Blockchains & Cryptocurrencies

Bitcoin Mechanics

Instructor: Abhishek Jain
Johns Hopkins University - Spring 2021

*Many slides based on NBFMG

Housekeeping

- Assignment 1 was released on 2/6; due on 2/22 end of day
- **Project groups and project idea due on 2/26 end of day**
    - List of project ideas and submission format will be up on course website tomorrow

Last Time

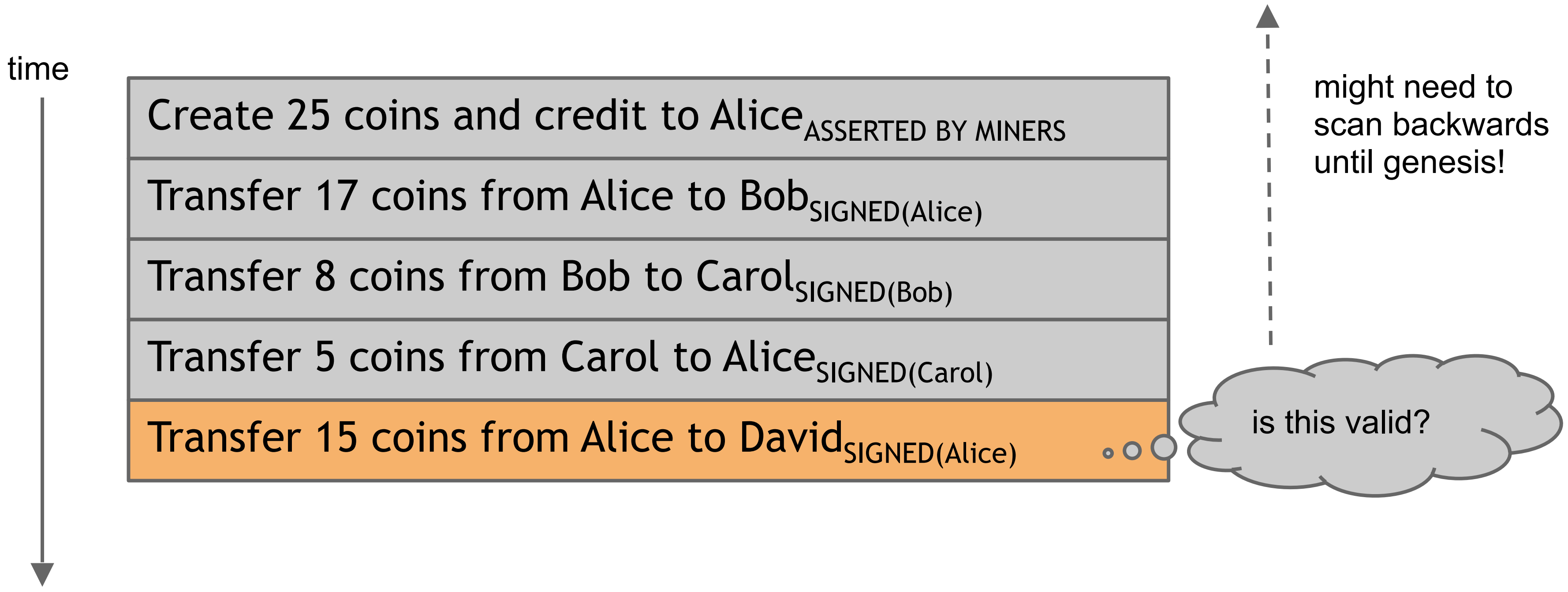
- Proof of Work (PoW) puzzles
- Consensus mechanism in Bitcoin using PoW
- Difficulty Parameter Adjustability
- Longest Chain Rule

