IT486 v3.0

Escrow transaction, Micro-payments, Multi-party lottery

What is a multi-sig address?

- Multi-Signature Address: An address that is associated with more than one private key
- *m*-of-*n* address - Associated with *n* private keys
  - Spending BTC requires signatures from at least *m* keys

What is a multi-sig address?

- Multi-Signature Address: An address that is associated with more than one private key
- *m*-of-*n* address - Associated with *n* private keys
  - Spending BTC requires signatures from at least *m* keys
- A multi-signature txn is one that spends funds from a multi-signature address

Fair transactions

- Alice wants to buy a product from an online vendor Bob
- Parties can cheat:
  - Alice can refuse to pay once Bob ships the product
  - Bob can take payment without shipping the product

Fair transactions

- Alice wants to buy a product from an online vendor Bob
- Parties can cheat:
  - Alice can refuse to pay once Bob ships the product
  - Bob can take payment without shipping the product
- Users of the protocol don't trust each other, but nevertheless they want to achieve a common goal

Fair transactions

- Alice wants to buy a product from an online vendor Bob
- Parties can cheat:
  - Alice can refuse to pay once Bob ships the product
  - Bob can take payment without shipping the product
- Users of the protocol don't trust each other, but nevertheless they want to achieve a common goal
- Solution: introduce a third-party arbitrator Judy. Don't want Arbitrator to be involved unless there's a dispute

Fair transactions via escrow

- Alice sends her payment to a 2-of-3 MULTISIG address. Any two of Alice, Bob, Judy must sign to spend funds from that address.

Alice

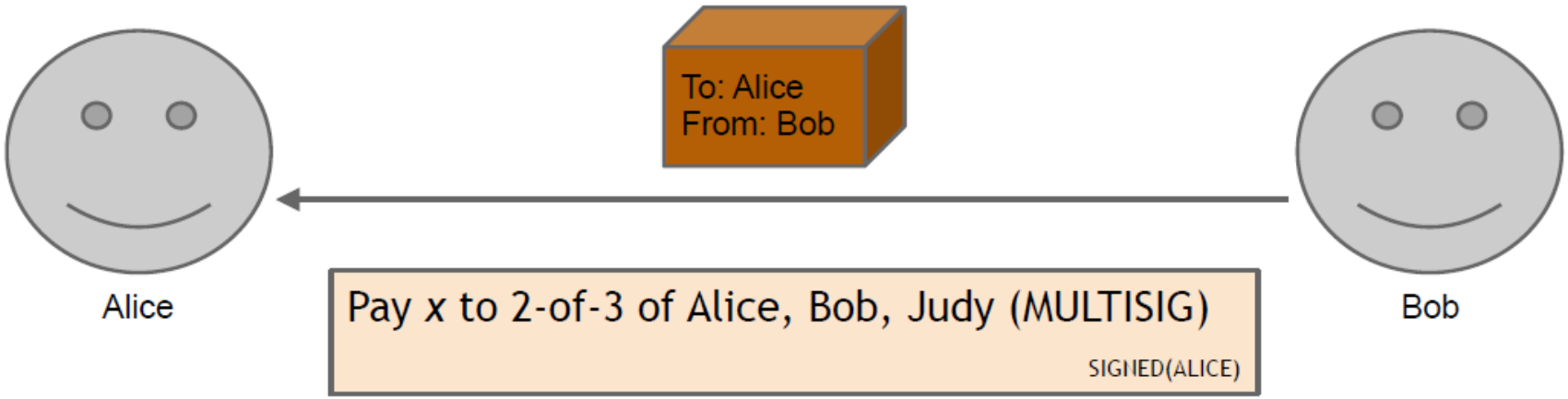
Pay x to 2-of-3 of Alice, Bob, Judy (MULTISIG)

SIGNED(ALICE)

Bob

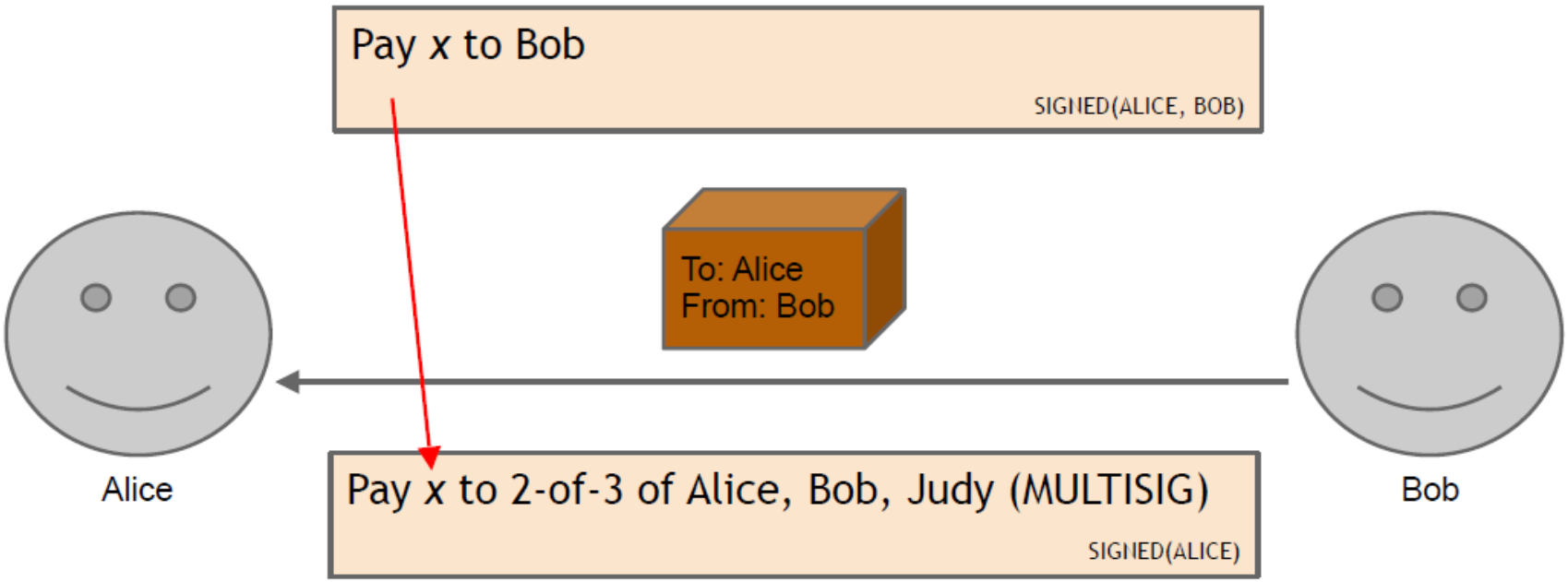
Fair transactions via escrow

- Once Bob sees this transaction included in the block chain, he sends the product to Alice

