共识协议与区块链系统

范磊

上海交通大学

比特币的技术基础

- 基于哈希函数的工作量证明算法
- 基于Markel树的交易记录结构
- 基于P2P网络的数据广播信道

利用哈希函数构造工作量证明

工作量证明（Proof of Work，PoW）

对于一个输入X，得到$Y=H(X)$。如果Y<T，那么X是一个有效的输入

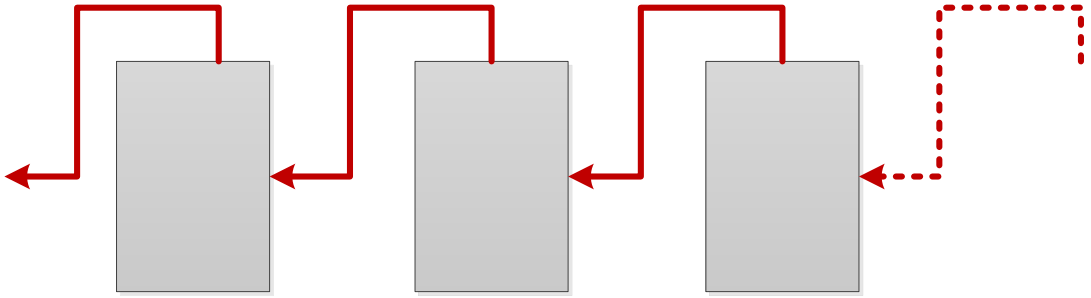
由于哈希函数是不可逆的，不能通过指定Y直接找到对应的X。因此找到有效输入的唯一方法是：

暴力搜索大量的X

找到一个有效的输入，意味着进行了大量的暴力搜索，也就是证明花费了相应的工作量

基于工作量证明构造区块链

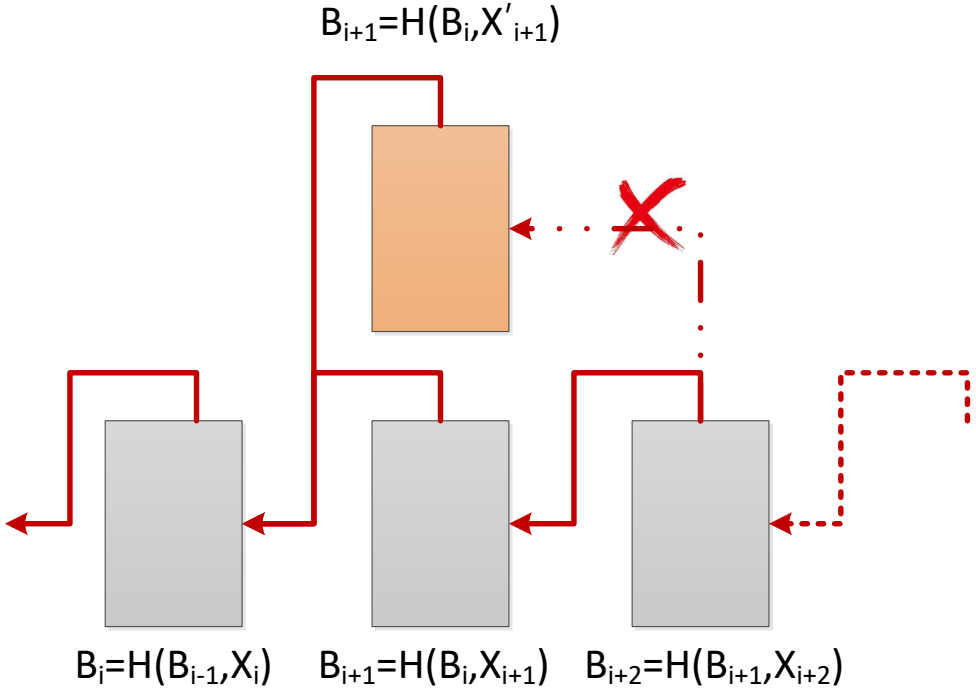
为了保证每次工作量证明都是全新的且不能更改，将每一个成功的工作量证明称为一个块，将工作量证明过程形成一个链条