Today

- Bitcoin Transaction Format
- Simple Smart-Contracts in Bitcoin

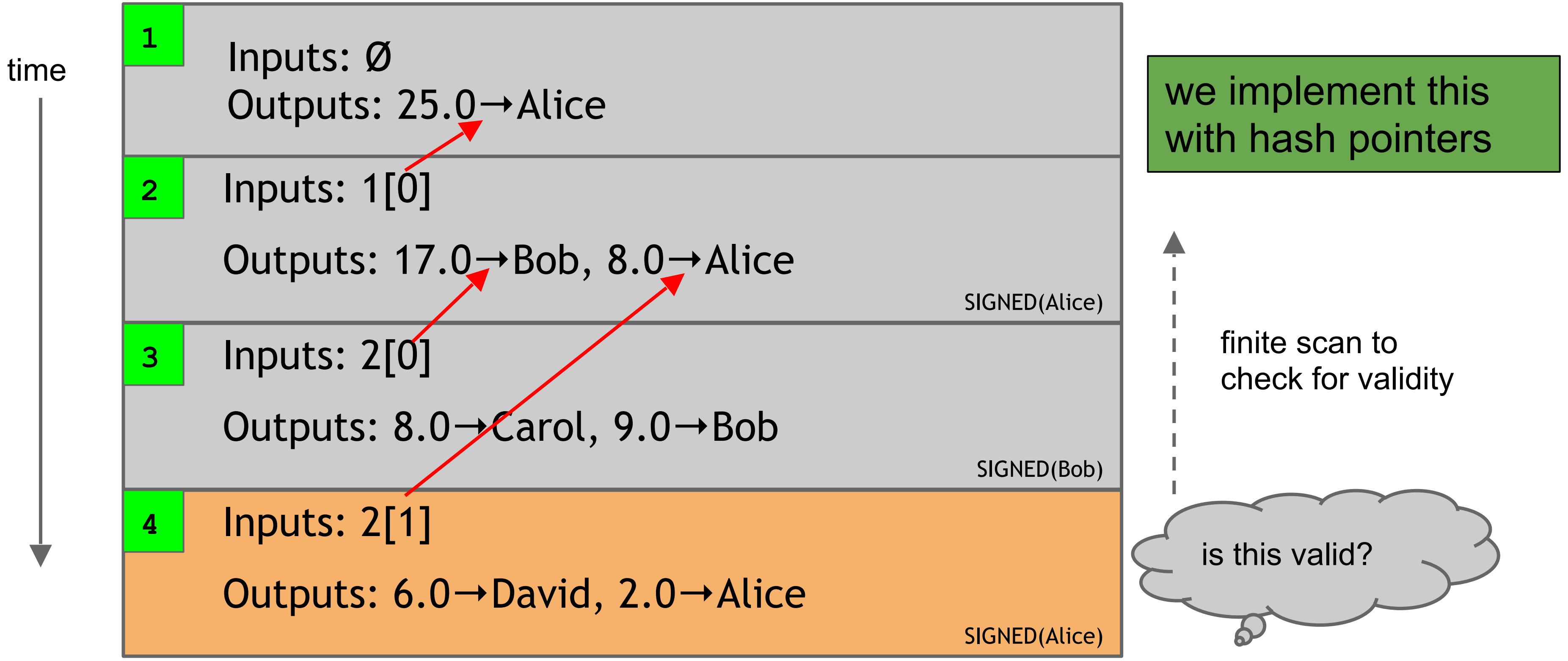
Bitcoin transactions

An <u>account-based</u> ledger (*not* Bitcoin)


time
Create 25 coins and credit to Alice$_{\text{ASSERTED BY MINERS}}$
Transfer 17 coins from Alice to Bob$_{\text{SIGNED(Alice)}}$
Transfer 8 coins from Bob to Carol$_{\text{SIGNED(Bob)}}$
Transfer 5 coins from Carol to Alice$_{\text{SIGNED(Carol)}}$
Transfer 15 coins from Alice to David$_{\text{SIGNED(Alice)}}$
is this valid?
might need to scan backwards until genesis!

SIMPLIFICATION: only one transaction per block

A transaction-based ledger (Bitcoin)



1
Inputs: Ø
Outputs: 25.0→Alice

2
Inputs: 1[0]
Outputs: 17.0→Bob, 8.0→Alice
SIGNED(Alice)

3
Inputs: 2[0]
Outputs: 8.0→Carol, 9.0→Bob
SIGNED(Bob)

4
Inputs: 2[1]
Outputs: 6.0→David, 2.0→Alice
SIGNED(Alice)

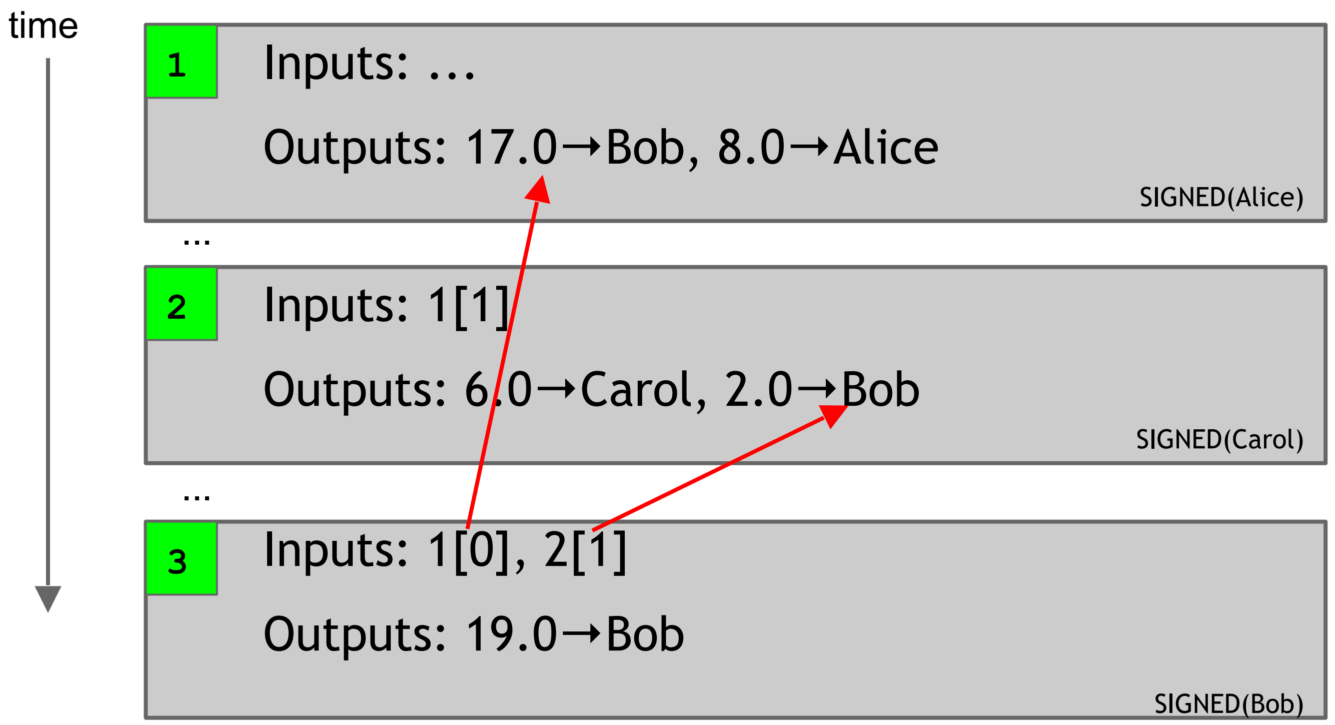
time
we implement this with hash pointers
finite scan to check for validity
is this valid?