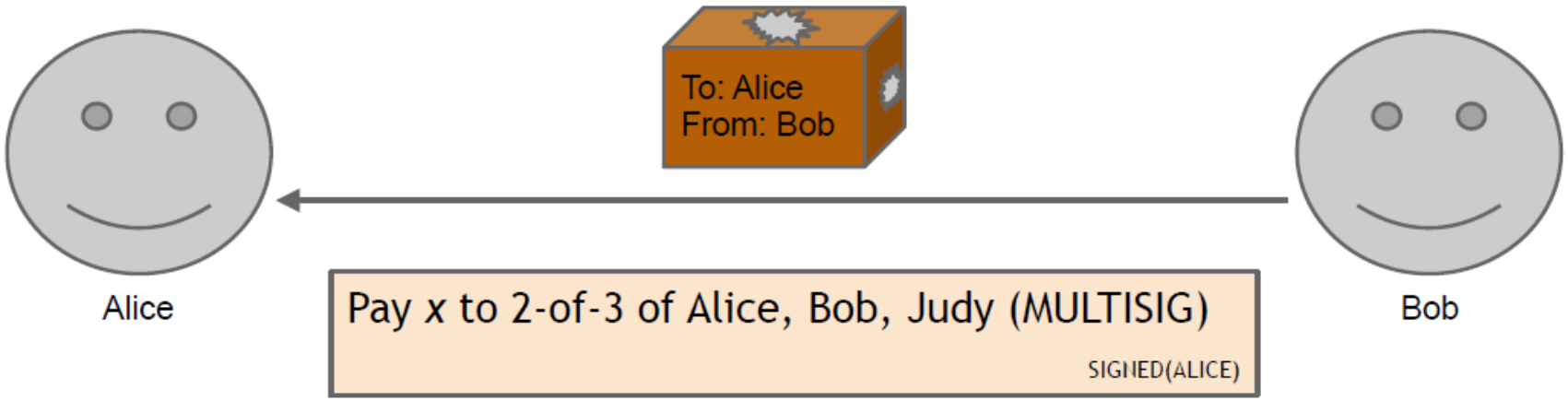


Assume both Alice and Bob are honest

- Both Alice and Bob sign a MULTISIG payment that goes to Bob

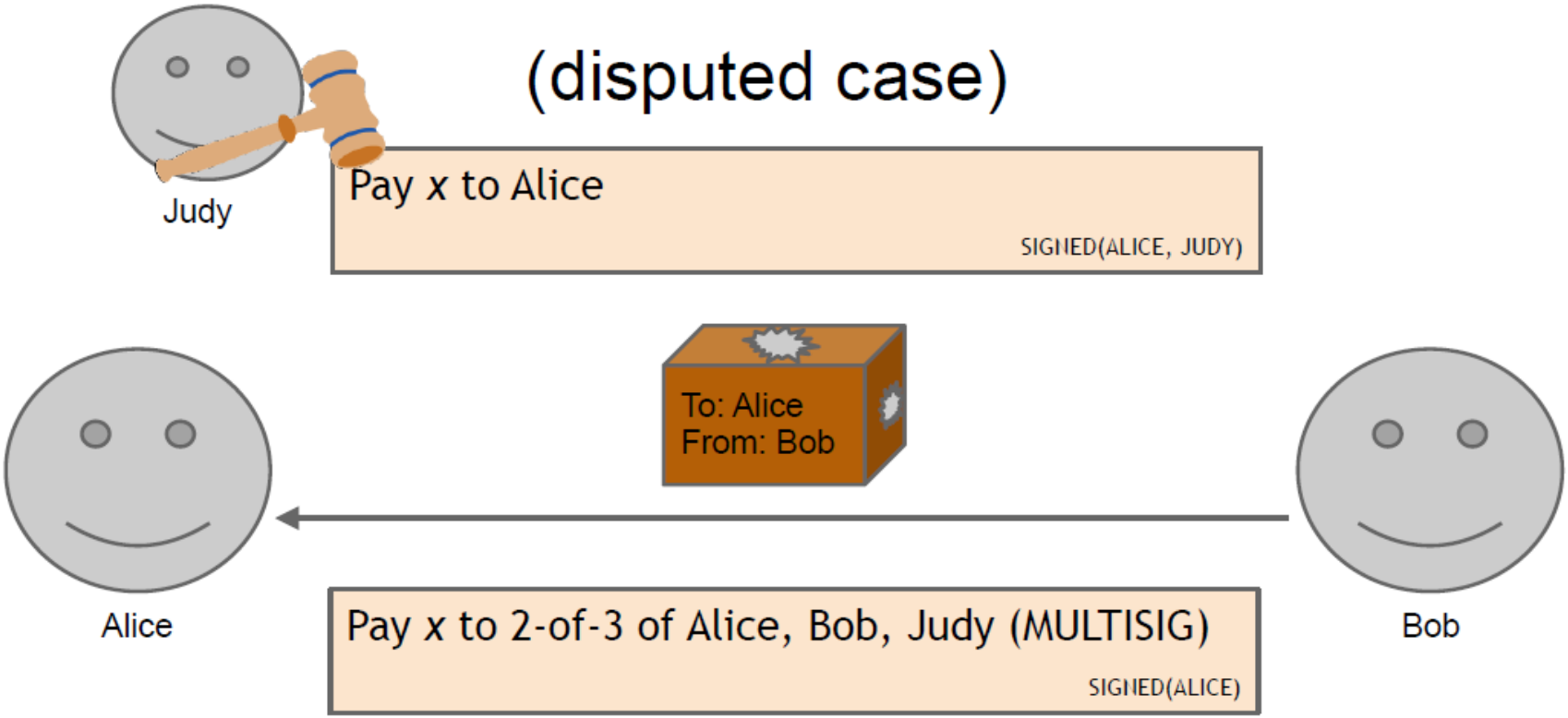


Bob sends damaged product



Dispute resolution

- Judy will decide and sign with the "good" person



Micropayments with bitcoin

- Bob is Alice's wireless provider and requires her to pay a fee (1 btc) for every minute of service
- BUT, Alice doesn't want to create a payment transaction every minute (avoid transaction fees!)

Micropayments with bitcoin

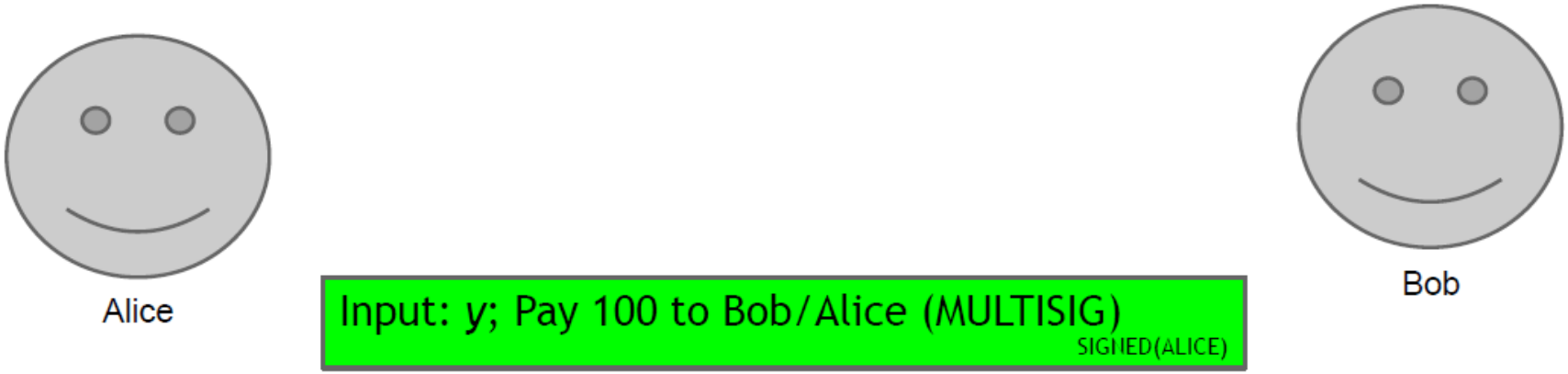
- Bob is Alice's wireless provider and requires her to pay a fee (1 btc) for every minute of service
- BUT, Alice doesn't want to create a payment transaction every minute (avoid transaction fees!)
- Solution Idea:
  - Instead of doing several transactions, do a single transaction for total payment at the end (and thus incur only a single transaction fee)

Micropayments with bitcoin

- Bob is Alice's wireless provider and requires her to pay a fee (1 btc) for every minute of service
- BUT, Alice doesn't want to create a payment transaction every minute (avoid transaction fees!)
- Solution Idea:
  - Instead of doing several transactions, do a single transaction for total payment at the end (and thus incur only a single transaction fee)
- How to implement it without a third party?

