

CS251 Fall 2020
(cs251.stanford.edu)



Bitcoin Mechanics

Dan Boneh

Note: HW#1 is posted on the course web site. Due Sep. 28.

Recap

(1) **SHA256**: a collision resistant hash function
that outputs 32-byte hash values

Applications:

- a binding commitment to one value: $\text{commit}(m) \rightarrow H(m)$
or to a list of values: $\text{commit}(m_1, \dots, m_n) \rightarrow \text{Merkle}(m_1, \dots, m_n)$
- Proof of work with difficulty D

Recap

(2) Digital signatures: (Gen, Sign, Verify)

$\text{Gen}() \rightarrow (\text{pk}, \text{sk}),$

$\text{Sign}(\text{sk}, m) \rightarrow \sigma,$ $\text{Verify}(\text{pk}, m, \sigma) \rightarrow \text{accept/reject}$

signing key



verification key



This lecture: Bitcoin mechanics

Oct. 2008: paper by Satoshi Nakamoto

Jan. 2009: Bitcoin network launched

Total market value:



This lecture: Bitcoin mechanics

Layer 3:

user facing tools (cloud servers)

Layer 2:

applications (DAPPs, smart contracts)

Layer 1.5:

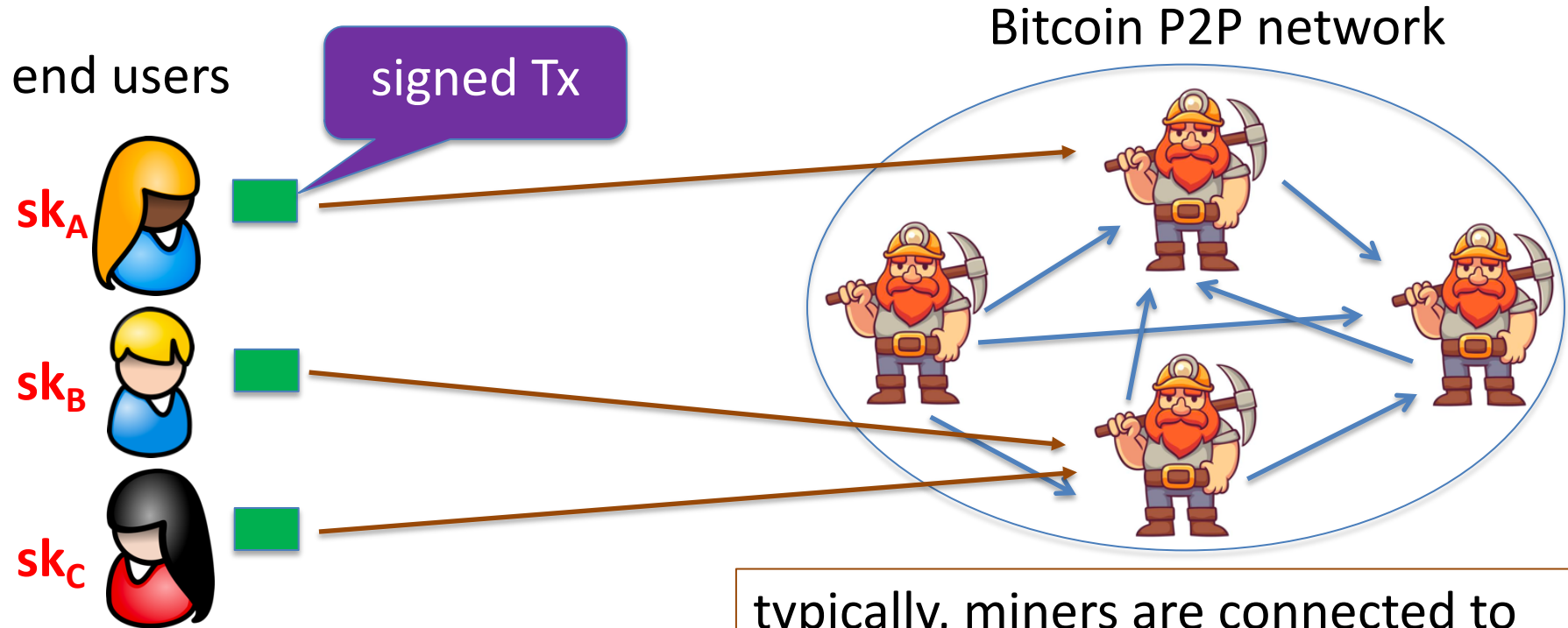
compute layer (blockchain computer)

Layer 1:

consensus layer

next week

First: overview of the Bitcoin consensus layer



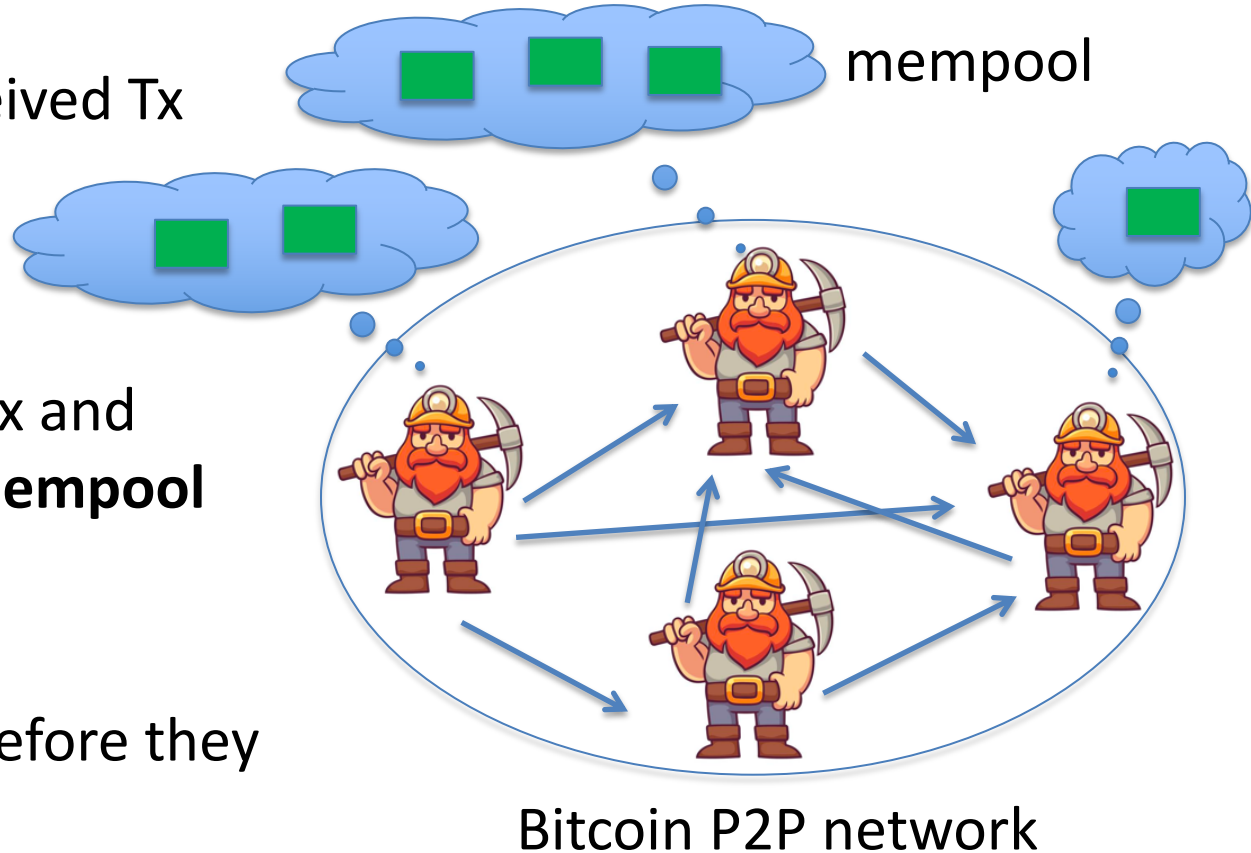
typically, miners are connected to eight other peers (anyone can join)

First: overview of the Bitcoin consensus layer

miners broadcast received Tx
to the P2P network

every miner:
validates received Tx and
stores them in its **mempool**
(unconfirmed Tx)

note: miners see Tx before they
are posted on chain



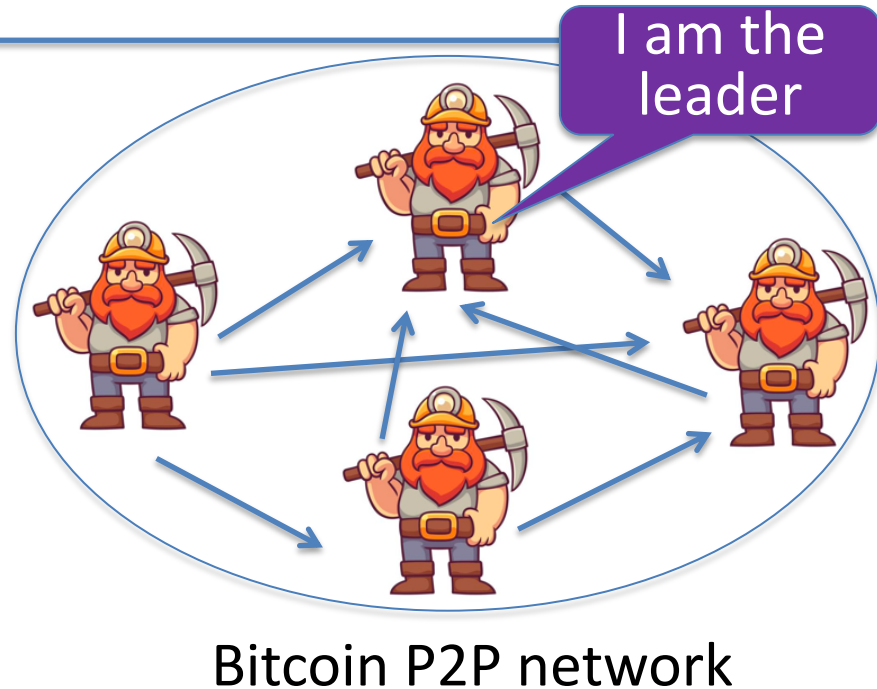
First: overview of the Bitcoin consensus layer

blockchain



Every 10 minutes:

- Each miner creates a candidate block from Tx in its mempool
- a “random” miner is selected (how: next week), and broadcasts its block to P2P network
- all miners validate new block