SIMPLIFICATION: only one transaction per block

Referencing Transactions

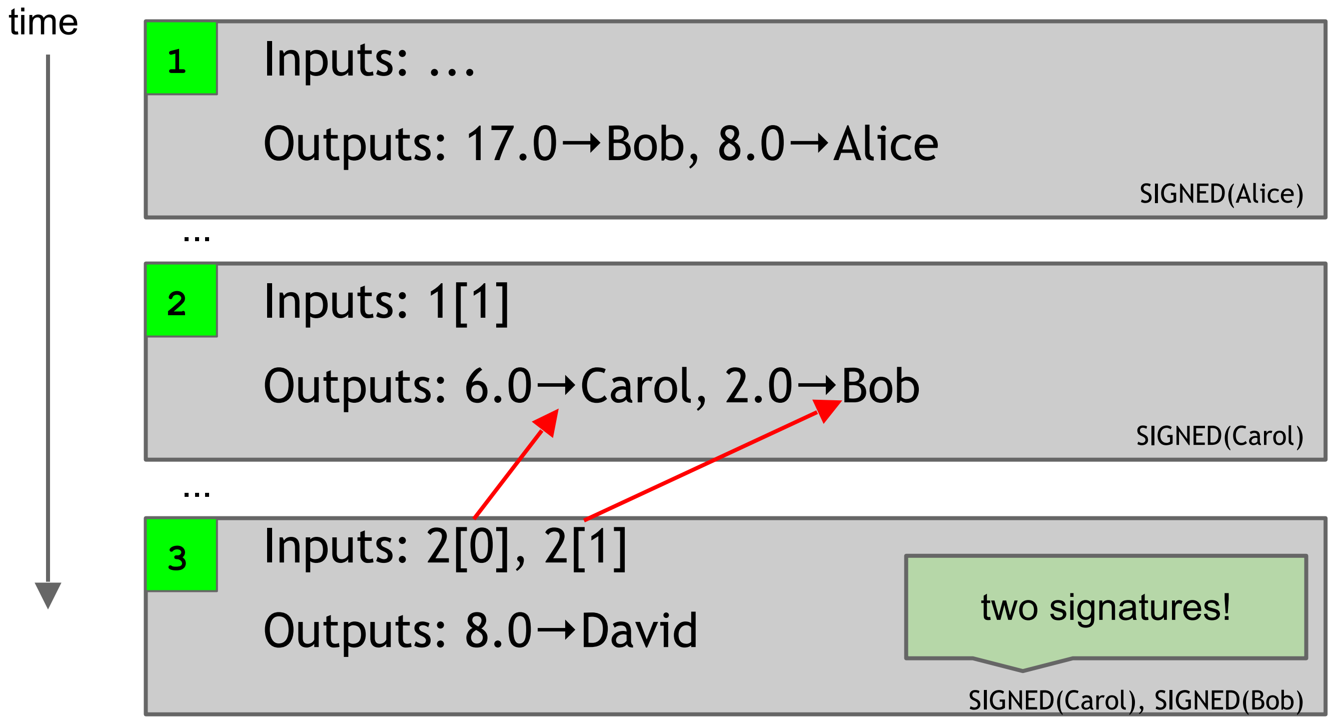
- Hash pointers for transactions
- Within a transaction, refer to a particular output via serial numbers

