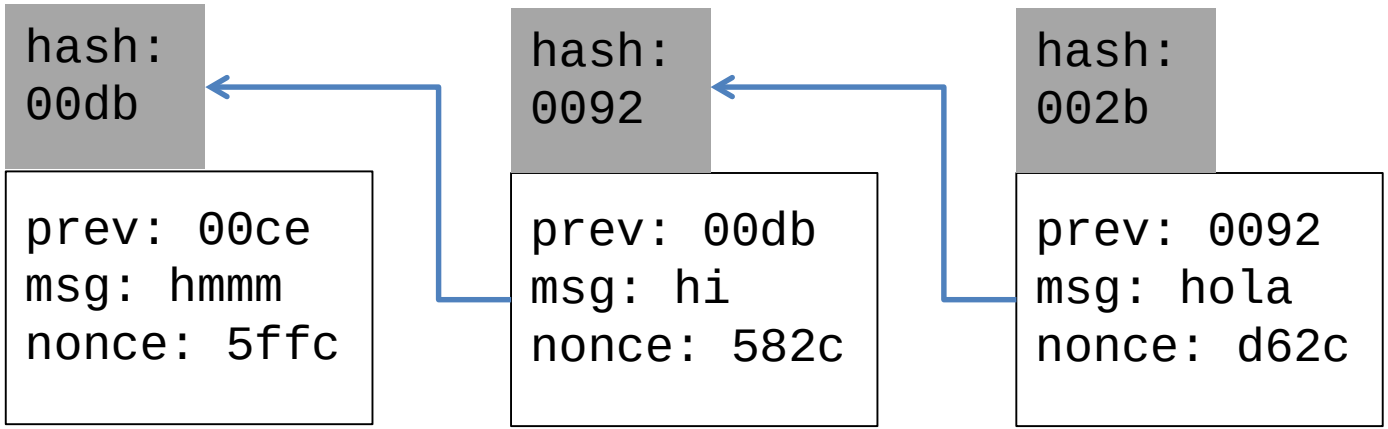
Cryptocurrency Engineering and Design

MAS.S62
2/19/2018 Lecture 4
Neha Narula

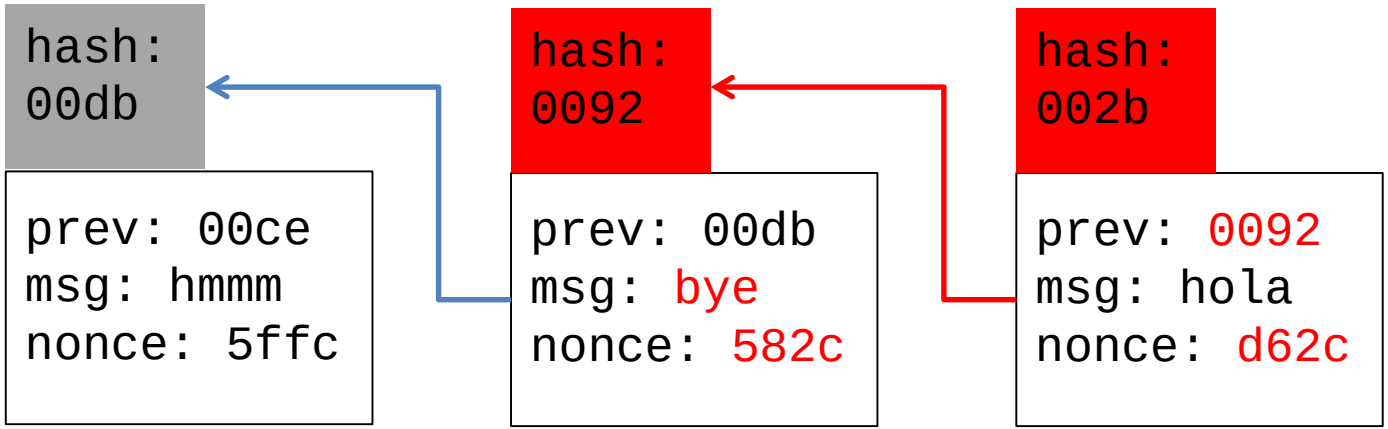
Recap

- Signatures
- Merkle trees
- RSA, ECDSA

Blockchain


hash:
00db
prev: 00ce
msg: hmmm
nonce: 5ffc
hash:
0092
prev: 00db
msg: hi
nonce: 582c
hash:
002b
prev: 0092
msg: hola
nonce: d62c

Blockchain


hash:
00db
prev: 00ce
msg: hmmm
nonce: 5ffc
hash:
0092
prev: 00db
msg: bye
nonce: 582c
hash:
002b
prev: 0092
msg: hola
nonce: d62c

What do we need in a transaction?

What do we need in a transaction?

- Amount
- User, authorization
- Who you’re paying

```
Who: Alice
Amount: $5
Payee: Bob
Auth: Sig_Alice(??)
```

What do we need in a transaction?

- Amount
- User, authorization
- Who you're paying

```
Who: Alice
Amount: $5
Payee: Bob
Auth: Sig_Alice(TXN)
```

What do we need in a transaction?

- Amount
- User, authorization
- Who you're paying

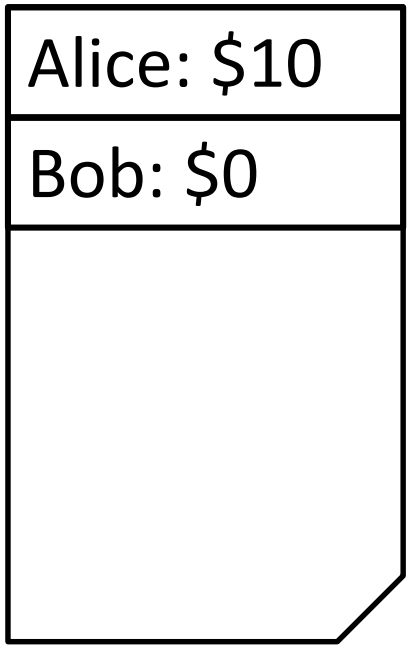
```
Who: Alice
Amount: $5
Payee: Bob
Auth: Sig_Alice(TXN-sig)
```

What do we need in a transaction?

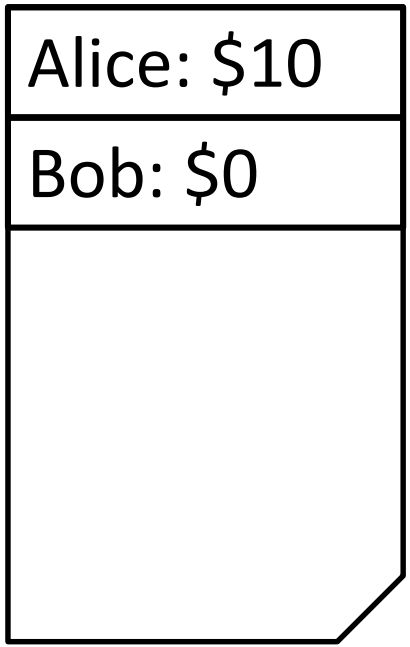
- Amount
- User, authorization
- Who you're paying

```
Who: Alice
Amount: $5
Payee: Bob
Auth: Sig_Alice(H(TXN-sig))
```

Account based model

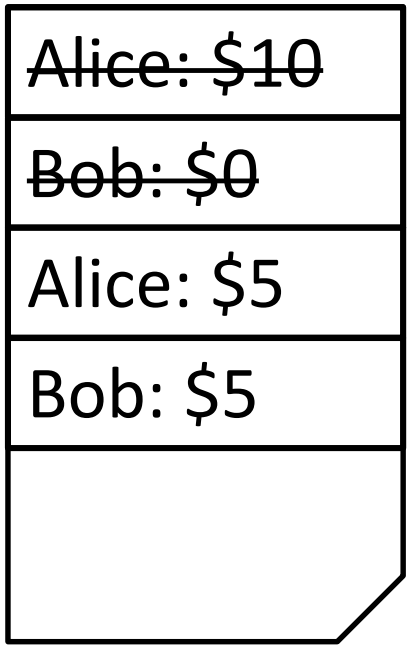


Account based model




```
Who: Alice
Amount: $5
Payee: Bob
Auth: Sig_Alice(TXN-sig)
```