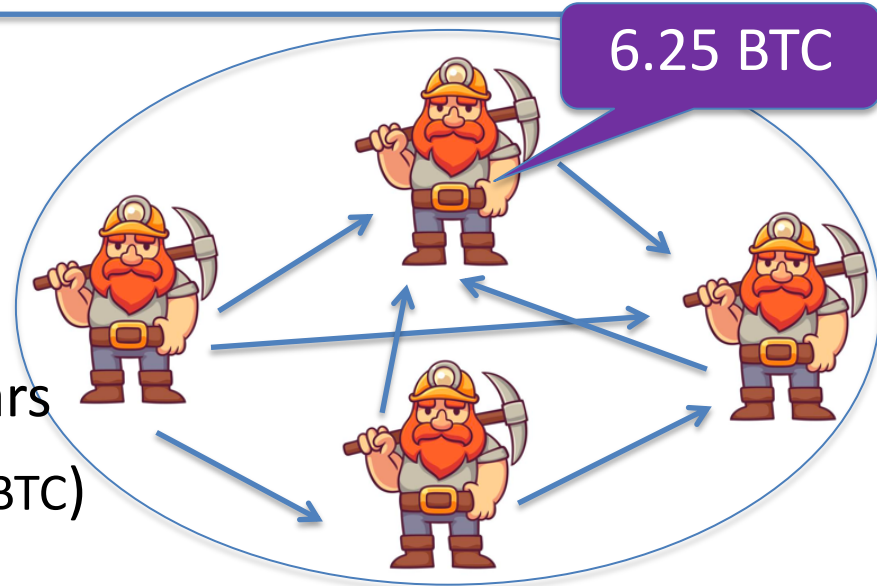
First: overview of the Bitcoin consensus layer

blockchain



Selected miner is paid 6.25 BTC
in **coinbase Tx** (first Tx in the block)

- only way new BTC is created
- block reward halves every four years
⇒ max 21M BTC (currently 18.5M BTC)



note: miner chooses order of Tx in block

Properties (very informal)

Next week:

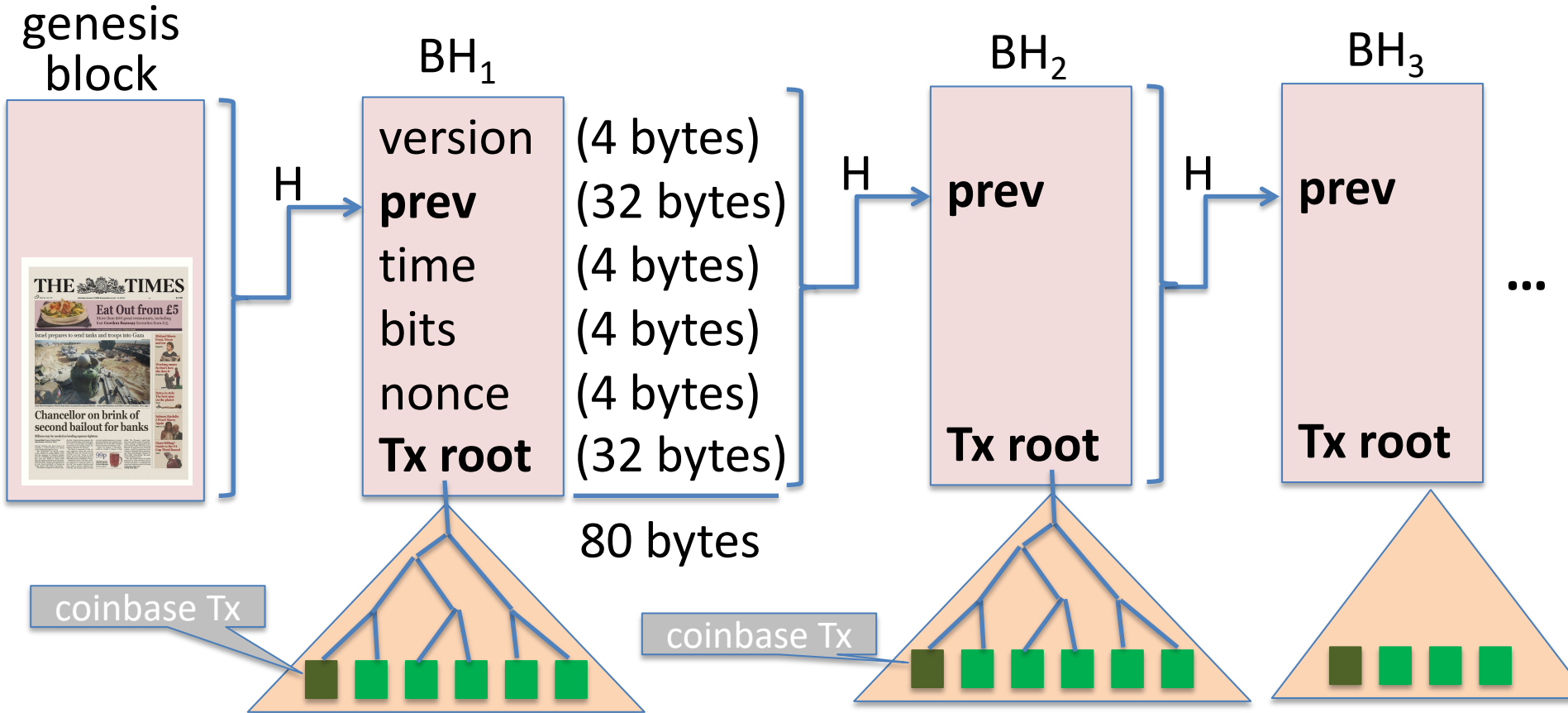
Persistence:

- to remove a block, need to convince 51% of mining power *

Liveness:

- to block a Tx from being posted, need to convince 51% of mining power **

Bitcoin blockchain: a sequence of block headers, 80 bytes each



Bitcoin blockchain: a sequence of block headers, 80 bytes each


time: time miner assembled the block. Self reported.
(block rejected if too far in past or future)

bits: proof of work difficulty
nonce: proof of work solution } for choosing a leader (next week)

Merkle tree: payer can give a short proof that Tx is in the block

new block every ≈ 10 minutes.