Merging value



SIMPLIFICATION: only one transaction per block

Joint payments



SIMPLIFICATION: only one transaction per block

The real deal: a classical Bitcoin transaction

metadata

input(s)

output(s)

```
{
    "hash":"5a42590fbe0a90ee8e8747244d6c84f0db1a3a24e8f1b95b10c9e050990b8b6b",
    "ver":1,
    "vin_sz":2,
    "vout_sz":1,
    "lock_time":0,
    "size":404,
    "in":[
      {
        "prev_out":{
          "hash":"3be4ac9728a0823cf5e2deb2e86fc0bd2aa503a91d307b42ba76117d79280260",
          "n":0
        },
            "scriptSig":"30440..."
      },
      {
        "prev_out":{
          "hash":"7508e6ab259b4df0fd5147bab0c949d81473db4518f81afc5c3f52f91ff6b34e",
          "n":0
        },
        "scriptSig":"3f3a4ce81...."
      }
    ],
    "out":[
      {
        "value":"10.12287097",
        "scriptPubKey":"OP_DUP OP_HASH160 69e02e18b5705a05dd6b28ed517716c894b3d42e OP_EQUALVERIFY OP_CHECKSIG"
      }
    ]
}
```

The real deal: transaction metadata

```
{
   "hash":"5a42590...b8b6b",       transaction hash
     "ver":1,                       housekeeping
     "vin_sz":2,
     "vout_sz":1,
     "lock_time":0,                 "not valid before"
     "size":404,                    housekeeping
...
}
```

more on this later...

The real deal: transaction inputs

```
"in":[
  {
    "prev_out":{
      "hash":"3be4...80260",
      "n":0
    },
 "scriptSig":"30440....3f3a4ce81"
  },
...
],
```

previous transaction

signature

(more inputs)

The real deal: transaction outputs

```
"out":[
     {
       "value":"10.12287097",
       "scriptPubKey":"OP_DUP OP_HASH160 69e...3d42e
OP_EQUALVERIFY OP_CHECKSIG"
     },
          ...
   ]
```

output value

recipient address??

more on this soon...

(more outputs)

Bitcoin scripts

Output "addresses" are really *scripts*

```
OP_DUP
OP_HASH160
69e02e18...
OP_EQUALVERIFY OP_CHECKSIG
```

Input "addresses" are *also* scripts

scriptSig

```
30440220...
0467d2c9...
```

scriptPubKey

```
OP_DUP
OP_HASH160
69e02e18...
OP_EQUALVERIFY OP_CHECKSIG
```

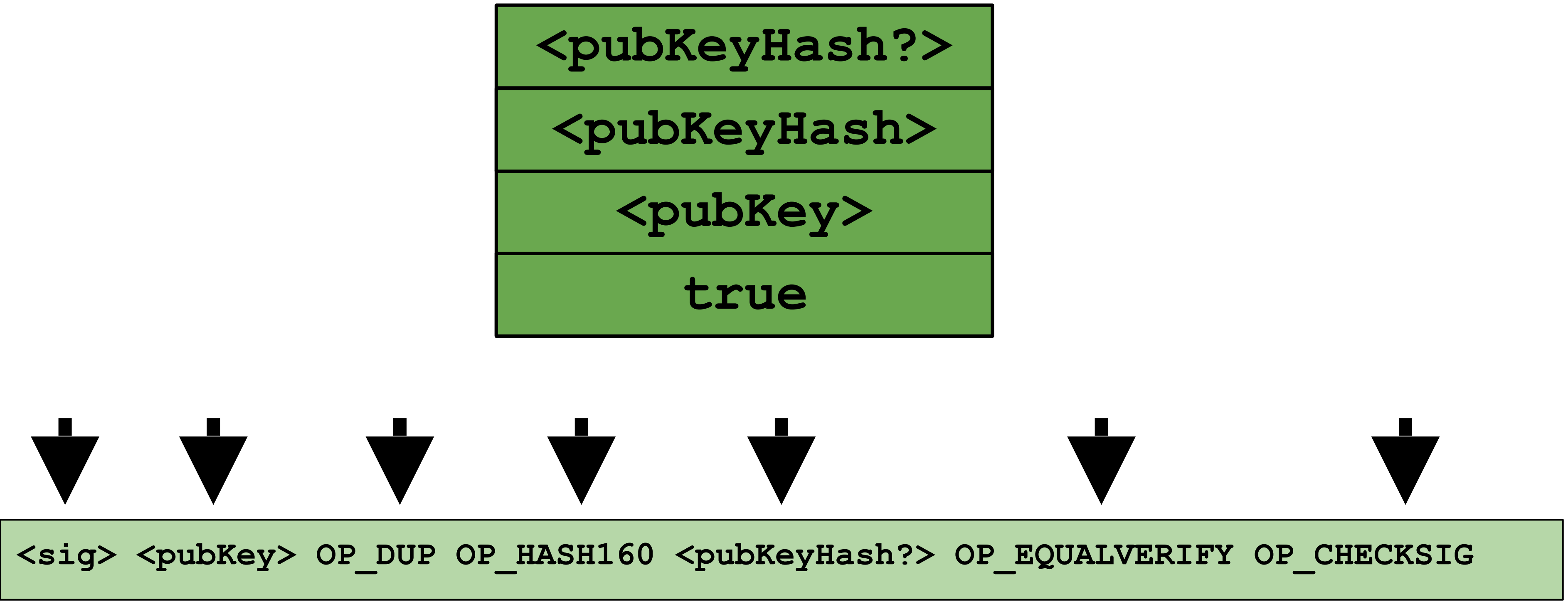
TO VERIFY: Concatenated script must execute completely with no errors

