Anonymity in Bitcoin
Tumbler/Mixer

Oct 9, 2019

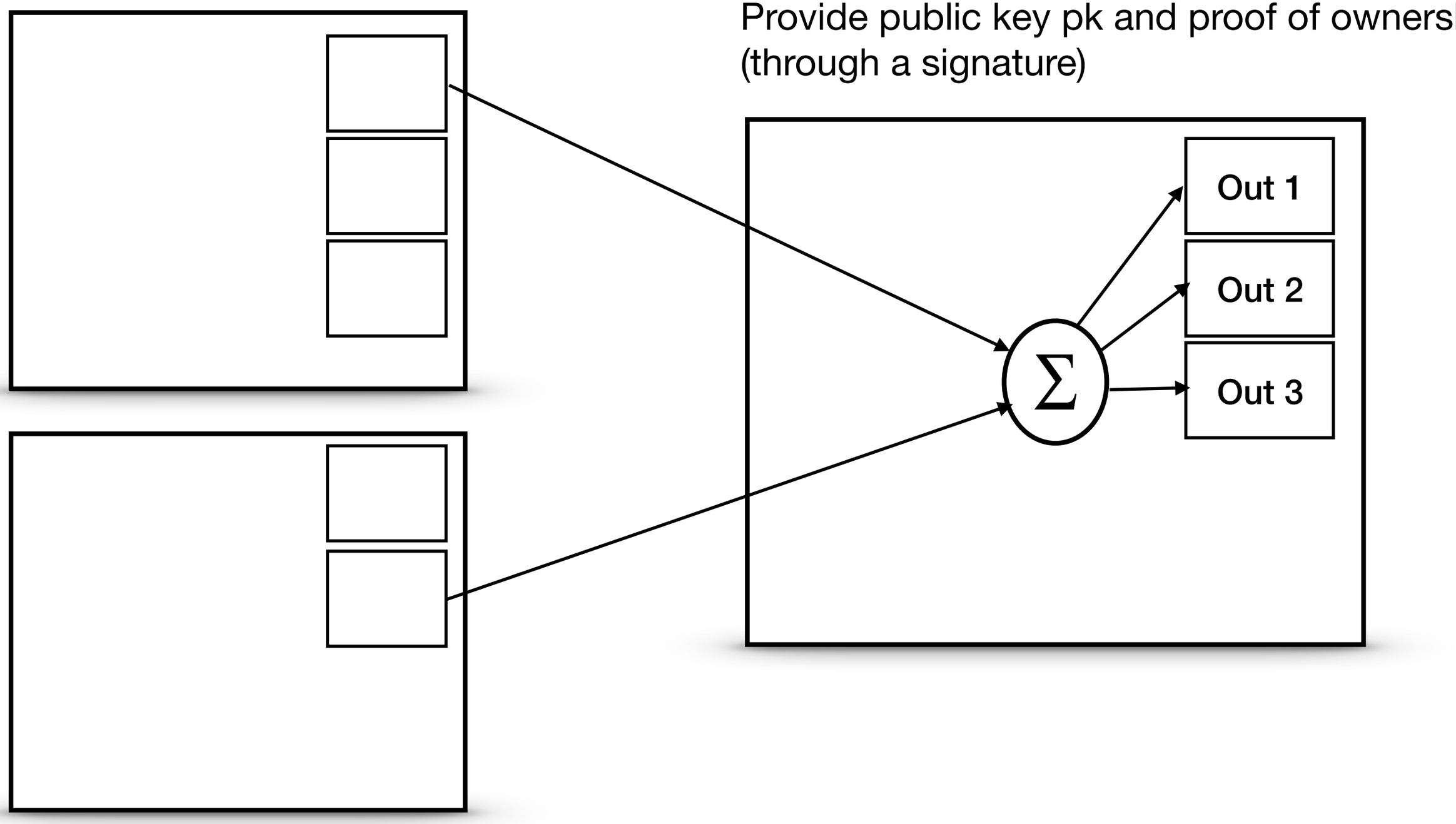
Anonymity and Pseudonymity

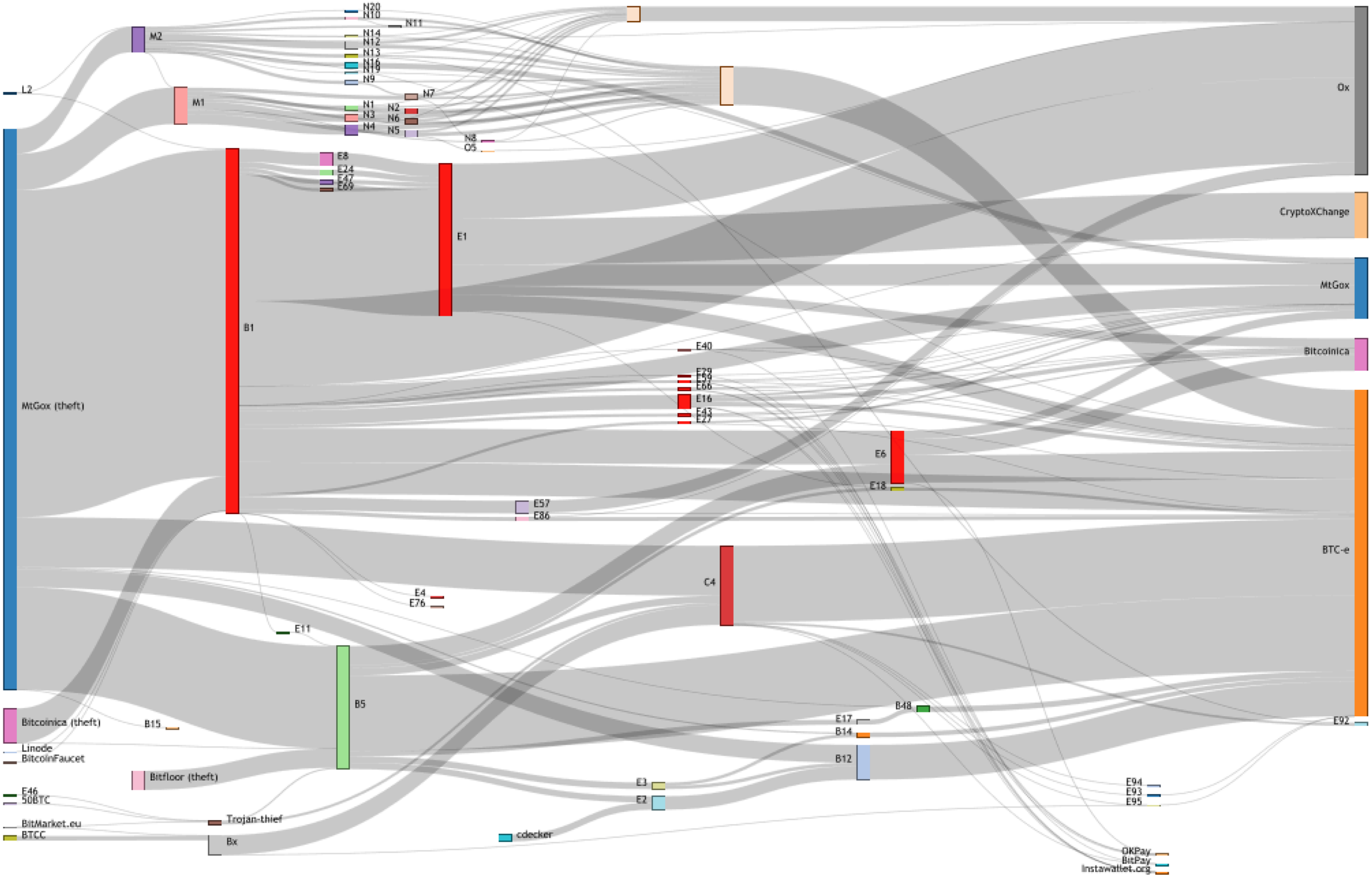
- anonymous = Nameless, unidentifiable

- pseudonymous = Fake name, still traceable

Tracing Bitcoin transactions

Normal redeem script:
Provide public key pk and proof of ownership
(through a signature)




L2
M2
M1
N20
N10
N11
N14
N12
N13
N16
N19
N9
N7
N1
N2
N3
N6
N4
N5
N8
O5
E8
E24
E47
E69
E1
B1
E40
E29
E59
E66
E16