Micropayments with bitcoin

- Alice sends a consolidated payment to a 2-of-2 MULTISIG address. Both Alice and Bob must sign to spend funds from that address



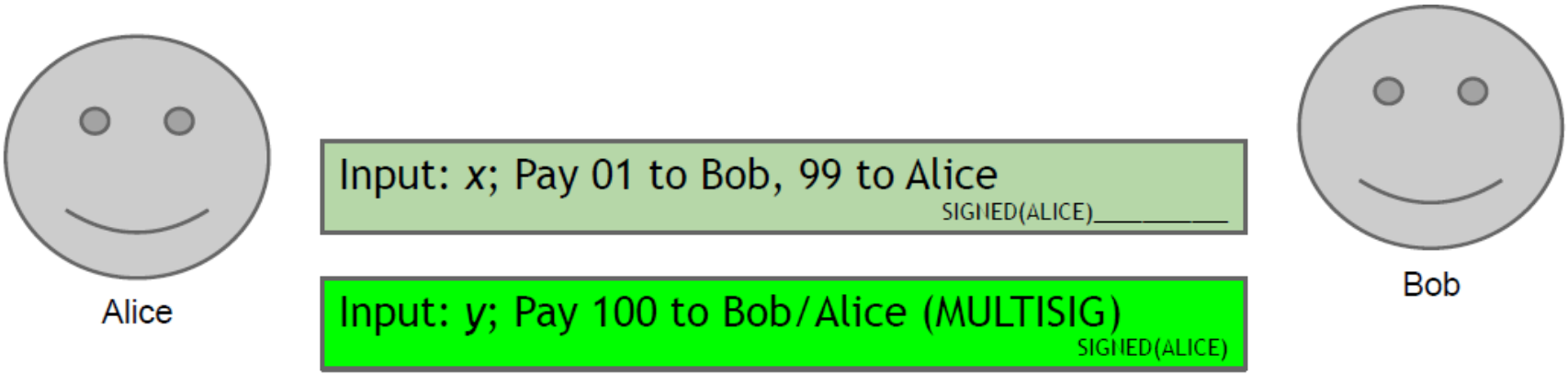
Micropayments with bitcoin

- After every minute, Alice creates a multisig tx that pays Bob the money that she owes him and rest to back to Alice. Bob doesn't sign and doesn't publish to the blockchain

Micropayments with bitcoin

- After every minute, Alice creates a multisig tx that pays Bob the money that she owes him and rest to back to Alice. Bob doesn't sign and doesn't publish to the blockchain
- At some time in the future, Alice stops signing, signifying end of service
- Bob understands that the last transaction is the final, and he publishes it to the blockchain

Micropayments with bitcoin



Micropayments with bitcoin

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)

Bob

Micropayments with bitcoin

Alice

Input: x; Pay 04 to Bob, 96 to Alice
SIGNED(ALICE)

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

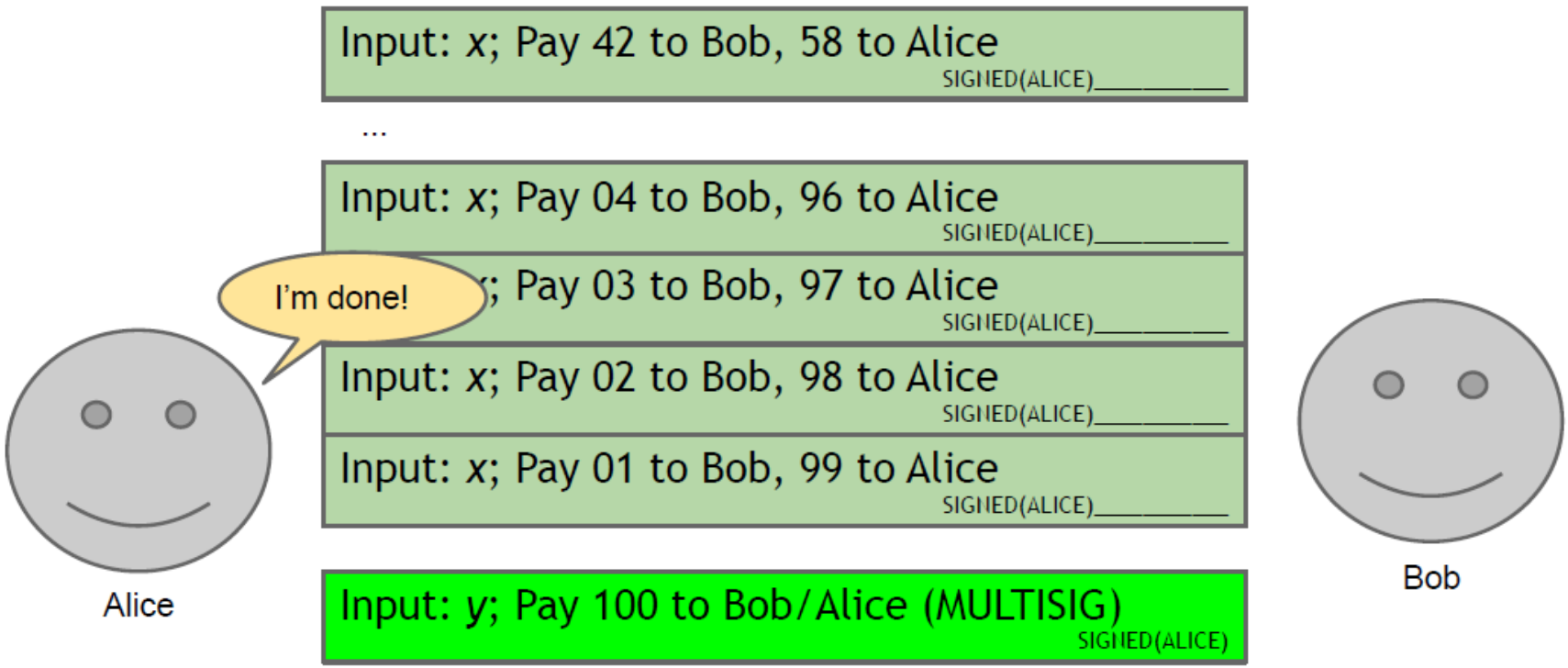
Input: x; Pay 02 to Bob, 98 to Alice
SIGNED(ALICE)

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

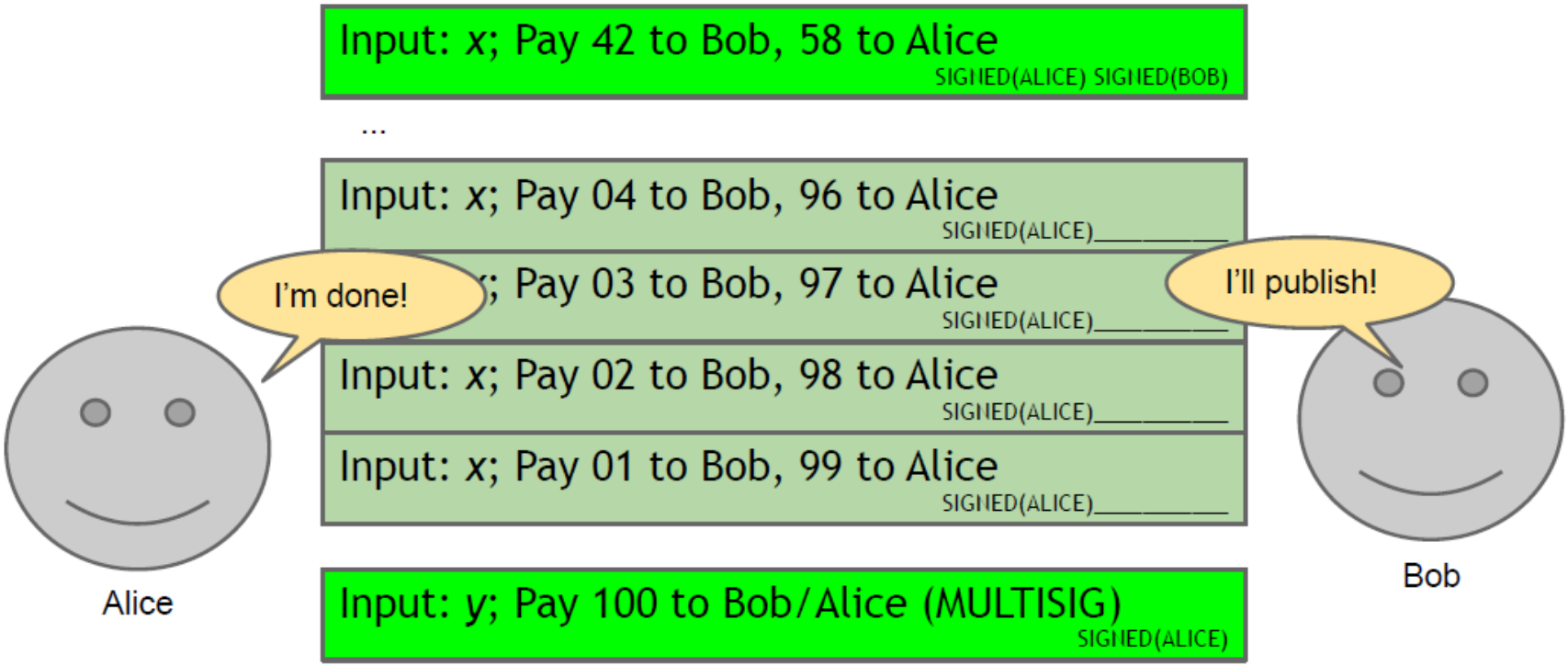
Input: y; Pay 100 to Bob/Alice (MULTISIG)
SIGNED(ALICE)