Bitcoin scripting language (“Script”)

Design goals

- Built for Bitcoin
- Simple, compact
- Support for cryptography
- Stack-based
- Limits on time/memory
- No looping

Bitcoin script execution example



```
<sig> <pubKey> OP_DUP OP_HASH160 <pubKeyHash?> OP_EQUALVERIFY OP_CHECKSIG
```

Bitcoin script instructions

256 opcodes total (15 disabled, 75 reserved)

- Arithmetic
- If/then
- Logic/data handling
- Crypto!
  - Hashes
  - Signature verification
  - Multi-signature verification

OP_CHECKMULTISIG

- Built-in support for joint signatures
- Specify ***n*** public keys
- Specify ***t***
- Verification requires ***t*** signatures

Bitcoin scripts in practice (“original”)

- Most nodes whitelist known scripts
- 99.9% are simple signature checks
- ~0.01% are MULTISIG
- ~0.01% are Pay-to-Script-Hash

More on this soon

- Remainder are errors, proof-of-burn

* numbers from NBFMG and slightly out of date

Proof-of-burn

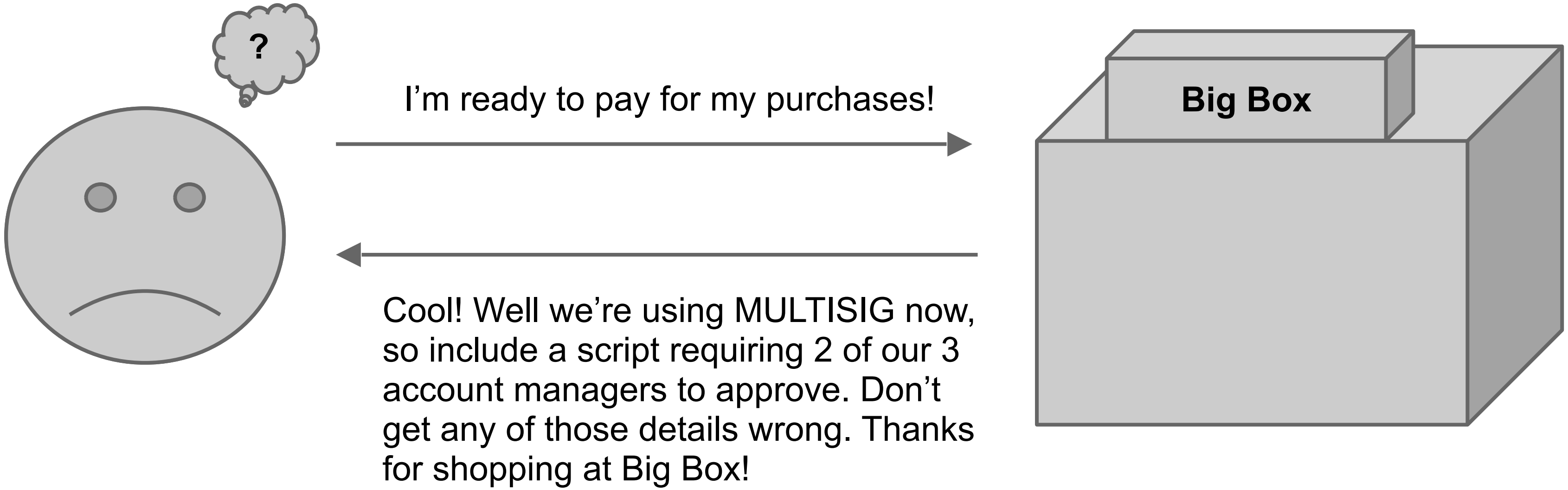
nothing’s going to redeem that ☹

OP_RETURN
<arbitrary data>

Proof-of-burn: Applications

- Can be used to publish arbitrary data on the blockchain (e.g., timestamping a document)
- Bootstrap Altcoins by requiring people to destroy bitcoins in order to get new altcoins

Should senders specify scripts?



Cool! Well we're using MULTISIG now, so include a script requiring 2 of our 3 account managers to approve. Don't get any of those details wrong. Thanks for shopping at Big Box!

