Blockchains & Cryptocurrencies

Mechanics of Bitcoin II

Instructor: Matthew Green
Johns Hopkins University - Fall 2024

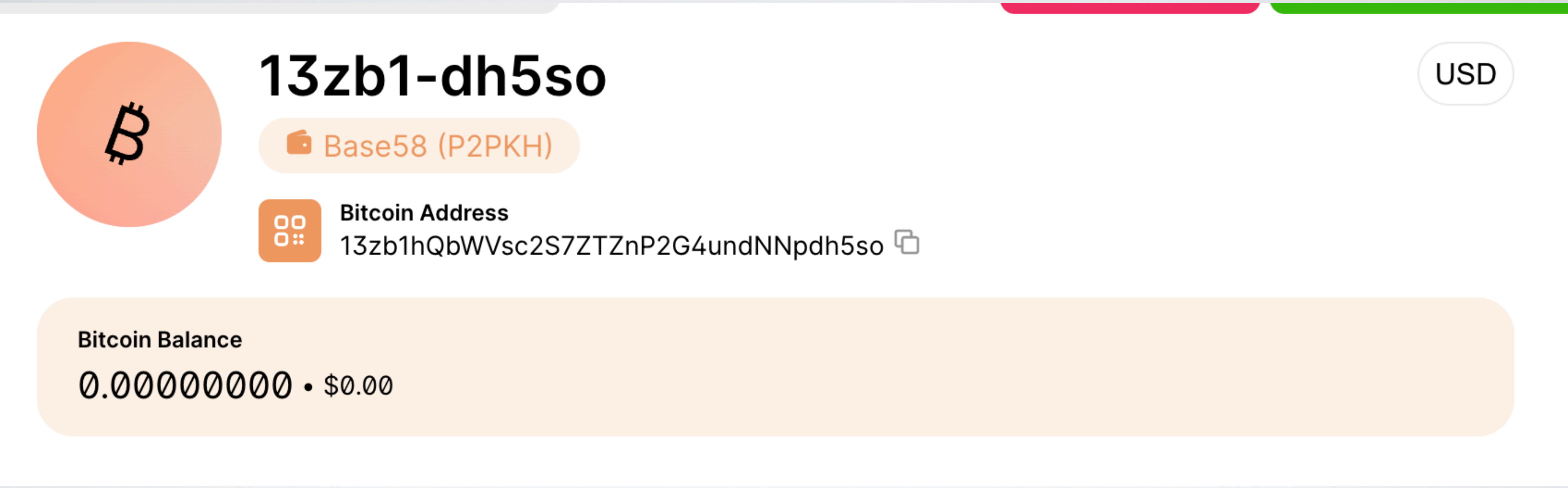
Many slides based on NBFMG

Housekeeping

- Pace of material?
- Office hours
- **Project groups and proposal is due 9/30 end of day**

News?

News?



