

Connecting Bitcoin Payment Layers

What?



Lightning



Liquid



Rootstock

Mainchain

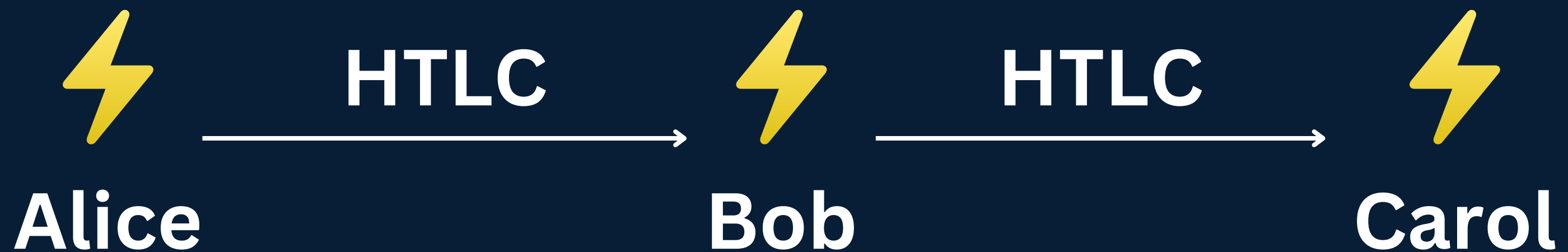
How?

HTLCs

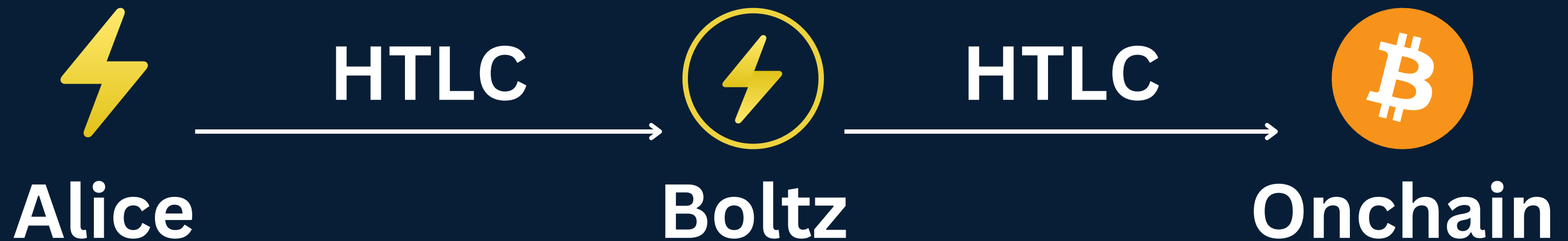
What do we need?

- Hash lock
- Time lock

Payments on Lightning



Swaps on Lightning



SegWit V0

```
OP_SIZE
OP_PUSHBYTES_1 20
OP_EQUAL
OP_IF
    OP_HASH160
    OP_PUSHBYTES_20 <preimage hash>
    OP_EQUALVERIFY
    OP_PUSHBYTES_33 <user public key>
OP_ELSE
    OP_DROP
    OP_PUSHBYTES_3 <timeout block height>
    OP_CLTV
    OP_DROP
    OP_PUSHBYTES_33 <Boltz public key>
OP_ENDIF
OP_CHECKSIG
```

SegWit V0

Hash lock



```
OP_SIZE
OP_PUSHBYTES_1 20
OP_EQUAL
OP_IF
  OP_HASH160
  OP_PUSHBYTES_20 <preimage hash>
  OP_EQUALVERIFY
  OP_PUSHBYTES_33 <user public key>
OP_ELSE
  OP_DROP
  OP_PUSHBYTES_3 <timeout block height>
  OP_CLTV
  OP_DROP
  OP_PUSHBYTES_33 <Boltz public key>
OP_ENDIF
OP_CHECKSIG
```