Bob

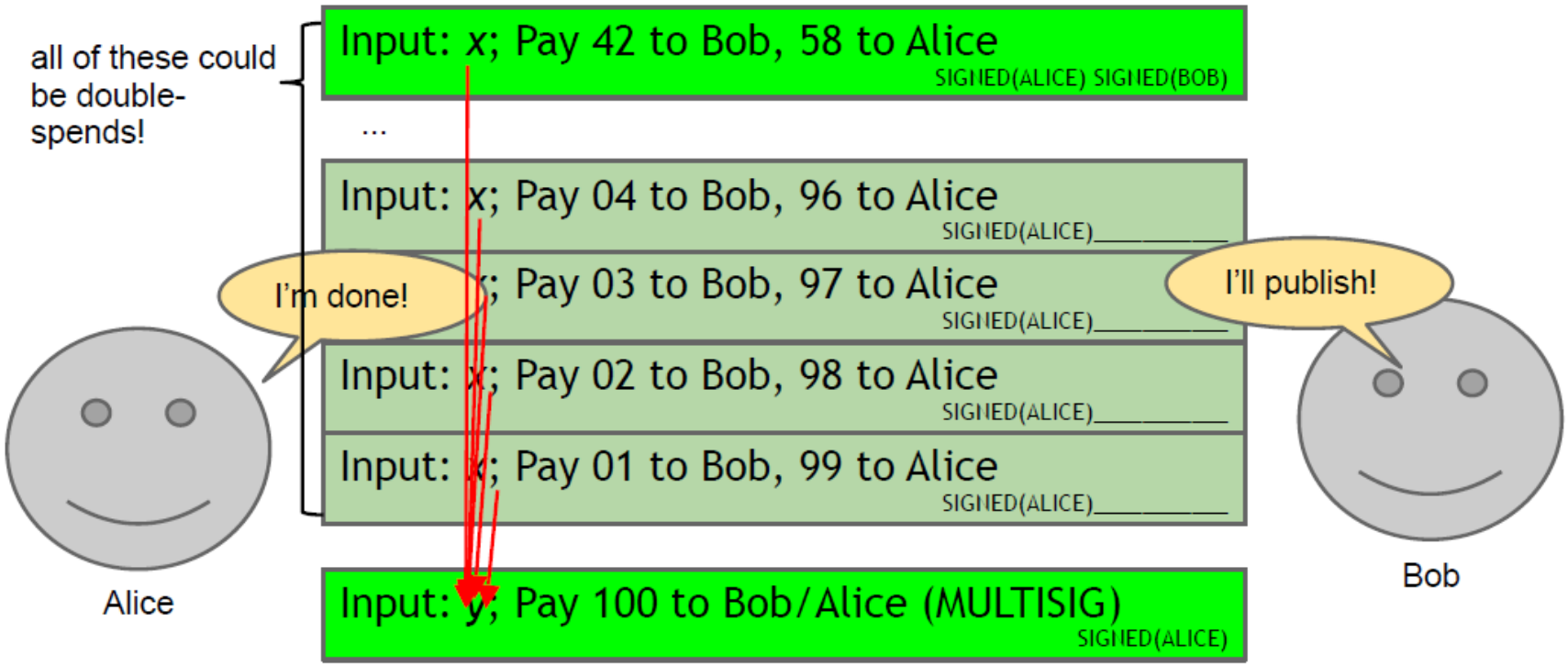
Micropayments with bitcoin



Micropayments with bitcoin

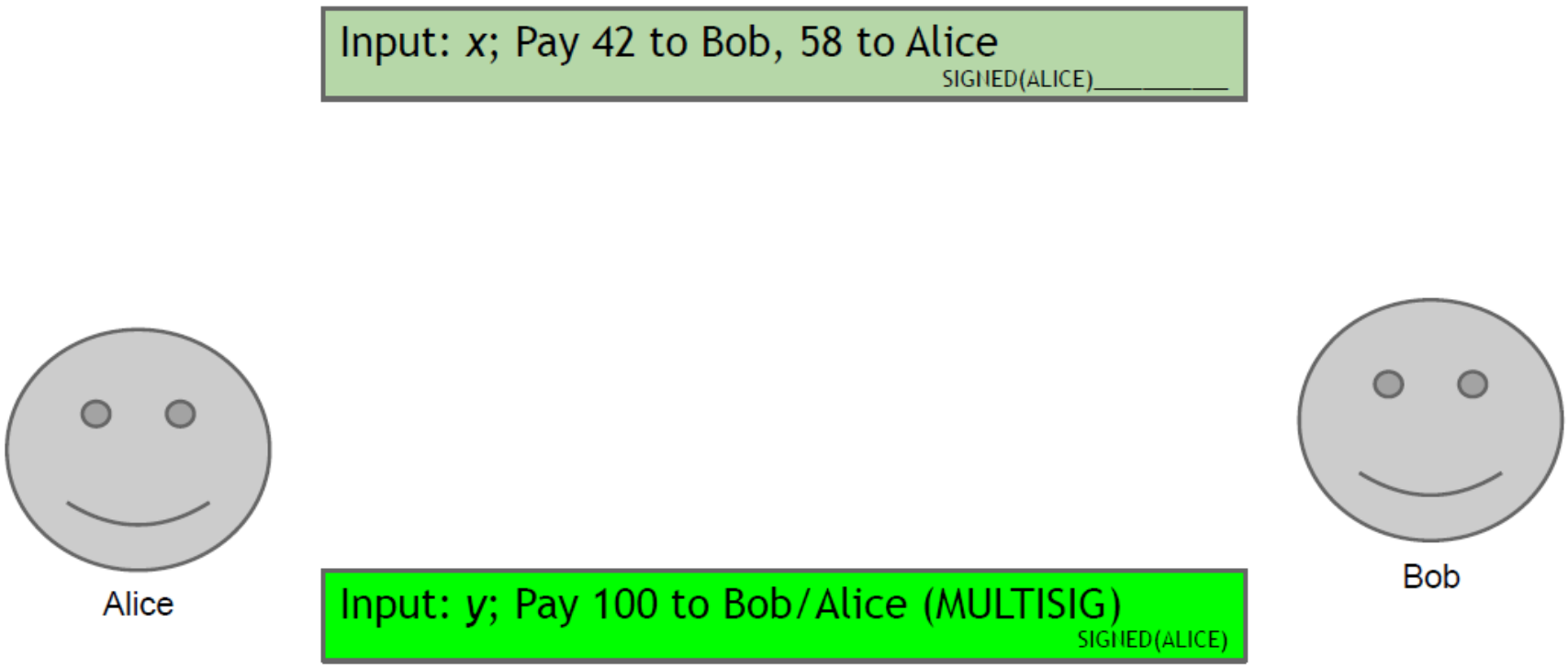


Micropayments with bitcoin



Micropayments with bitcoin

- What if Bob never signs the last transaction. Alice loses her money!



Solution: Lock Time

- Before Alice starts using Bob's service, Alice and Bob both sign a tx refunding all of Alice's money back to her, but the refund is "locked" until some time in the future
- When this "lock time" comes, Alice signs this refund tx to publish it on blockchain to reclaim the refund

Micropayments with bitcoin

What if Bob never signs??