13zb1-dh5so

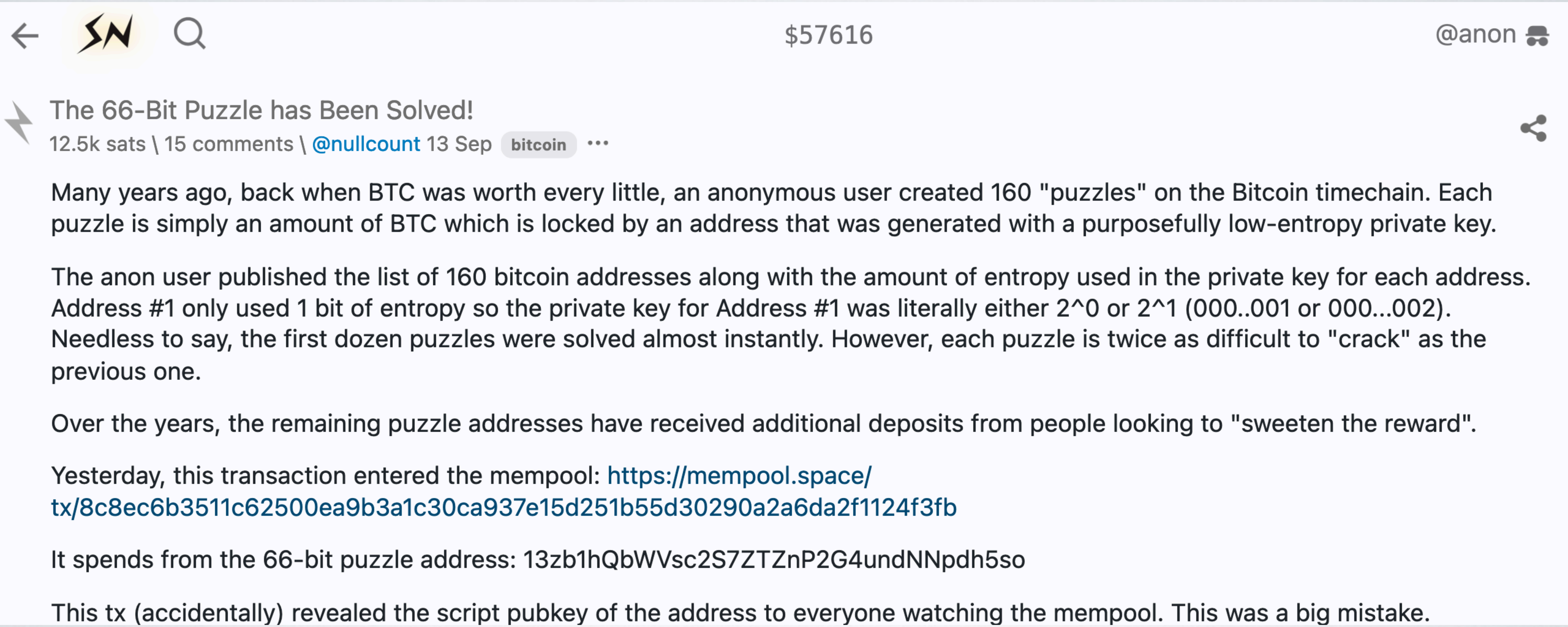
Base58 (P2PKH)

USD

Bitcoin Address
13zb1hQbWVsc2S7ZTZnP2G4undNNpdh5so

Bitcoin Balance
0.00000000 • $0.00

News?



$57616

@anon

The 66-Bit Puzzle has Been Solved!

12.5k sats \ 15 comments \ @nullcount 13 Sep bitcoin •••

Many years ago, back when BTC was worth every little, an anonymous user created 160 "puzzles" on the Bitcoin timechain. Each puzzle is simply an amount of BTC which is locked by an address that was generated with a purposefully low-entropy private key.

The anon user published the list of 160 bitcoin addresses along with the amount of entropy used in the private key for each address. Address #1 only used 1 bit of entropy so the private key for Address #1 was literally either 2^0 or 2^1 (000..001 or 000...002). Needless to say, the first dozen puzzles were solved almost instantly. However, each puzzle is twice as difficult to "crack" as the previous one.

Over the years, the remaining puzzle addresses have received additional deposits from people looking to "sweeten the reward".

Yesterday, this transaction entered the mempool: https://mempool.space/tx/8c8ec6b3511c62500ea9b3a1c30ca937e15d251b55d30290a2a6da2f1124f3fb

It spends from the 66-bit puzzle address: 13zb1hQbWVsc2S7ZTZnP2G4undNNpdh5so

This tx (accidentally) revealed the script pubkey of the address to everyone watching the mempool. This was a big mistake.

