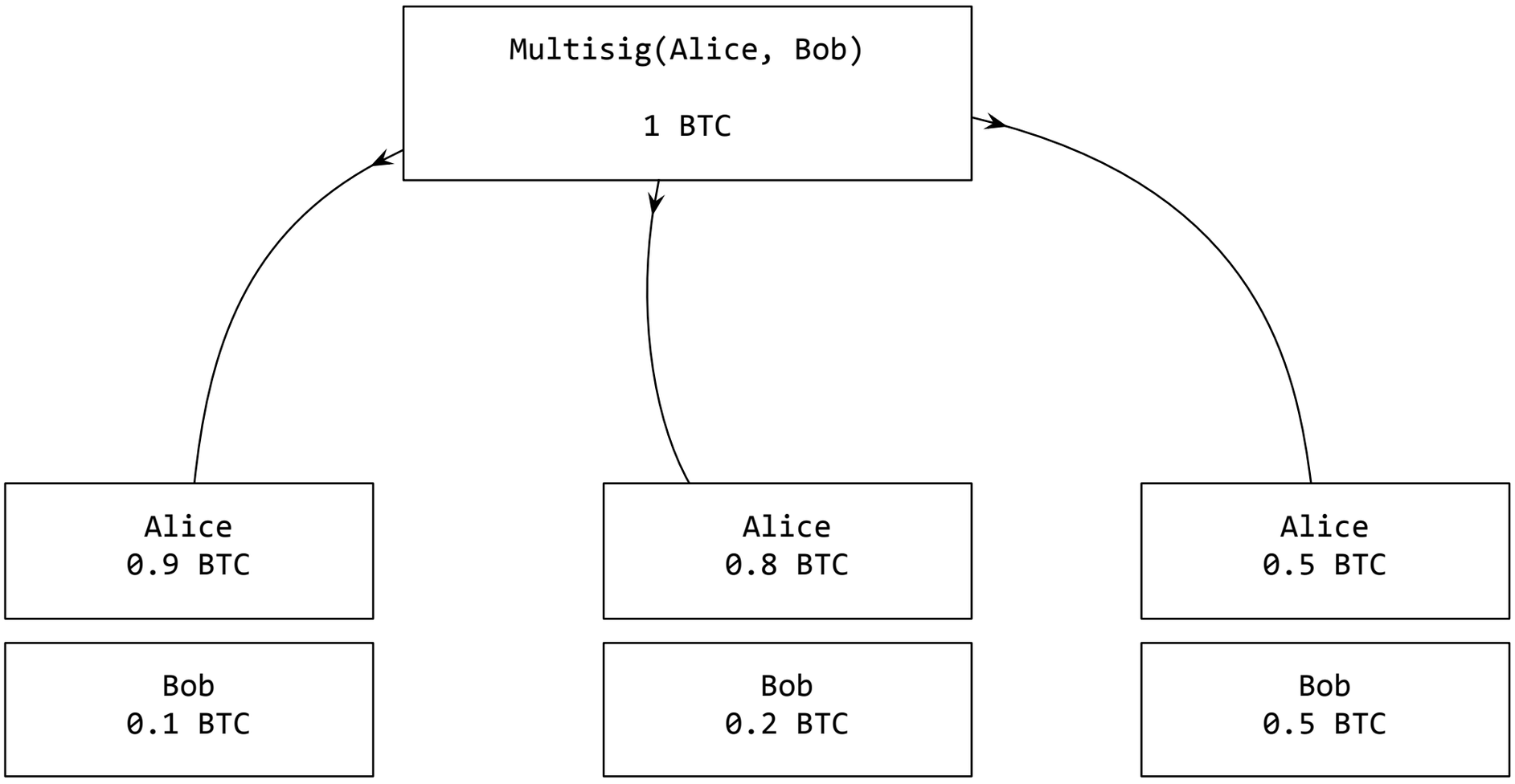
Broadcast This

- This opens a channel

Send Transactions



Send Transactions



Closing

- Bob picks on of these
  - (presumably the best one)
- And broadcasts it
- Alice can’t
  - Why?

Timeout Clause

- Why?

This Channel

- One way
- Bob can’t (shouldn’t) pay Alice
  - Why?
- Only Alice Pays Bob

Bidirectional How?