Account based model

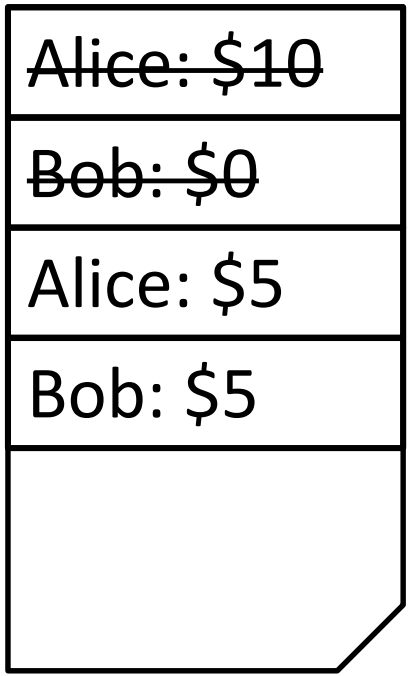


Who: Alice
Amount: $5
Payee: Bob
Auth: Sig_{Alice}(TXN-sig)

Account based model

- Store list of accounts and balances
- A transaction is valid if there is enough balance in the account
- Sender debited, receiver credited

Replay attacks




```
Who: Alice
Amount: $5
Payee: Bob
Auth: Sig_Alice(TXN-sig)
```

Replay attacks

~~Alice: $10~~

~~Bob: $0~~

Alice: $5

Bob: $5

Who: Alice
Amount: $5
Payee: Bob
Auth: Sig_{Alice}(TXN-sig)

Replay attacks

~~Alice: $10~~

~~Bob: $0~~

~~Alice: $5~~

~~Bob: $5~~

Alice: $0

Bob: $10

Who: Alice
Amount: $5
Payee: Bob
Auth: Sig_{Alice}(TXN-sig)

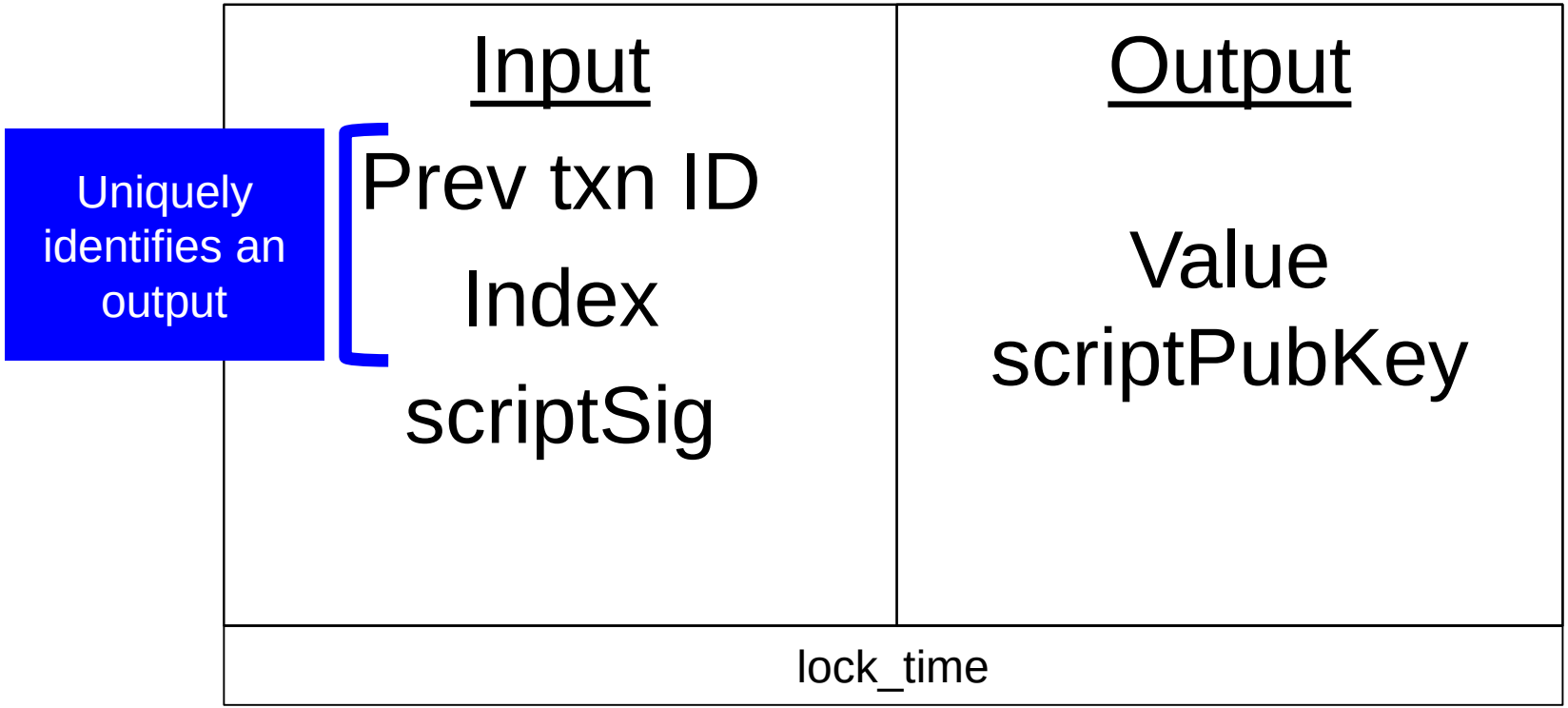
Unspent Transaction Outputs

- All coins are not the same
- Refer to specific coins when spending
- Coins are consumed; create new ones
- A coin can only be spent once