$B_i=H(B_{i-1},X_i)$ $B_{i+1}=H(B_i,X_{i+1})$ $B_{i+2}=H(B_{i+1},X_{i+2})$

每个块的输入包含上一个块输出信息和一个随机数X，通过尝试随机数X形成新的块

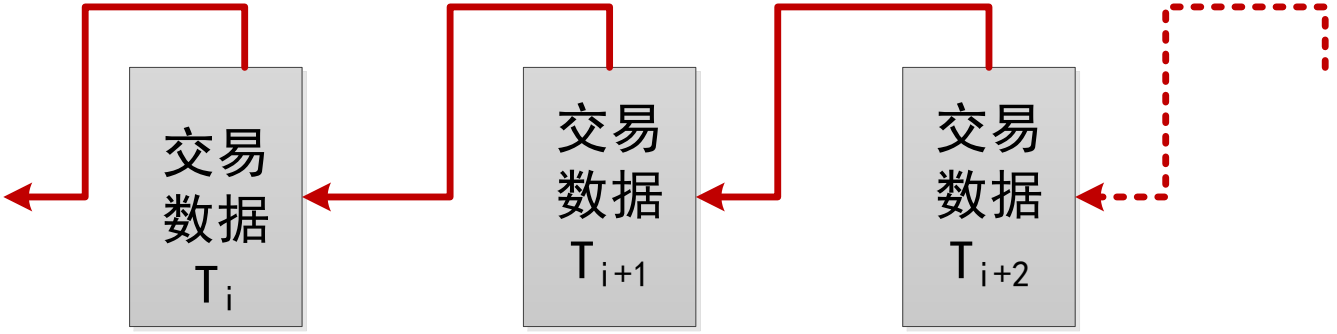
区块链的不可更改性



- 假如链中某一区块被修改了输入，由哈希函数抗碰撞性可知，其输出将发生变化，则下一区块输入将发生变化，链条被打破。
- 修改一个区块，则必须修改后续的所有区块。

交易数据与区块链的结合

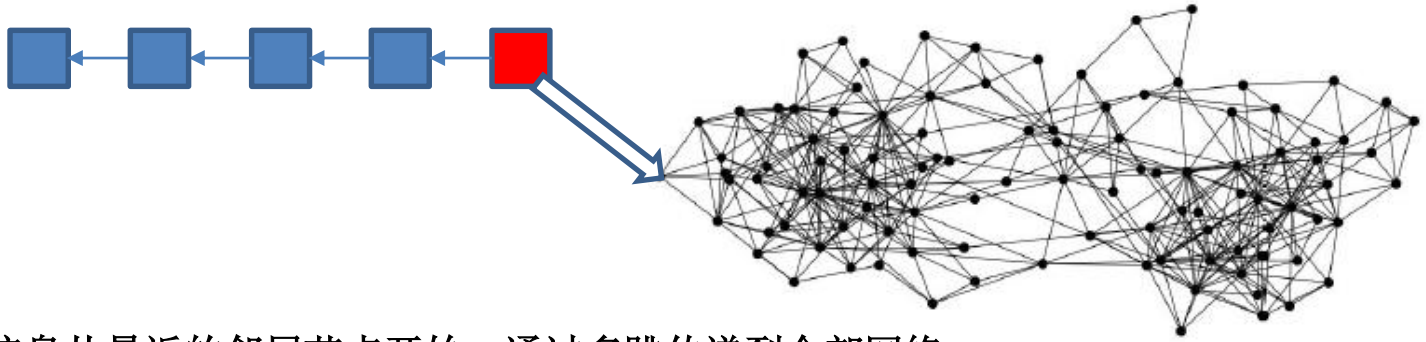
用户生成区块时，将自己当前网上的交易数据作为输入的一部分包含进去



- 包含进区块的交易数据不可更改，否则将破坏区块的链条连接
- 对交易数据的完整性校验可检验其是否满足区块的输出实现

区块链网络的通信方式

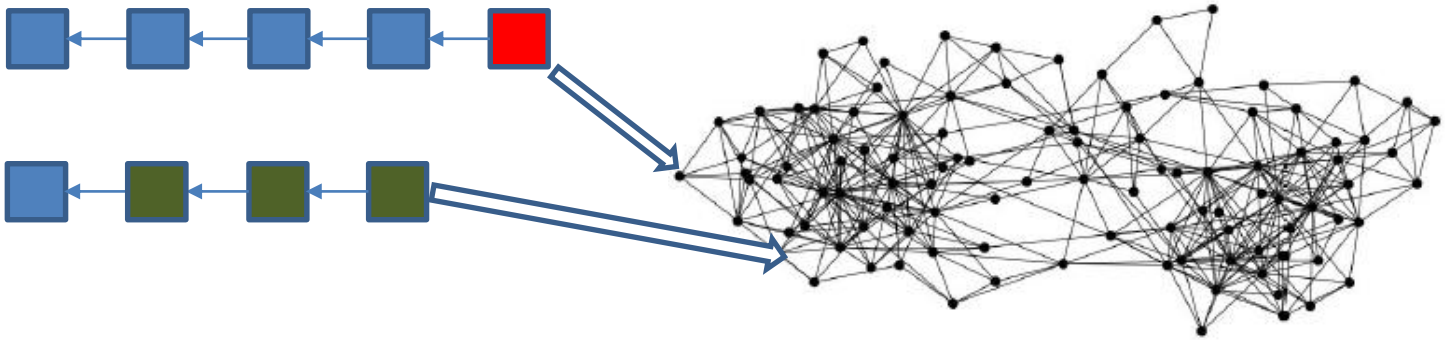
比特币及区块链使用基于P2P网络的广播通信，新生产区块以及交易数据通过节点间的泛洪传递到所有节点