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

Alice demands a timed refund transaction before starting

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

Alice

Bob

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

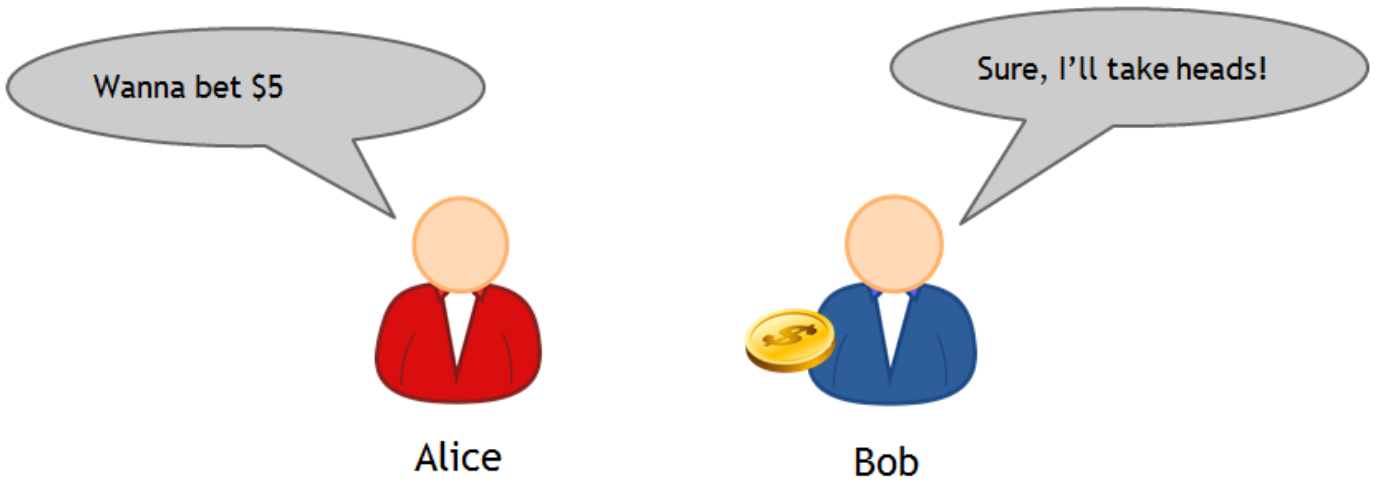
lock_time


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

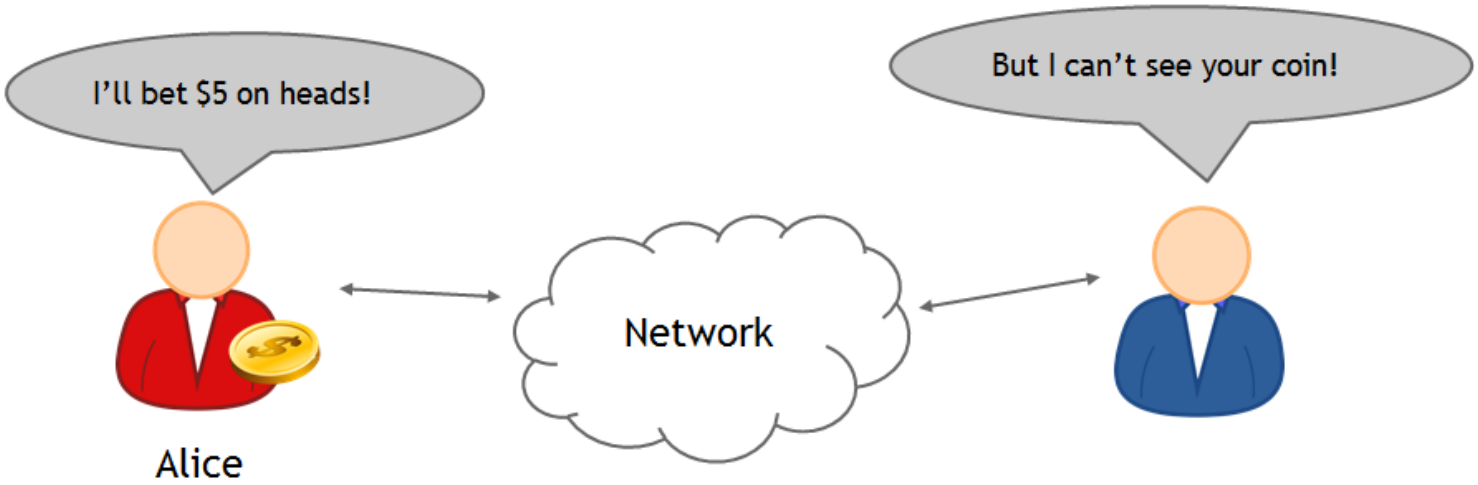

Block index or real-world timestamp before which this transaction can't be published

Multi-party lotteries: offline version

The outcome is fair, but both parties have to trust the other will actually pay up



Multi-party lotteries: online version



- How to generate randomness that both parties agree is fair?
- How to force the party who loses to pay up?

Hash commitments

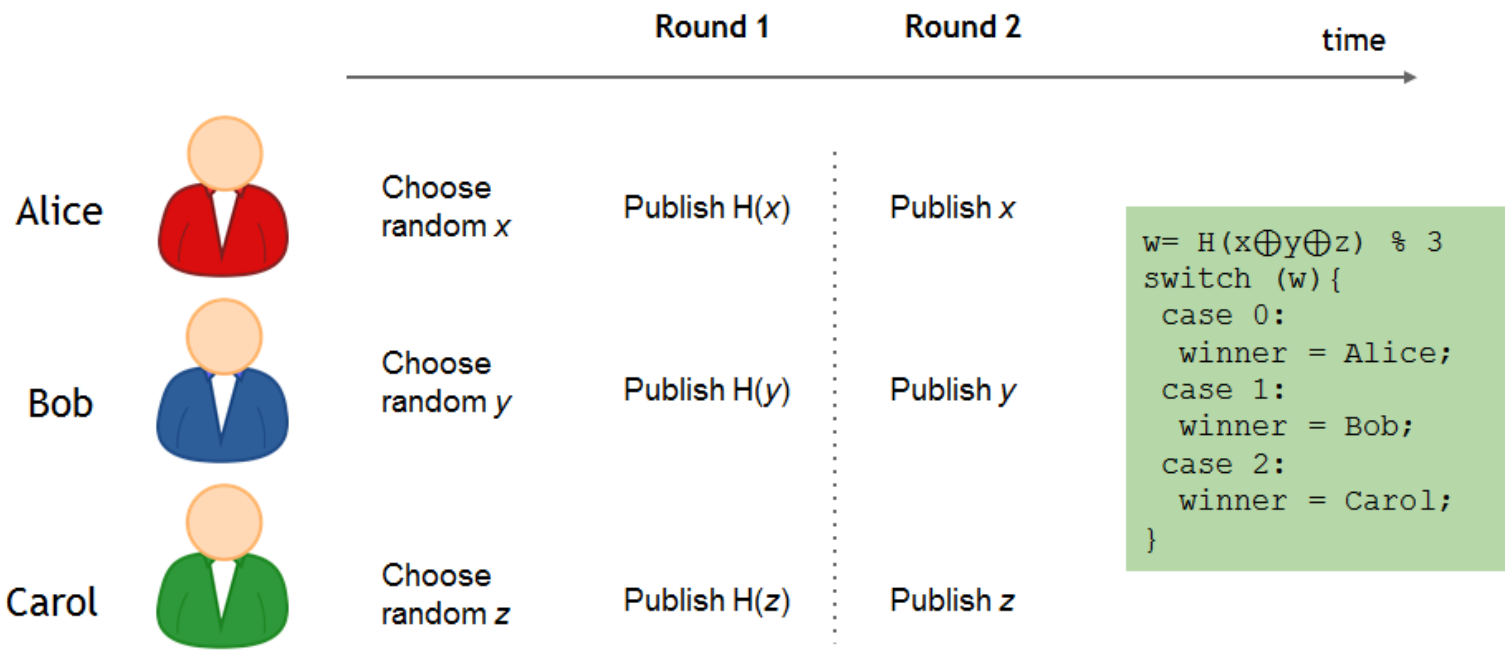
Recall: Publishing $H(x)$ is a commitment to x

- Can't find an $x' \neq x$ later such that $H(x') = H(x)$
- $H(x)$ reveal no information about x (assuming the space of possible x values is big).

