Cryptocurrencies

Brice Minaud

email: brice.minaud@ens.fr
website: www.di.ens.fr/brice.minaud/init-crypto.html

Initiation à la Cryptologie, ENS/MPRI, 2019-2020

Meta information

Exam: Monday, May 25, 2pm to Wednesday 27, 5pm.

Register here:

https://www.di.ens.fr/david.pointcheval/cours.html

All other info for this course, including past lectures/TAs:

https://www.di.ens.fr/brice.minaud/init-crypto.html

New info:

- Write in .pdf/.txt, or if needed scan *legible* handwritten text.
- Upload to cloud server (see David's page) by deadline.

Roadmap

1. Before bitcoin: electronic cash.
2. Bitcoin.
3. Limitations, Anonymity vs Pseudonymity.
4. Other cryptocurrencies: Monero, Zcash.

Electronic Cash

Electronic cash

Electronic Money: credit cards etc.

$\neq$ Electronic Cash: not traceable.

For now, consider traditional “bank-based” money.

First goal: unforgeability. Impossible for third party to forge coins.

Unforgeability

Idea: bank signs the coin.

Similar to traditional bank notes.

"Signature"

Unforgeability

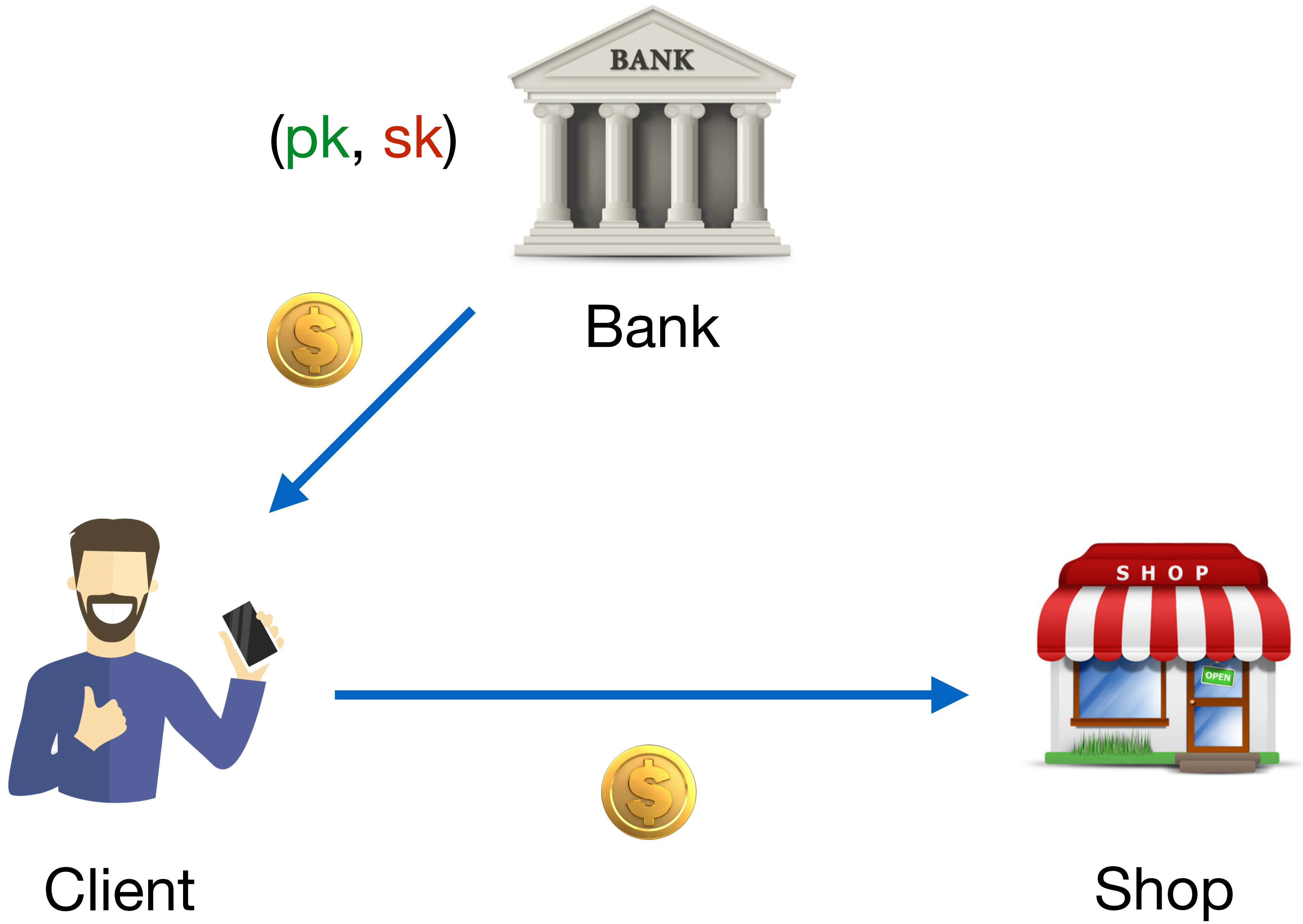
Cryptographic signatures:

- (publick key, secret key) pair.
- Only signer who knows secret key can sign.
- Anybody can check signature using public key.

The bank has a public key/secret key pair (pk, sk).

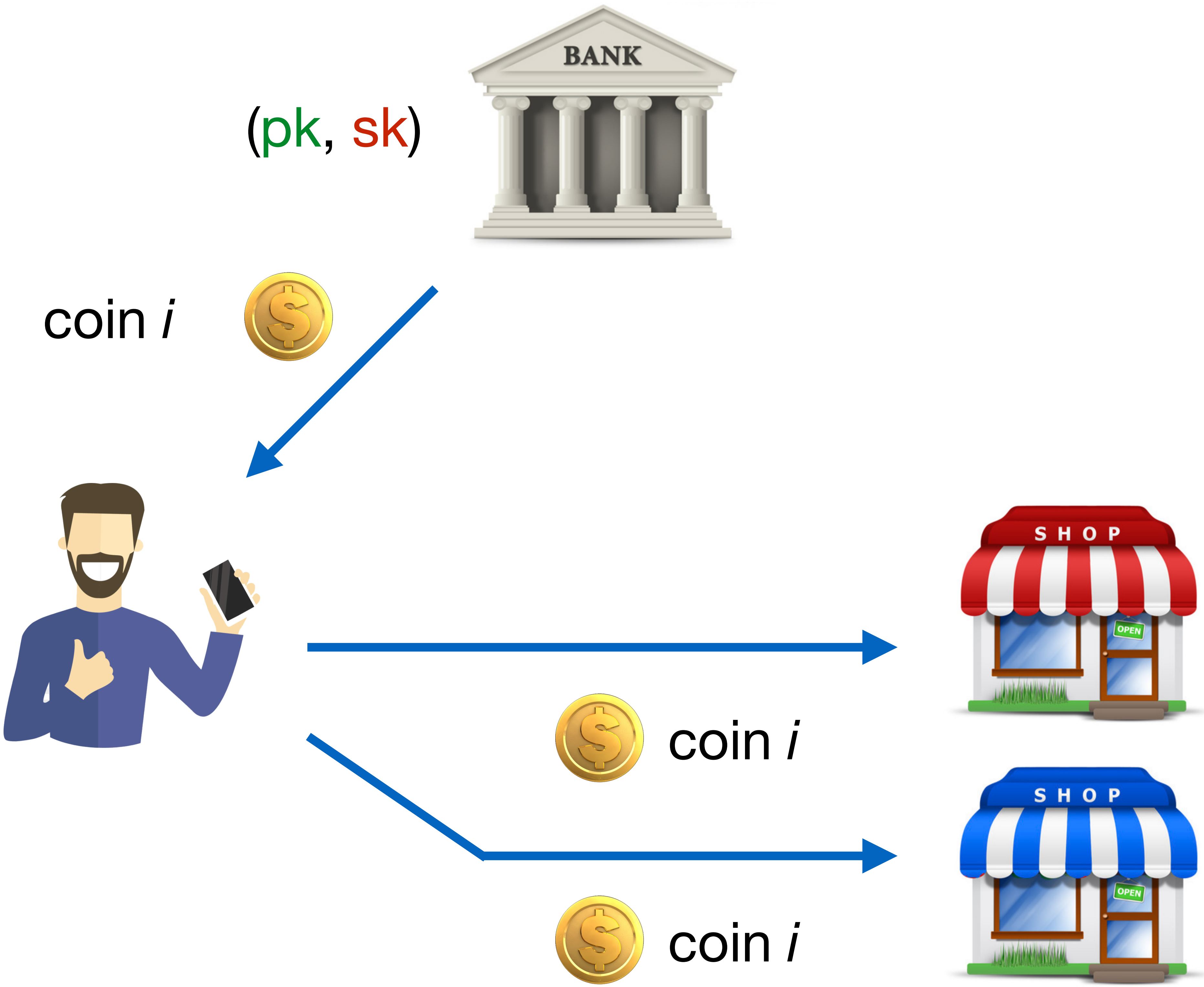
To issue a coin, bank signs the message “coin-ID”.

Setup



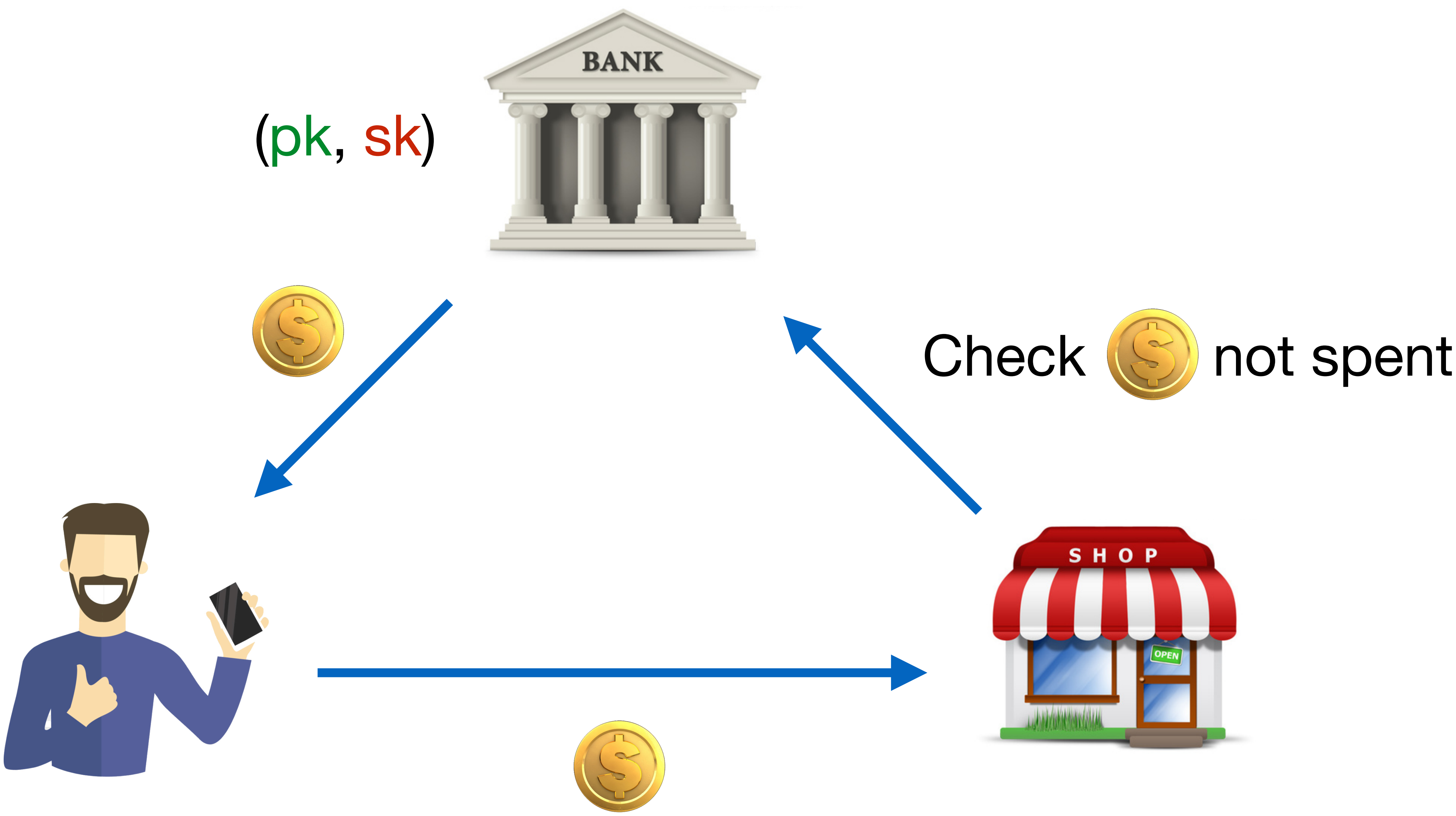
Signed coin = (coin-ID, sign_{sk}(coin-ID))