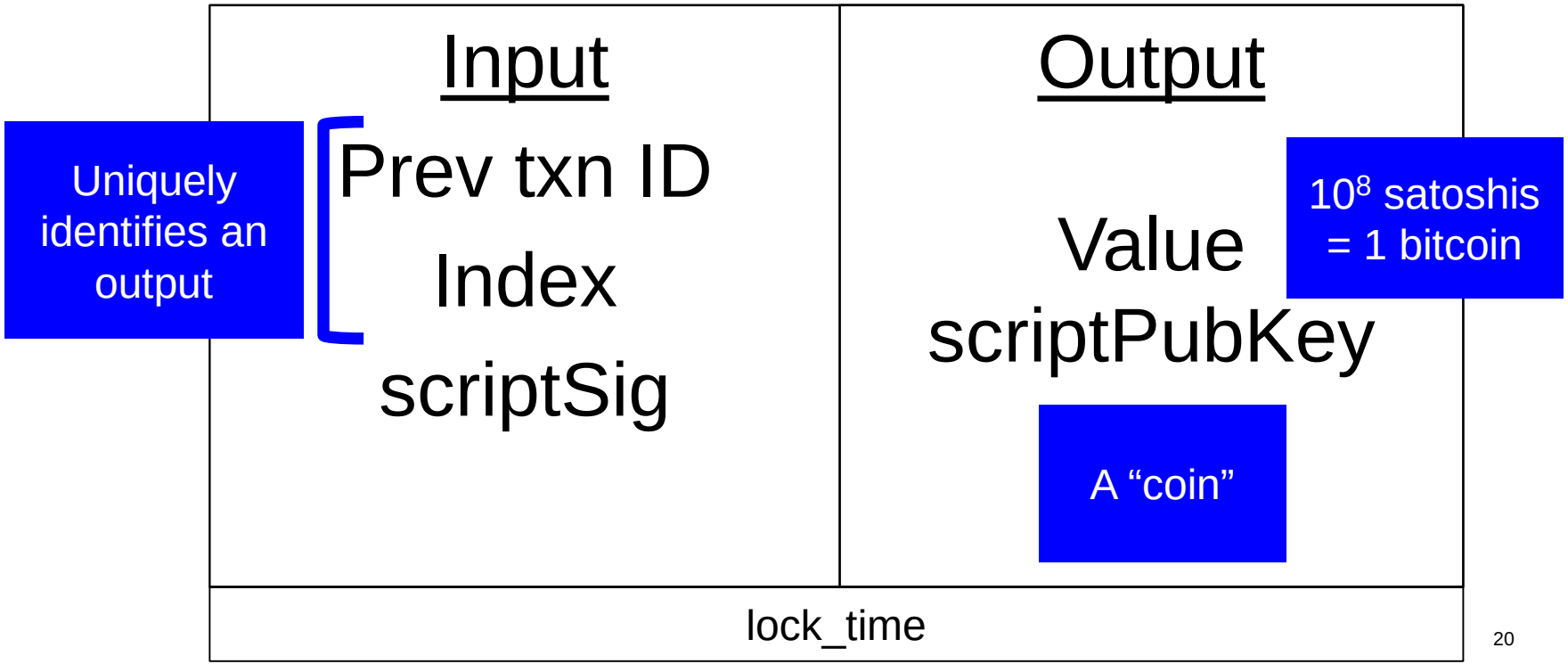
Transaction format

Input	Output
Prev txn ID Index scriptSig	Value scriptPubKey
lock_time	

Transaction format



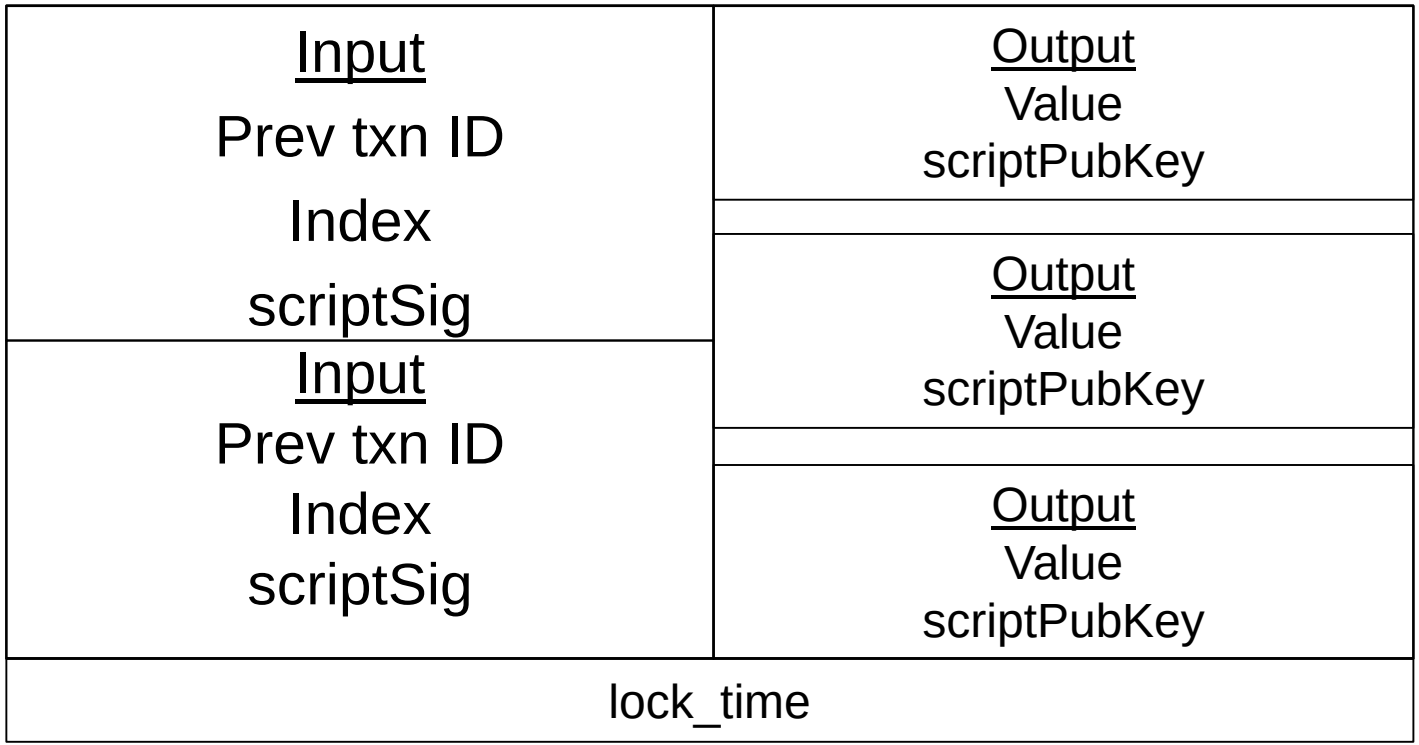
Transaction format



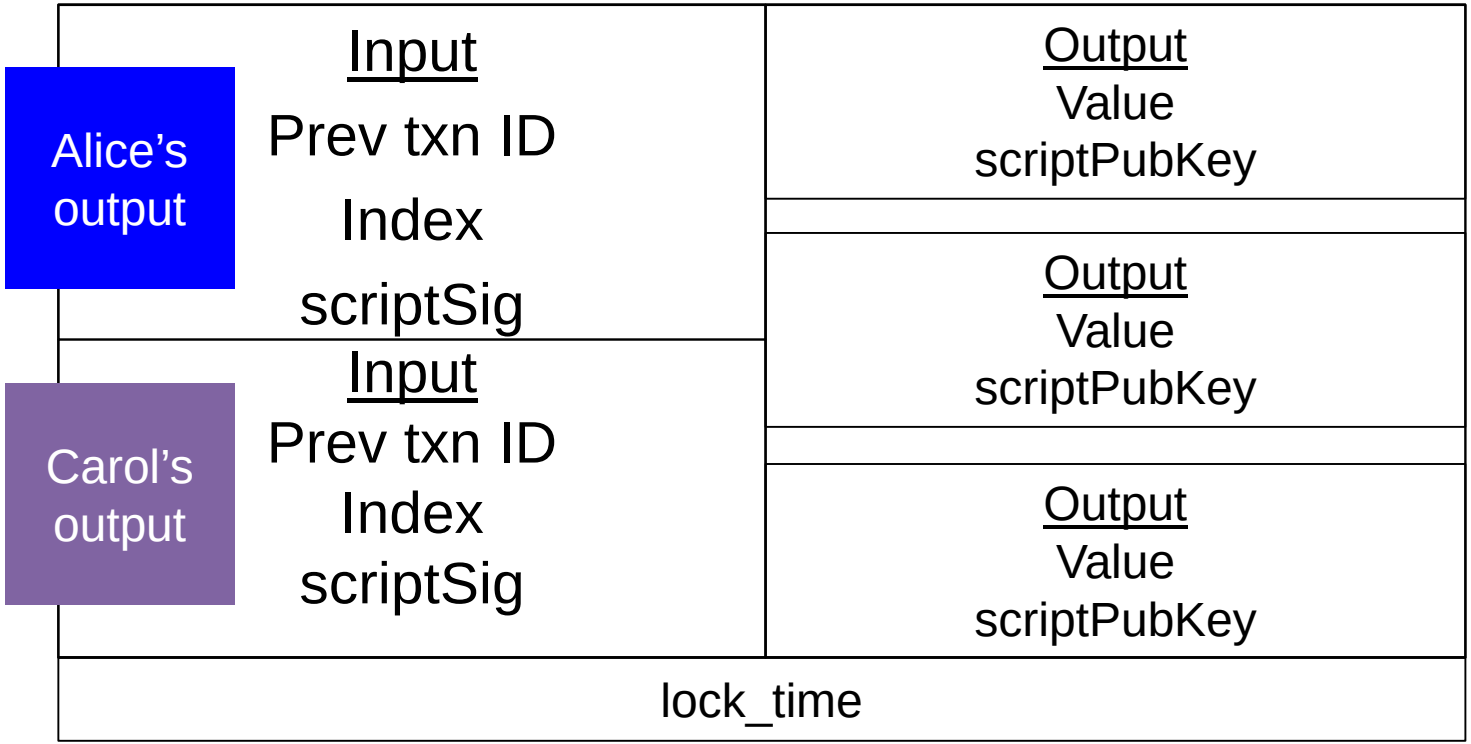
ScriptSigs and scriptPubkeys

- ScriptPubkeys are predicates
- ScriptSigs help satisfy the predicates
- When can you spend a coin? You know how to produce a satisfying scriptSig