- 信息从最近的邻居节点开始，通过多跳传递到全部网络
- 各个节点接收信息可能延迟、错序以及丢失部分信息

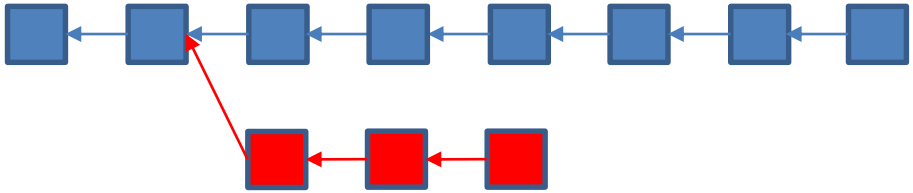
区块链竞争裁决

无中心分布式网络中，各个节点所产生与接收的区块是可能不同的。产生不同区块链后，用户选择长度最长的区块链。

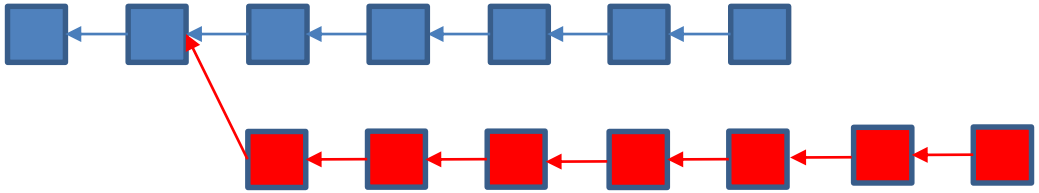


- 区块链的长度蕴含了不同的工作量，越长的区块链包含越多的工作量
- 由于网络传播的不同步，网络中不同的用户在同一时刻可能选择不同的区块链

诚实用户计算量占优



恶意用户计算量占优



比特币区块链结构

创世块

H

BH_1

version (4 bytes)
prev (32 bytes)
time (4 bytes)
bits (4 bytes)
nonce (4 bytes)
Tx root (32 bytes)

80 bytes

H

BH_2

prev

Tx root

H

BH_3

prev

Tx root

...

coinbase Tx

coinbase Tx

比特币区块链结构：区块头-80字节

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**

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

new block every ≈10 minutes.

区块出块信息

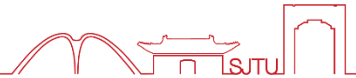
Tx data

Height	Mined	Miner	Size	#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

区块信息

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

交易例子(block 648493)　[2826 Tx]



COINBASE (Newly Generated Coins)

Tx0

0.00000000 BTC

1CK6KHY6MHgYvmRQ4PAafKYDrg1ejbH1cE　7.14047154 BTC

OP_RETURN　0.00000000 BTC

OP_RETURN　0.00000000 BTC

7.14047154 BTC

3PuJbxJS1pKxf8EdVR18yBkD1fPAbgUtyw　0.72333974 BTC

input

Tx1

0.00005800 BTC

1E5Ao1VUnA5BhffvXf2Xmud6avUgwkFnJv　0.00917379 BTC

bc1qr8k3e0vx06lpu3j7m858pa2ak9tyr56ttwvefk　0.61504199 BTC

bc1qdrxve8kua3yz5dgx6wf3u95ngh0d3e648...　0.09290152 BTC

14ZhjuXpQ5jCDjtAy7ZMu3hfEQCWewzLw7　0.00616444 BTC

outputs

0.72328174 BTC

17MWze4Z1uP1jnvqvj7SAnGtxcoVq11H8A　0.05000000 BTC

Tx2

0.00192000 BTC

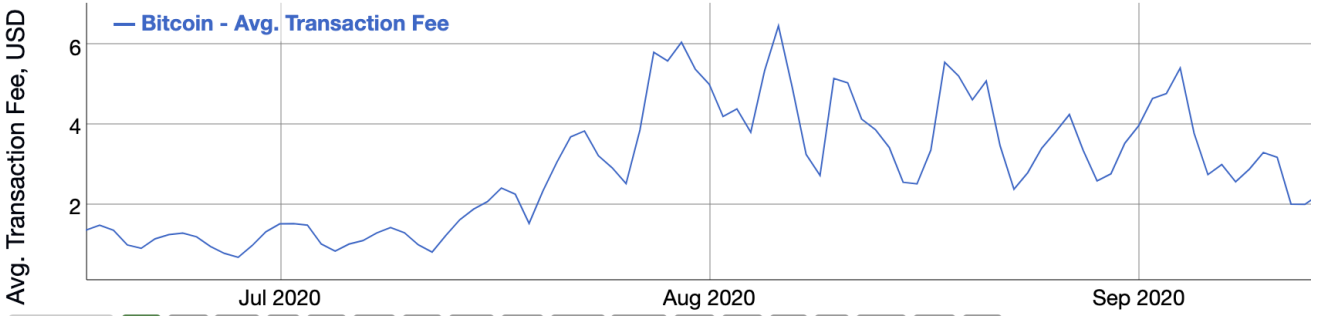
3G3C2RFQ8gsf77EQpdR4ZReChWFKEHhxVU　0.04808000 BTC