- Issues:
  - Can’t revoke old transactions
  - Bitcoin only has timelock, no expiry
  - Only way to invalidate is to spend with another txn
    - What is the point of the channel then

Trick

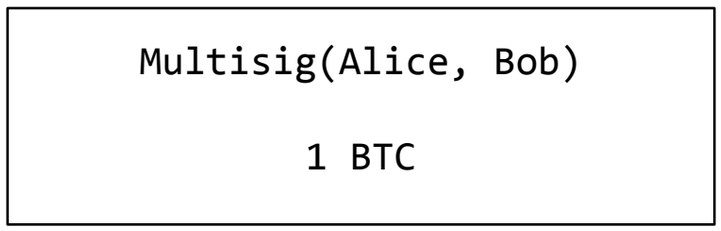
- Change the primitives
- Previously:
  - Multisig(Alice, Bob) | (Alice & timelock(X))

Trick

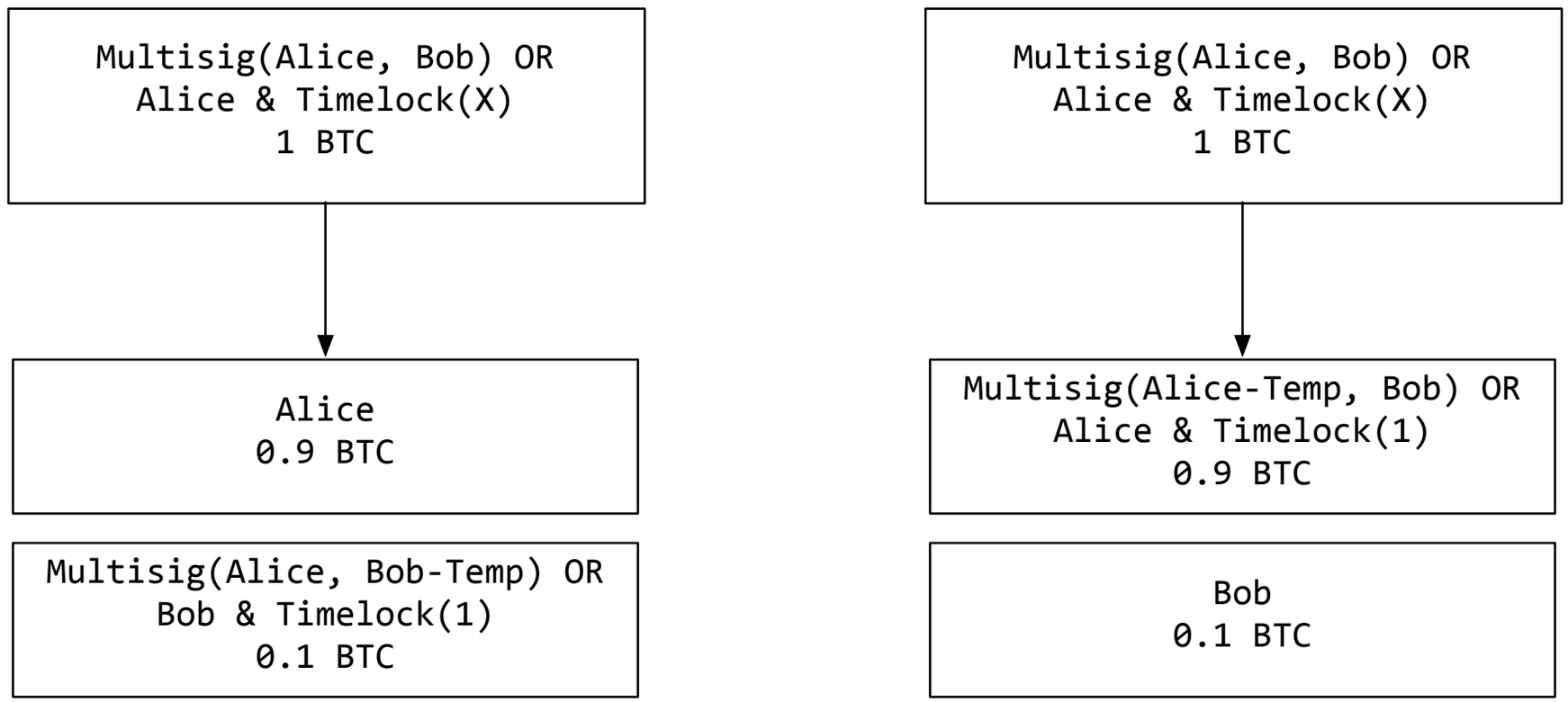
- Now:
- Temporary key for transaction.
- Timelock of 1 day
- Alice, Bob, Bob-Temp

Open

- Same



Alice & Bob States



Multisig(Alice, Bob) OR
Alice & Timelock(X)
1 BTC

Alice
0.8 BTC

Multisig(Alice, Bob-Temp2) OR
Bob & Timelock(1)
0.2 BTC

Multisig(Alice, Bob) OR
Alice & Timelock(X)
1 BTC

Multisig(Alice-Temp2, Bob) OR
Alice & Timelock(1)
0.8 BTC

Bob
0.2 BTC

Alice
0.9 BTC

Multisig(Alice, Bob-Temp) OR
Bob & Timelock(1)
0.1 BTC

Multisig(Alice-Temp, Bob) OR
Alice & Timelock(1)
0.9 BTC

Bob
0.1 BTC

Revoke

Revoke Old

- Alice sends Bob Alice-Temp
- Bob sends Alice Bob-Temp
- Why?
- In what sequence?

You Have

- A Sequence of Payments!