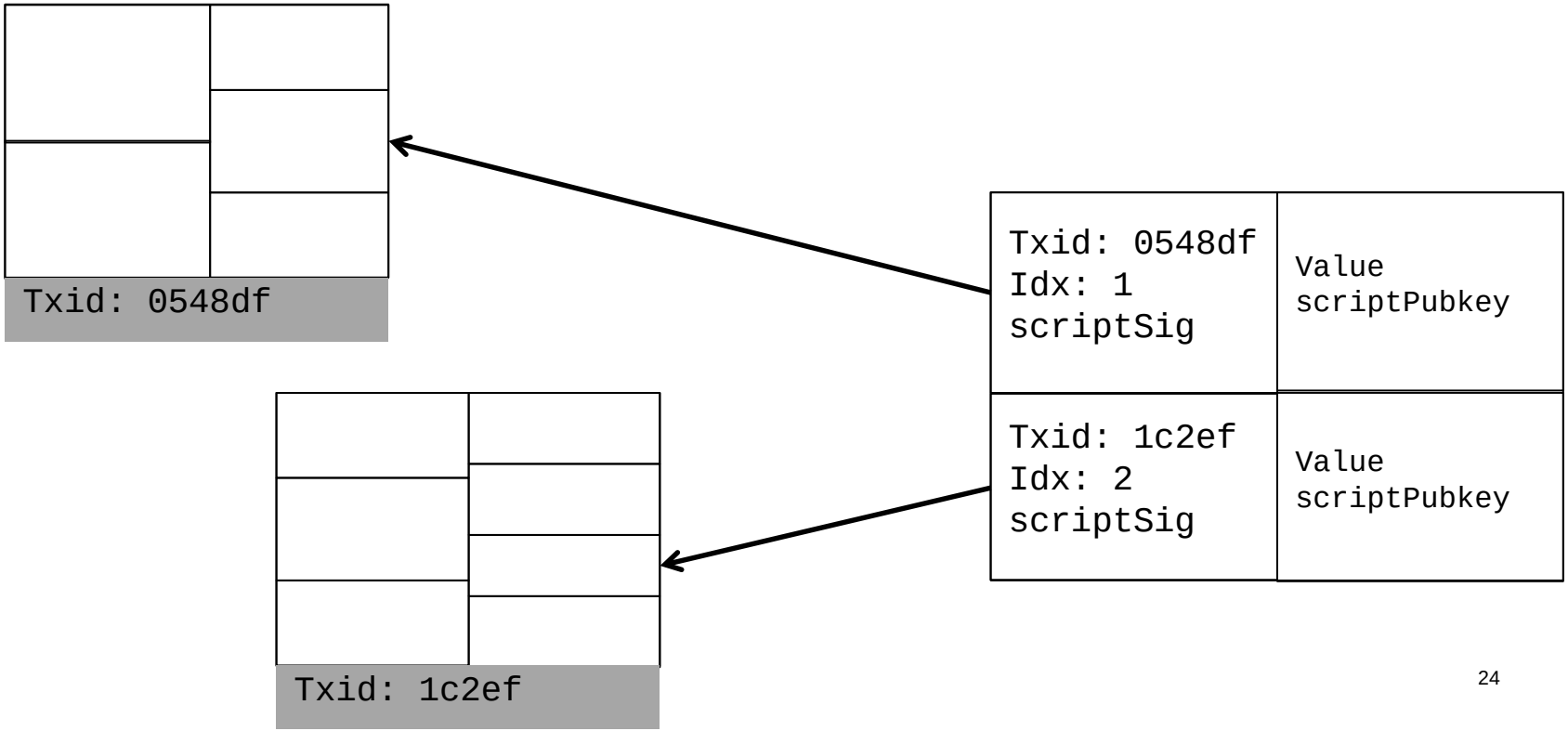
Multiple inputs and outputs



Inputs and outputs are independent



Transactions



```
"txid" : "c80b343d2ce2b5d829c2de9854c7c8d423c0e33bda264c40138d834aab4c0638",
"hash" : "c80b343d2ce2b5d829c2de9854c7c8d423c0e33bda264c40138d834aab4c0638",
"size" : 85,
"vsize" : 85,
"version" : 1,
"locktime" : 0,
"vin" : [
    {
        "txid" : "3f4fa19803dec4d6a84fae3821da7ac7577080ef75451294e71f9b20e0ab1e7b",
        "vout" : 0,
        "scriptSig" : {
            "asm" : "",
            "hex" : ""
        },
        "sequence" : 4294967295
    }
],
"vout" : [
    {
        "value" : 49.99990000,
        "n" : 0,
        "scriptPubKey" : {
            "asm" : "OP_DUP OP_HASH160 cbc20a7664f2f69e5355aa427045bc15e7c6c772 OP_EQUALVERIFY OP_CHECKSIG",
            "hex" : "76a914cbc20a7664f2f69e5355aa427045bc15e7c6c77288ac",
            "reqSigs" : 1,
            "type" : "pubkeyhash",
            "addresses" : [ "mz6KvC4aoUeo6wSxtiVQTo7FDwPnkp6URG” ]
        }
    }
]
```

Consensus rules

- Sum(inputs) <= Sum(outputs)
  - One exception: coinbase transactions
  - Why not equal? Fees!
- For every input, Eval(scriptSig+scriptPubKey) == true
- Output has not already been spent
- lock_time

Pay to Pubkey Hash (P2PKH)

- Idea: Send money to a pubkey
- Pubkeys are big, a hash of a pubkey is only 32 bytes (+1 byte for prefix)
- scriptPubkey: instructions on how to verify a signature of a pubkey that is hashed
- scriptSig: signature, pubkey

Pay to Pubkey Hash (P2PKH)

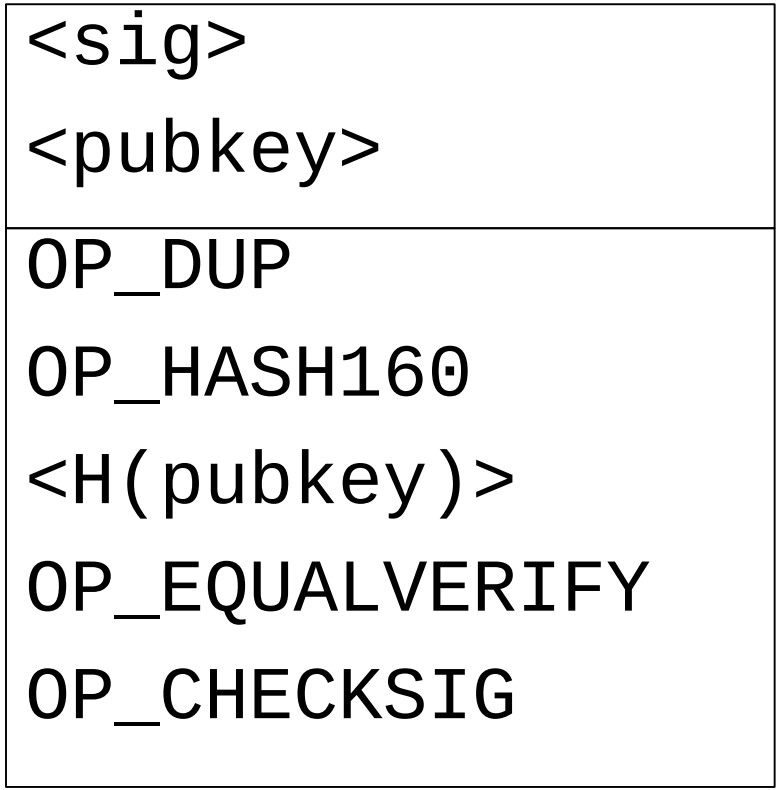
ScriptPubkey:

```
OP_DUP
OP_HASH160
<H(pubkey)>
OP_EQUALVERIFY
OP_CHECKSIG
```