Idea: use the hash of redemption script

<

< <signature>

O

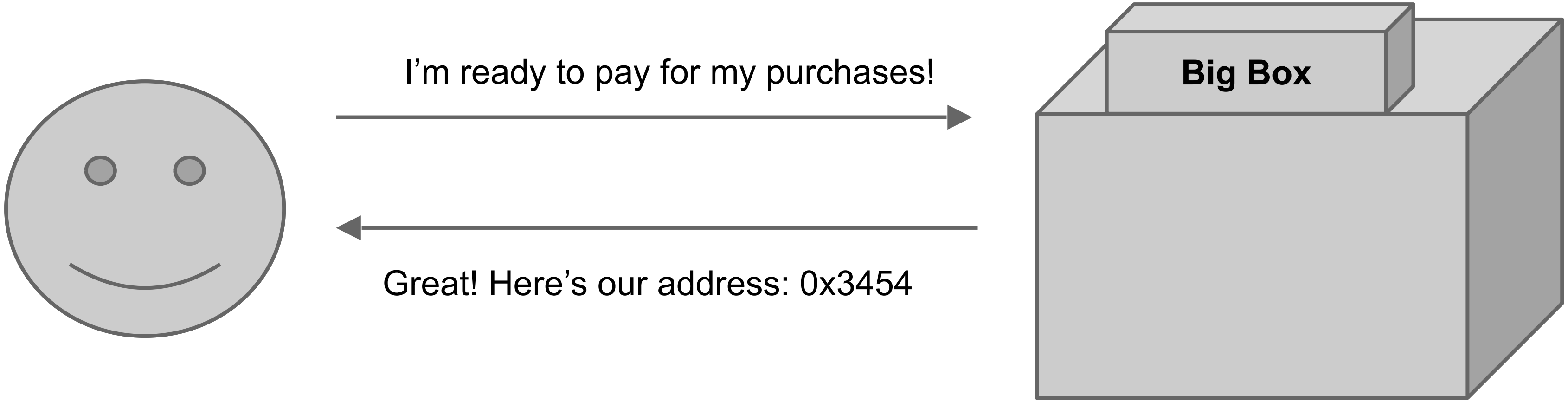
<pubkey>

< OP_CHECKSIG

O

“Pay to Script Hash”