Review

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

Proof-of-burn

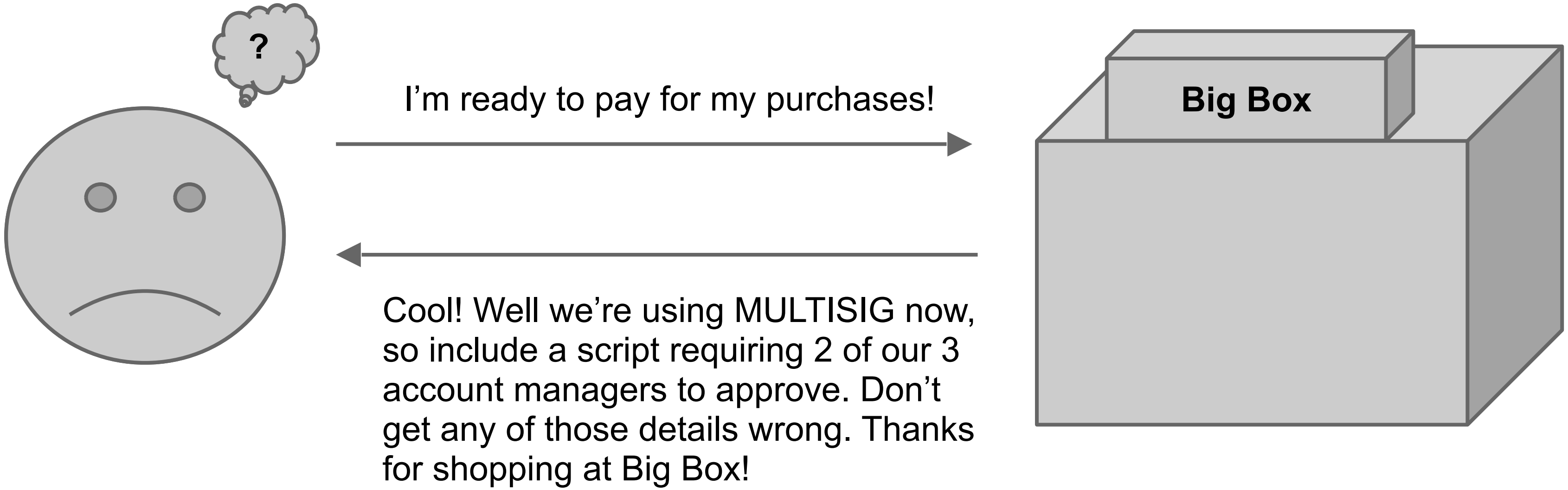
nothing's going to redeem that ☹

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!

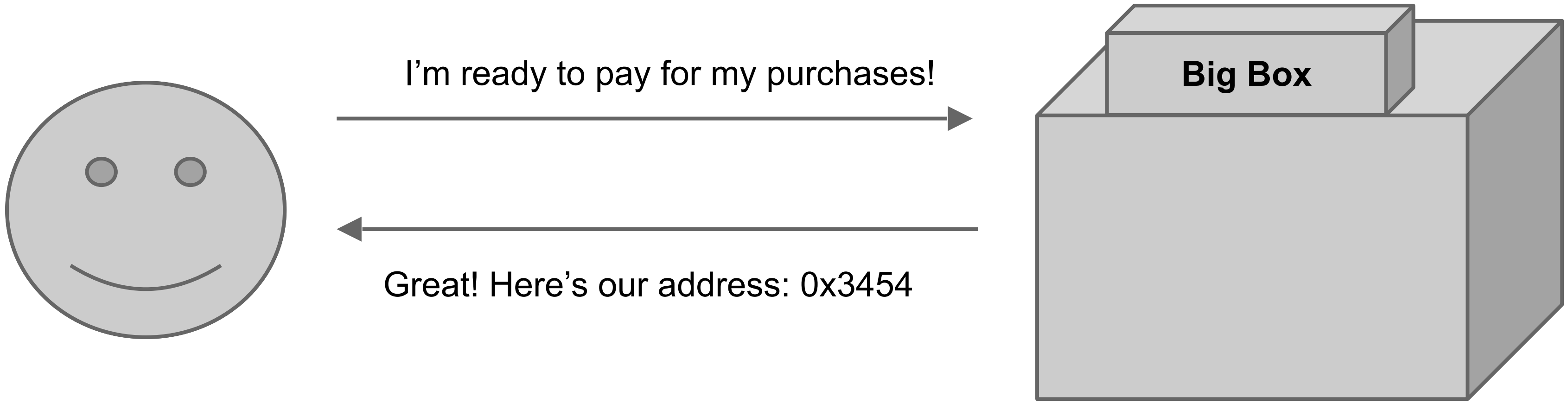
Idea: use the hash of redemption script

```
<
<
<signature>

O
<
O
<pubkey>
OP_CHECKSIG
```