Problem: double spending



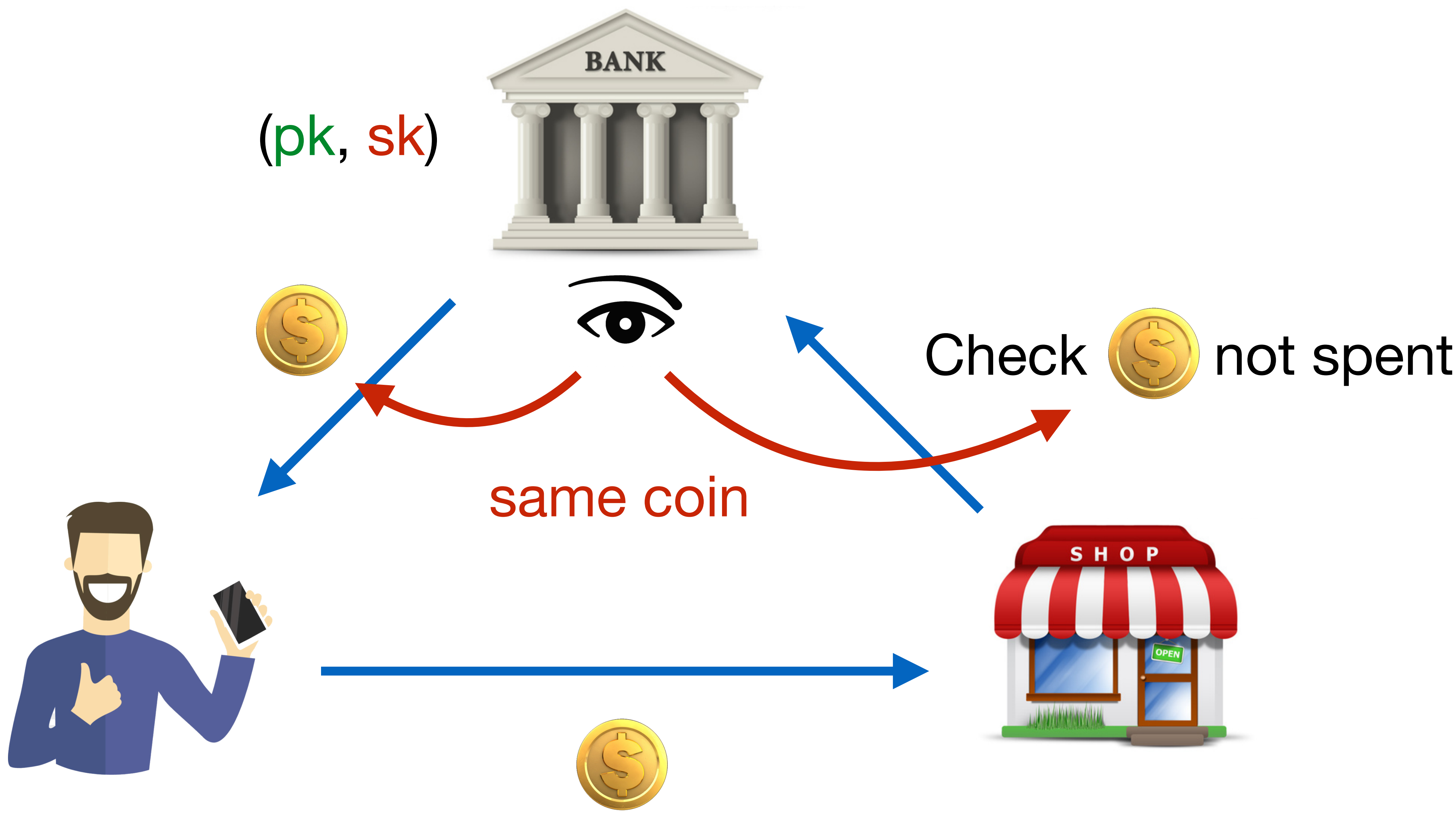
Fundamental problem with electronic money.

Solution



Signed coin = (**coin-ID**, sign_{sk}(coin-ID))

Problem: traceability



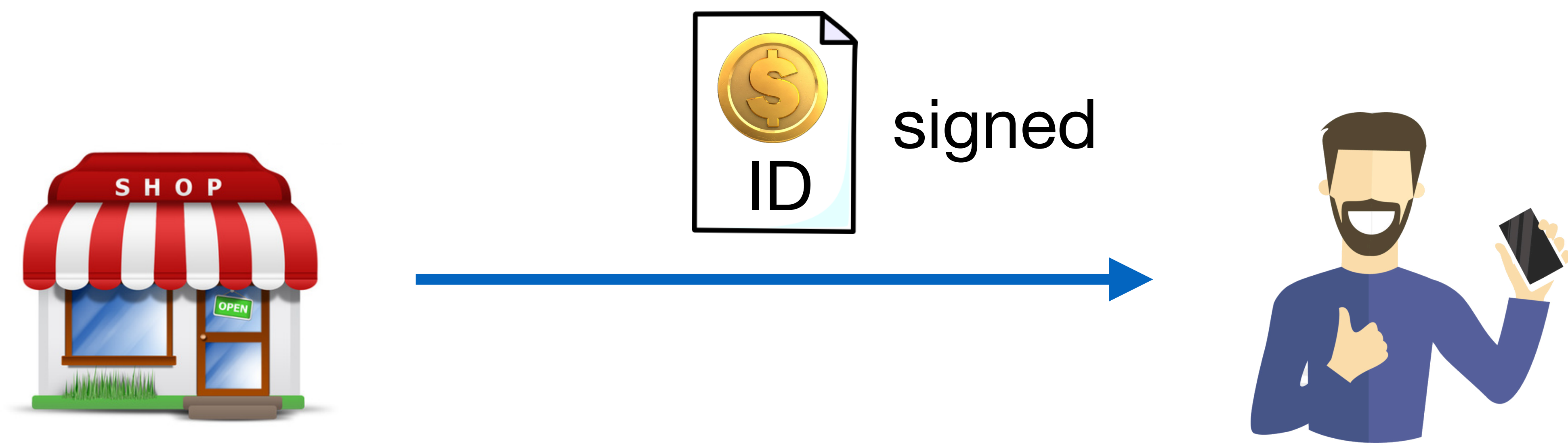
Signed coin = (**coin ID**, sign_{sk}(coin ID))

Traceability: this is electronic money, not cash.

Solution: blind signatures

Idea: bank signs coin-ID *without knowing* coin-ID.

Current naive solution:

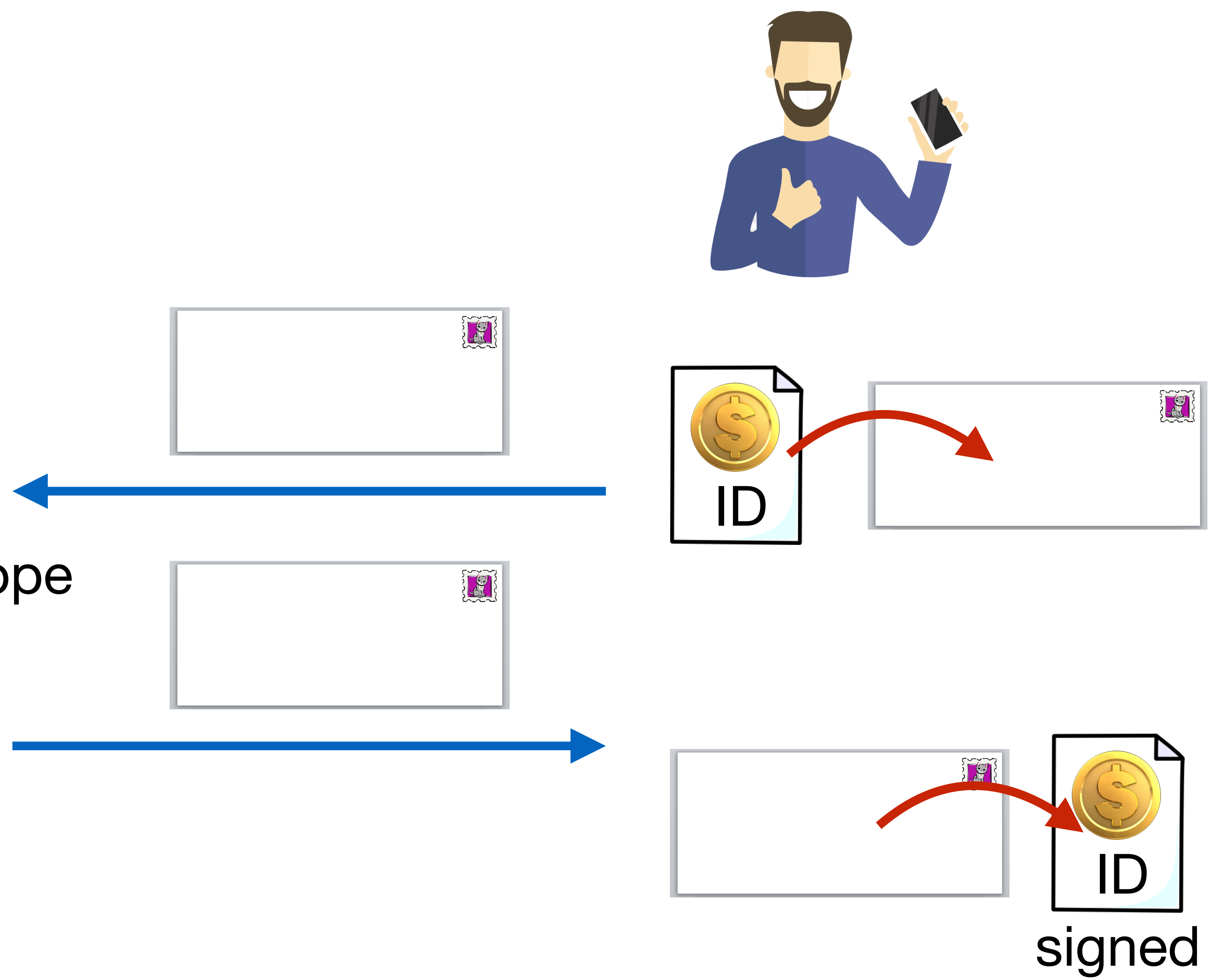


coin-ID ← $\{0,1\}^n$

$\sigma \leftarrow \text{sign}_{sk}(\text{coin-ID})$

Solution: blind signatures

With blind signatures:



Sign *through* envelope

(carbon copy)

Solution: blind signatures

Electronic version:

coin-ID $\leftarrow \{0,1\}^n$

envelop(coin-ID)

$\sigma' \leftarrow$ sign$_{sk}$(envelop(coin-ID))

σ'

$\sigma \leftarrow$ develop(σ')

We want: develop ○ sign ○ envelop = sign

RSA signatures

- Select a pair of random primes p, q. Set $N = pq$.
- Select integers d, e such that $de = 1 \bmod (p-1)(q-1)$.
  - The public key is $pk = (e,N)$.
  - The secret key is $sk = d$.