Pay to script hash


I'm ready to pay for my purchases!
Great! Here's our address: 0x3454
Big Box

Applications of Bitcoin scripts

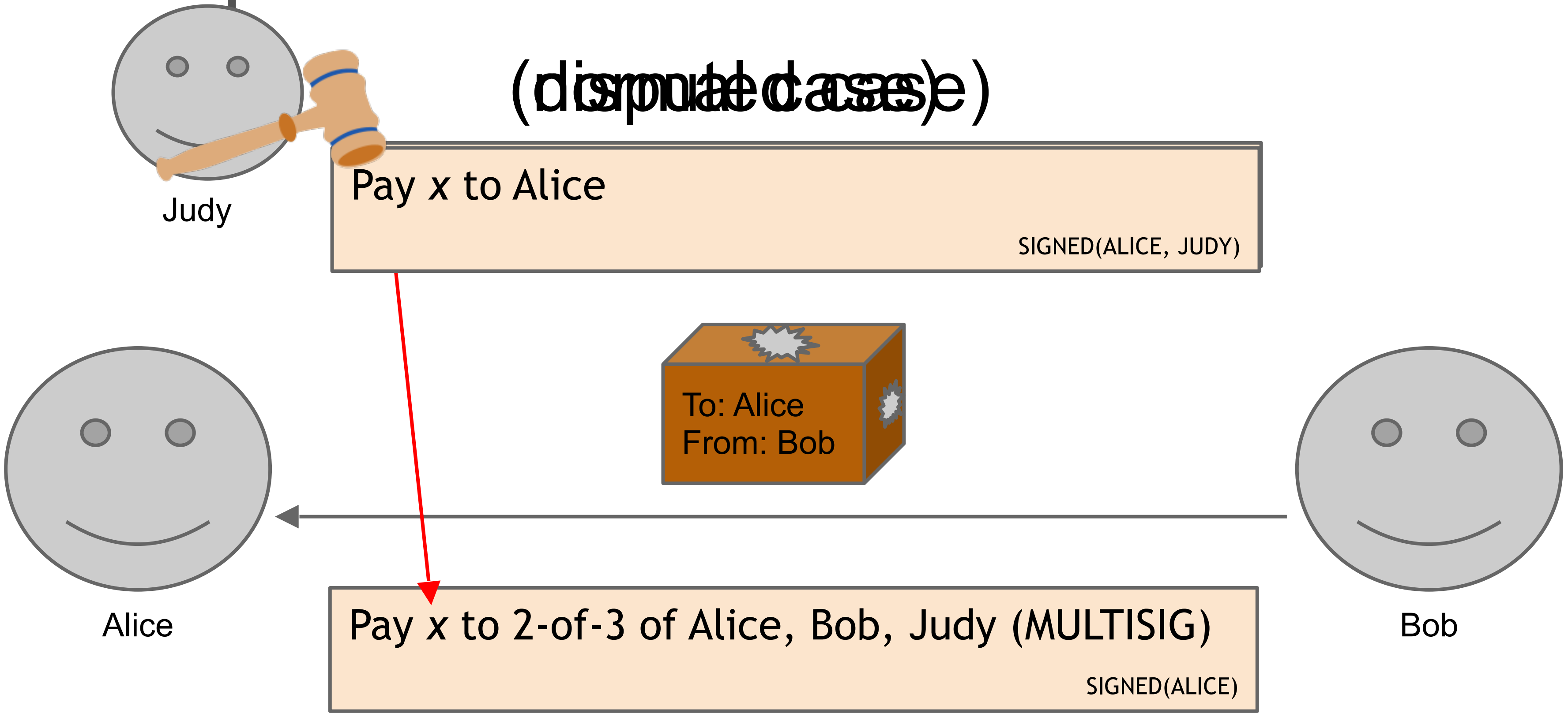
Example 1: “Fair” transactions

- Problem: Alice wants to buy a product from an online vendor Bob
- Alice doesn’t want to pay until after Bob ships
- Bob doesn’t want to ship until after Alice pays

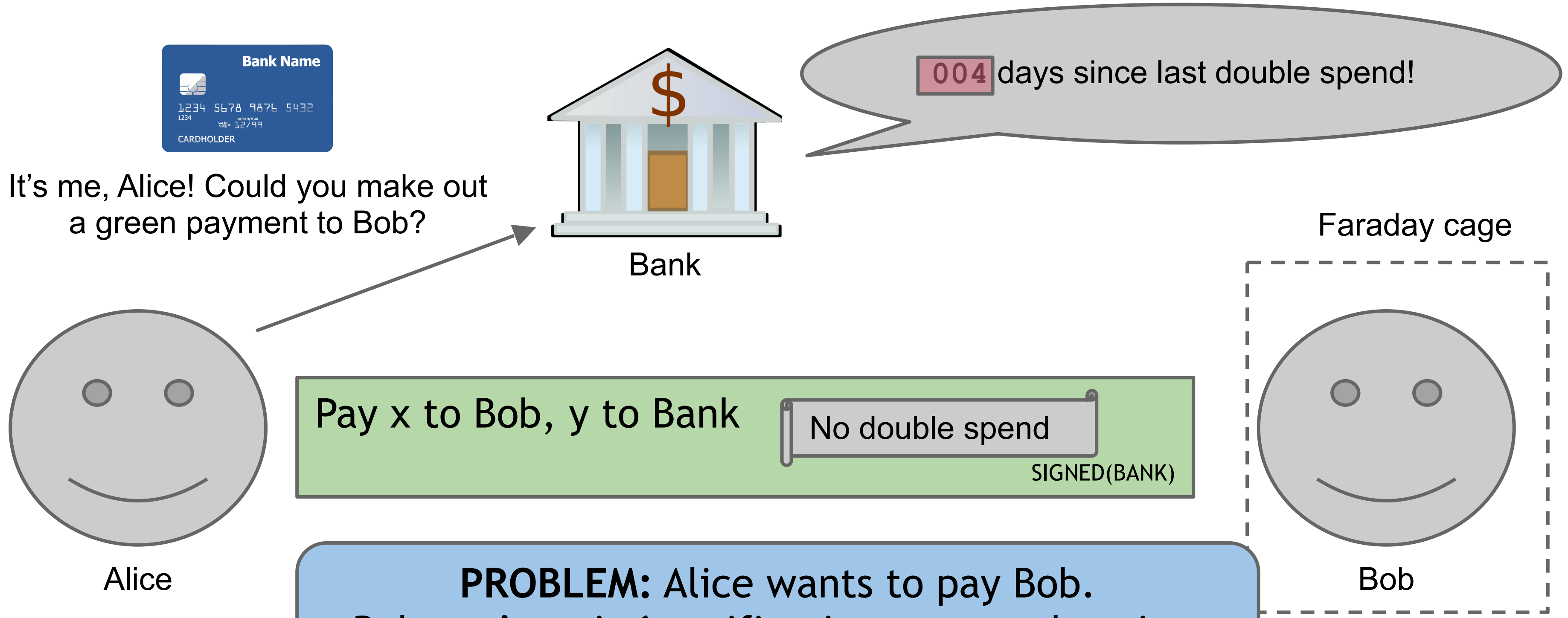
Example 1: Fair transactions via Escrow

(disputed case)

(normal case)



Example 2: Green addresses



PROBLEM: Alice wants to pay Bob.
Bob can't wait 6 verifications to guard against double-spends, or is offline completely.