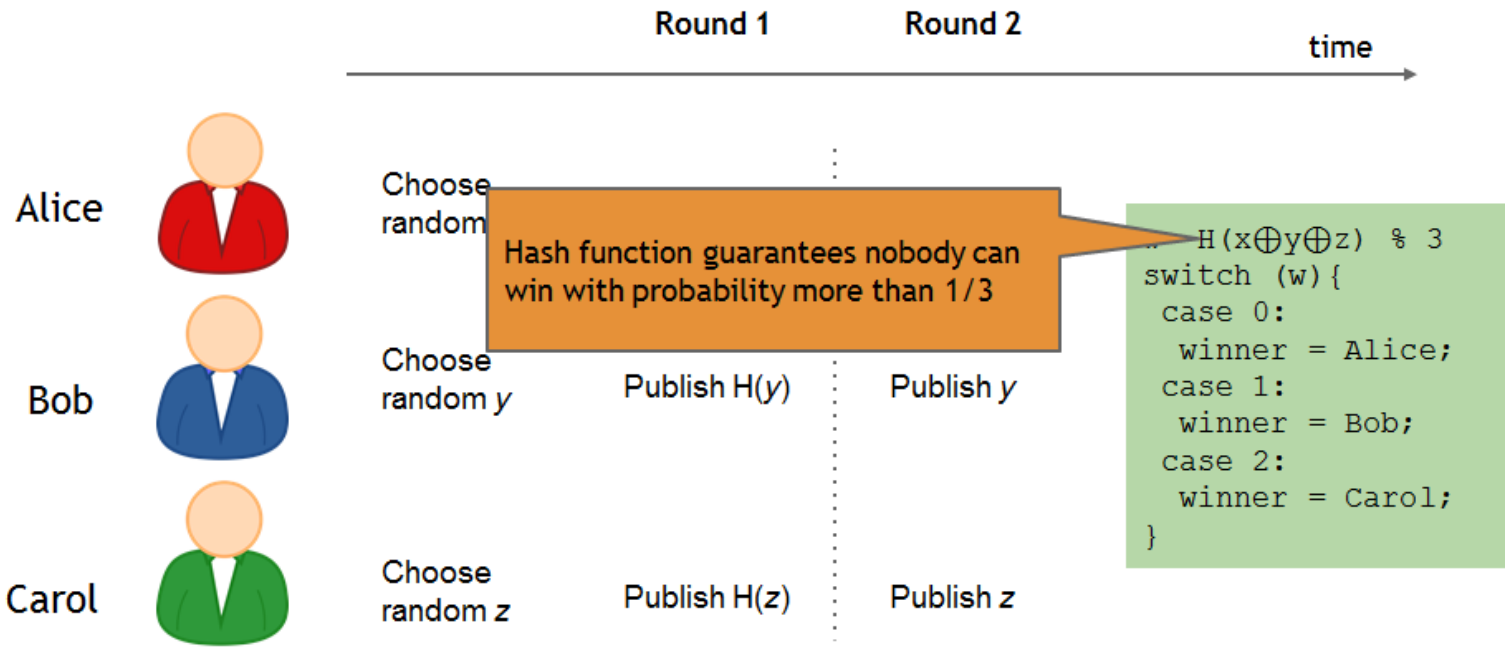
A lottery with hash commitments

- Round 1: The parties publish hash commitments of their secret nonces
- Round 2: The parties open their commitments by revealing their nonces
- At the end, the parties compute a pre-agreed function of their inputs to determine the winner