“Pay to Script Hash”

Pay to script hash


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

Block size limits and Segwit

Applications of Bitcoin scripts

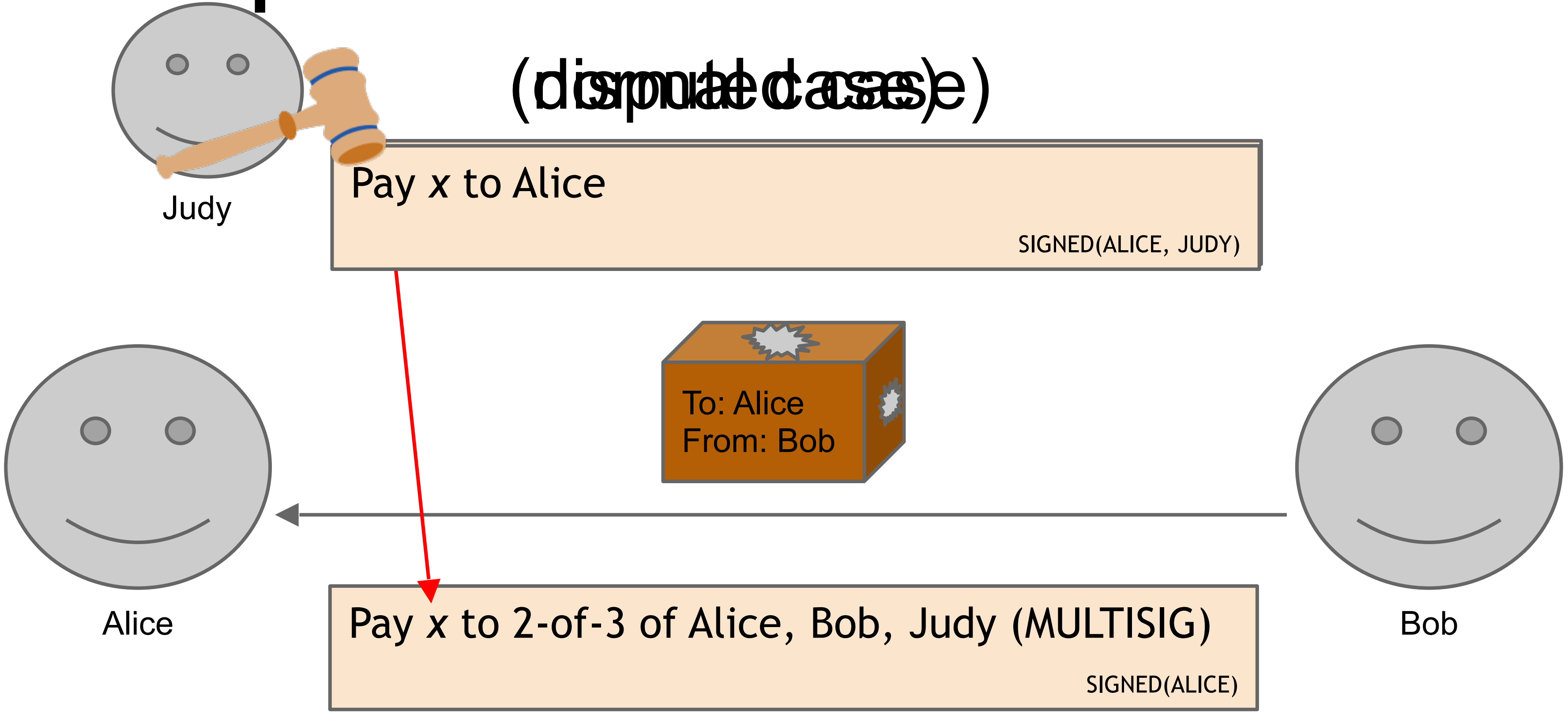
Example 1: “Fair” transactions

- 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

(normal case)