SegWit V0

Hash lock



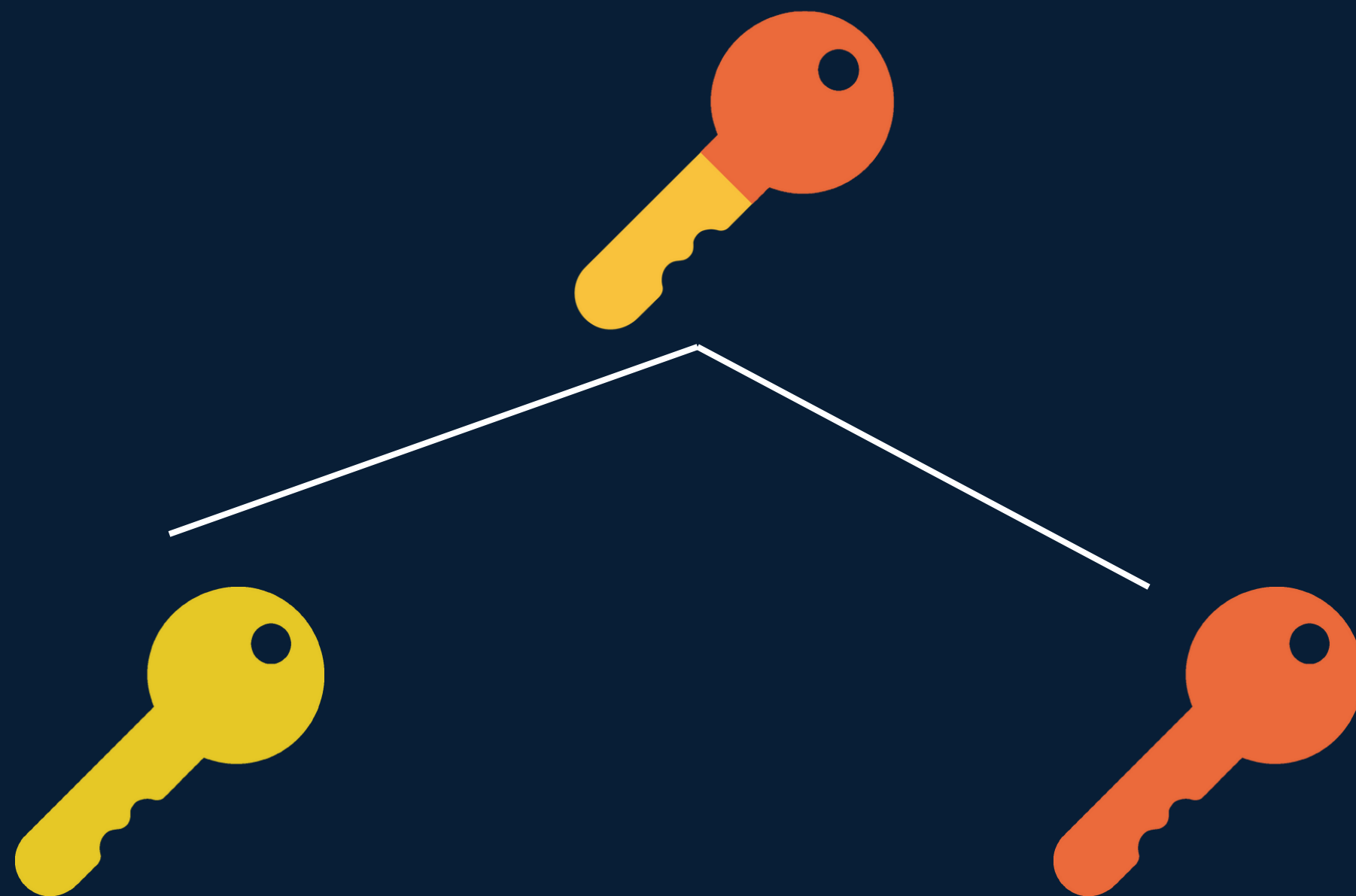
```
OP_SIZE
OP_PUSHBYTES_1 20
OP_EQUAL
OP_IF
  OP_HASH160
  OP_PUSHBYTES_20 <preimage hash>
  OP_EQUALVERIFY
  OP_PUSHBYTES_33 <user public key>
OP_ELSE
  OP_DROP
  OP_PUSHBYTES_3 <timeout block height>
  OP_CLTV
  OP_DROP
  OP_PUSHBYTES_33 <Boltz public key>
OP_ENDIF
OP_CHECKSIG
```