0.04808000 BTC

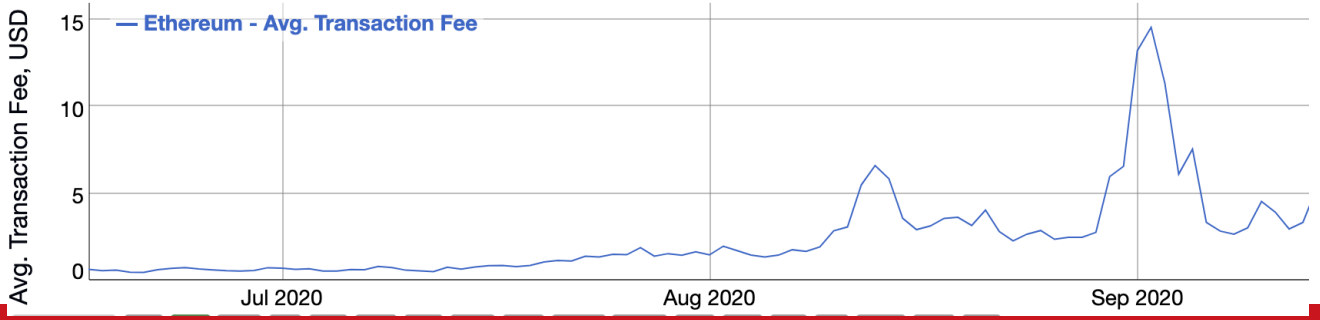
交易费将打入区块的Coinbase账号中

交易费

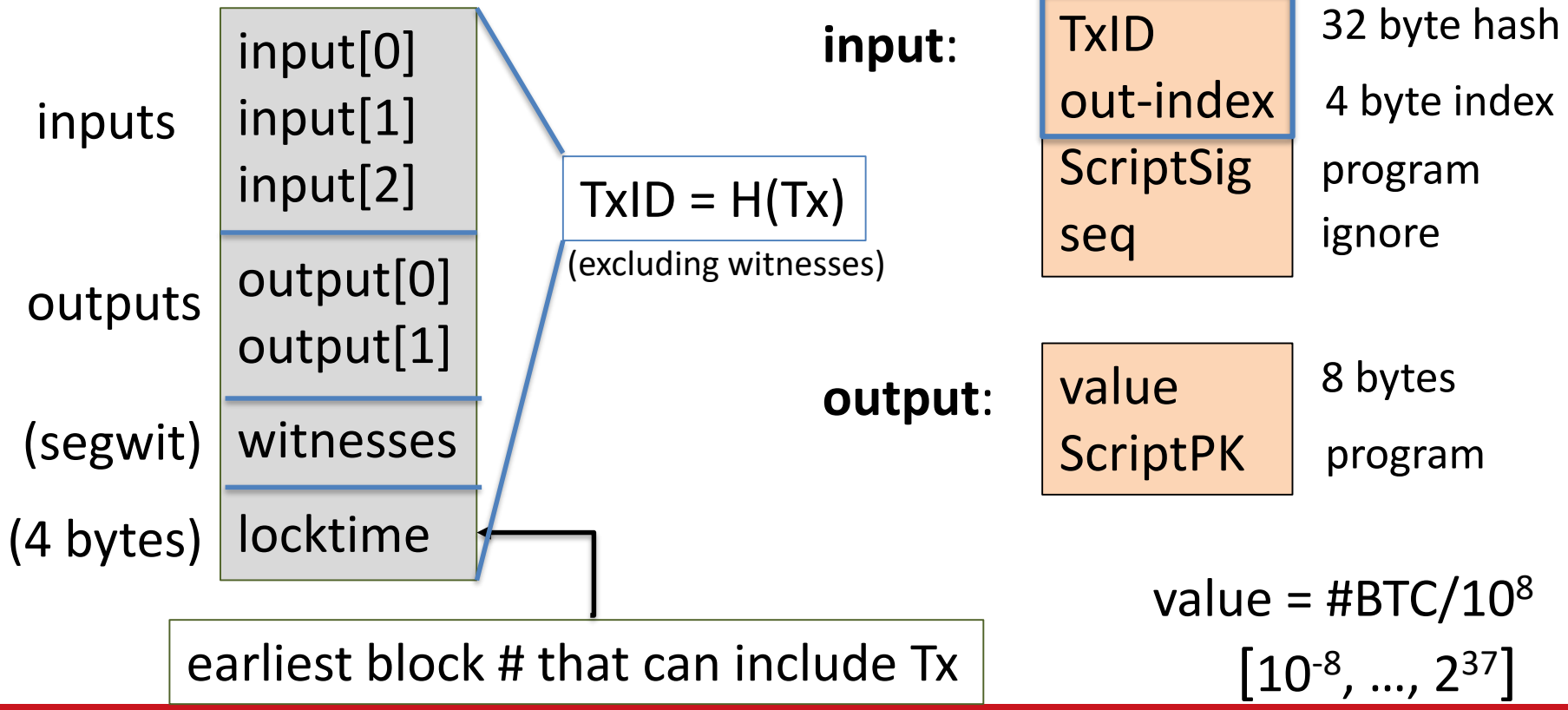
Bitcoin average Tx fees in USD



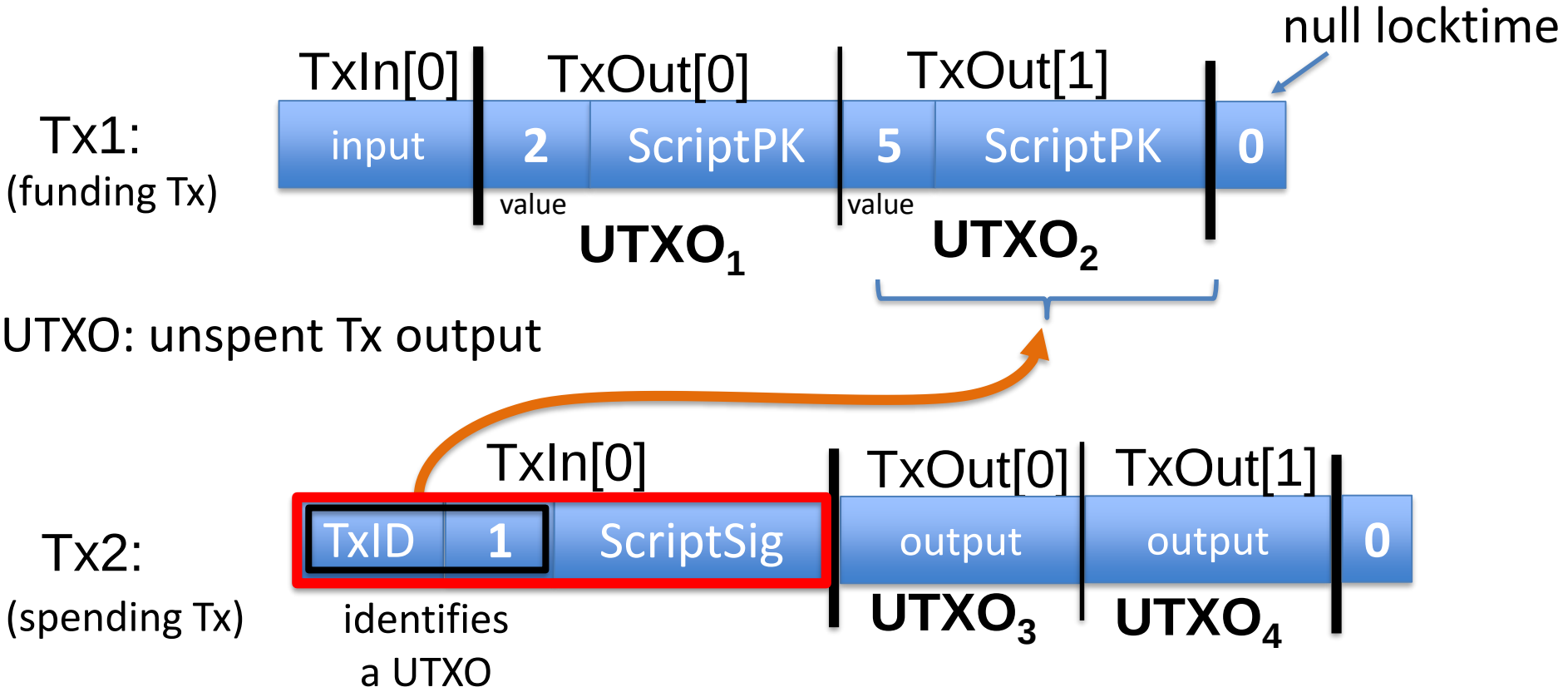