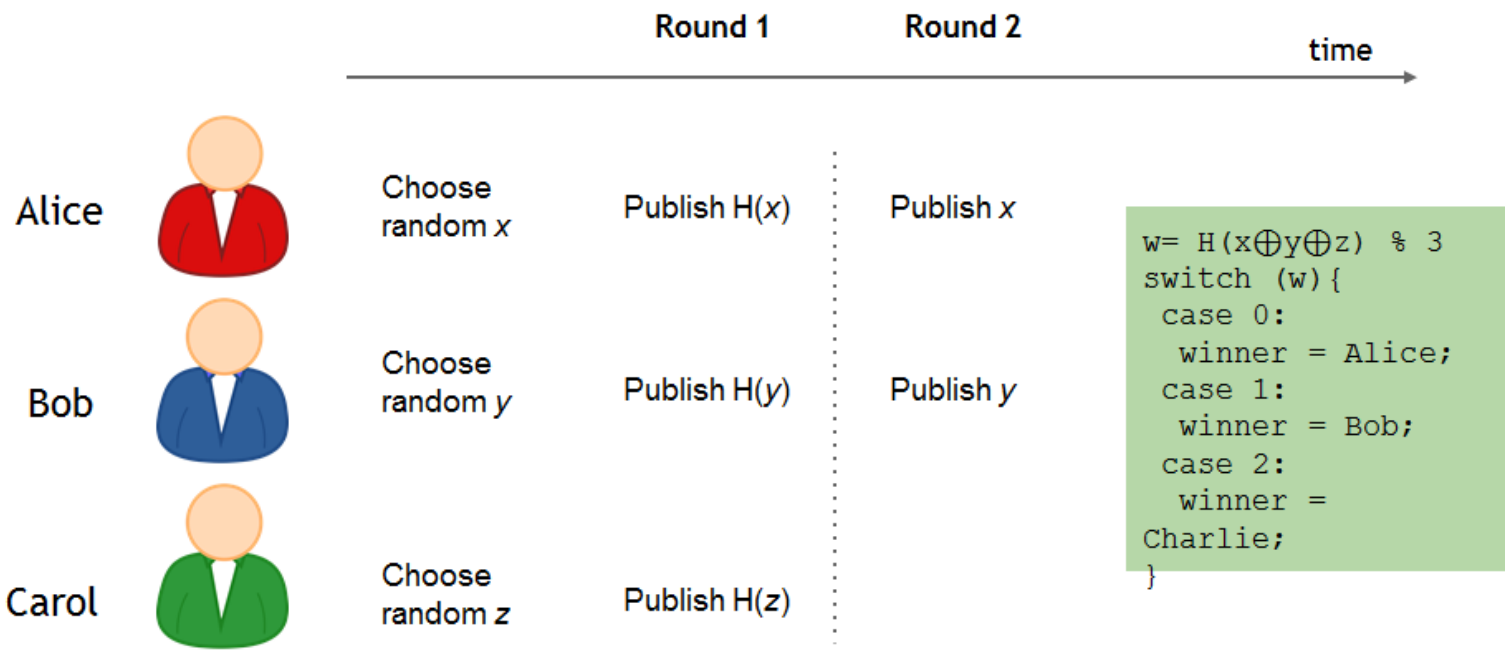
A lottery with hash commitments



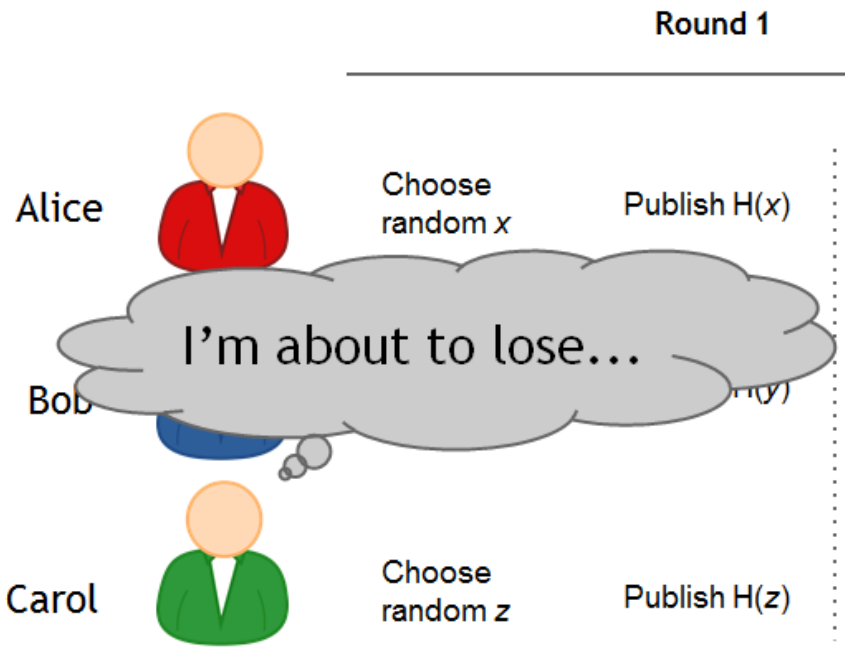
A lottery with hash commitments



A lottery with hash commitments

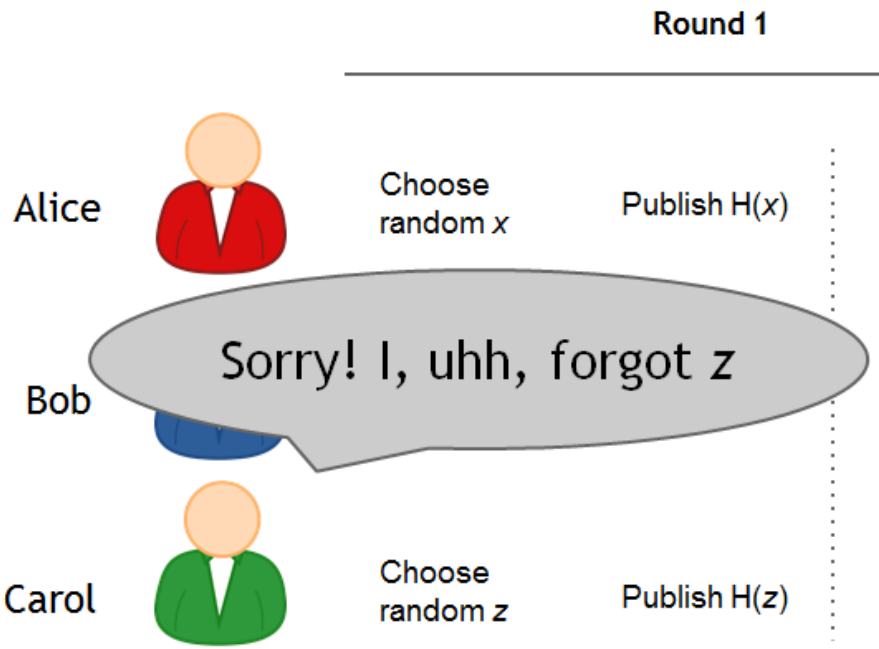


A lottery with hash commitments



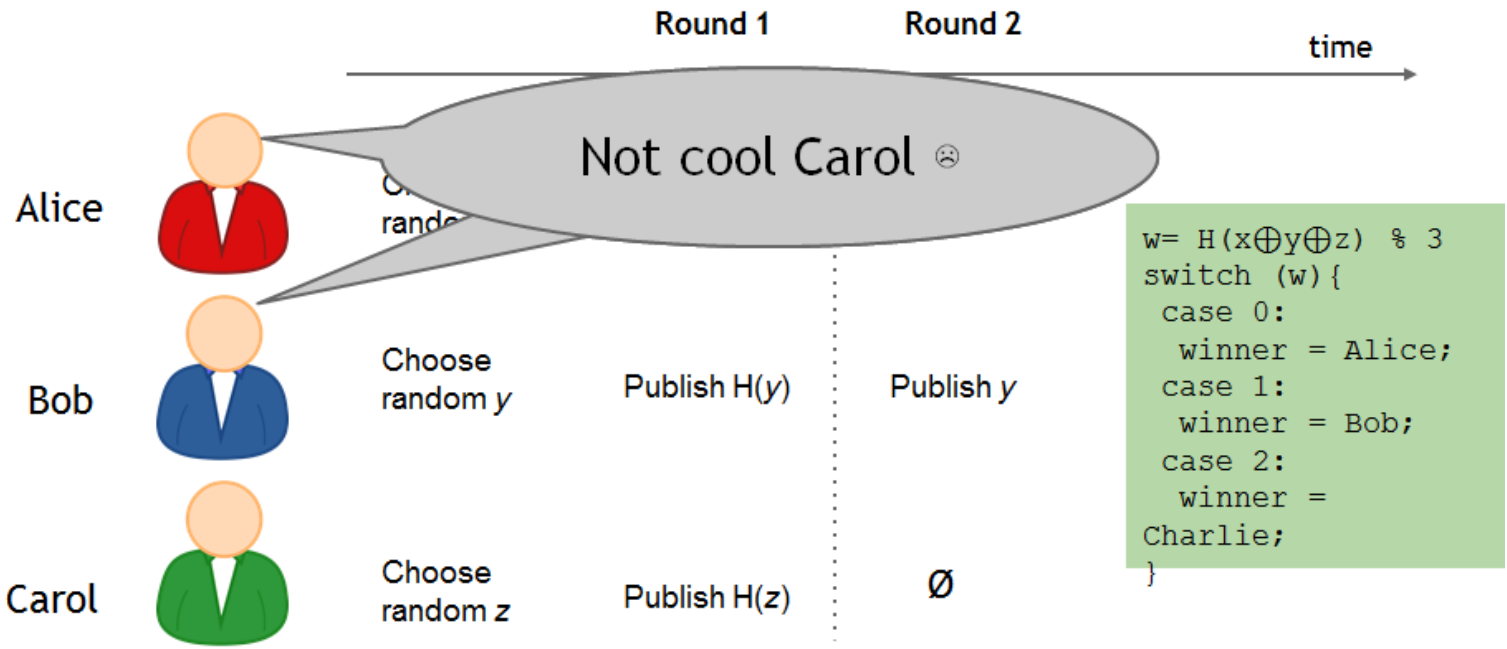

```
w= H(x⊕y⊕z) % 3
switch (w){
 case 0:
  winner = Alice;
 case 1:
  winner = Bob;
 case 2:
  winner =
Charlie;
}
```

A lottery with hash commitments




```
w= H(x⊕y⊕z) % 3
switch (w){
 case 0:
  winner = Alice;
 case 1:
  winner = Bob;
 case 2:
  winner =
Charlie;
}
```

A lottery with hash commitments



Timed hash commitments

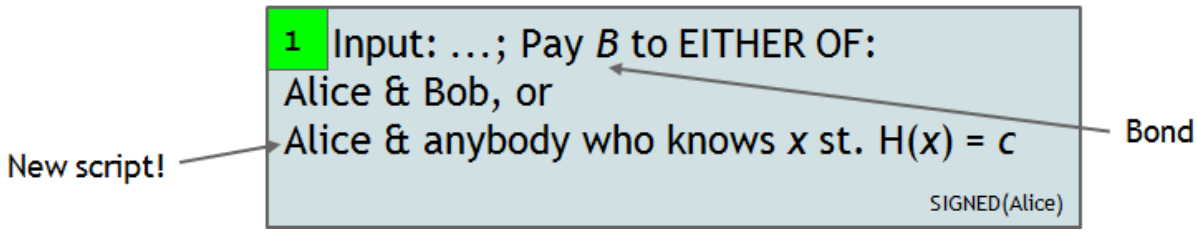
- Problem: What if Alice runs away and does not reveal her number x?

Timed hash commitments

- Problem: What if Alice runs away and does not reveal her number x?
- Solution Idea: Alice deposits a bond which she loses if she does not reveal x within a certain time

Time hash commitments

- Alice creates a transaction (bond) that can be spent in 2 ways: (1) by both Alice and Bob signing, or (2) Alice reveals her secret number x

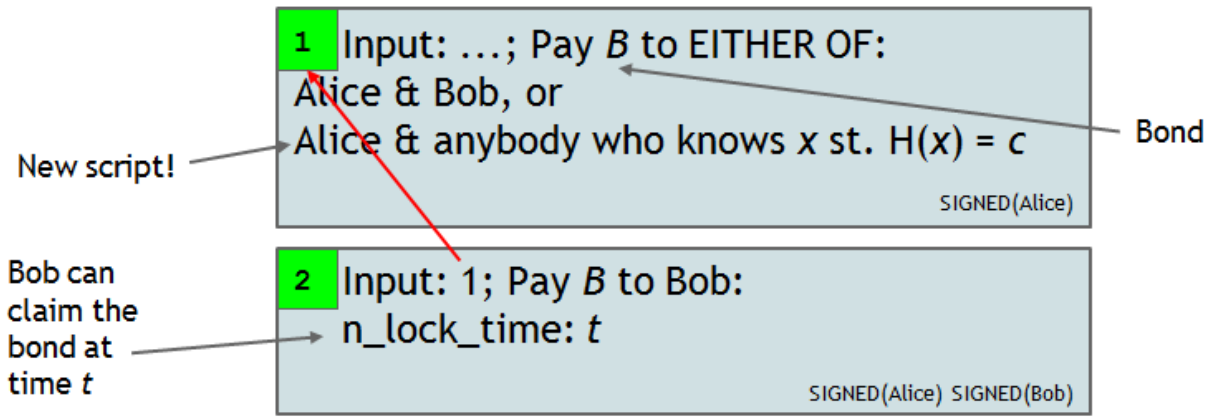


- locking script:

```
OP_IF
    AlicePubK OP_CHECKSIGVERIFY BobPubK OP_CHECKSIG
OP_ELSE
    AlicePubK OP_CHECKSIGVERIFY OP_HASH H(x) OP_EQUAL
OP_ENDIF
```