An example (Sep. 2020)

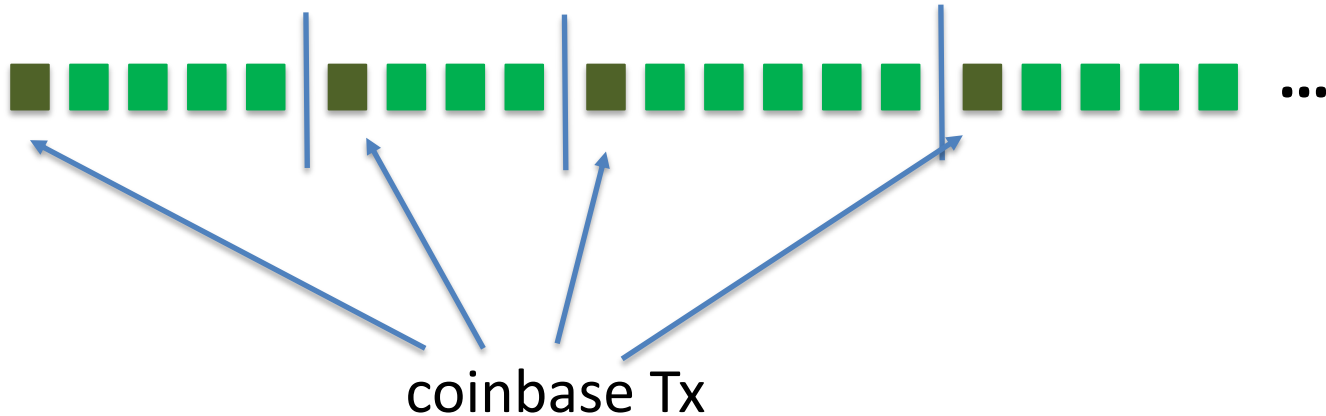
Height	Mined	Miner	Size	Tx data	#Tx
648494	17 minutes	Unknown	1,308,663 bytes		1855
648493	20 minutes	SlushPool	1,317,436 bytes		2826
648492	59 minutes	Unknown	1,186,609 bytes		1128
648491	1 hour	Unknown	1,310,554 bytes		2774
648490	1 hour	Unknown	1,145,491 bytes		2075
648489	1 hour	Poolin	1,359,224 bytes		2622

Block 648493

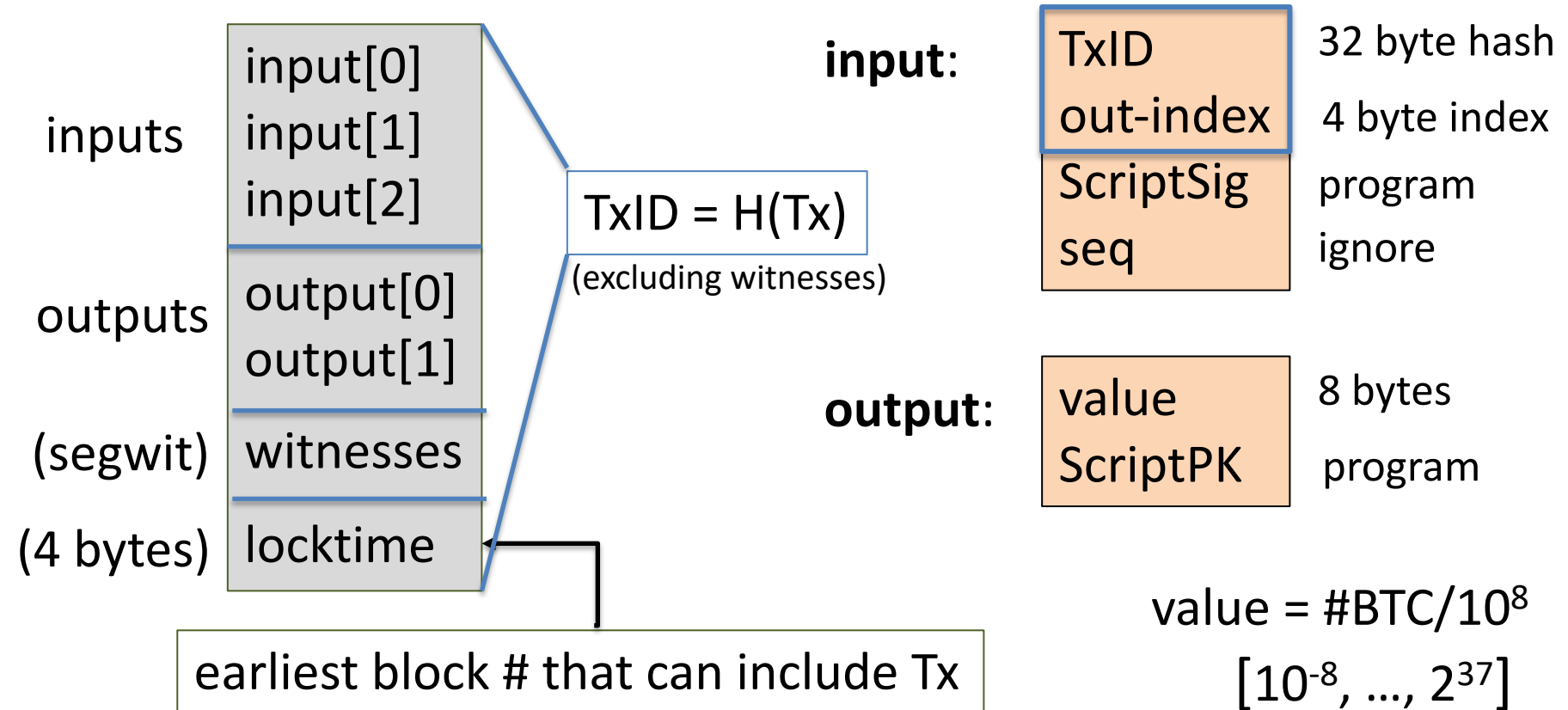
Timestamp	2020-09-15 17:25	
Height	648493	
Miner	SlushPool	(from coinbase Tx)
Number of Transactions	2,826	
Difficulty (D)	17,345,997,805,929.09	(adjusts every two weeks)
Merkle root	350cbb917c918774c93e945b960a2b3ac1c8d448c2e67839223bbcf595baff89	
Transaction Volume	11256.14250596 BTC	
Block Reward	6.25000000 BTC	
Fee Reward	0.89047154 BTC	