Sign: for a message m, the signature is:

$$\sigma = m^d \bmod N.$$

Verify: for a message $m \in [1,N\text{-}1]$, signature σ, check:

$$m = \sigma^e \bmod N.$$

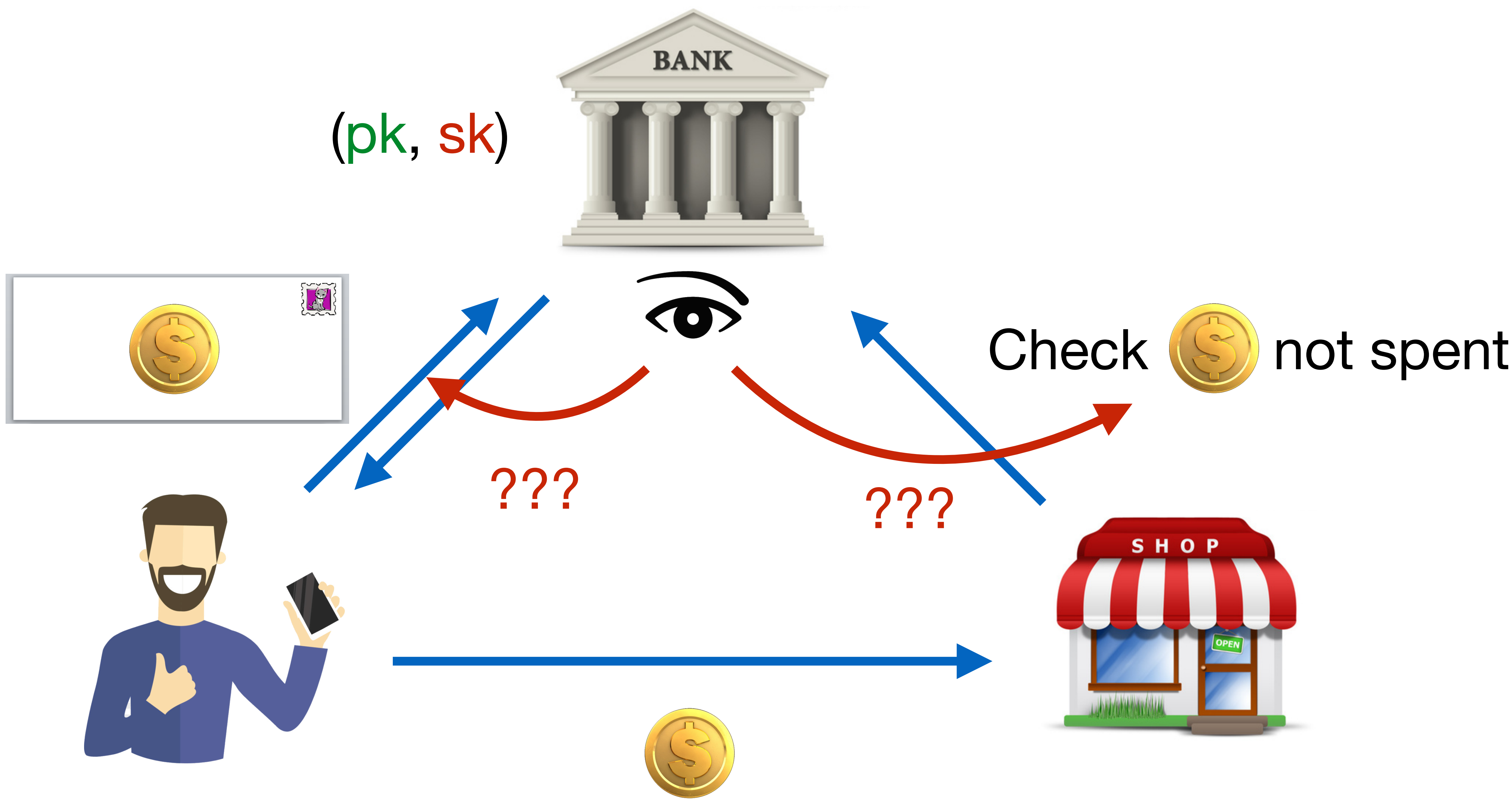
Envelop: pick $r \leftarrow [1,N\text{-}1]$ uniformly, output envelop(m) = $m \cdot r^e$.

Develop: develop(σ') = $\sigma' \cdot r^{-1}$.

Indeed, sign(envelop(m)) = $(m \cdot r^e)^d = m^d \cdot r$.

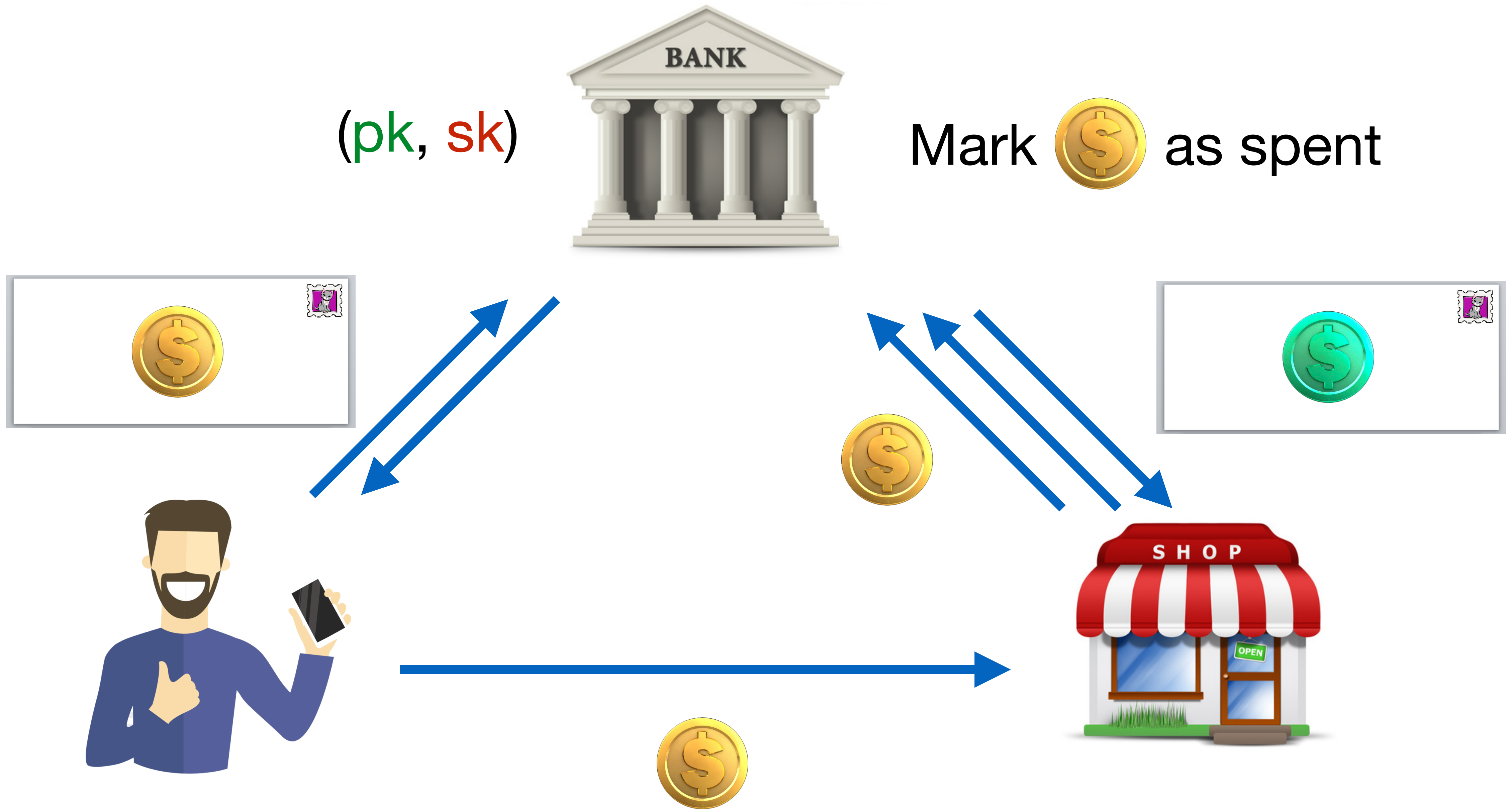
$\Rightarrow$ develop(sign(envelop(m))) = m^d = sign(m)

Chaum ’83: untraceable payments



Signed coin = (coin-ID, sign_{sk}(coin-ID))

Chaum ’83: untraceable payments



Signed coin = (coin-ID, sign_{sk}(coin-ID))

Signed coin = (coin-ID’, sign_{sk}(coin-ID’))

Chaum '83: untraceable payments

This is electronic cash.

Unforgeability: signatures.

Untraceability: blind signatures.

But requires central authority.

Bitcoin: decentralized system.

- Trust: no trust required on central authority.
- Economics: no possibility for authority to mint coins at will.

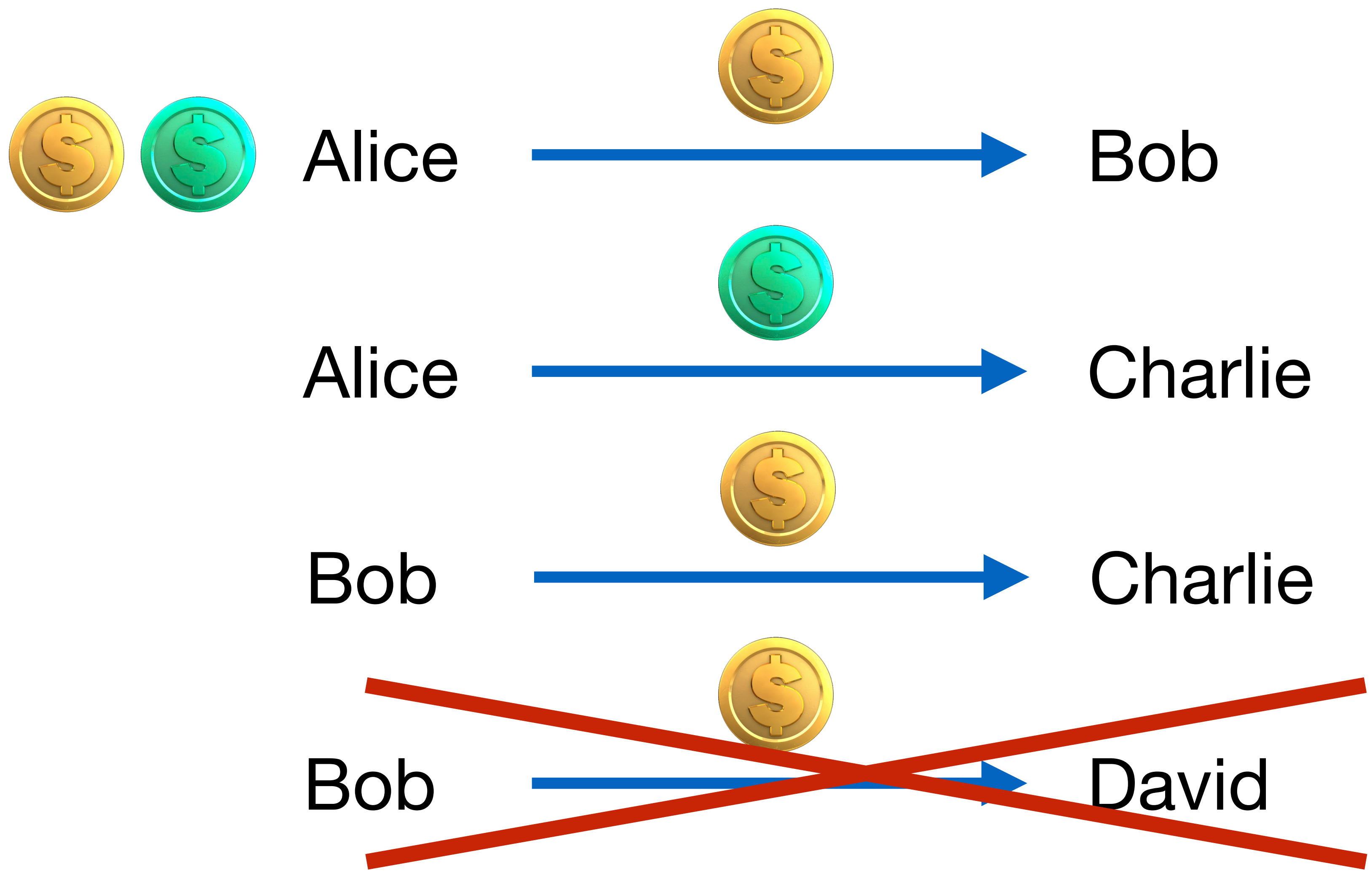
Bitcoin

Public ledger

No bank → who checks validity of transactions? (no double spending)

Idea: just publish all transactions! Everybody can check.