E43
E27
E6
E18
E57
E86
C4
E4
E76
E11
B5
Ox
CryptoXChange
MtGox
Bitcoinica
BTC-e
E92
MtGox (theft)
Bitcoinica (theft)
B15
Linode
BitcoinFaucet
Bitfloor (theft)
E46
50BTC
BitMarket.eu
BTCC
Trojan-thief
Bx
cdecker
E3
E2
B48
E17
B14
B12
E94
E93
E95
OKPay
BitPay
Instawatbet.org

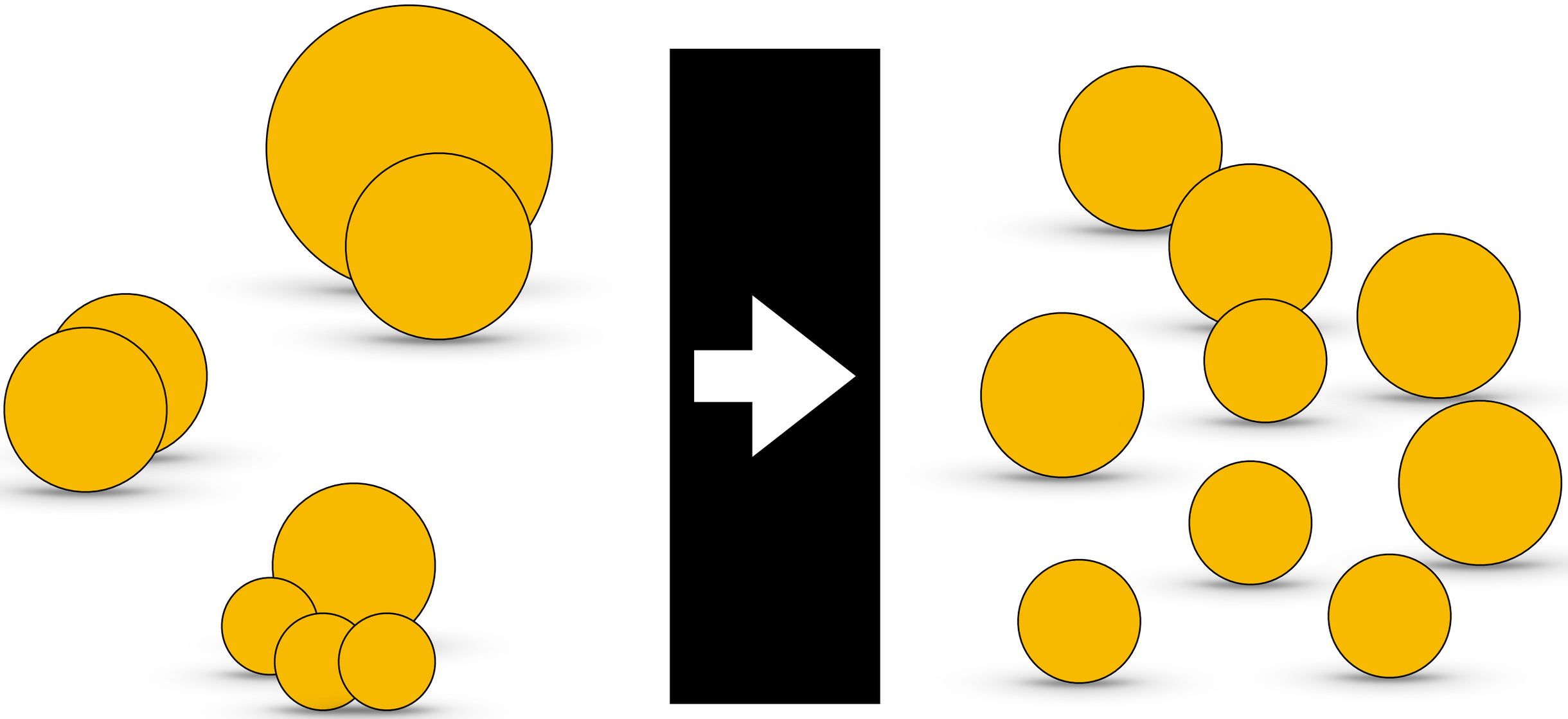
Privacy problems

- Your family can detect where you spend your money
- Your employer might detect unfavorable donations
- Every business partner knows all other