Ethereum average Tx fees in USD



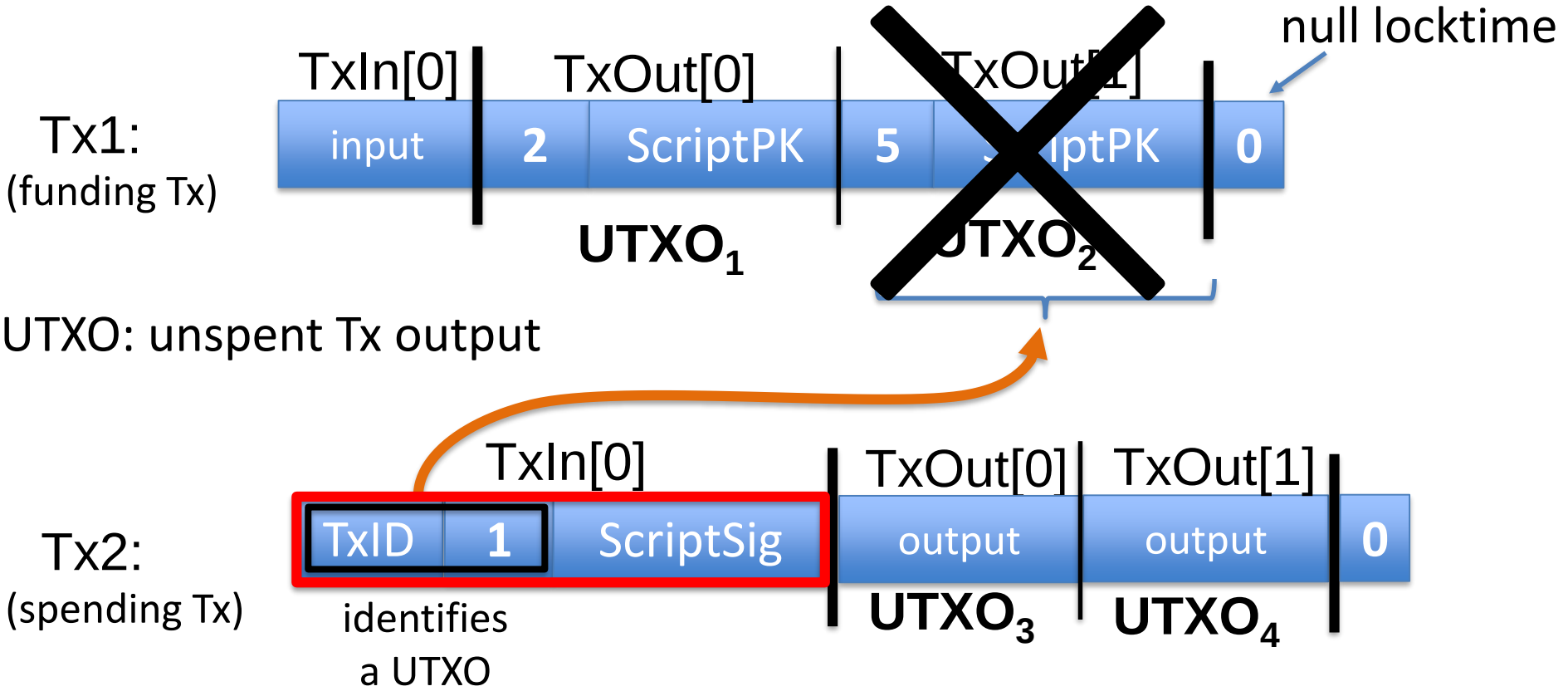
交易结构


inputs
input[0]
input[1]
input[2]
outputs
output[0]
output[1]
(segwit)
witnesses
(4 bytes)
locktime
TxID = H(Tx)
(excluding witnesses)
earliest block # that can include Tx
input:
TxID
out-index
ScriptSig
seq
32 byte hash
4 byte index
program
ignore
output:
value
ScriptPK
8 bytes
program
value = #BTC/10^8
[10^-8, …, 2^37]