Public ledger:



Public ledger

How to prevent people from writing any transaction they want?

An account is a (public key, secret key) pair for signature scheme.

Pseudo-anonymity: account is just a key.

Alice  Bob

Alice: (pk_A, sk_A).

Bob: (pk_B, sk_B).

Ledger: $pk_A \longrightarrow pk_B$

$+ \text{sign}_{sk_A}(pk_A \rightarrow pk_B)$

Accounts

Ledger: $pk_A \longrightarrow pk_B$

$+ \text{sign}_{skA}(pk_A \rightarrow pk_B)$

How do you know pk_A has the money?

Comes from previous transaction (**tx**) in the ledger (chain).

$pk_C \longrightarrow pk_A$

$+ \text{sign}_{skA}(pk_C \rightarrow pk_A)$

$pk_A \longrightarrow pk_B$

$+ \text{sign}_{skA}(pk_A \rightarrow pk_B)$

Fungibility

One transaction:

Input amount	Input key		Output key	Output amount
3	pk_A	→	pk_D	2
1	pk_B		pk_E	1.5
1	pk_C		pk_A	1.5
$\Sigma = 5$		↔		$\Sigma = 5$

+ signatures with sk_A, sk_B, sk_C.

Payback: pk_A is giving the change back to itself.

Public ledger, revisited

Ledger is a chain of transactions.

3	pk_A	pk_D	2
1	pk_B	pk_E	1.5
1	pk_C	pk_A	1.5
+ sign sk_A, sk_B, sk_C.			

tx 1

1.5	pk_A	pk_H	2
2	pk_F	pk_I	1
0.5	pk_G	pk_A	1
+ sign sk_A, sk_F, sk_G.			

tx 2

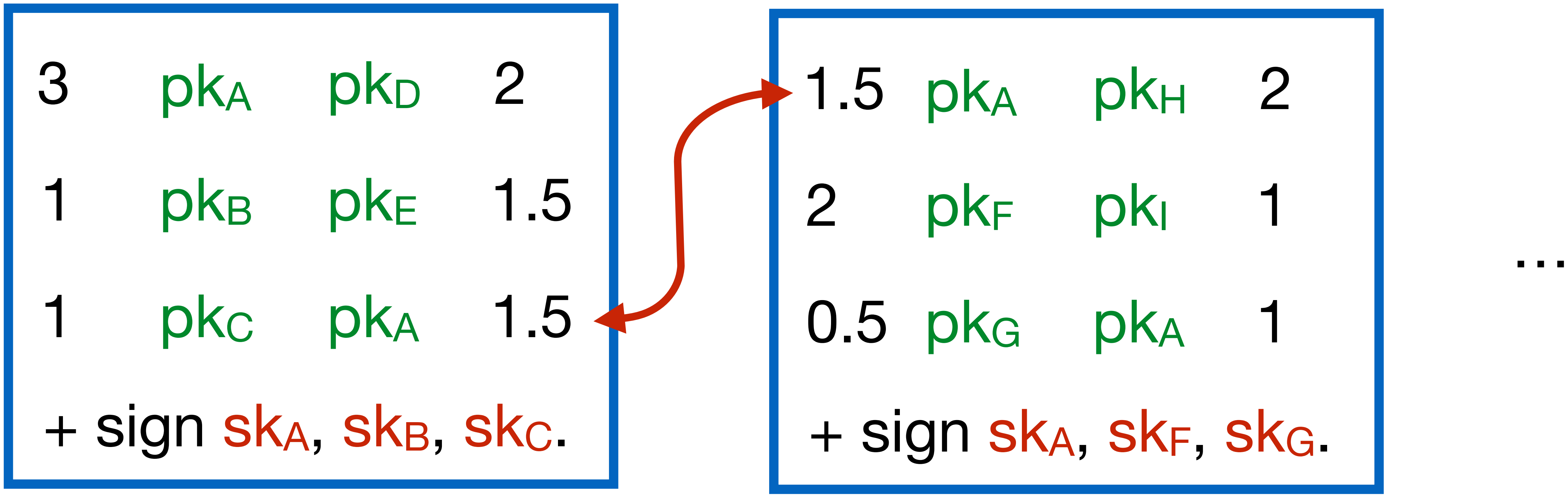
...

No real notion of account: every tx input links to previous unspent tx output (utxo).

To receive money, user can create new “account” (pk, sk) as destination, for every tx.

Public ledger, revisited

Ledger is a chain of transactions.



tx 1 tx 2

Assume for now there are some atomic coins somewhere.

As long as everybody agrees on state of ledger, this just works!

⇒ Whole problem is agreement.

Bitcoin can be viewed as an agreement protocol.

Agreement

How to ensure everybody agrees on state of ledger?

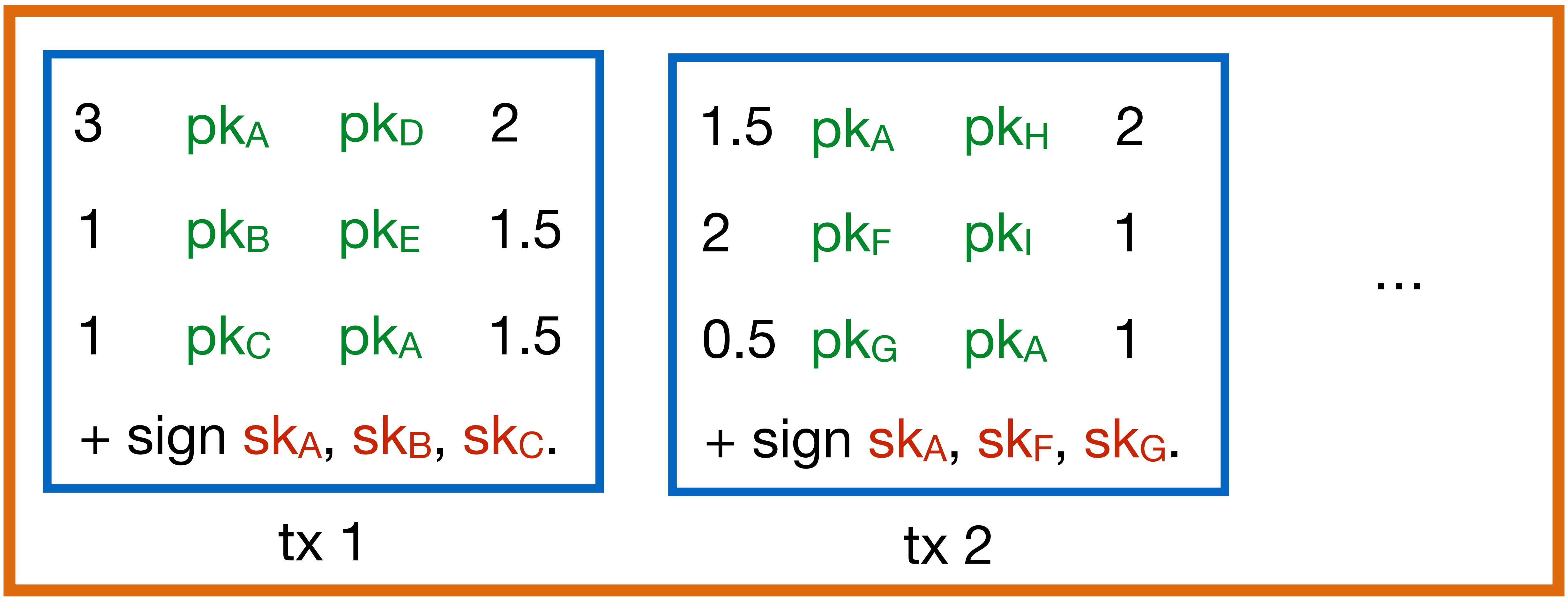
Two components:

1. Blockchain.

2. Mining.

The blockchain

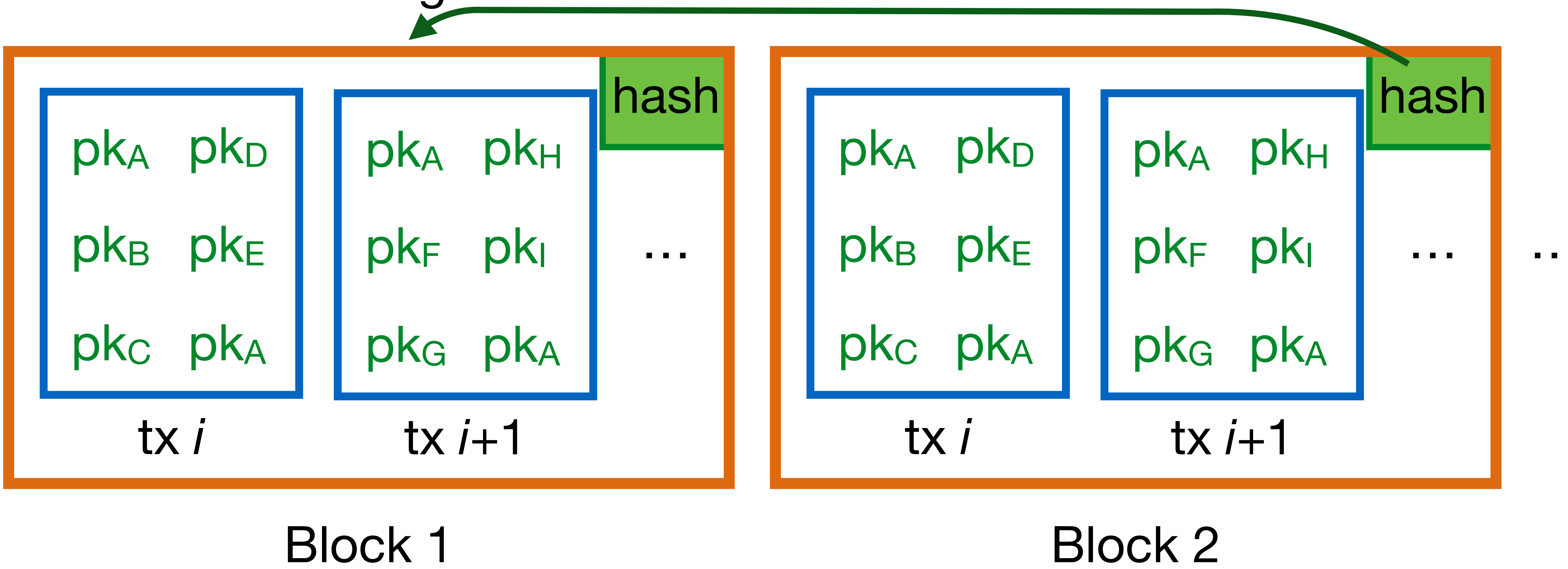
Transactions are arranged into blocks.