Address reuse is discouraged, but not always possible

Mixers

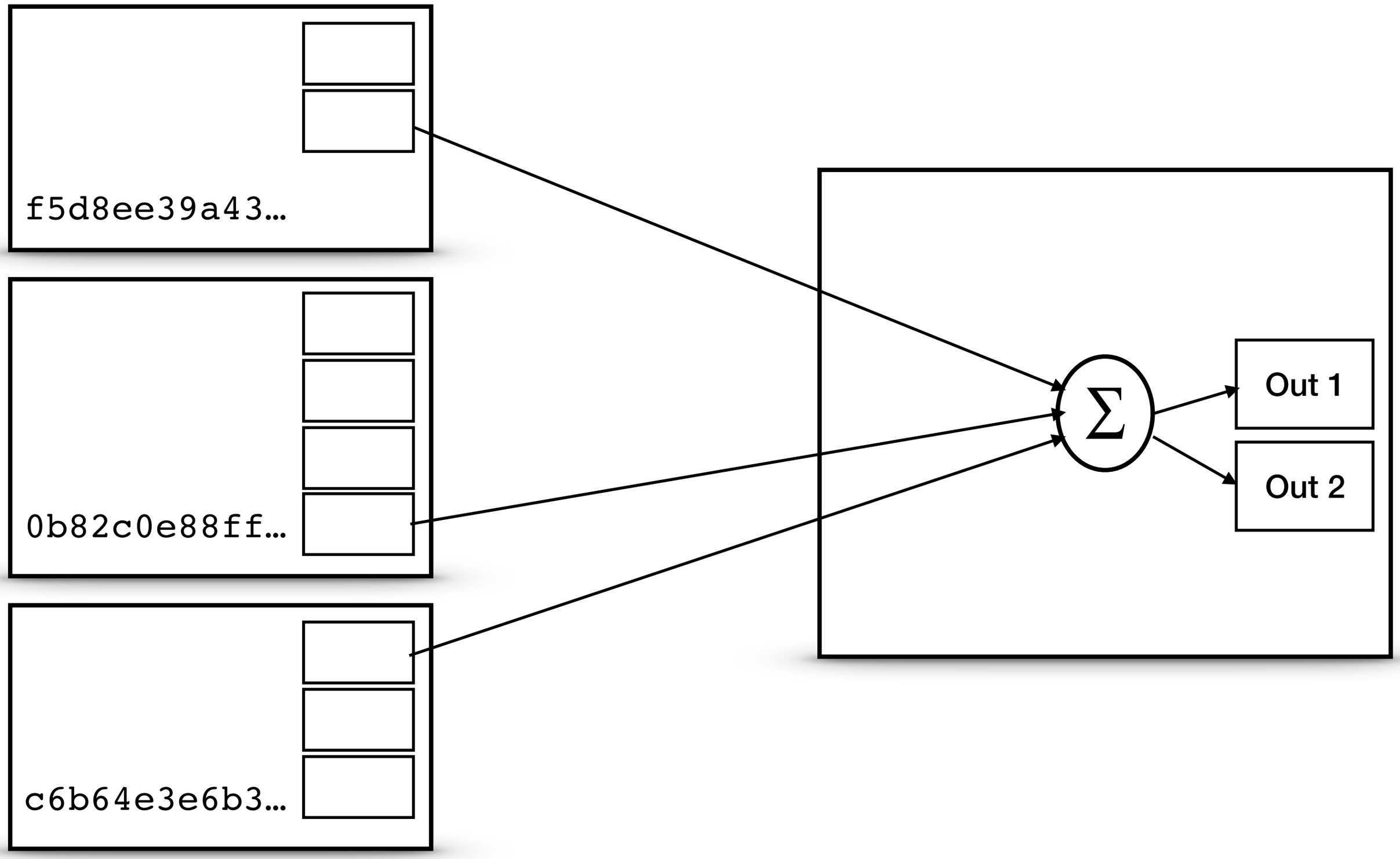
- Mixing many different inputs and outputs reduces traceability



Mixers in Bitcoin

- Mixers are not a first class citizen in Bitcoin
    - Bitcoin is flexible
- Many different varieties exist to disassociate inputs and outputs
    - Most popular one is CoinJoin

Bitcoin transaction



Transaction Details

Input1: **scriptSig:**
Transaction: f5d8ee39a43… 304502206e21…
Transaction Output: 1 43b0b82c0e88…

Input2: **scriptSig:**
Transaction: 0b82c0e88ff… 304502206e21…
Transaction Output: 4 43b0b82c0e88…

Input3: **scriptSig:**
Transaction: c6b64e3e6b3… 304502206e21…
Transaction Output: 0 43b0b82c0e88…

Output1:
value: 5000000000
OP_DUP OP_HASH160 304371705fa… OP_EQUALVERIFY OP_CHECKSIG

Output2:
value: 2300530000
OP_DUP OP_HASH160 3b24a405fa… OP_EQUALVERIFY OP_CHECKSIG

Transaction Details

Input1:
Transaction: f5d8ee39a43…
Transaction Output: 1

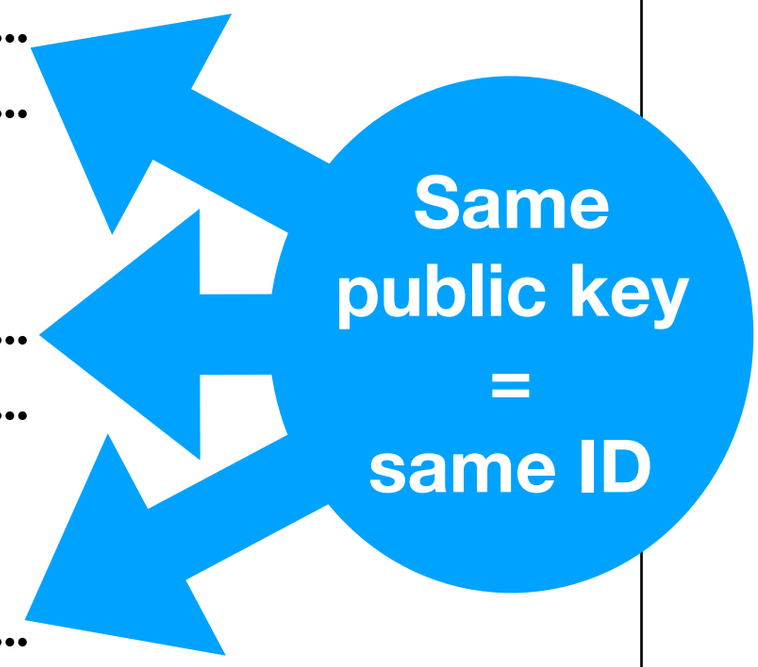
scriptSig:
304502206e21…
43b0b82c0e88…

Input2:
Transaction: 0b82c0e88ff…
Transaction Output: 4

scriptSig:
304502206e21…
43b0b82c0e88…

Input3:
Transaction: c6b64e3e6b3…
Transaction Output: 0

scriptSig:
304502206e21…
43b0b82c0e88…



Output1:
value: 5000000000
OP_DUP OP_HASH160 304371705fa… OP_EQUALVERIFY OP_CHECKSIG