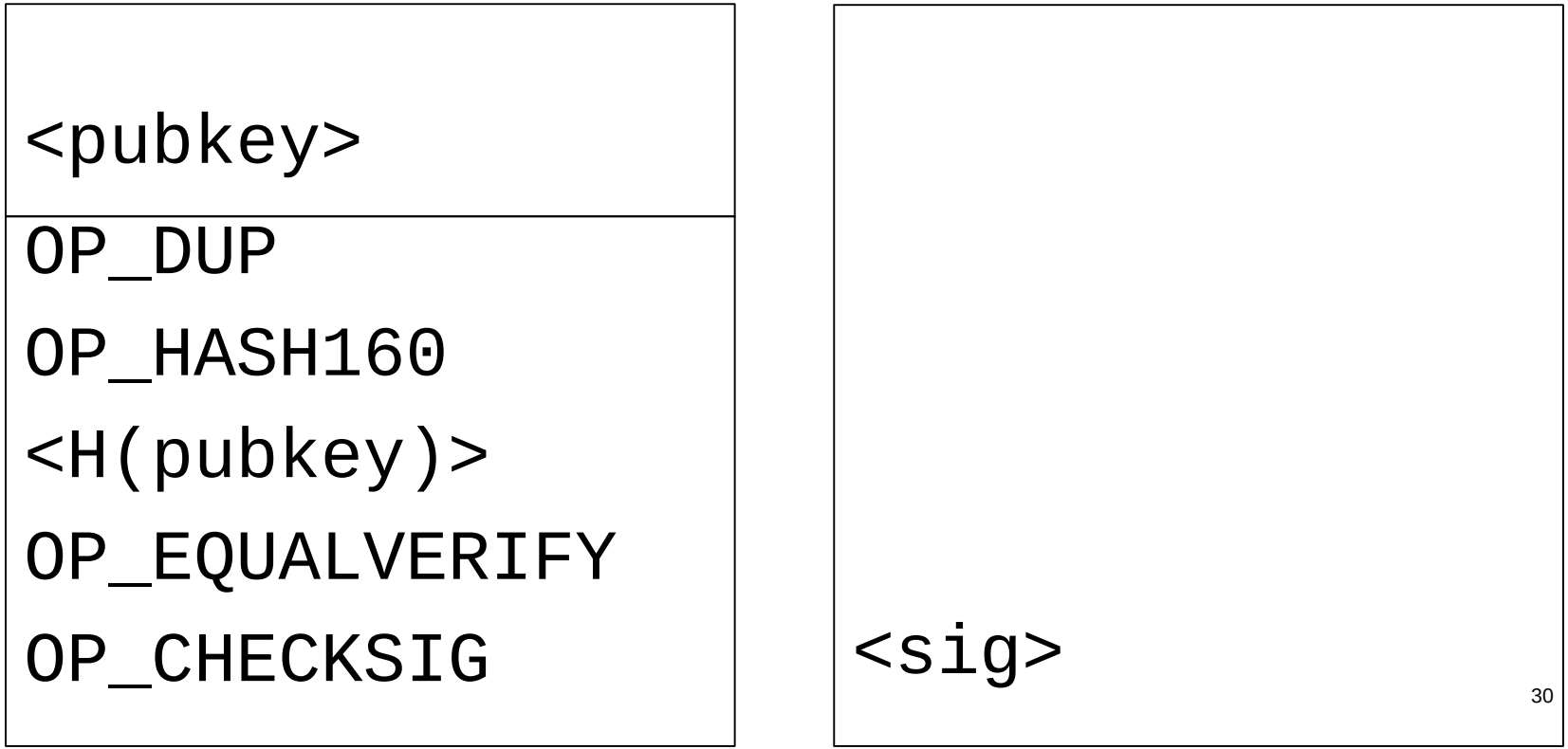
ScriptSig:

```
<sig>
<pubkey>
```

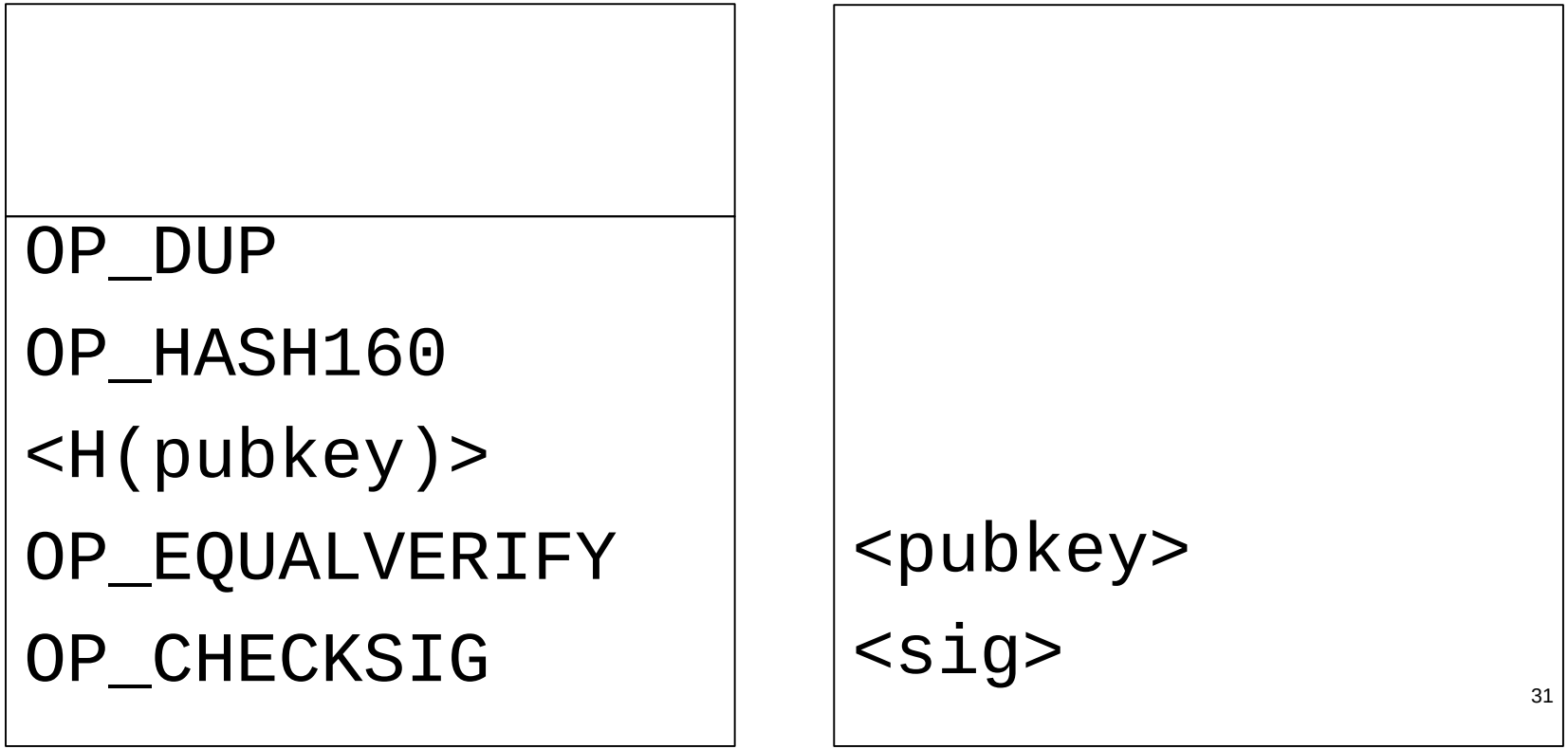
Pay to Pubkey Hash (P2PKH)



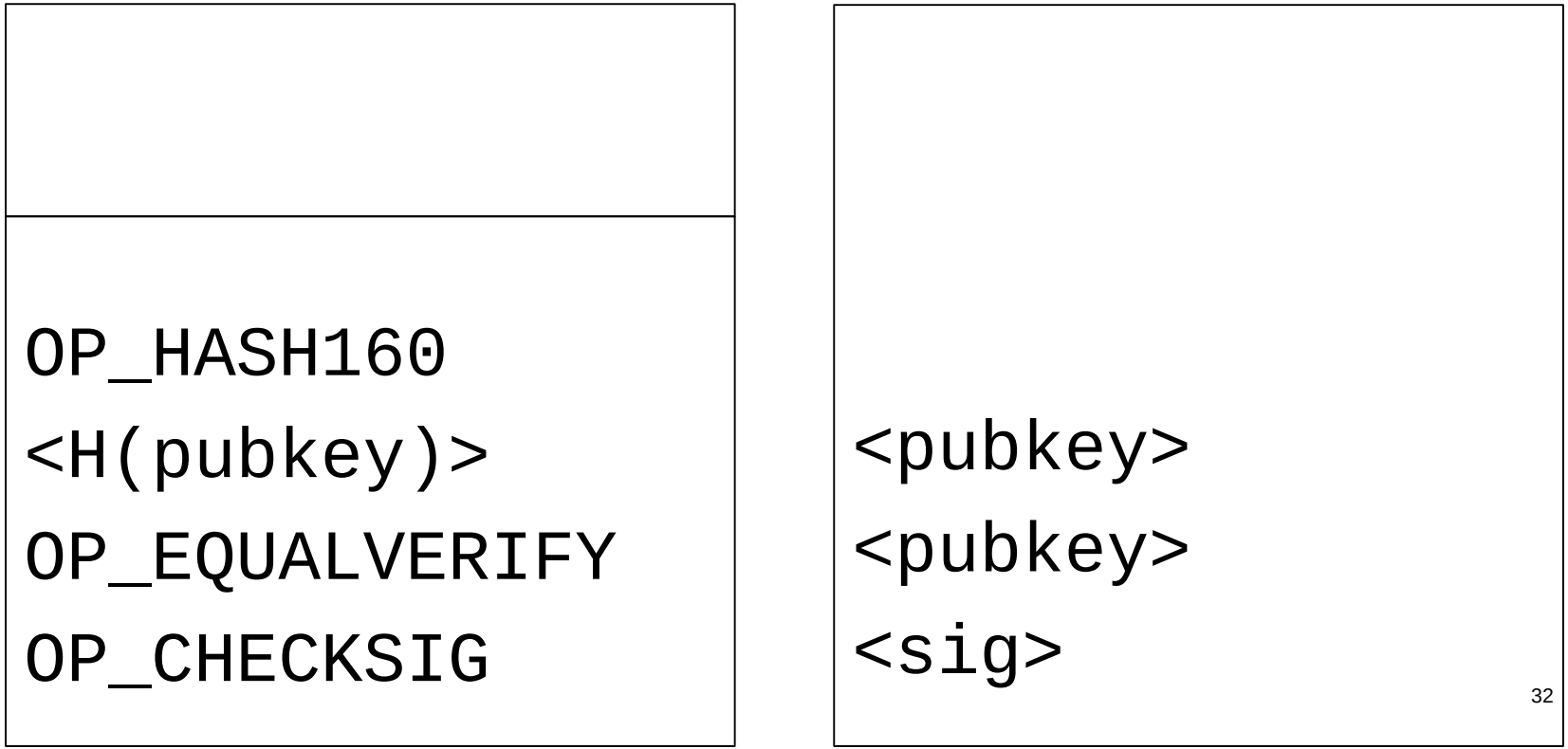
Pay to Pubkey Hash (P2PKH)