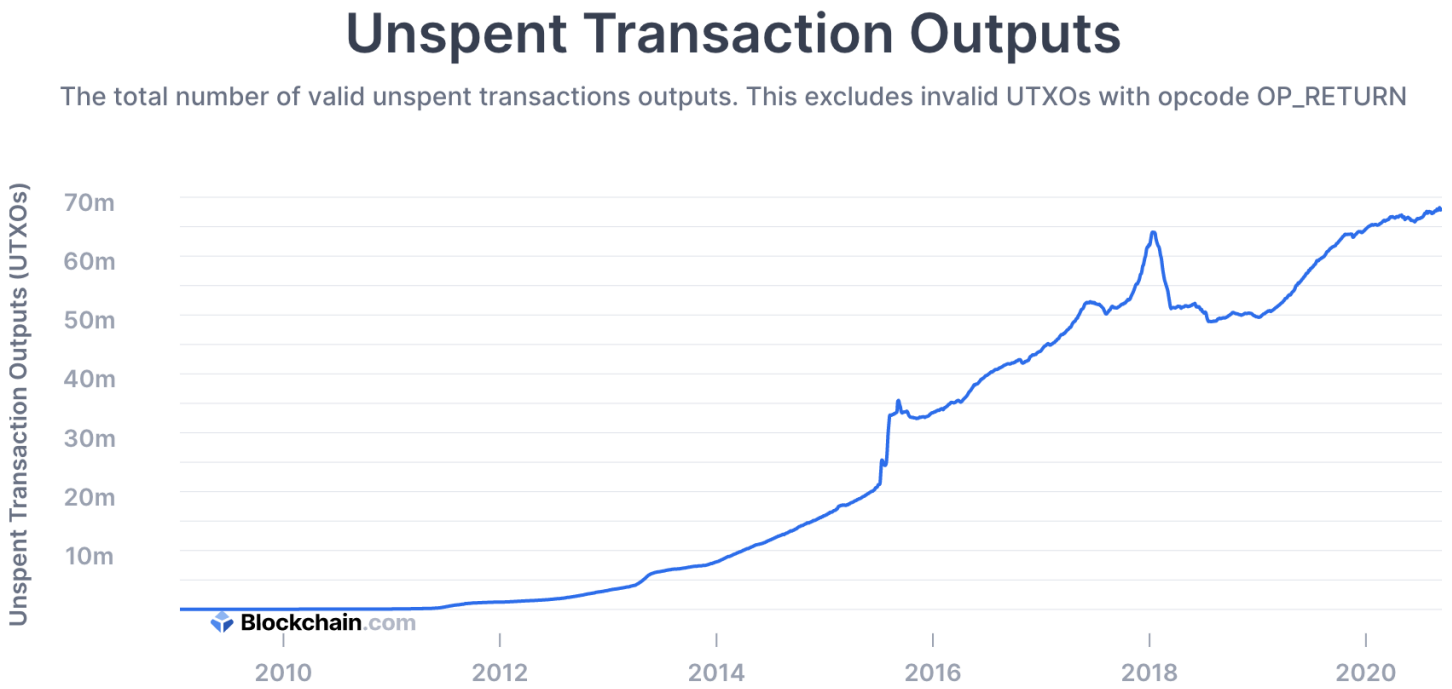
交易例子


null locktime
TxIn[0]
TxOut[0]
TxOut[1]
Tx1:
(funding Tx)
input
2
ScriptPK
5
ScriptPK
0
value
value
UTXO1
UTXO2
UTXO: unspent Tx output
TxIn[0]
TxOut[0]
TxOut[1]
Tx2:
(spending Tx)
TxID
1
ScriptSig
output
output
0
identifies
a UTXO
UTXO3
UTXO4

交易例子


Tx1:
(funding Tx)
TxIn[0]
TxOut[0]
TxOut[1]
null locktime
input
2
ScriptPK
5
ScriptPK
0
UTXO1
UTXO2
UTXO: unspent Tx output
Tx2:
(spending Tx)
TxIn[0]
TxOut[0]
TxOut[1]
TxID
1
ScriptSig
output
output
0
identifies
a UTXO
UTXO3
UTXO4

UTXO集合的数量



Sep. 2020: miners need to store ≈70M UTXOs in memory

交易验证

矿工验证交易的每一个输入:

1. 交易有效性代码 ScriptSig | ScriptPK 返回 true
2. TxID | index 在当前的UTXO 集合中
3. 输入总值 ≥ 输出总值

交易存入代码:
在何种条件下可以使用UTXO

交易花费代码:
证明UTXO使用条件满足

交易完成（写入区块）后,矿工将该UTXO从UTXO集合中删除

具体的交易结构

from UTXO
(Bitcoin script)

Value	0.05000000 BTC
Pkscript	OP_DUP OP_HASH160 45b21c8a0cb687d563342b6c729d31dab58e3a4e OP_EQUALVERIFY OP_CHECKSIG
Sigscript	304402205846cace0d73de82dfbdeba4d65b9856d7c1b1730eb401cf4906b2401a69bdc90220589d36d36be64e774c8796b96c011f29768191abeb7f56ba20ffb0351280860c01 03557c228b080703d52d72ead1bd93fc72f45c4ddb4c2b7a20c458e2d069c8dd9e

from TxInp[0]

比特币脚本

基于栈的计算模型，非图灵完备

比特币交易脚本支持的操作码样例:

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

比特币脚本

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

比特币脚本

4. arithmetic:

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

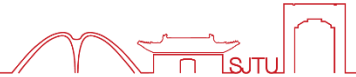
5. crypto:

   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

脚本的例子



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

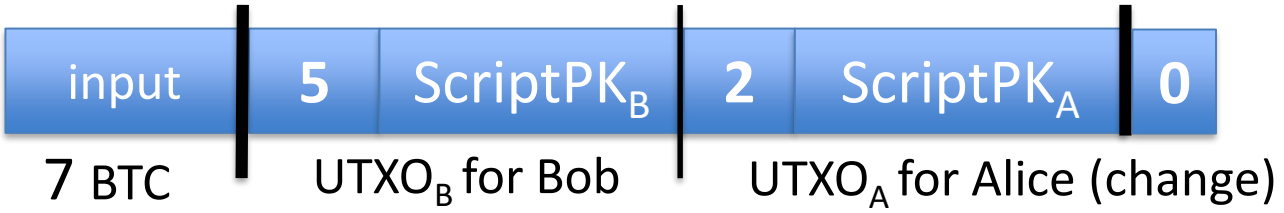
stack:		
	empty	init
	<sig> <pk>	push values
	<sig> <pk> <pk>	DUP
	<sig> <pk> <hash>	HASH256
	<sig> <pk> <hash> <pkhash>	push value
	<sig> <pk>	EQVERIFY
	1	CHECKSIG
⇒ 结束		verify(pk, Tx, sig)

交易类型: (1) P2PKH

pay to public key hash

Alice 向 Bob 支付 5 BTC:

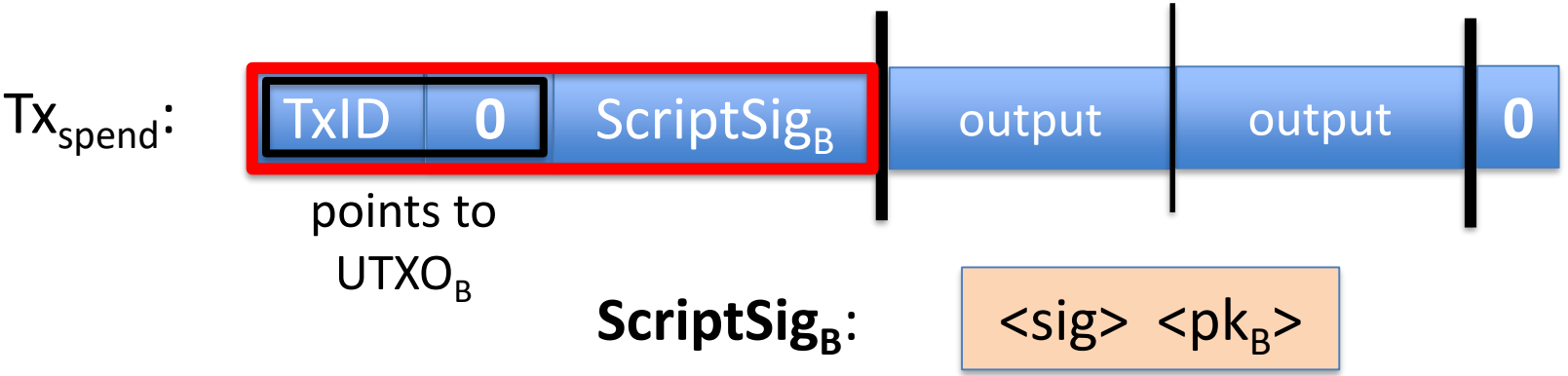
- step 1: Bob generates sig key pair $(pk_B, sk_B) \leftarrow Gen()$
- step 2: Bob computes his Bitcoin address as $Addr_B \leftarrow H(pk_B)$
- step 3: Bob sends $Addr_B$ to Alice
- step 4: Alice creates Tx:



$ScriptPK_B$: DUP HASH256 **<Addr$_B$>** EQVERIFY CHECKSIG

交易类型: (1) P2PKH

Bob 花费其拥有的UTXO: 生成交易Tx_{spend}



<sig> = Sign(sk_B, Tx) where Tx = (Tx_{spend} excluding all ScriptSigs) (SIGHASH_ALL)

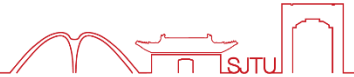
Miners validate that $ScriptSig_B$ | $ScriptPK_B$ returns true

交易类型: (1) P2PKH

scriptPubKey: OP_DUP OP_HASH160 <pubKeyHash> OP_EQUALVERIFY OP_CHECKSIG
scriptSig: <sig> <pubKey>

Stack	Script	Description
Empty.	<sig> <pubKey> OP_DUP OP_HASH160 <pubKeyHash> OP_EQUALVERIFY OP_CHECKSIG	scriptSig and scriptPubKey are combined.
<sig> <pubKey>	OP_DUP OP_HASH160 <pubKeyHash> OP_EQUALVERIFY OP_CHECKSIG	Constants are added to the stack.
<sig> <pubKey> <pubKey>	OP_HASH160 <pubKeyHash> OP_EQUALVERIFY OP_CHECKSIG	Top stack item is duplicated.
<sig> <pubKey> <pubHashA>	<pubKeyHash> OP_EQUALVERIFY OP_CHECKSIG	Top stack item is hashed.
<sig> <pubKey> <pubHashA> <pubKeyHash>	OP_EQUALVERIFY OP_CHECKSIG	Constant added.
<sig> <pubKey>	OP_CHECKSIG	Equality is checked between the top two stack items.
true	Empty.	Signature is checked for top two stack items.

P2PKH总结



- Alice 在UTXO中指定接收者的pk
- 接收者的pk不会被泄露，直到该UTXO花费掉
- 接收者的地址不能被更改

交易类型: (2) P2SH

pay to script hash

转账人指定对付的脚本(不仅仅是一个公钥地址)

用法: 收款人发布兑换脚本的hash← 特殊的Bitcoint地址.
付款人将资金转入此特殊地址

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

ScriptSig to spend: $<sig_1>$ $<sig_2>$ … $<sig_n>$ <redeem script>

付款人可以为何时可以使用UTXO指定复杂的条件

交易类型: (2) P2SH

矿工验证:

(1) <ScriptSig> ScriptPK = true ←收款人提供正确的脚本

(2) ScriptSig = true ← 脚本条件满足

交易类型: (2) P2SH例子——multisig

目标: 花费一个 UTXO 要求 t-out-of-n 签名

兑换脚本 2-out-of-3: (付款人设置)

<2> <PK_1> <PK_2> <PK_3> <3> CHECKMULTISIG

hash gives P2SH address

ScriptSig to spend: (收款人) <0> <sig2> <sig3> <redeem script>

谢谢