Output2:
value: 2300530000
OP_DUP OP_HASH160 3b24a405fa… OP_EQUALVERIFY OP_CHECKSIG

Transaction Details

Input1:
Transaction: f5d8ee39a43…
Transaction Output: 1

scriptSig:
b022100e2acb…
ae2ac980643b…

Input2:
Transaction: 0b82c0e88ff…
Transaction Output: 4

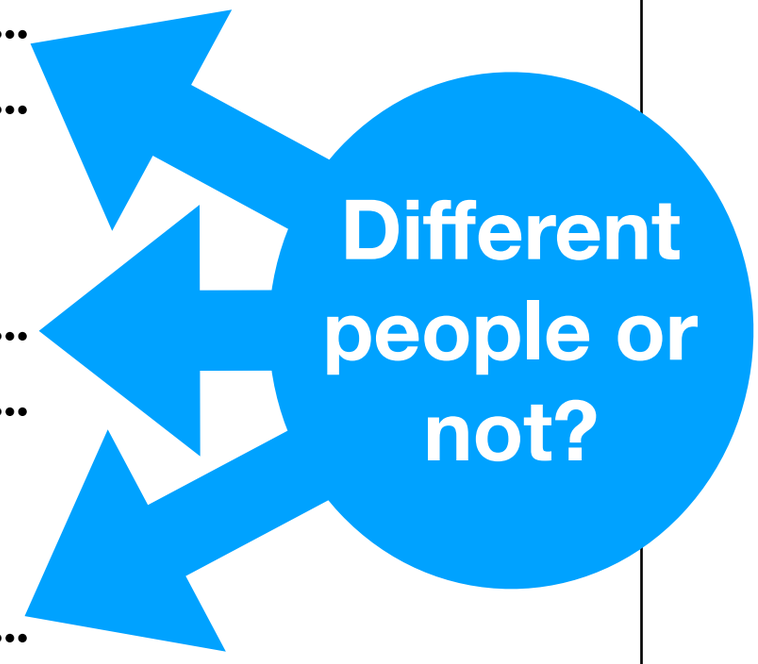
scriptSig:
80643b0b82ca…
467f11e8c0e8…

Input3:
Transaction: c6b64e3e6b3…
Transaction Output: 0

scriptSig:
8d9e14466dad…
222eed3ee373…

Output1:
value: 5000000000
OP_DUP OP_HASH160 304371705fa… OP_EQUALVERIFY OP_CHECKSIG

Output2:
value: 2300530000
OP_DUP OP_HASH160 3b24a405fa… OP_EQUALVERIFY OP_CHECKSIG



CoinJoin Details

- Many different parties create one single transaction

- How can that work?

Bad approach

- Naïve way: Give your money to a bank and hope that the money will be returned

CoinJoin Details

- Trusting other parties with your money is not neccessary

- ScriptSig signatures are sufficiently well designed

Transaction Details

Input1:
Transaction: f5d8ee39a43…
Transaction Output: 1

scriptSig:
b022100e2acb…
ae2ac980643b…

Input2:
Transaction: 0b82c0e88ff…
Transaction Output: 4

scriptSig:
80643b0b82ca…
467f11e8c0e8…

Input3:
Transaction: c6b64e3e6b3…
Transaction Output: 0

scriptSig:
8d9e14466dad…
222eed3ee373…

Output1:
value: 5000000000
OP_DUP OP_HASH160 304371705fa… OP_EQUALVERIFY OP_CHECKSIG

Output2:
value: 2300530000
OP_DUP OP_HASH160 3b24a405fa… OP_EQUALVERIFY OP_CHECKSIG

What are these signatures?

Signatures

- $s = \text{sign}(sk, \text{document})$

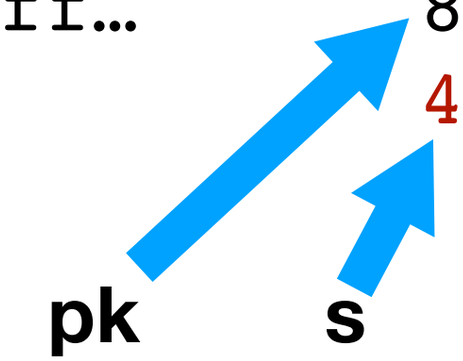
- $\text{verify}(pk, s, \text{document}) \in \{\text{True}, \text{False}\}$

Signatures

- $s = \text{sign}(sk, \text{document})$
- $\text{verify}(pk, s, \text{document}) \in \{\text{True}, \text{False}\}$

Input2:
Transaction: 0b82c0e88ff…
Transaction Output: 4

scriptSig:
80643b0b82ca…
467f11e8c0e8…

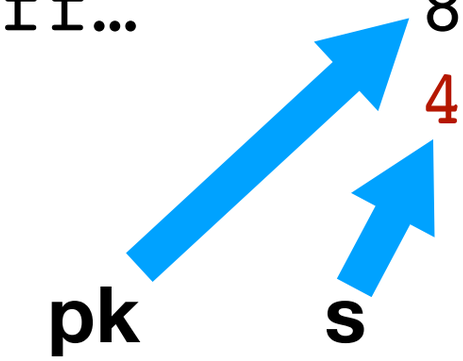


Signatures

- $s = \text{sign}(sk, \text{document})$
- $\text{verify}(pk, s, \text{document}) \in \{\text{True}, \text{False}\}$

Input2:
Transaction: 0b82c0e88ff…
Transaction Output: 4

scriptSig:
80643b0b82ca…
467f11e8c0e8…



Where is the document ?