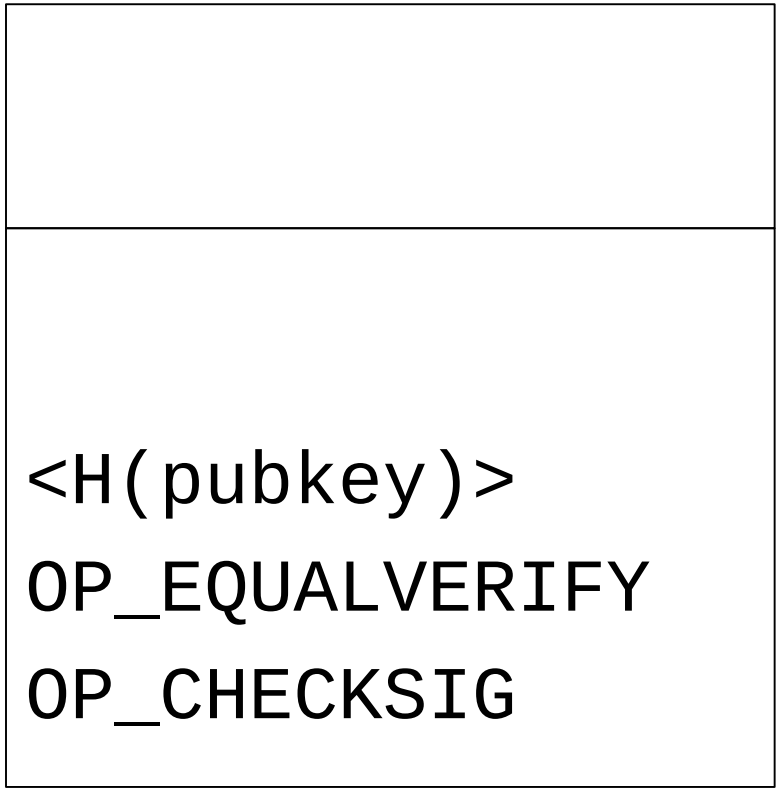
Pay to Pubkey Hash (P2PKH)

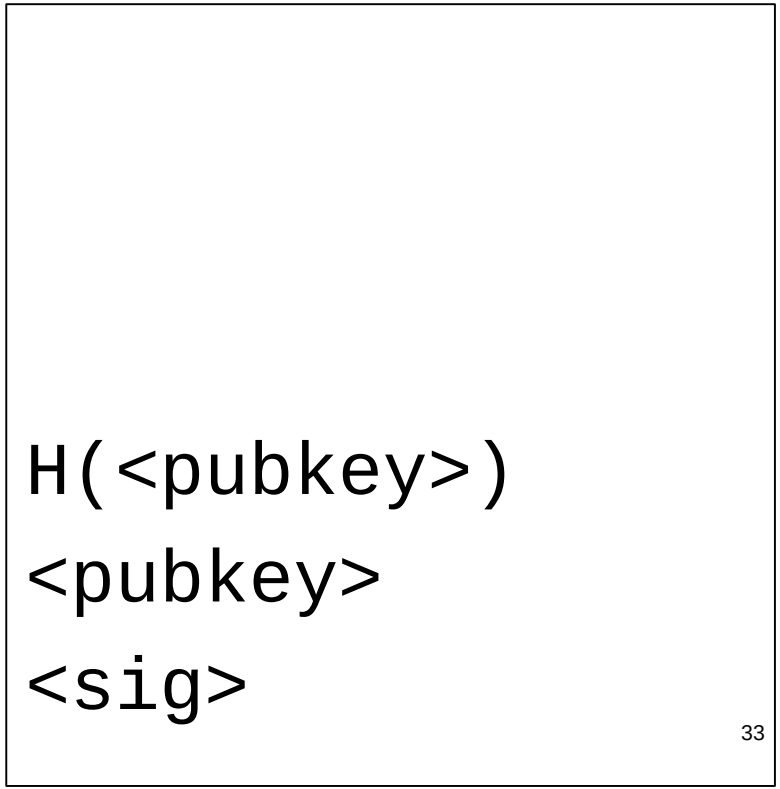


Pay to Pubkey Hash (P2PKH)

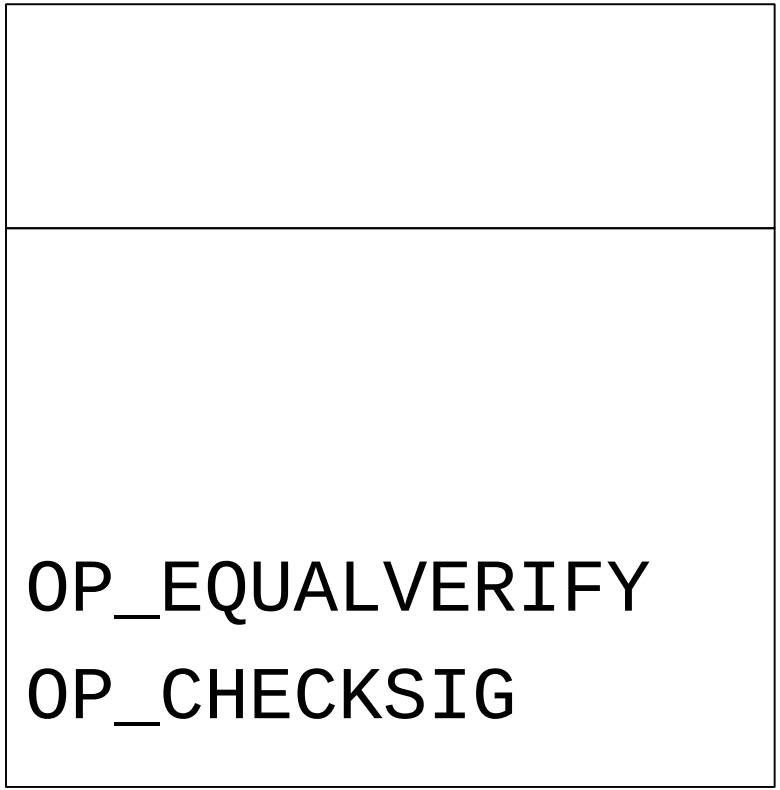


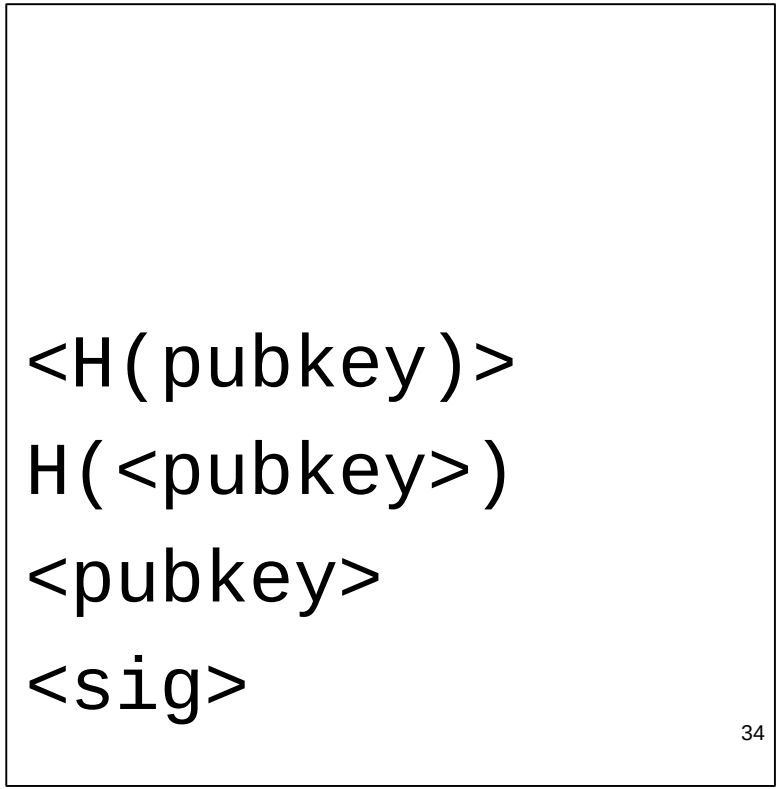
Pay to Pubkey Hash (P2PKH)