Example 3: Micro-payments

- Pay-as-you-go WIFI: Alice wants to pay WIFI provider (Bob) for each minute of WIFI service. But she doesn't want to incur a transaction fee for every minute
- Similarly, pay-as-you-go online subscriptions
- Ad-free websites

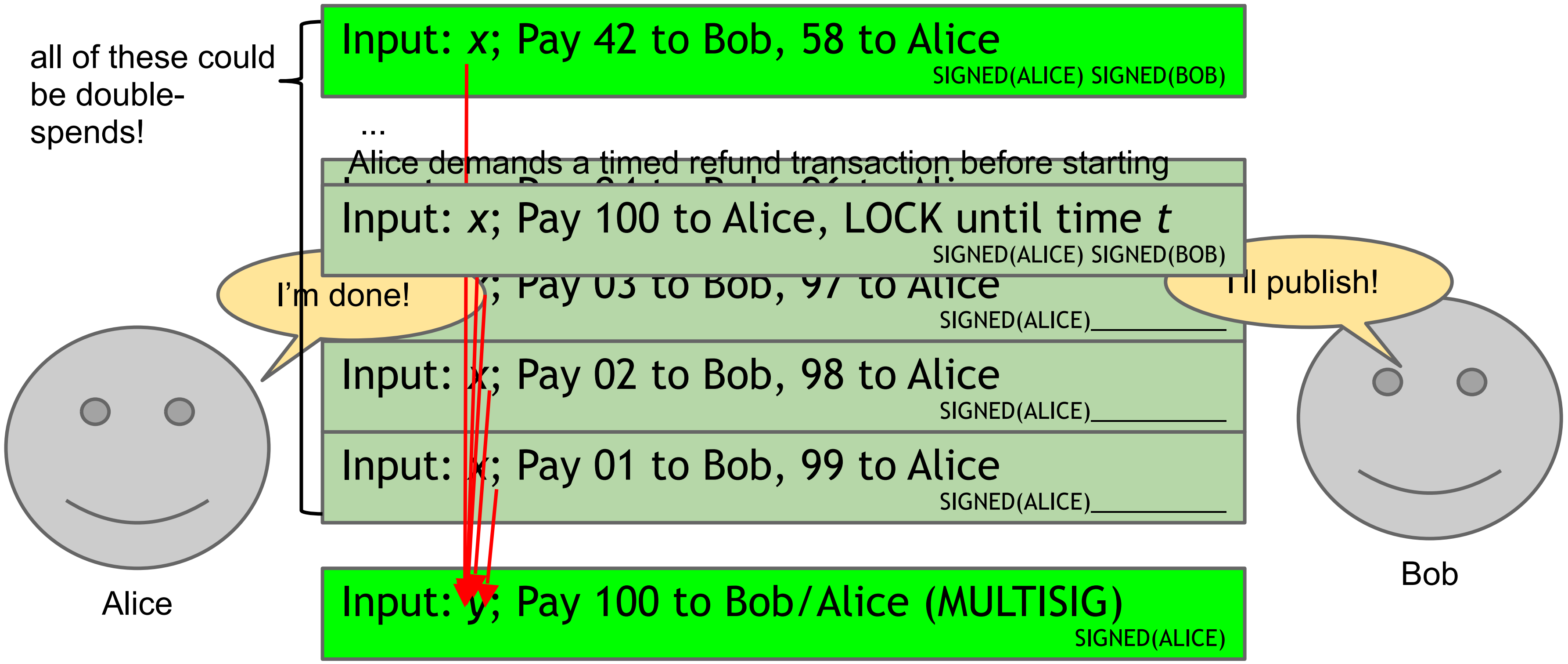
Example 3: Micro-payments with Bitcoin

- Main Idea: Instead of doing several transactions, do a single transaction for total payment (and thus incur only a single transaction fee)
- *How to implement it?*

Example 3: Micro-payments with Bitcoin

What if Bob never signs??

Alice demands a timed refund transaction before starting



Input: *x*; Pay 42 to Bob, 58 to Alice
SIGNED(ALICE) SIGNED(BOB)

...

Input: *x*; Pay 100 to Alice, LOCK until time *t*
SIGNED(ALICE) SIGNED(BOB)

Input: *x*; Pay 03 to Bob, 97 to Alice
SIGNED(ALICE)__________

Input: *x*; Pay 02 to Bob, 98 to Alice
SIGNED(ALICE)__________

Input: *x*; Pay 01 to Bob, 99 to Alice
SIGNED(ALICE)__________

Input: *y*; Pay 100 to Bob/Alice (MULTISIG)
SIGNED(ALICE)

lock_time

```
{
    "hash":"5a42590...b8b6b",
      "ver":1,
      "vin_sz":2,
      "vout_sz":1,
      "lock_time":315415,
      "size":404,
...
}
```

Block index or real-world timestamp before which this transaction can’t be published

Micro-payments from Cryptocurrencies

Some recent constructions, that achieve better properties

- Pass, shelat [CCS'16]
- Chiesa, Green, Liu, Miao, Miers, Mishra [EUROCRYPT'17]

More advanced scripts

- Fair multiplayer lotteries and fair multiparty computation [Andrychowichz-Dziembowski-Malinowski-Mazurek, S&P'14; Bentov-Kumaresan, CRYPTO'14]
- Hash pre-image challenges

“Smart contracts”

Later: More powerful smart contracts with Ethereum
(Turing-complete scripting language)