This lecture

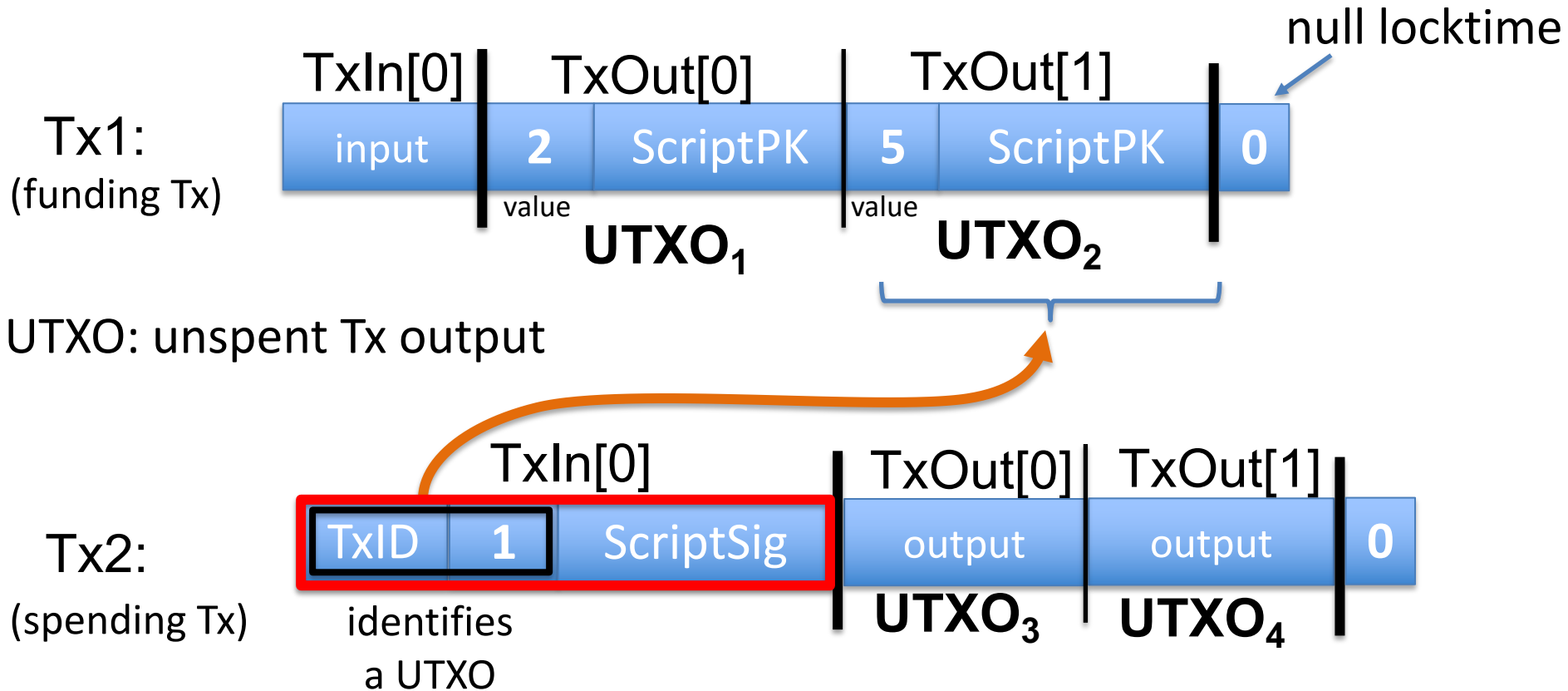
View the blockchain as a sequence of Tx (append-only)



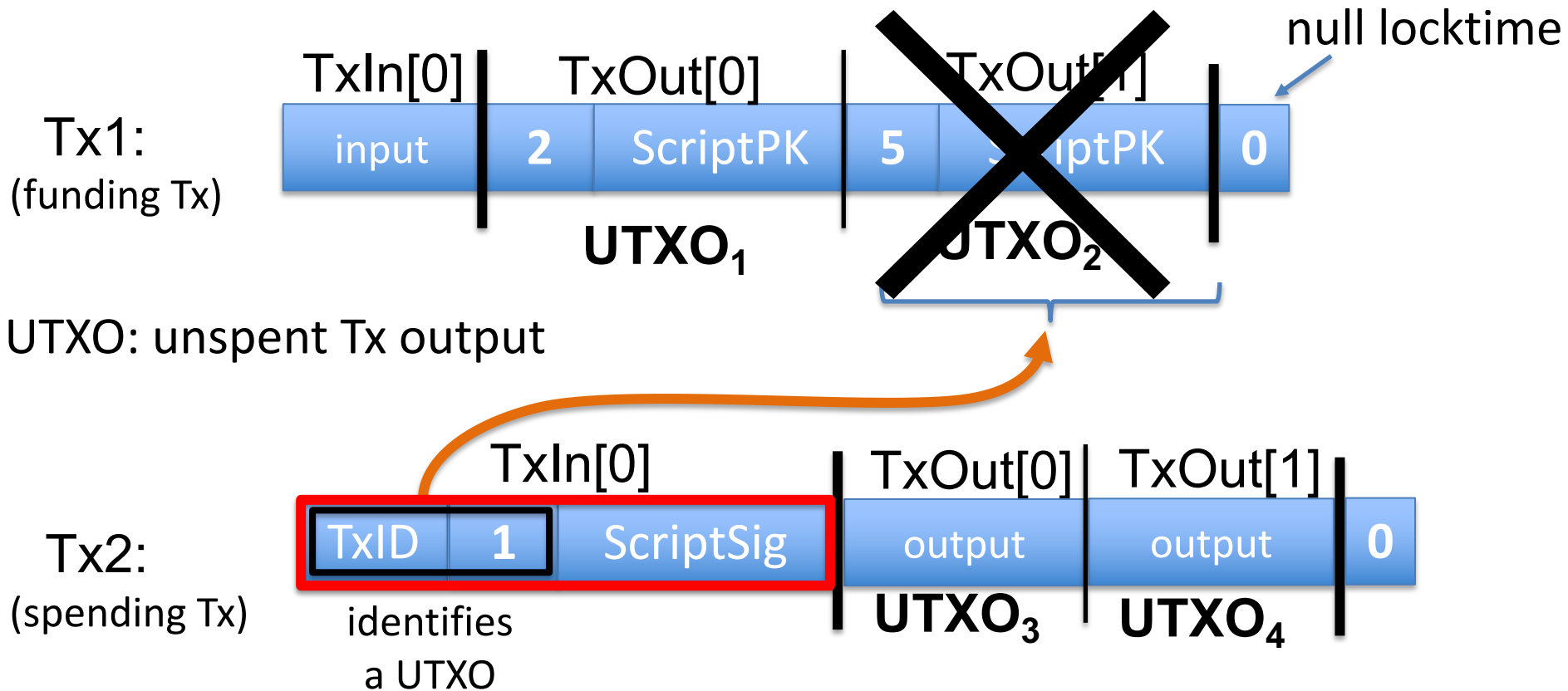
Tx structure (non-coinbase)