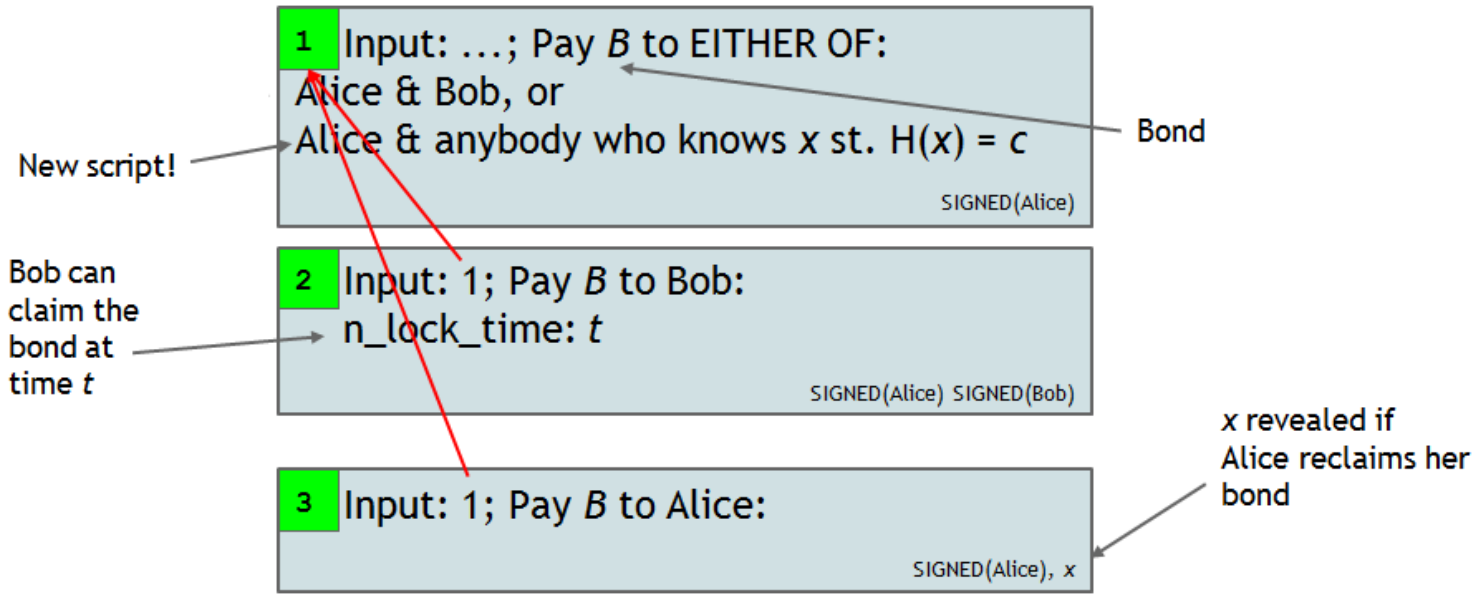
Timed hash commitments

- Alice and Bob both sign a transaction that pays the bond to Bob, which can only be spent after time t
- In txn 2 input Alice and Bob use the following unlocking script:
  - `BobSignature AliceSignature 1`



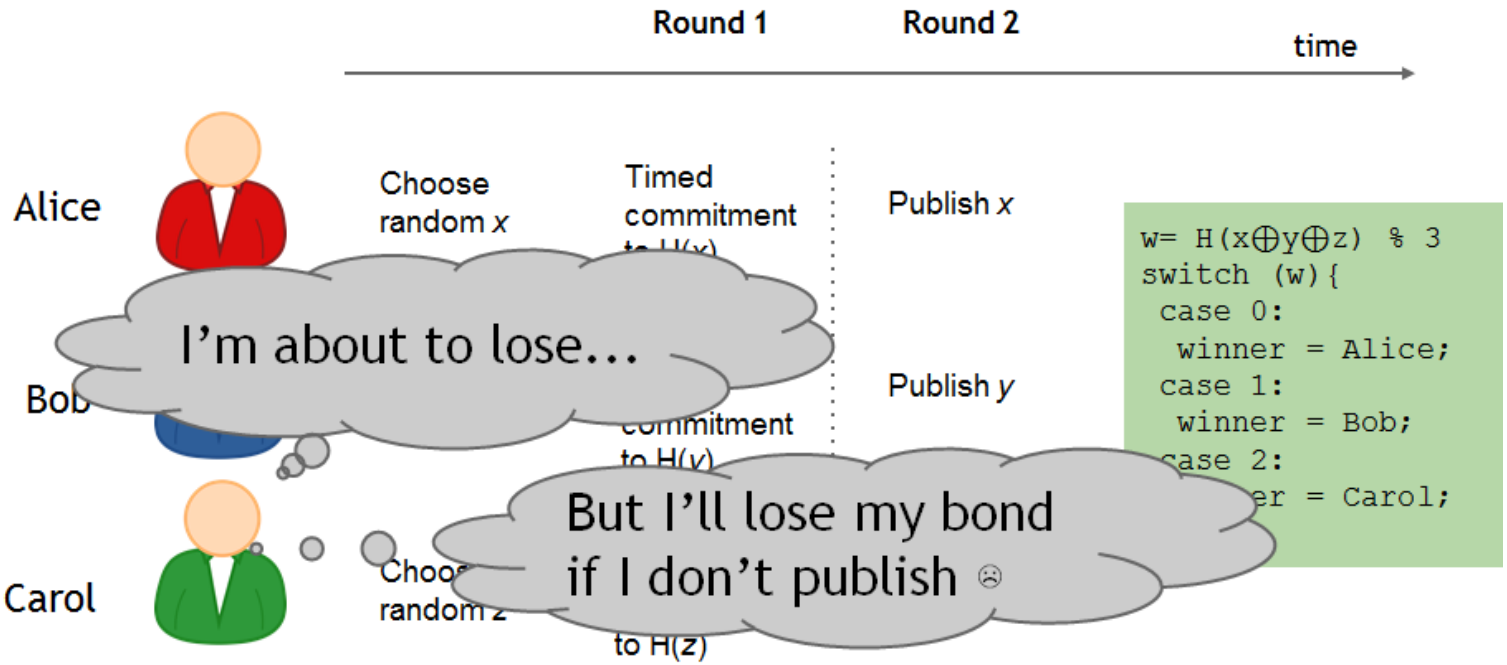
Timed hash commitments

- Alice creates txn 3 to claim her bond before the deadline
- To do so, she is forced to reveal her secret number x, as evident from the required unlocking script:
  - `x AliceSignature 0`



Timed hash commitments

- If Alice aborts without revealing her secret value x, she will forfeit her bond



Lottery with timed commitments

- Pro
  - implementable using Bitcoin without any change
- Cons
  - bond must be higher than the bet
  - not very efficient: $O(N^2)$ timed hash commitments (reasonable for a small number of players)

Required Reading

- Andrychowicz, Dziembowski, Malinowski, Mazurek, Secure Multiparty Computations on Bitcoin, CACM, 2016
- Can be found in the class folder or at the following link
  - https://doi.org/10.1145/2896386

In-class Exercise

Suppose we want Alice to irreversibly give Bob coins that Bob can only spend after time T. Consider the following protocol: Alice creates and signs a transaction tx paying BTC to Bob, with lock time T. Then Alice gives tx to Bob, who verifies it. Bob can't submit until time T.

(a) Bob does not trust Alice. Explain the problem with the proposed protocol.

(b) Suggest a simple way to modify the protocol to fix this problem.