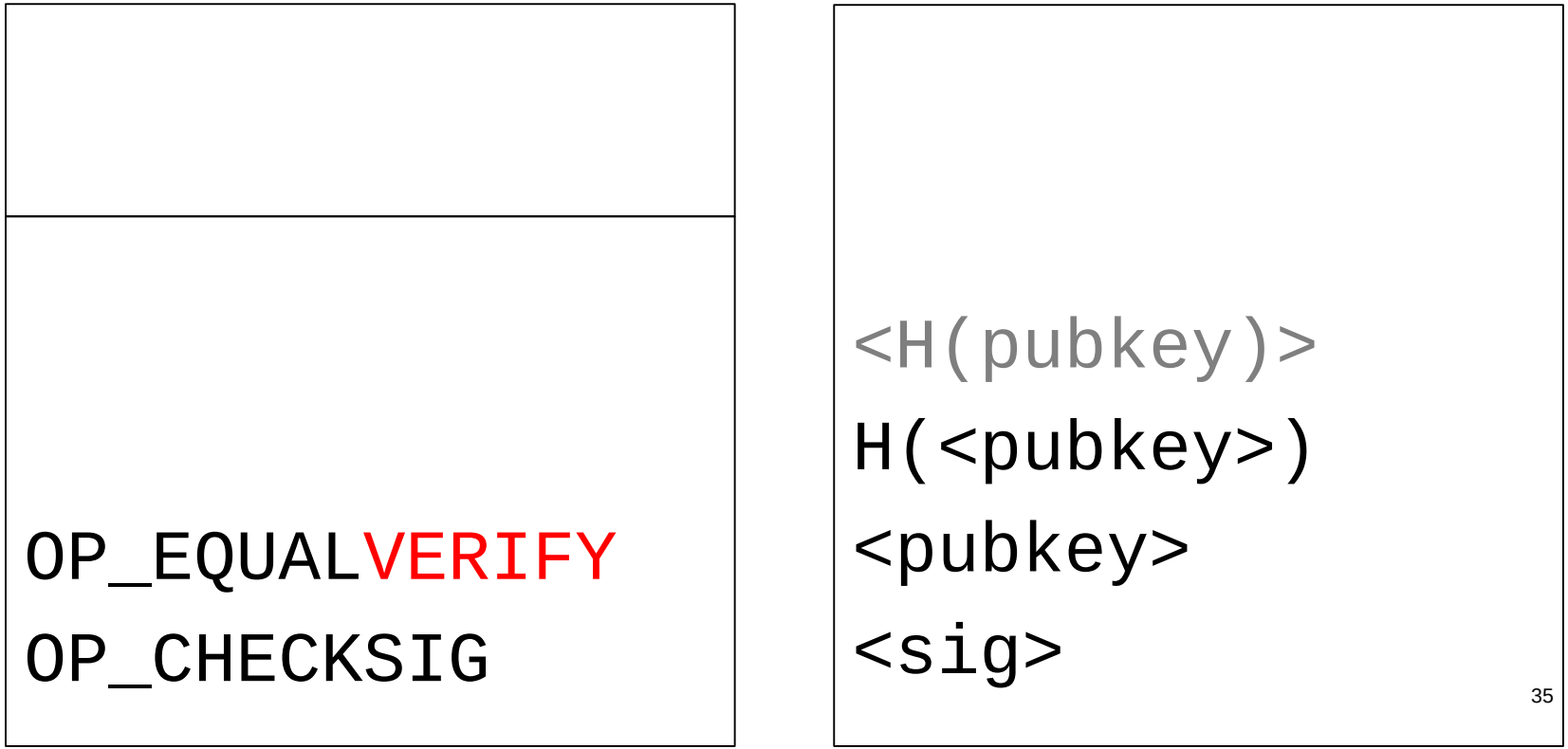


Pay to Pubkey Hash (P2PKH)

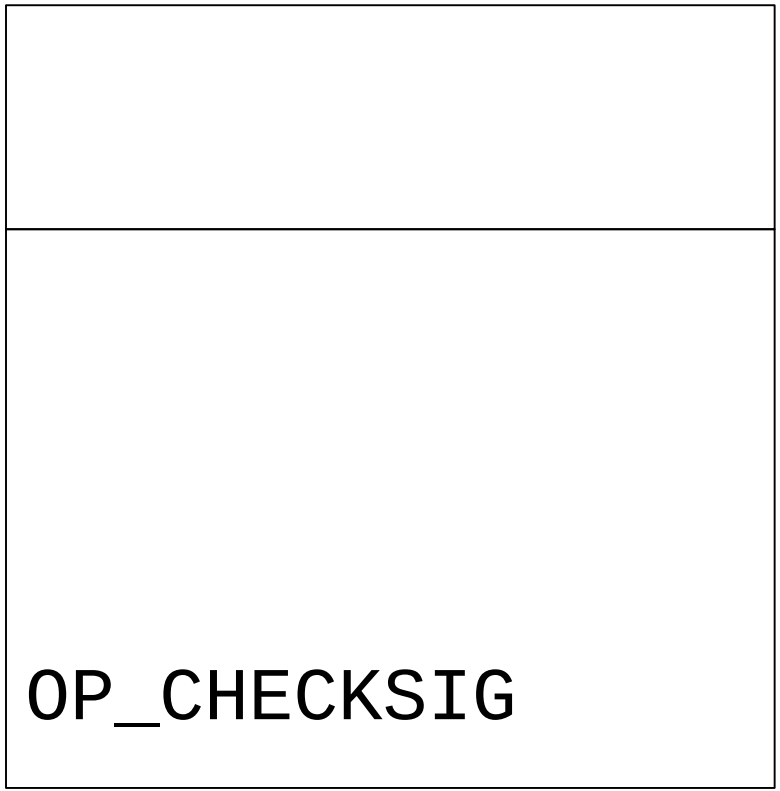


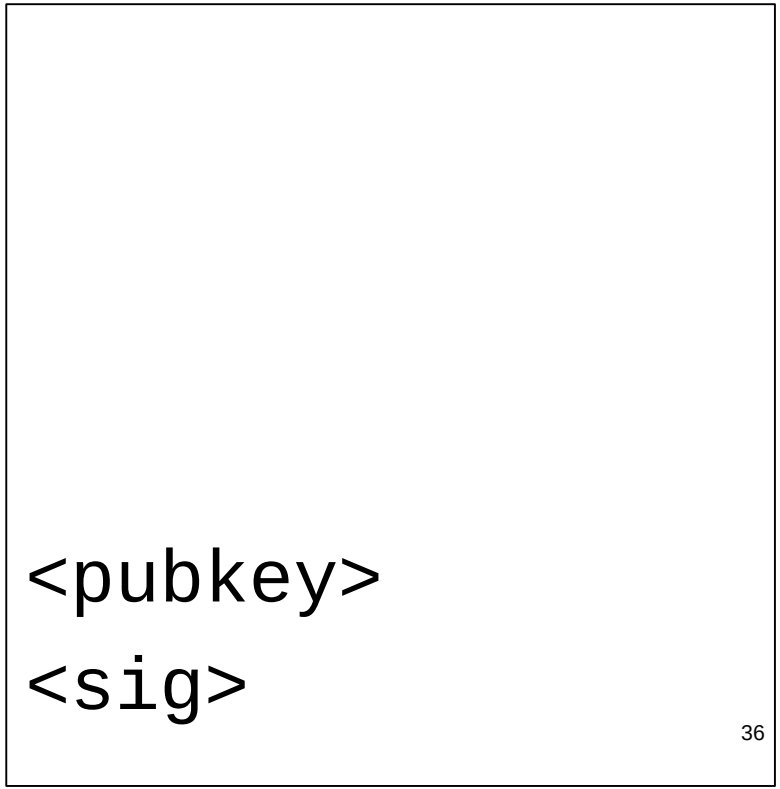


Pay to Pubkey Hash (P2PKH)