One block

The blockchain

Blocks are arranged into a chain.



Each new block contains hash(previous block).

Cryptographic hash function

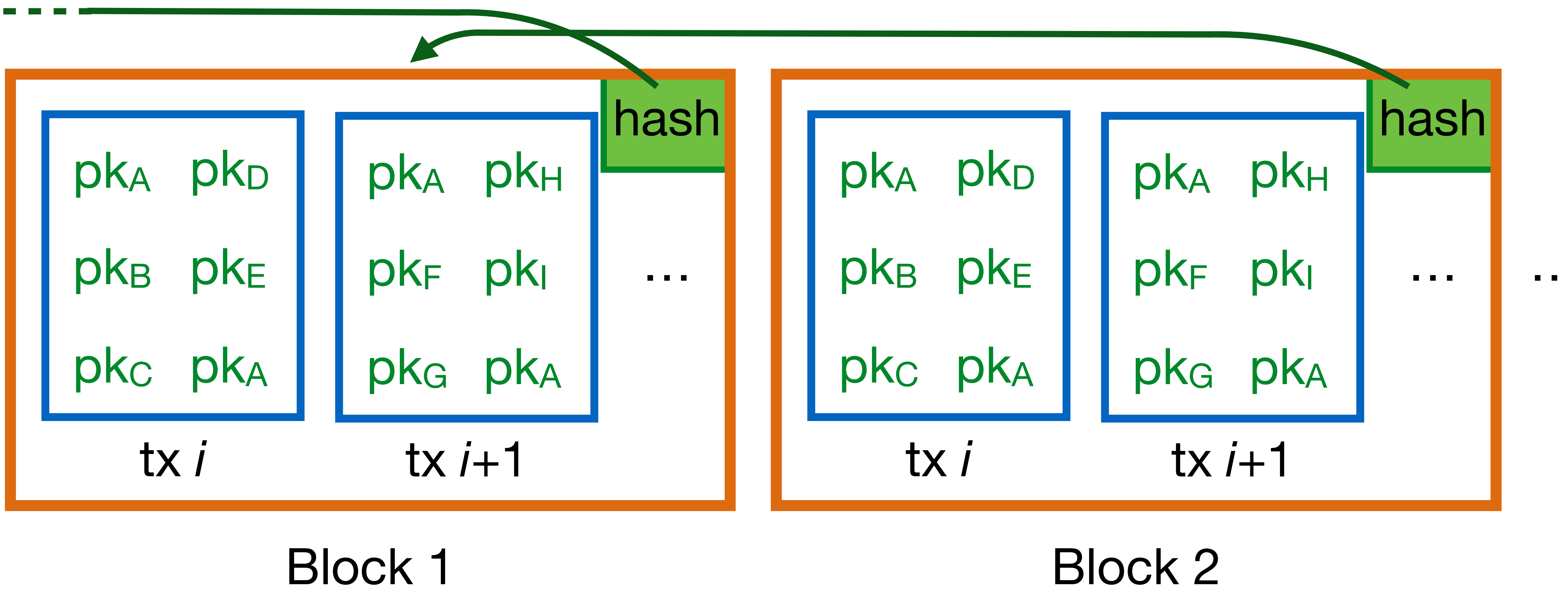
Hash function H: $\{0,1\}^* \rightarrow \{0,1\}^n$.

Preimage resistance: for uniform $y \in \{0,1\}^n$, hard to find x such that $H(x) = y$.

Collision resistance: hard to find $x \neq y \in \{0,1\}^*$ such that $H(x) = H(y)$.

⇒ a hash value $H(x)$ uniquely determines its input x (in a computational sense). It is very short (e.g. 256 bits).

The blockchain



Each new block contains hash(previous block).

⇒ by induction, hash uniquely identifies entire preceding chain (in a computational sense).

Mining

Now the problem is ‘just’ to agree on the next block.

Idea: any user can propose the next block.

But two more ingredients…

- Proof of work: proposing next block is difficult, so not too many users propose at the same time.
- Forks: how to resolve conflicts.

Proof of work

Bitcoin proof of work: when adding a block B, user must provide value r such that hash(B,r) begins with n zeros.

Requires 2^n hash computations on average.

Hash function for bitcoin: SHA-256.

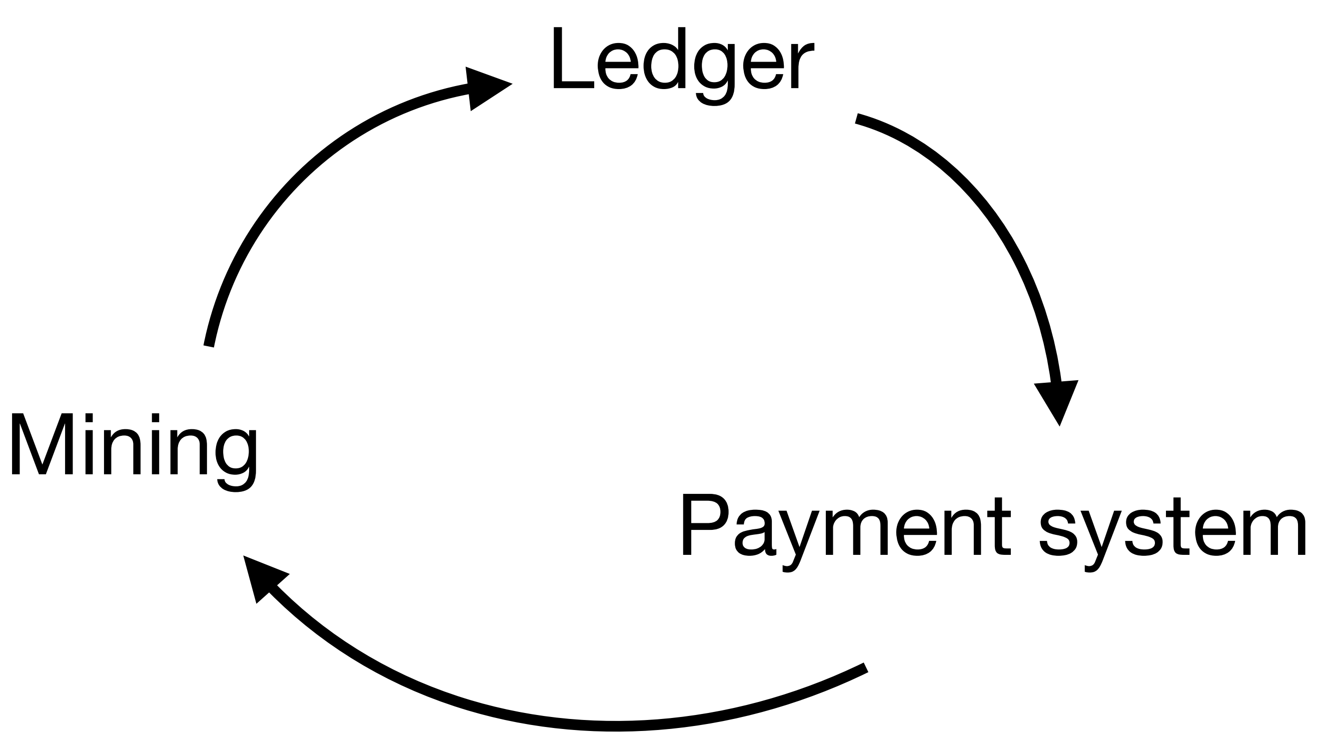
Proof of work

The difficulty (#hashes required to find new block) is adjusted every 2 weeks.

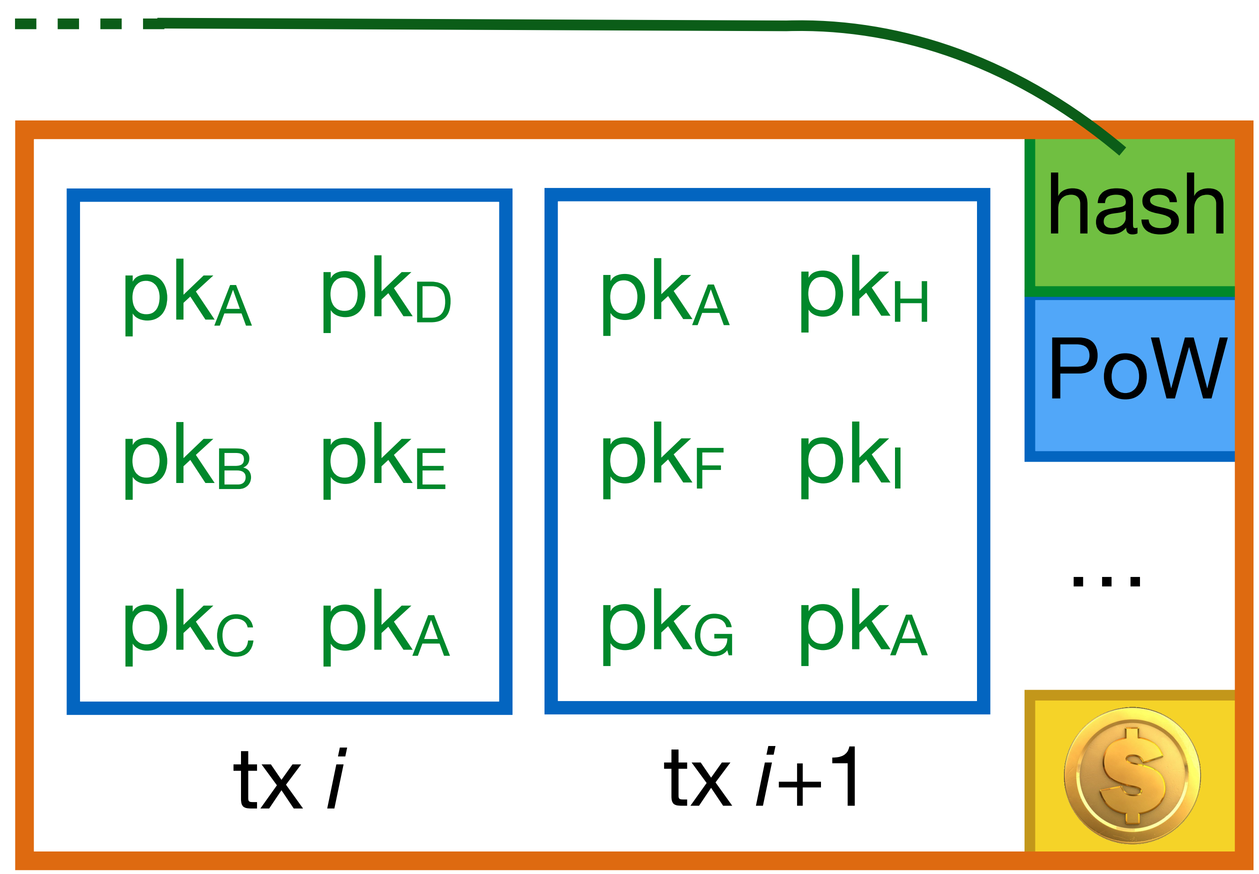
For Bitcoin, about 2^{76} hashes per block today…

How to incentivize miners?

Give them bitcoins!