The document to sign:

Input1: **scriptSig:**
Transaction: f5d8ee39a43… b022100e2acb…
Transaction Output: 1 ae2ac980643b…

Input2: **scriptSig:**
Transaction: 0b82c0e88ff… 80643b0b82ca…
Transaction Output: 4 467f11e8c0e8…

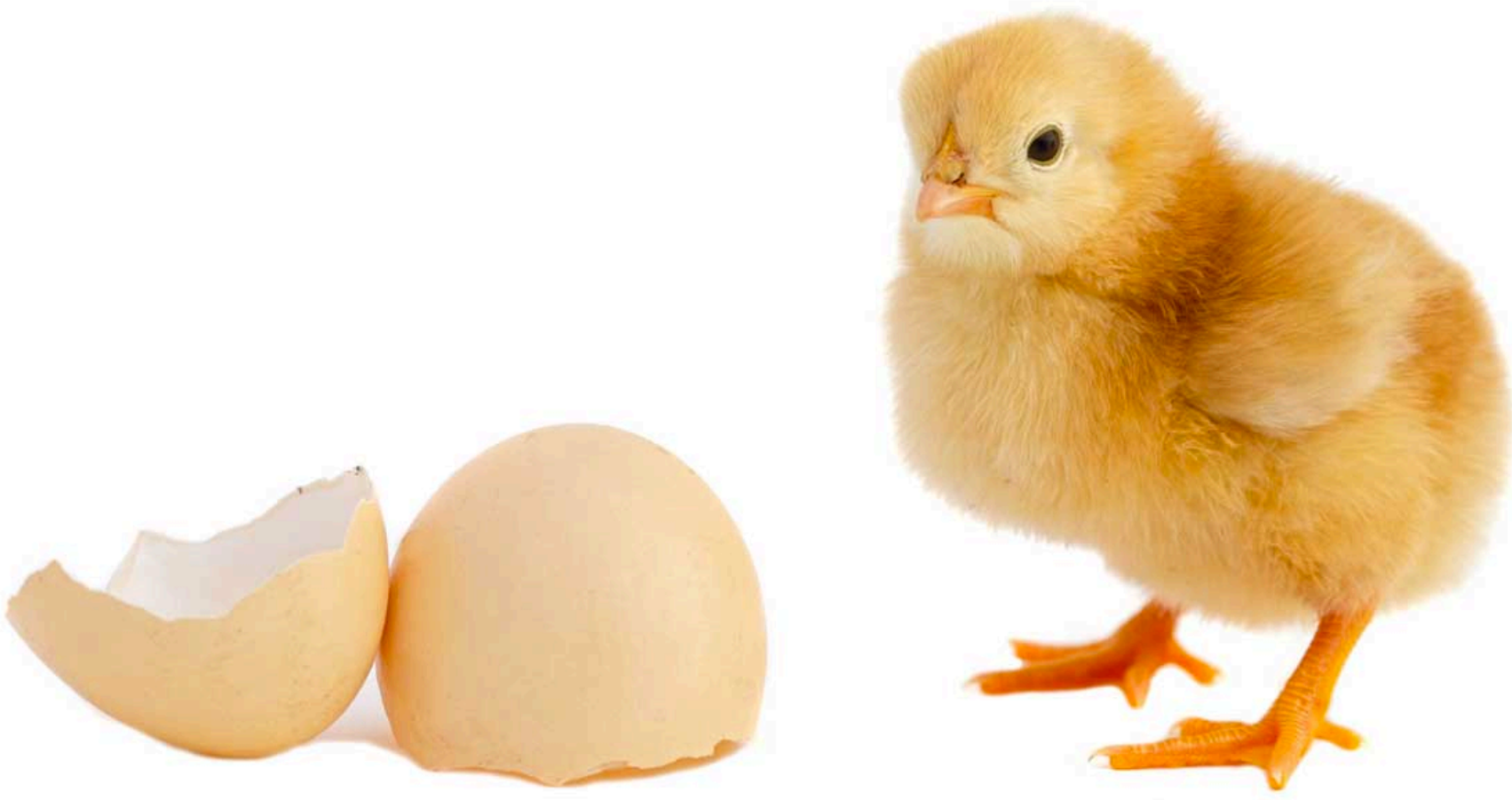
Input3: **scriptSig:**
Transaction: c6b64e3e6b3… 8d9e14466dad…
Transaction Output: 0 222eed3ee373…

Output1:
value: 5000000000
OP_DUP OP_HASH160 304371705fa… OP_EQUALVERIFY OP_CHECKSIG

Output2:
value: 2300530000
OP_DUP OP_HASH160 3b24a405fa… OP_EQUALVERIFY OP_CHECKSIG

Nearly…

- The signature cannot be part of the document itself



The actual document:

Input1: **scriptSig:**
Transaction: f5d8ee39a43…
Transaction Output: 1

Input2: **scriptSig:**
Transaction: 0b82c0e88ff…
Transaction Output: 4

Input3: **scriptSig:**
Transaction: c6b64e3e6b3…
Transaction Output: 0

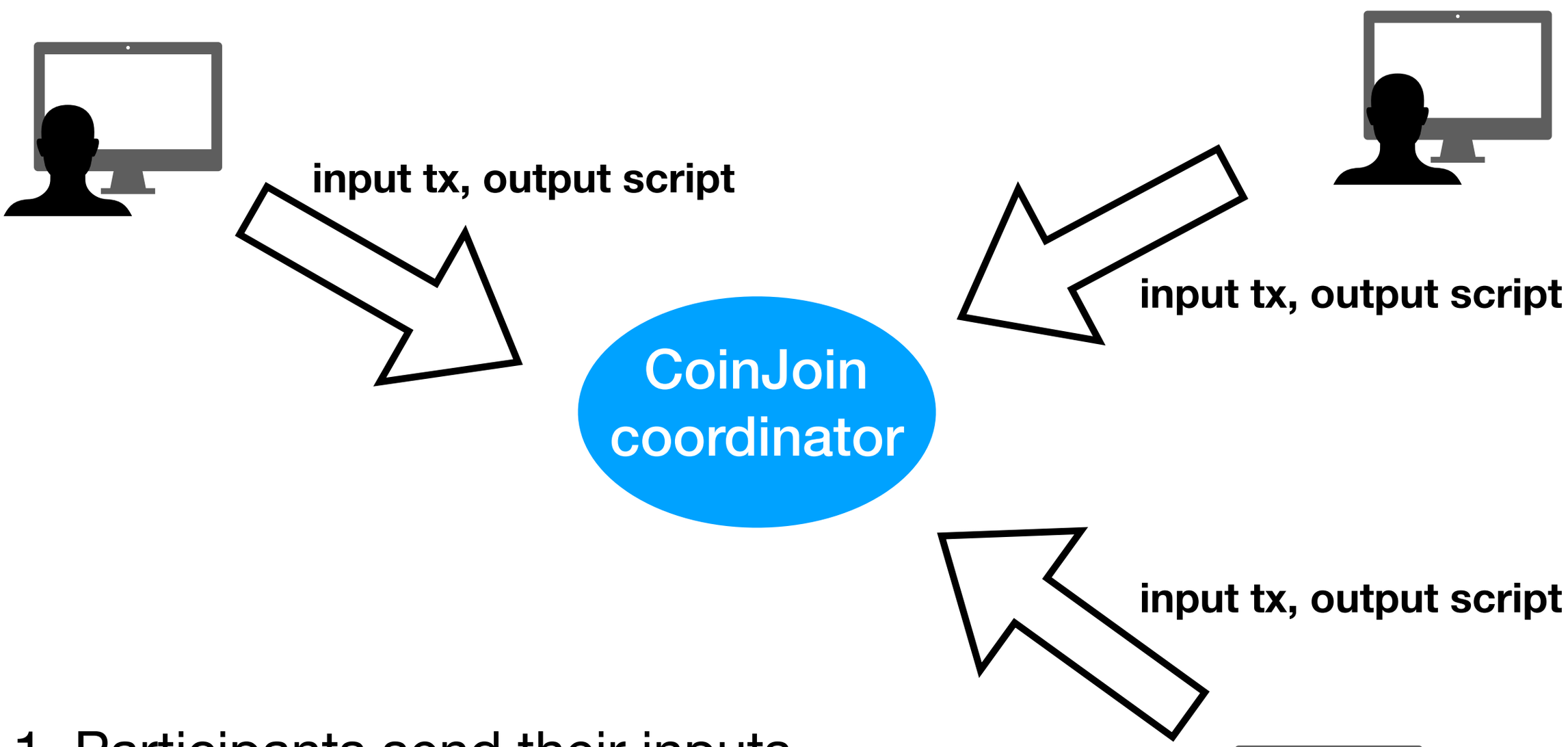
Output1:
value: 5000000000
OP_DUP OP_HASH160 304371705fa… OP_EQUALVERIFY OP_CHECKSIG

Output2:
value: 2300530000
OP_DUP OP_HASH160 3b24a405fa… OP_EQUALVERIFY OP_CHECKSIG

Signing a bitcoin transaction

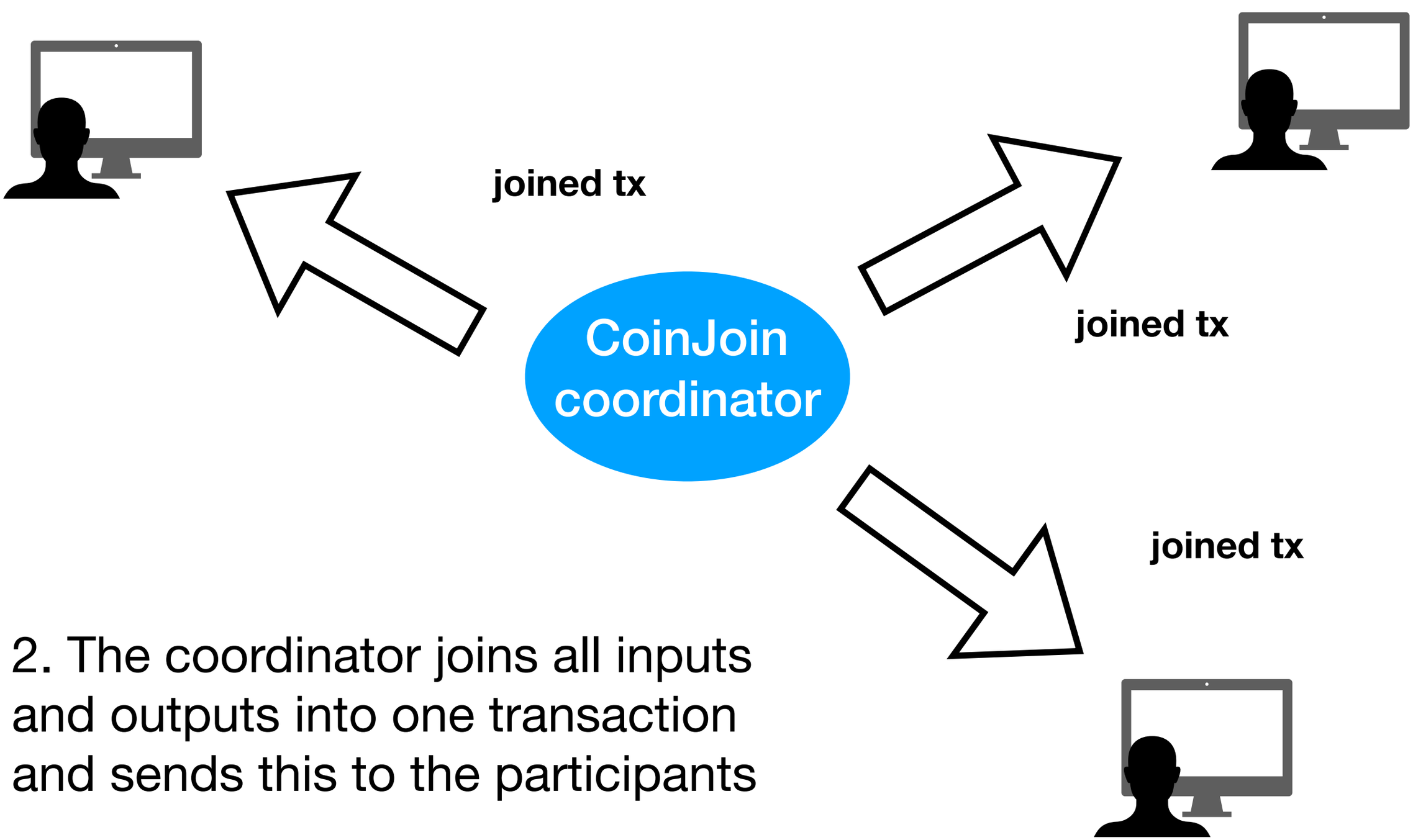
1. Create the transaction, with all inputs and all outputs
2. Remove the scriptSig field
3. Compute s=sign(sk,tx without scriptSig)
4. Insert signatures