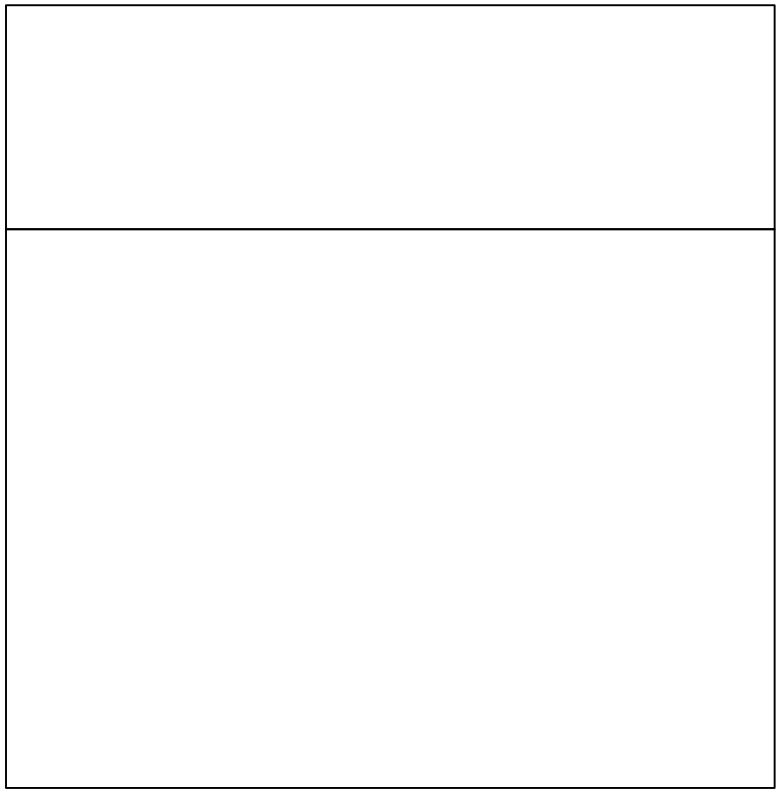


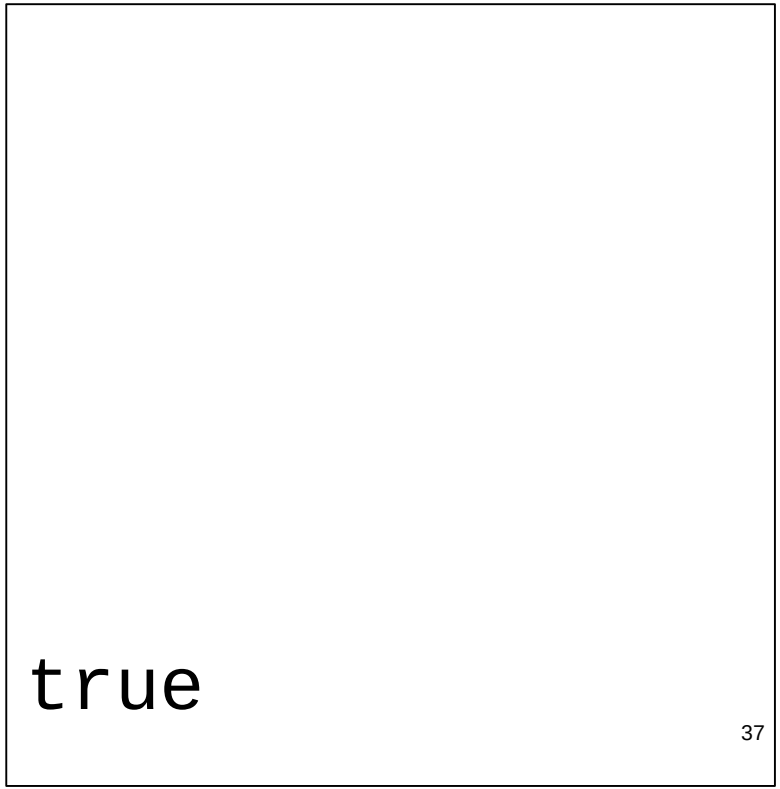
Pay to Pubkey Hash (P2PKH)





Pay to Pubkey Hash (P2PKH)



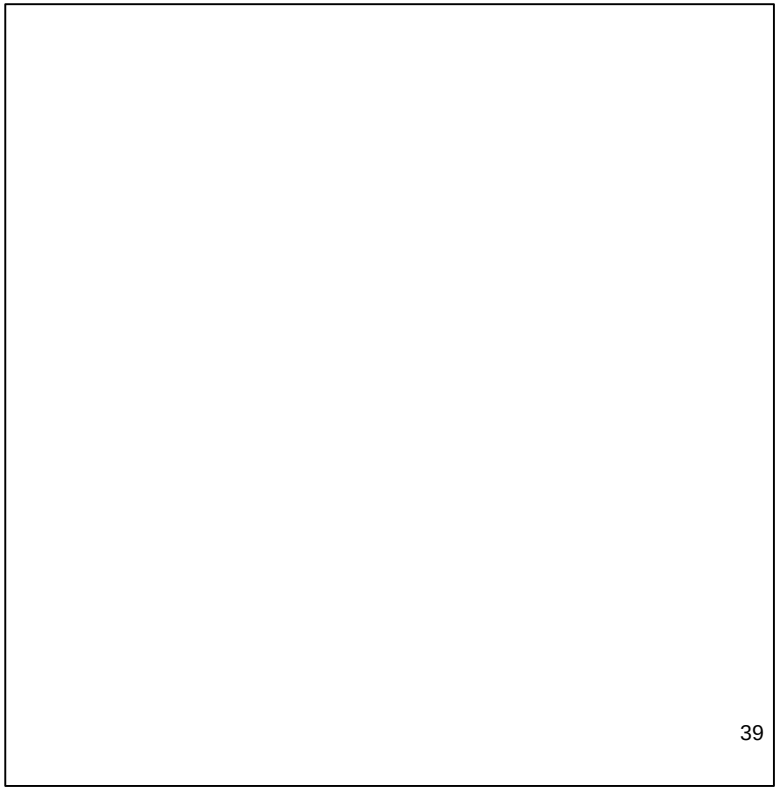


Unspendable output

OP_RETURN
<whatever>

Anyone can spend output

OP_TRUE
<empty>



Benefits of UTXOs

- Help with replay attacks:
  - State order the number of unspent coins, not all accounts
- Privacy (can generate new pubkeys)

Downsides of UTXOs

- Complex
- Fungibility: blacklisting coins

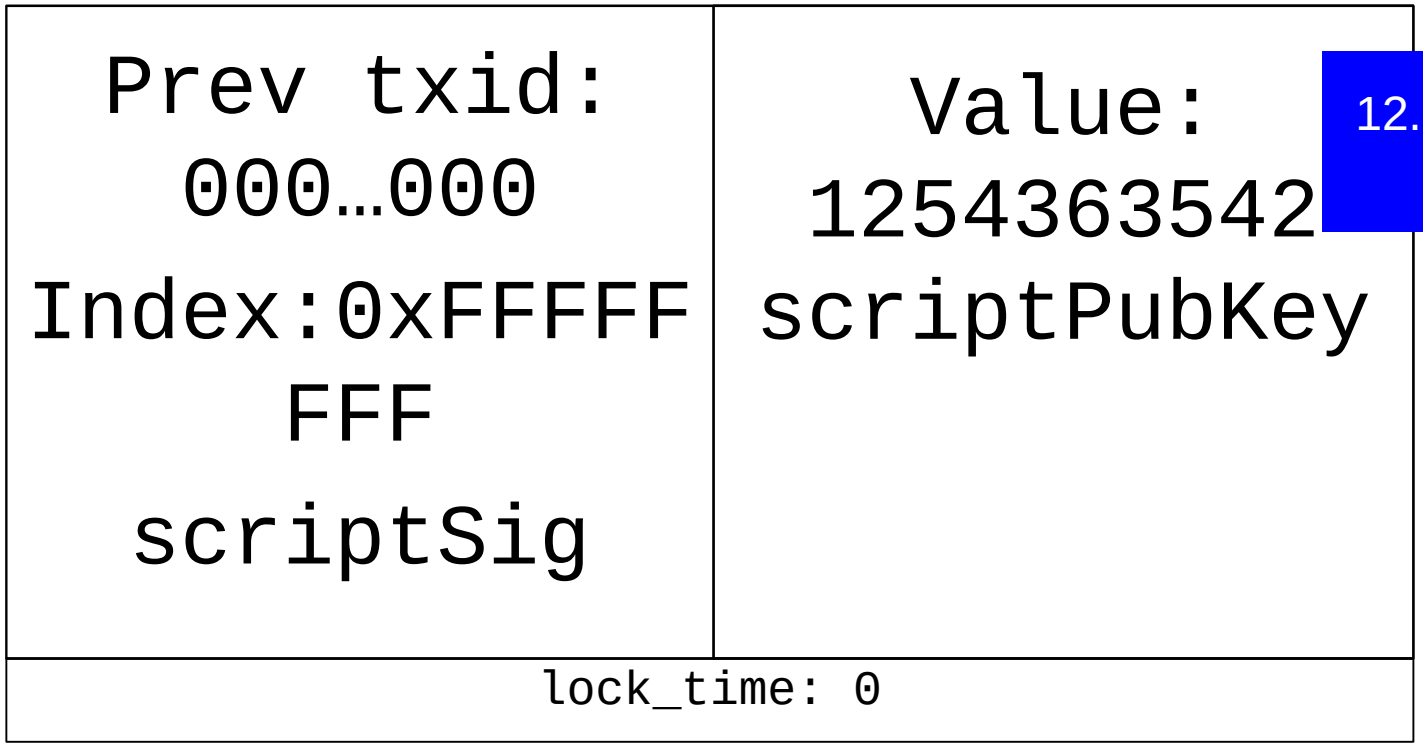
UTXO set

- Every Bitcoin node computes this from the blockchain
- Represents valid set of coins
- ~60M UTXOs
- ~3GB

Coinbase transaction

Prev txid: 000…000 Index:0xFFFFFFFF scriptSig	Value: 1254363542 scriptPubKey
lock_time: 0	

Coinbase transaction



12.5 BTC + fees

MAS.S62 Cryptocurrency Engineering and Design

Spring 2018