(disputed case)



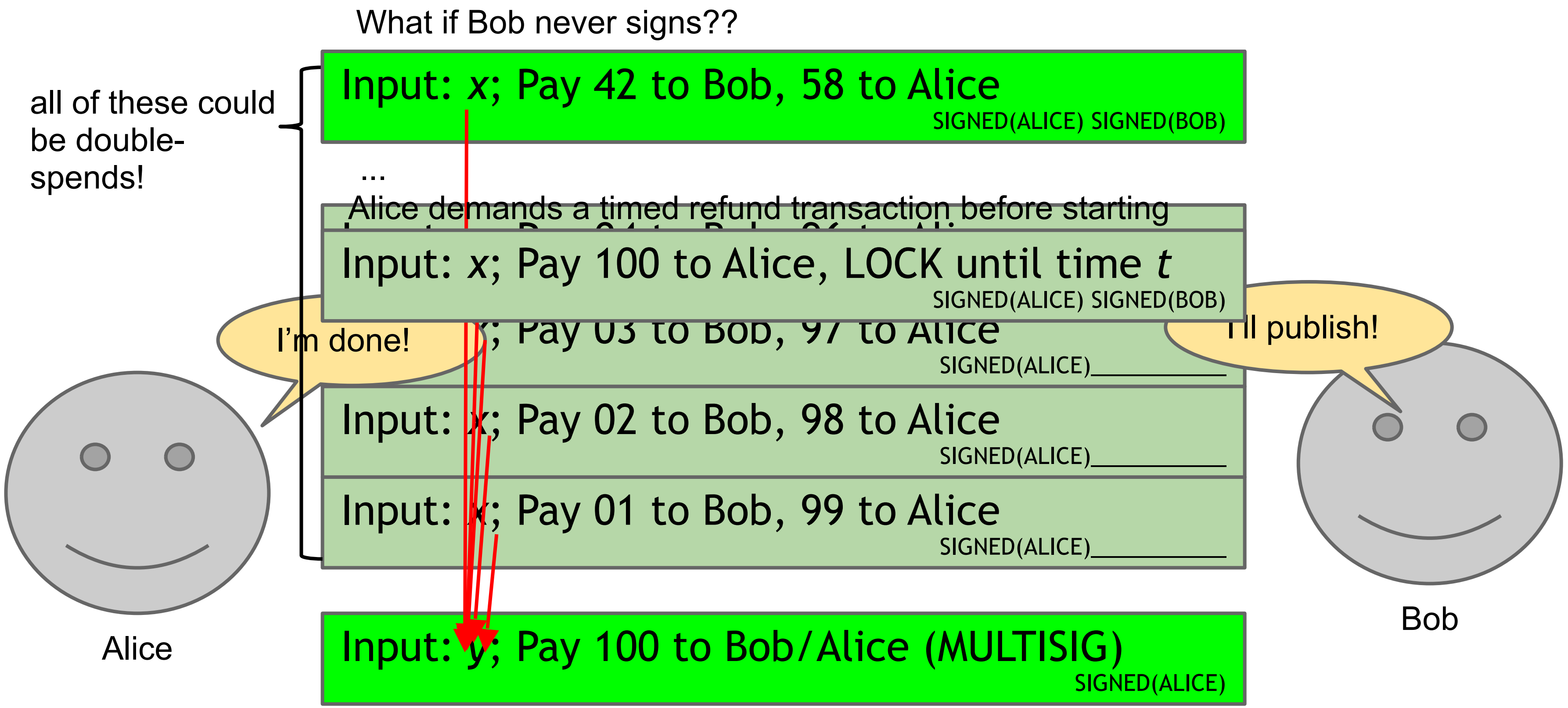
Example 2: Micro-payments

- 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 2: 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 2: Micro-payments with Bitcoin


What if Bob never signs??
all of these could be double-spends!
Input: x; Pay 42 to Bob, 58 to Alice
SIGNED(ALICE) SIGNED(BOB)
...
Alice demands a timed refund transaction before starting
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)
I'm done!
I'll publish!
Alice
Bob

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

More 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)