Need To

- Keep track of all temp keys
- Data structure called GGM

Network

- Multiple bidirectional channels
  - Alice <-> Bob <-> Manuel <-> Silvio

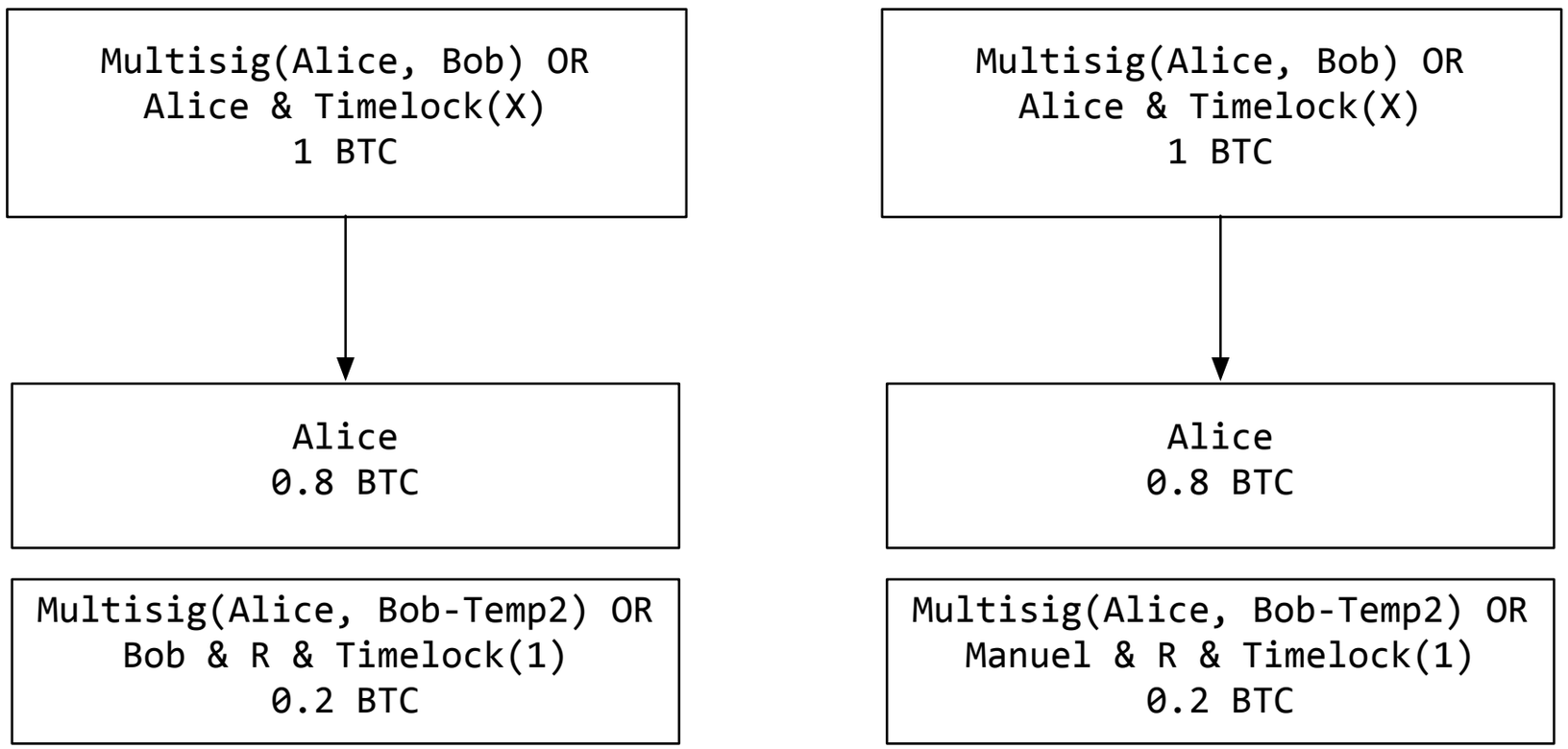
Alice -> Manuel

- Alice Pays Bob
- Bob Pays Manuel
  - Thus Alice -> Manuel
- Make this work?

Hash/Time Locked Contract

- Nonce!

Hash/Time Locked Contract



Alex and Manuel communicate
What do they communicate?
Manuel knows R
Sends Alice H(R)
Test is knowledge of R

Sidechains

Sidechain

- Take your bitcoin to a different blockchain
- Use it there
- Move it back
- Trustless 2-way peg

Why?

- Innovations not possible on BTC
- Using the ledges for a variety of things?

Examples

- Liquid
  - Connects exchanges and payment providers
  - Avoid going to the chain for transfers
  - Large payments, large providers

Peg

- A transaction in 1st chain locks coins
- Reference in 2nd chain
- (Some kind of swap)
- 2-way means both directions
- Some exchange rate

Verification

- Centralized authority
- N of m authorities (federated peg)
- Simplified Payment Verification

SPV

- Show miners have mined blocks on top of the block containing the locked transaction
- Build a merkle tree and this is a “proof”
- Can counter this proof showing a longer chain without this transaction

Questions?