Example




Example



Validating Tx2

Miners check (for each input):

program from funding Tx:
under what conditions
can UTXO be spent



1. The program **ScriptSig | ScriptPK** returns true
2. **TxID | index** is in the current UTXO set
3. $\text{sum input values} \geq \text{sum output values}$

After Tx2 is posted, miners remove UTXO_2 from UTXO set

An example (block 648493)

[2826 Tx]

COINBASE (Newly Generated Coins)



1CK6KHY6MHgYvmRQ4PAafKYDrg1ejbH1cE

7.14047154 BTC

OP_RETURN

0.00000000 BTC

OP_RETURN

0.00000000 BTC

Tx0

0.00000000 BTC

7.14047154 BTC

3PuJbxJS1pKxf8EdVR18yBkD1fPAbgUtyw

0.72333974 BTC



1E5Ao1VUnA5BhffvXf2Xmud6avUgwkFnJv

0.00917379 BTC

bc1qr8k3e0vx06lpu3j7m858pa2ak9tyr56ttwvefk 0.61504199 BTC

bc1qdrxve8kua3yz5dgx6wf3u95ngh0d3e648... 0.09290152 BTC

14ZhuXpQ5jCDjtAy7ZMu3hfEQCWewzLw7 0.00616444 BTC

outputs

input

Tx1

0.00005800 BTC

0.72328174 BTC

17MWze4Z1uP1jnvqvj7SAnGtxcoVq11H8A

0.05000000 BTC



3G3C2RFQ8gsf77EQpdR4ZReChWFKEHhxVU

0.04808000 BTC

Tx2

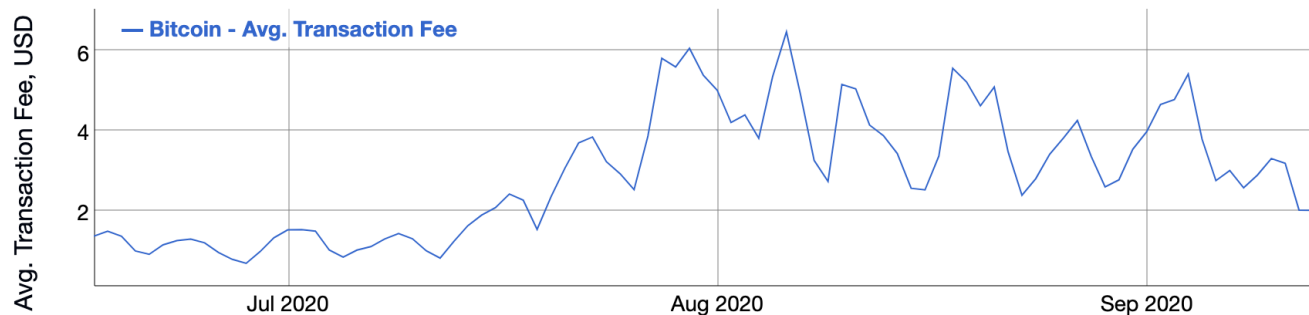
0.00192000 BTC

0.04808000 BTC

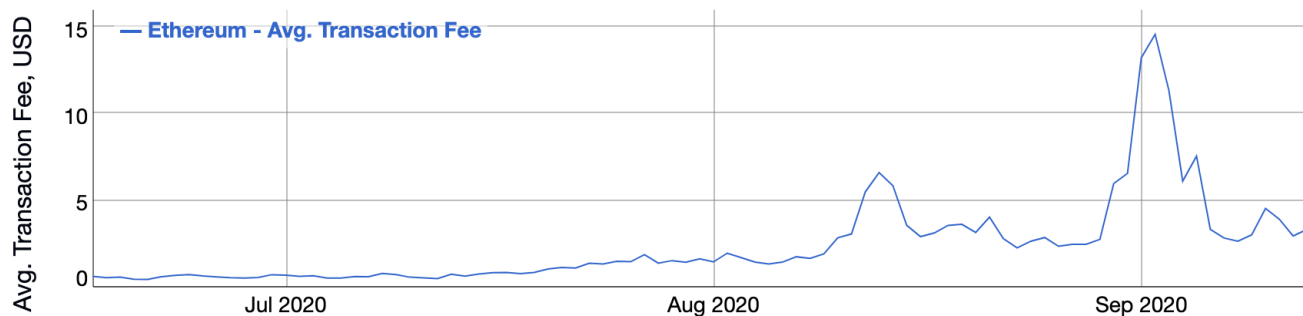
sum of fees in block added to coinbase Tx