A block



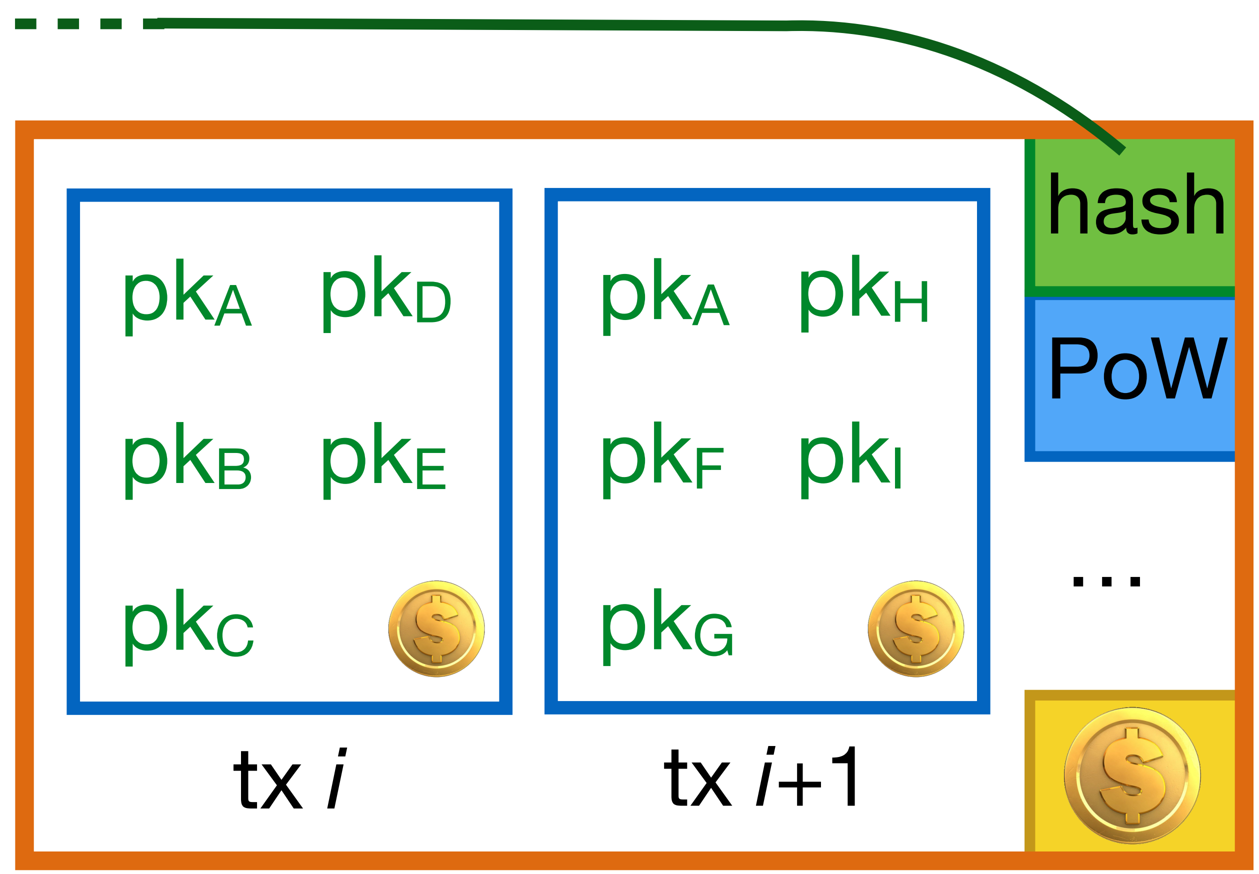
Block

Each new block affords C bitcoins.

Currently C = 12.5, halved every four years. Happened last week!

This is how all fresh bitcoins happen.

Fees



Block

In addition, miner collects fees from each tx.

Total block size is limited to ~1Mb.

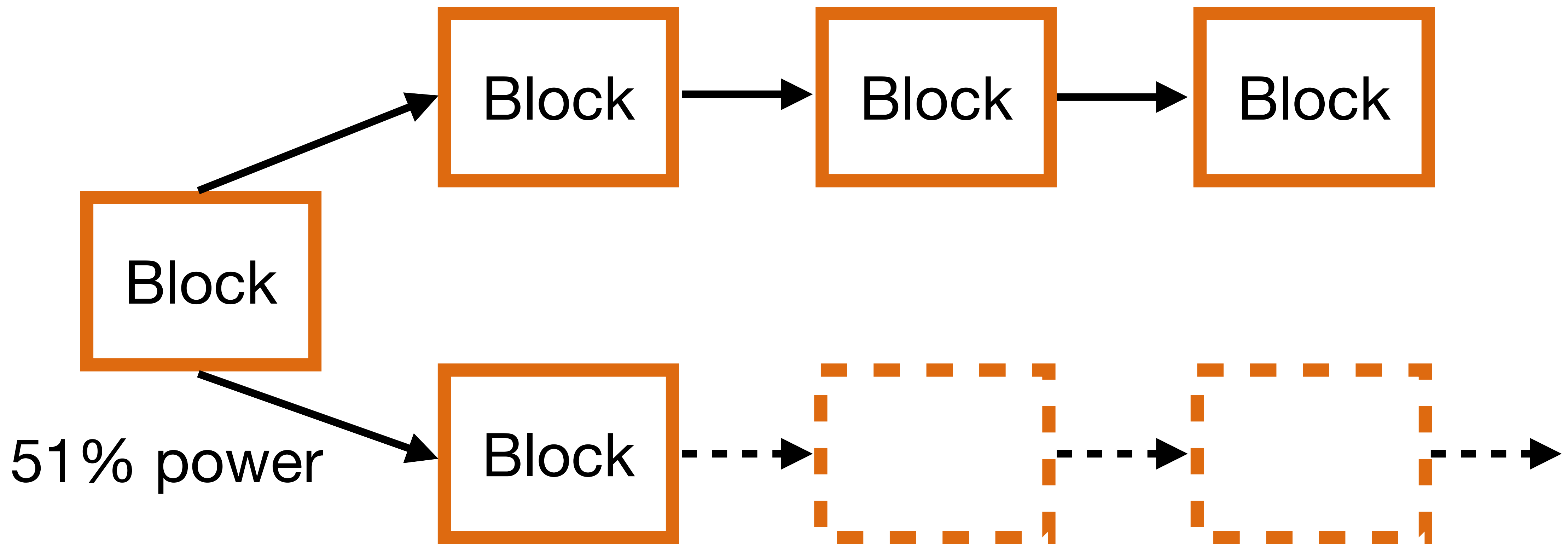
Published tx’s with higher BC/byte get prioritized by miners.

Forks

How to resolve conflicts?

Idea: mine on the longest chain.

Limitation: fails if 51% of mining power colludes.



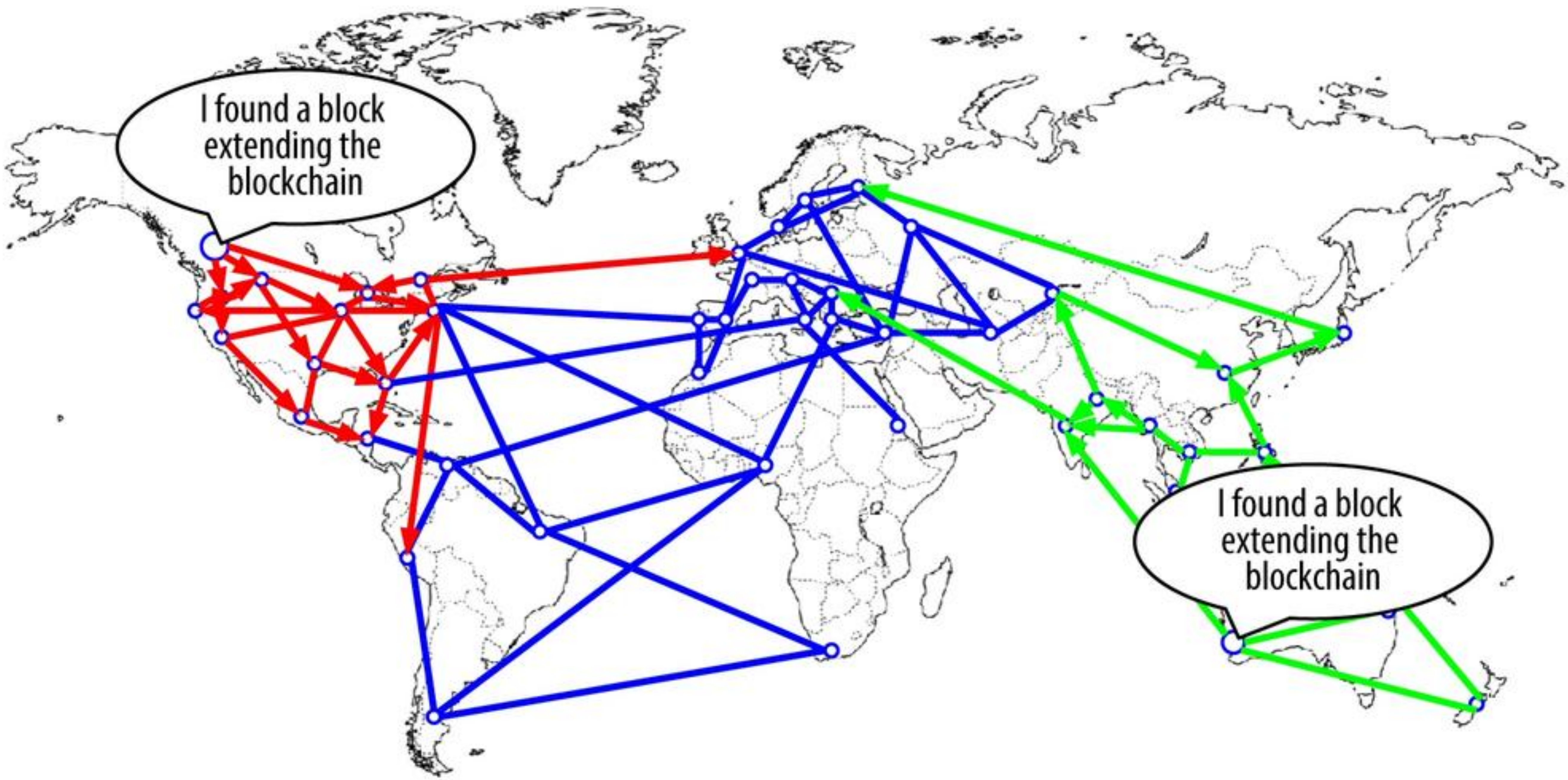
Trust assumption of BC is trust on honest majority.