CoinJoin



1. Participants send their inputs and outputs to a central coordinator

CoinJoin



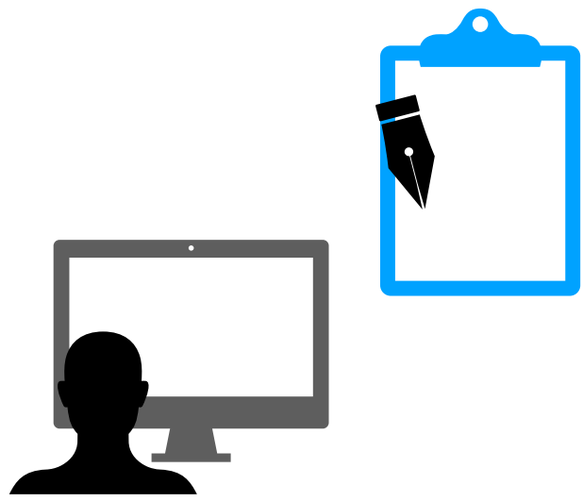
2. The coordinator joins all inputs and outputs into one transaction and sends this to the participants

CoinJoin

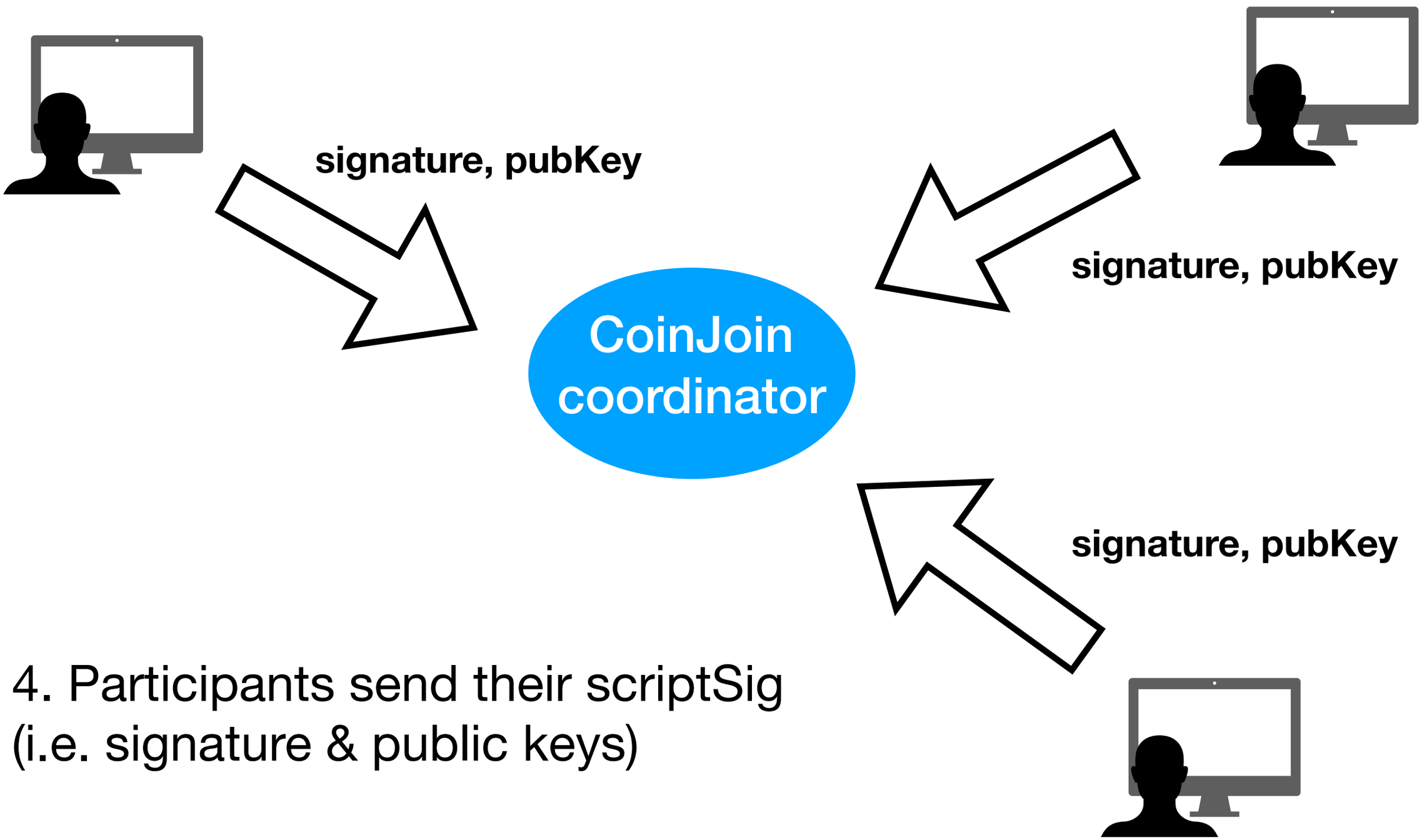
CoinJoin coordinator

3. Each participant creates a signature

Transaction valid only if all participants sign it

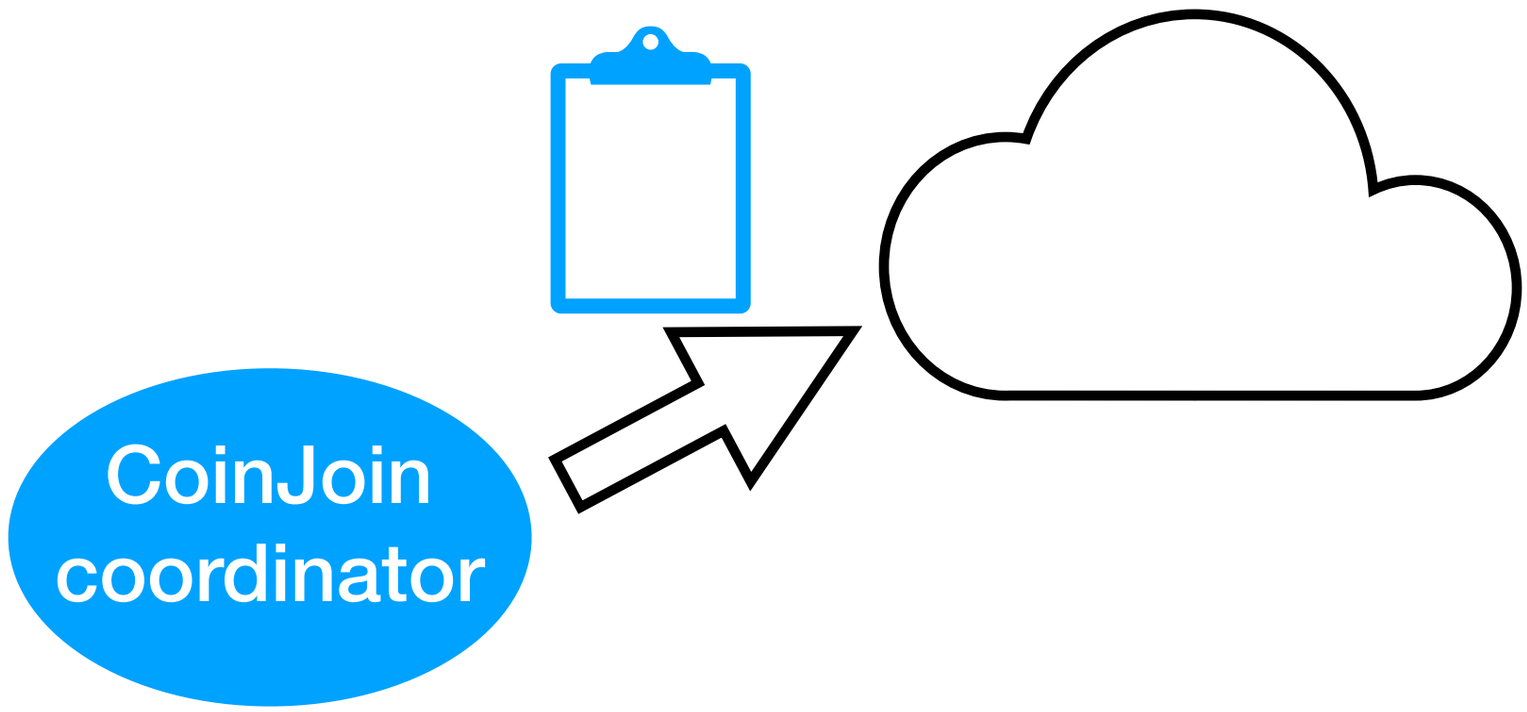


CoinJoin



4. Participants send their scriptSig (i.e. signature & public keys)

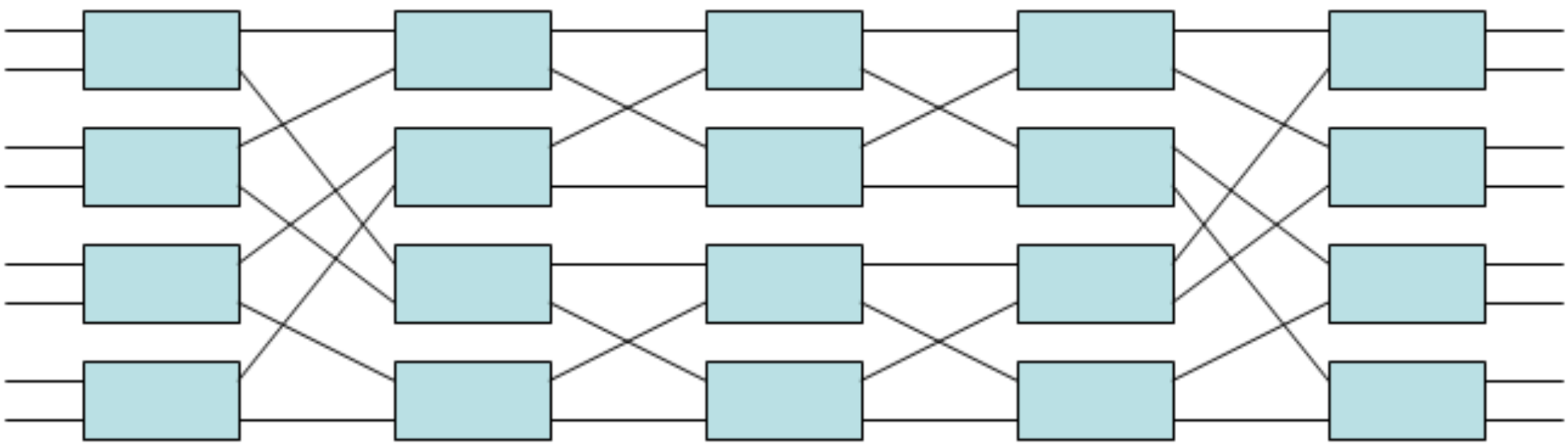
CoinJoin



5. CoinJoin coordinator publishes transaction

Anonymity through mixing

- Mixing does not guarantee anonymity
  - Size of the *anonymity set* important
- If small, use multiple rounds of mixing



CoinJoin limitation

- In the given implementation, the server learns the mapping input -> output
- One person can refuse to sign (DoS attack vector)
- CoinJoin transaction themselves are tainted

TumbleBit

- More complicated implementations exist
- In RSA, signing a document = same mathematical operation as decryption
    - Possible to devise a scheme where the coordinators does not learn anything about the input-output mapping

- Round 1: Clients send Bitcoins to a server in exchange for an anonymous voucher
- Round 2: Clients use the voucher to redeem Bitcoins

    - Related: Atomic Swaps

DoS Attack on CoinJoin

- Transactions can easily be blocked
  - If a client does not sign, a new transaction can be signed without security risks
- CoinJoin servers might be attacked

CoinJoin is tainted

- CoinJoin transactions are significantly more involved in criminal activities

    - Pure participation in CoinJoin can be seen negatively

CoinJoin can be detected

- CoinJoin might seems like a normal transaction, but network analysis can detect CoinJoins
  - Number of input/outputs
  - Origins
  - etc.