Tx fees

Bitcoin average Tx fees in USD



Ethereum average Tx fees in USD



Focusing on Tx2: TxInp[0]

from UTXO
(Bitcoin script)

Value 0.05000000 BTC

Pkscript
OP_DUP
OP_HASH160
45b21c8a0cb687d563342b6c729d31dab58e3a4e
OP_EQUALVERIFY
OP_CHECKSIG

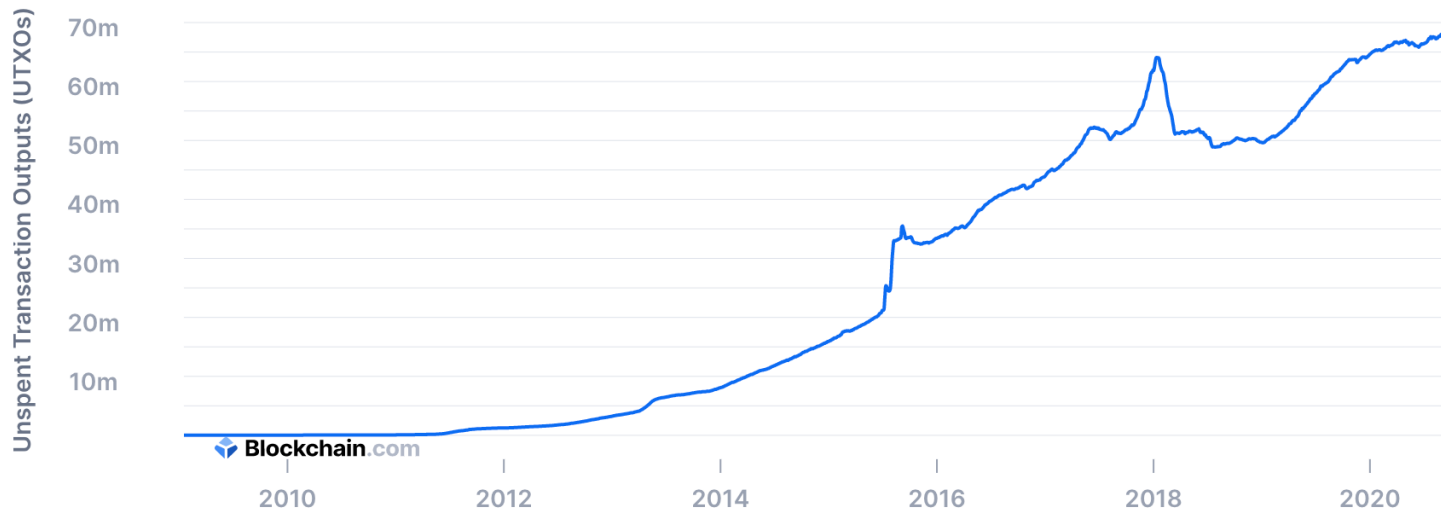
Sigscript
304402205846cace0d73de82dfbdeba4d65b9856d7c1b1730eb401cf4906b2401a69b
dc90220589d36d36be64e774c8796b96c011f29768191abeb7f56ba20ffb0351280860
c01
03557c228b080703d52d72ead1bd93fc72f45c4ddb4c2b7a20c458e2d069c8dd9e

from TxInp[0]

All value in Bitcoin is held in UTXOs

Unspent Transaction Outputs

The total number of valid unspent transactions outputs. This excludes invalid UTXOs with opcode OP_RETURN



Sep. 2020: miners need to store $\approx 70\text{M}$ UTXOs in memory

Bitcoin Script

A stack machine. Not Turing Complete: no loops.

Quick survey of op codes:

1. **OP_TRUE** (OP_1), **OP_2**, ..., **OP_16**: push value onto stack

81

82

96

2. **OP_DUP**: push top of stack onto stack

118

Bitcoin Script

3. control:

99 **OP_IF** <statements> **OP_ELSE** <statements> **OP_ENDIF**

105 **OP_VERIFY**: abort fail if top = false

106 **OP_RETURN**: abort and fail

what is this for? ScriptPK = [OP_RETURN, <data>]

136 **OP_EQVERIFY**: pop, pop, abort fail if not equal

Bitcoin Script

4. arithmetic:

OP_ADD, OP_SUB, OP_AND, ...: pop two items, add, push

5. crypto:

OP_SHA256: pop, hash, push

OP_CHECKSIG: pop sig, pop pk, verify sig. on Tx, push 0 or 1

6. Time: **OP_CheckLockTimeVerify** (CLTV):

fail if value at the top of stack > Tx locktime value.