Forks

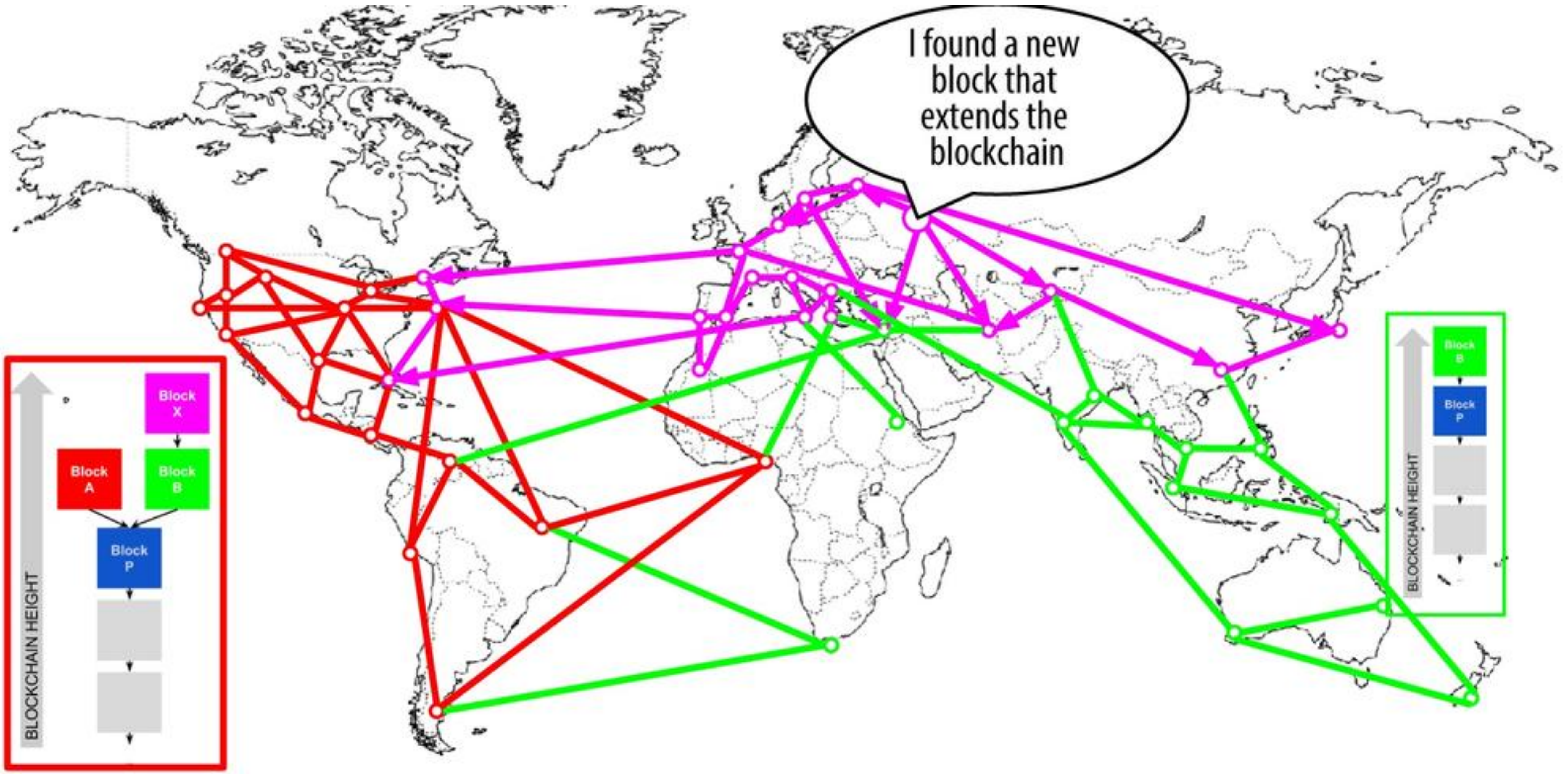
How to resolve conflicts?

Borrowed from https://www.oreilly.com/library/view/mastering-bitcoin/9781491902639/ch08.html

Forks

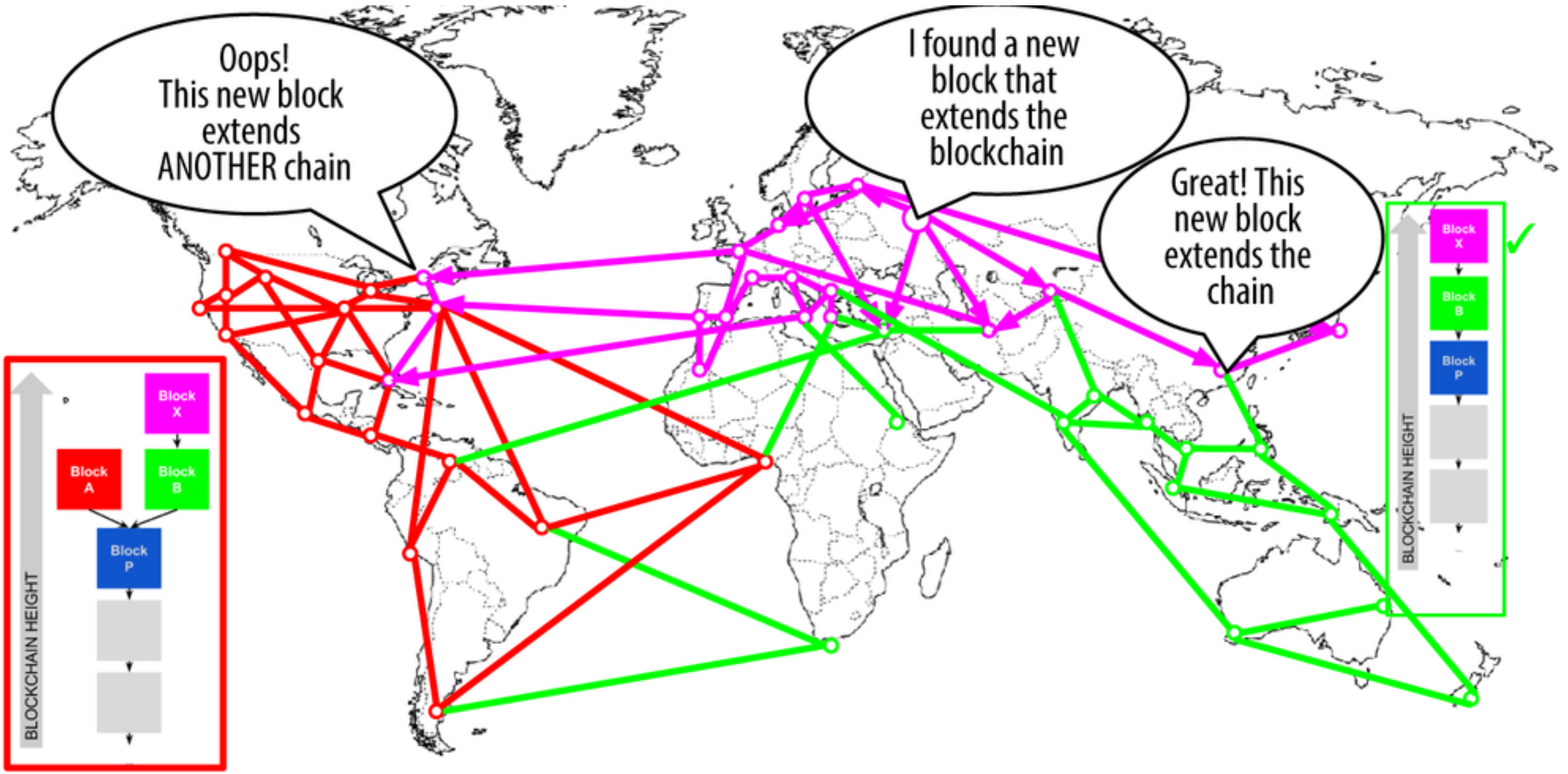

I found a block extending the blockchain
I found a block extending the blockchain

Forks



I found a new block that extends the blockchain
Block X
Block A
Block B
Block P
BLOCKCHAIN HEIGHT
Block B
Block P
BLOCKCHAIN HEIGHT

Forks



Need to wait for a few blocks to confirm transaction (1-6).

A few numbers

New block every ~10 min. Block size: 1 Mb. ~2000 tx/block.

Currently 18.5m BC mined. Out of 21m total.

Current total blockchain size: ~250Gb, about +60Gb/year.

Transaction fees: order is roughly $1/transaction (very variable).

Limitations of bitcoin

Quantitative issues:

- Long confirmation: 10+ minutes.
- Expensive for small transactions.
- Scalability is questionable. Whole tx history stored.

Qualitative issues:

- Pseudo-anonymity.
- Proof of Work = huge energy waste.

Most of these problems have ‘solutions’ within Bitcoin.

Other cryptocurrencies also offer alternatives.

Beyond Bitcoin

Proof-of-Work alternatives

Problems with PoW:

- Energy waste
- Advantage to ASICs.

Challenge: avoid Sybil attacks.

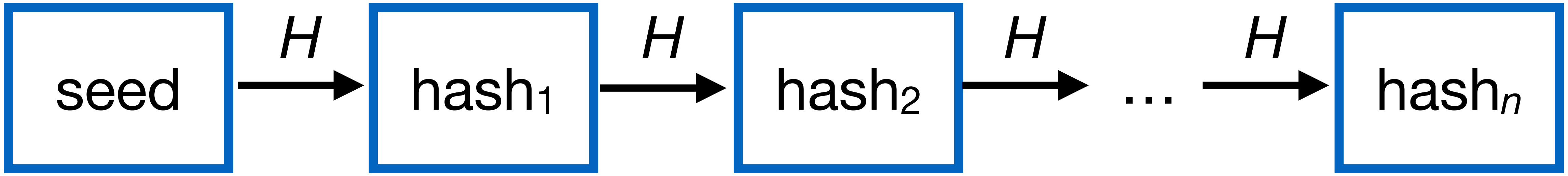
Alternatives:

- Proof of Space: memory-hard functions.

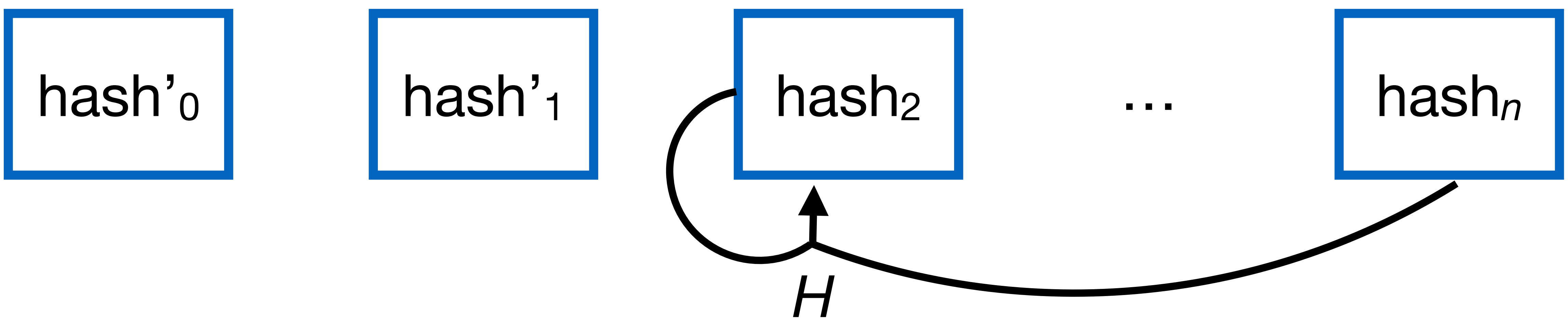
  Does not favor dedicated circuits as much.

Memory-hard function

Step 1: fill fixed amount of memory with randomness.



Step 2: replace each cell with hash(current cell, random cell).



Step 3: repeat step 2 several times.

Step 4: output hash of memory.

Proof-of-Work alternatives

Problems with PoW:

- Energy waste
- Advantage to ASICs.