Time lock



Taproot

Musig2



Taptree



```
graph TD; Taptree --> HashLock[Hash lock]; Taptree --> TimeLock[Time lock];
```

Hash lock

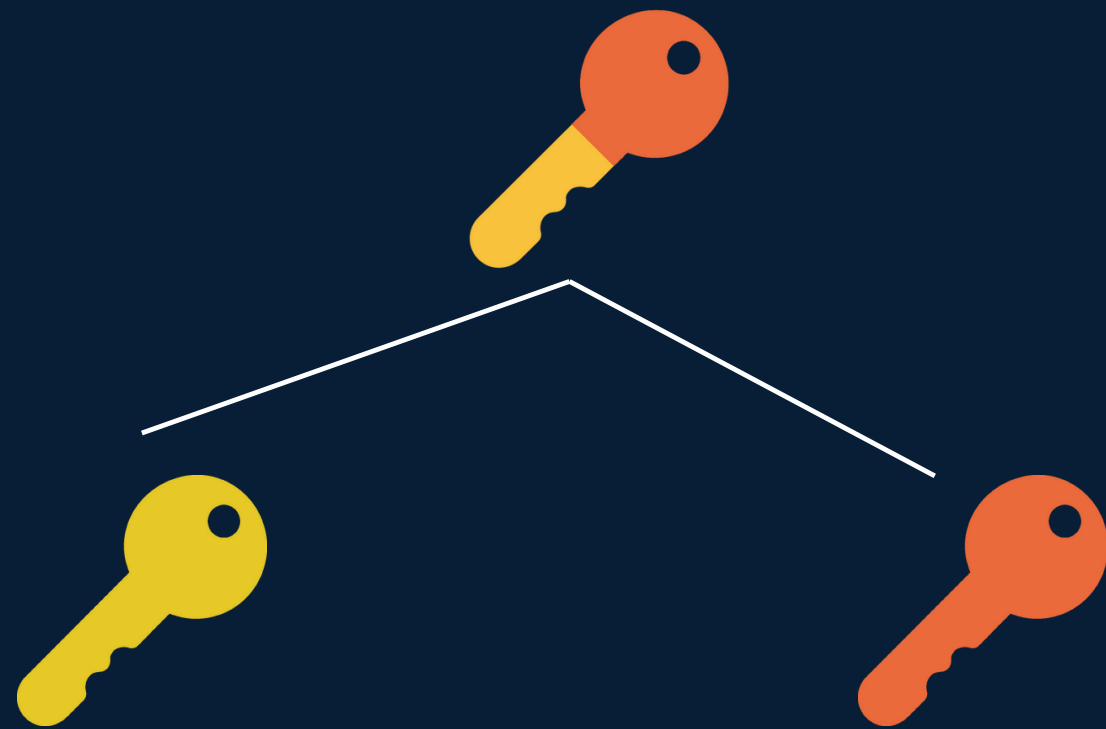
```
OP_SIZE  
OP_PUSHTBYTES_1 20  
OP_EQUALVERIFY  
OP_HASH160  
OP_PUSHTBYTES_20 <preimage hash>  
OP_EQUALVERIFY  
OP_PUSHTBYTES_33 <user public key>  
OP_CHECKSIG
```

Time lock

```
OP_PUSHTBYTES_33 <Boltz public key>  
OP_CHECKSIGVERIFY  
OP_PUSHTBYTES_3 <timeout block height>  
OP_CLTV
```

Ways to spend

Key path



Script path

Hash lock

```
OP_SIZE
OP_PUSHTO 1 20
OP_EQUALVERIFY
OP_HASH160
OP_PUSHTO 20 <preimage hash>
OP_EQUALVERIFY
OP_PUSHTO 33 <user public key>
OP_CHECKSIG
```

Time lock

```
OP_PUSHTO 33 <Boltz public key>
OP_CHECKSIGVERIFY
OP_PUSHTO 3 <timeout block height>
OP_CLTV
```