usage: UTXO can specify min-time when it can be spent

Example: a common script

<sig> <pk> **DUP** **HASH256** <pkhash> **EQVERIFY** **CHECKSIG**

stack: empty

<sig> <pk>

<sig> <pk> <pk>

<sig> <pk> <hash>

<sig> <pk> <hash> <pkhash>

<sig> <pk>

1

⇒ successful termination

init

push values

DUP

HASH256

push value

EQVERIFY

CHECKSIG

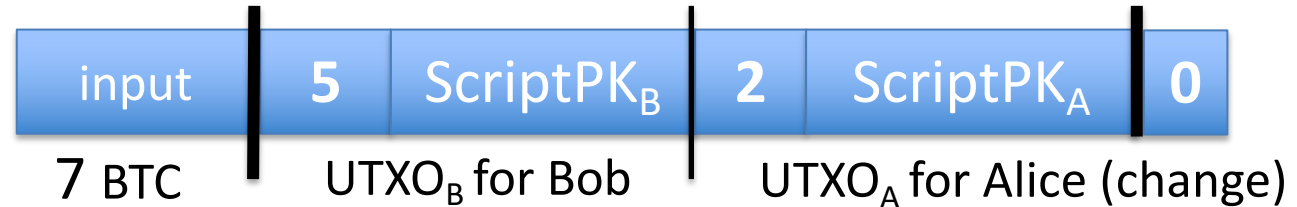
verify(pk, Tx, sig)

Transaction types: (1) P2PKH

pay to public key hash

Alice want to pay Bob 5 BTC:

- step 1: Bob generates sig key pair $(pk_B, sk_B) \leftarrow \text{Gen}()$
- step 2: Bob computes his Bitcoin address as $Addr_B \leftarrow H(pk_B)$
- step 3: Bob sends $Addr_B$ to Alice
- step 4: Alice creates Tx:

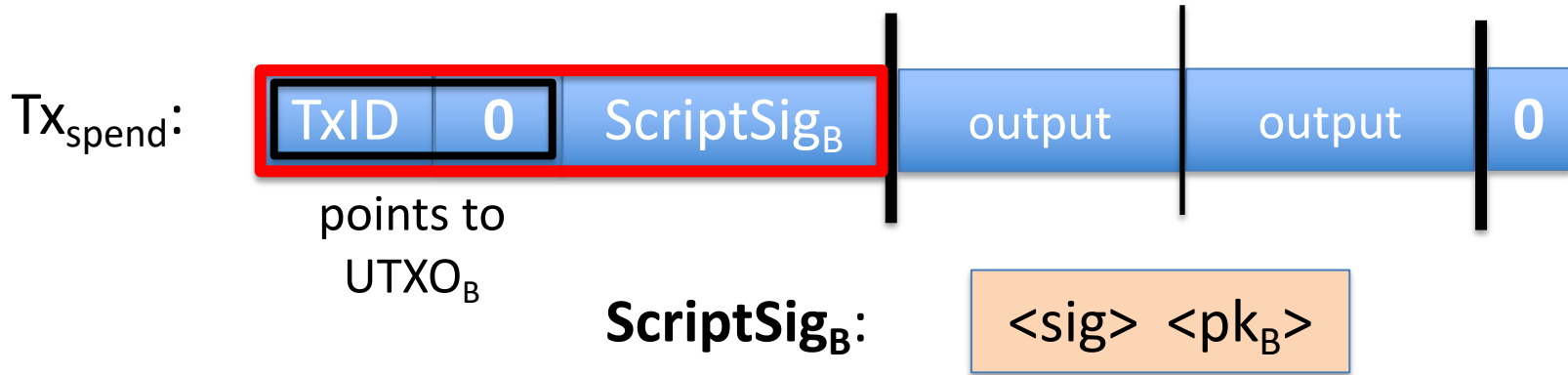


ScriptPK_B:

DUP HASH256 <Addr_B> EQVERIFY CHECKSIG

Transaction types: (1) P2PKH

Later, when Bob wants to spend his UTXO: create a Tx_{spend}



$\langle \text{sig} \rangle = \text{Sign}(\text{sk}_B, Tx)$ where $Tx = (Tx_{\text{spend}} \text{ excluding all ScriptSigs})$ (SIGHASH_ALL)

Miners validate that $\text{ScriptSig}_B \mid \text{ScriptPK}_B$ returns true

P2PKH: comments

- Alice specifies recipient's pk in UTXO_B
- Recipient's pk is not revealed until UTXO is spent
(some security against attacks on pk)
- Miner cannot change $\langle \text{Addr}_B \rangle$ and steal funds:
invalidates the signature that created the UTXO_B

Segregated Witness

ECDSA malleability:

- given (m, sig) anyone can create (m, sig') with $\text{sig} \neq \text{sig}'$
- ⇒ miner can change sig in Tx and change $\text{TxID} = \text{SHA256}(\text{Tx})$
- ⇒ Tx issuer cannot tell what TxID is, until Tx is posted
- ⇒ leads to problems and attacks

Segregated witness: signature is moved to witness field in Tx
 $\text{TxID} = \text{Hash}(\text{Tx without witnesses})$

Transaction types: (2) P2SH: pay to script hash

(pre SegWit in 2017)

Let's payer specify a redeem script (instead of just pkhash)

Usage: payee publishes $\text{hash}(\text{redeem script}) \leftarrow \text{Bitcoin addr.}$
payer sends funds to that address

ScriptPK in UTXO: `HASH160 <H(redeem script)> EQUAL`

ScriptSig to spend: `<sig1> <sig2> ... <sign> <redeem script>`

payer can specify complex conditions for when UTXO can be spent

P2SH

Miner verifies:

- (1) $\langle \text{ScriptSig} \rangle \text{ScriptPK} = \text{true}$ \leftarrow payee gave correct script
- (2) $\text{ScriptSig} = \text{true}$ \leftarrow script is satisfied

Example P2SH: multisig

Goal: spending a UTXO requires t-out-of-n signatures

Redeem script for 2-out-of-3: (set by payer)

`<2> <PK1> <PK2> <PK3> <3> CHECKMULTISIG`

 hash gives P2SH address

ScriptSig to spend: (by payee)

`<0> <sig1> <sig3> <redeem script>`

END OF LECTURE

Next lecture: interesting scripts,
wallets, and how to manage crypto assets