Alternatives:

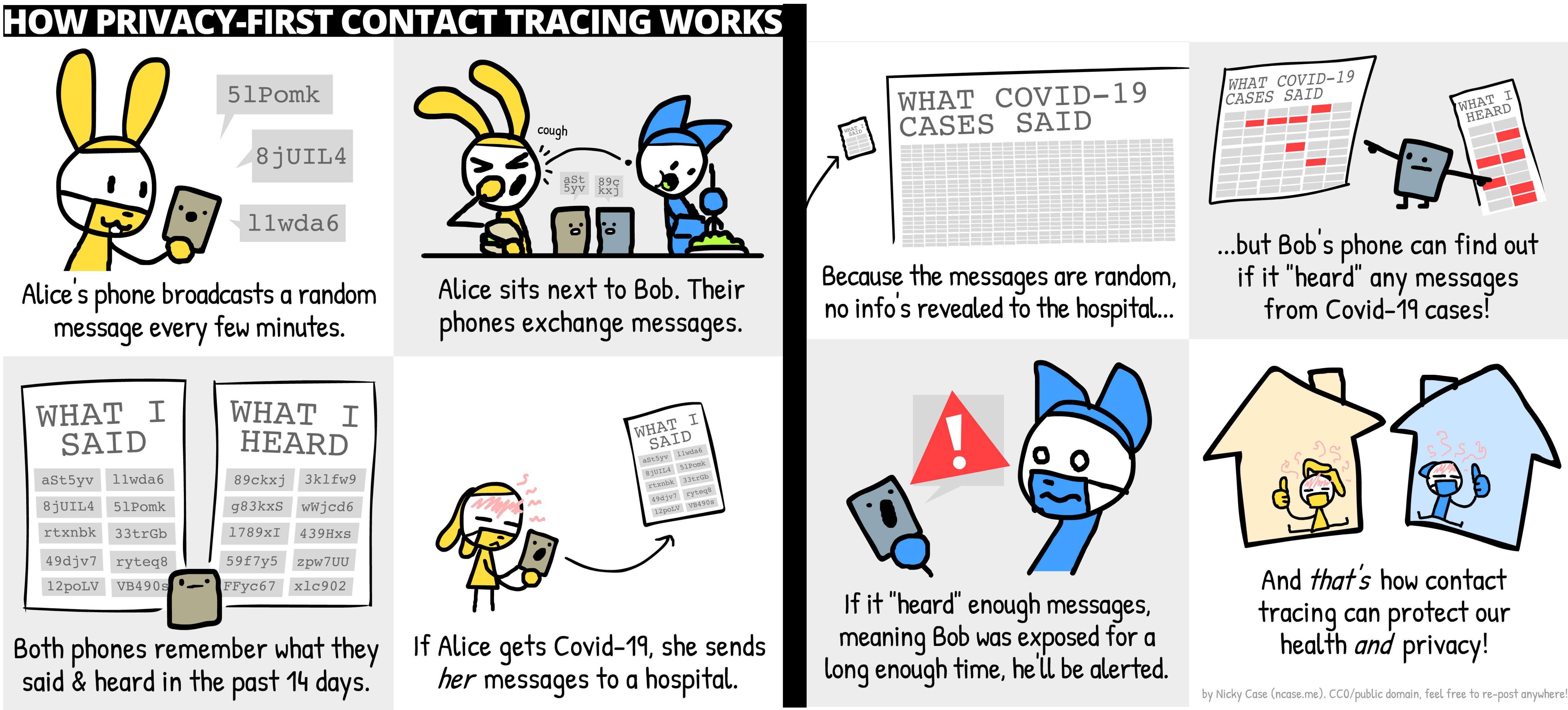
- Proof of Space: memory-hard functions.

  Does not favor dedicated circuits as much.
- Proof of Stake: choose random user based on amount of currency owned.

  No energy waste.

Anonymity vs pseudonymity

Let's talk about Covid for a moment!



p. 1 p. 2

Source: https://github.com/DP-3T/documents/tree/master/public_engagement/cartoon

Anonymity vs pseudonymity

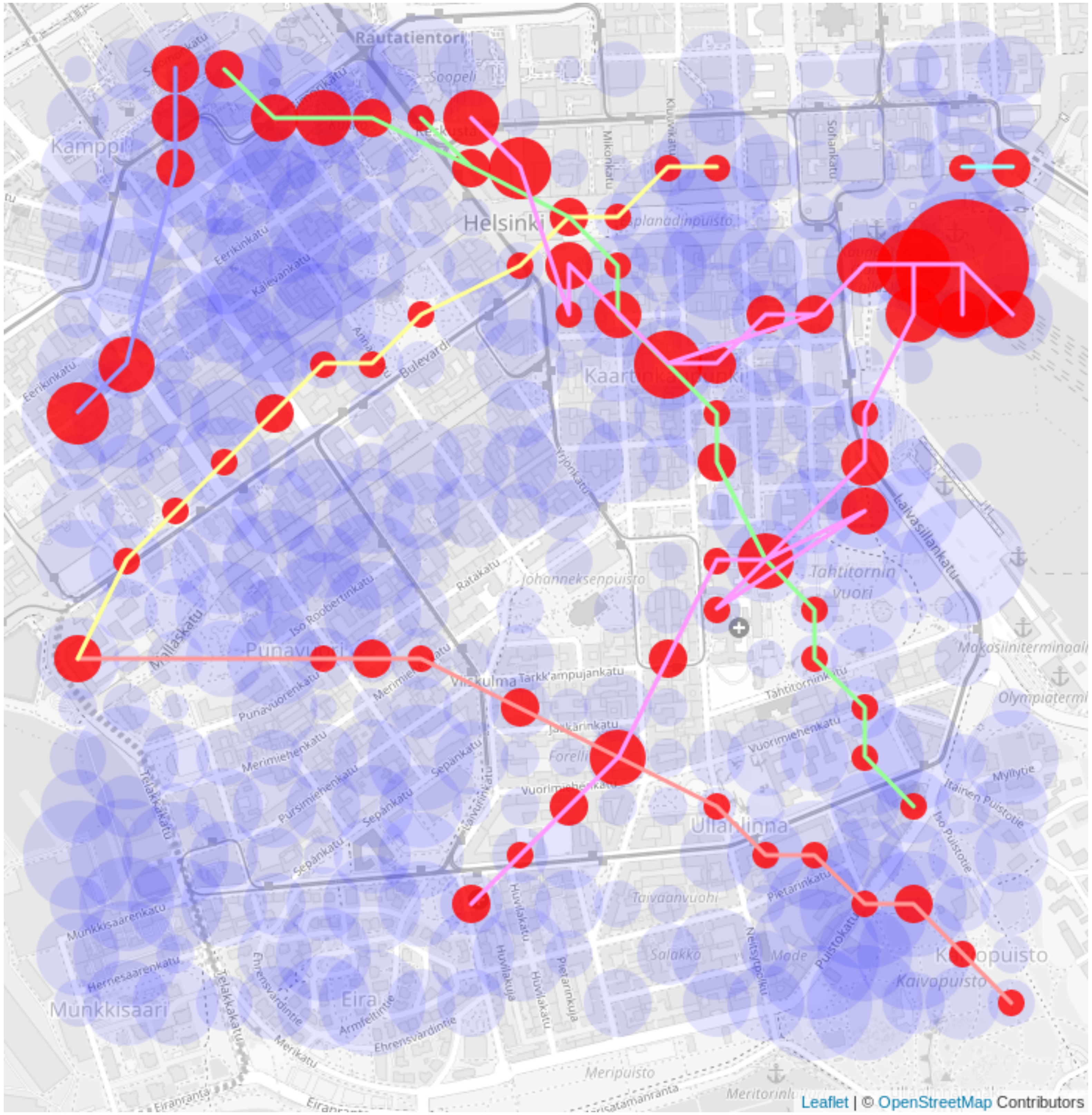
Risks:

- Single point of contact → identify sick person.
- Use single points of contact on purpose: test if sick.
- Create false alarms on purpose. Buy/sell this service.
- …

Source: https://risques-tracage.fr/docs/risques-tracage.pdf

Anonymity and pseudonymity

Another example: trace movement of sick person.



Source: https://github.com/oseiskar/corona-sniffer

Bitcoin and anonymity

Whole transaction graph is public!

Can trace transactions. See e.g. Ron and Shamir 2012.

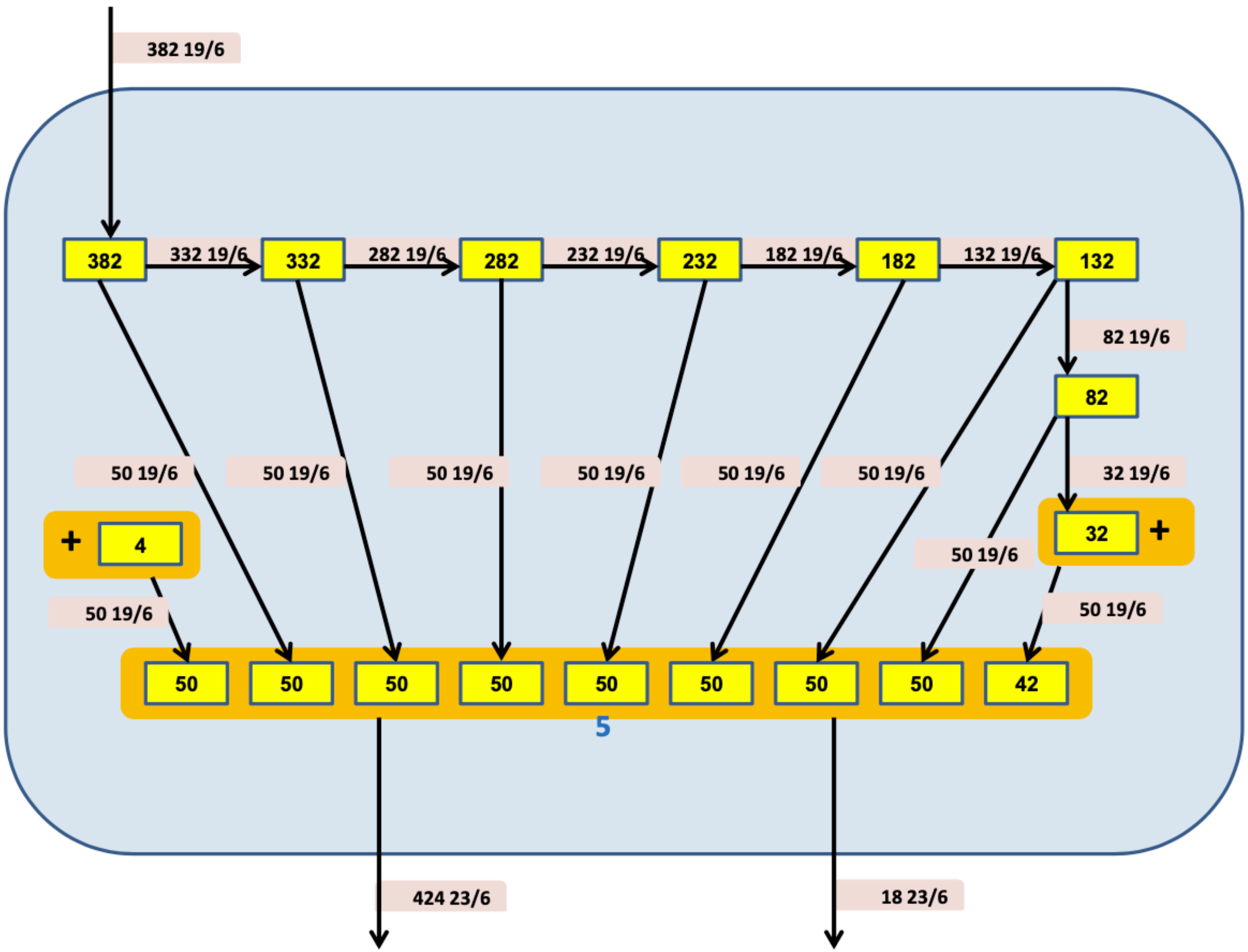
3	pk_A	pk_D	2
1	pk_B	pk_E	1.5
1	pk_C	pk_A	1.5

+ sign sk_A, sk_B, sk_C.

Probably same person

Bitcoin and anonymity

Suspicious activity.



Stronger anonymity

Monero:

- Stealth addresses: anonymity of recipient.
- Ring signatures: anonymity of sender.
- Homomorphic commitments: confidentiality of amounts.

Zcash:

- Zero-knowledge proofs: anonymity of all quantities.

Zcash

Encrypted

3	pk_A	pk_D	2
1	pk_B	pk_E	1.5
1	pk_C	pk_A	1.5

+ sign sk_A, sk_B, sk_C.

+ ZK proof of validity

In addition, each shielded tx gives hash(recipient, amount, rho, r).

Each used tx gives matching “nullifier” hash(spending key, rho).