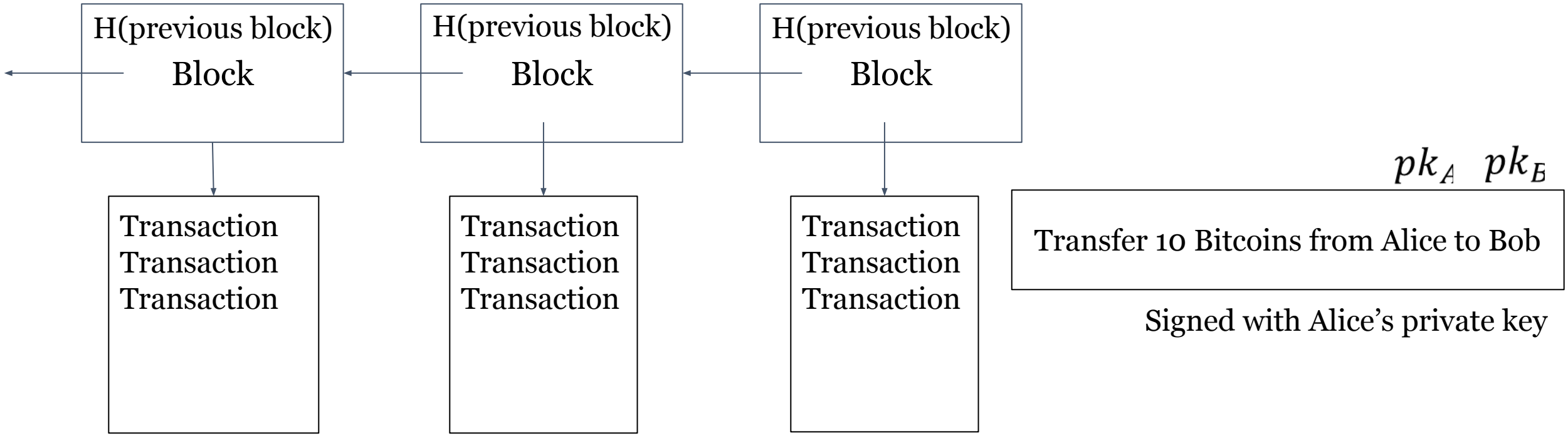
Privacy-preserving smart contract

Blockchain



- Append-only authenticated list
- A random party is selected to propose the next block (mining)
- Everyone checks the data of the new block is **valid**
- More than 50% honest parties → consensus without a centralized trusted party

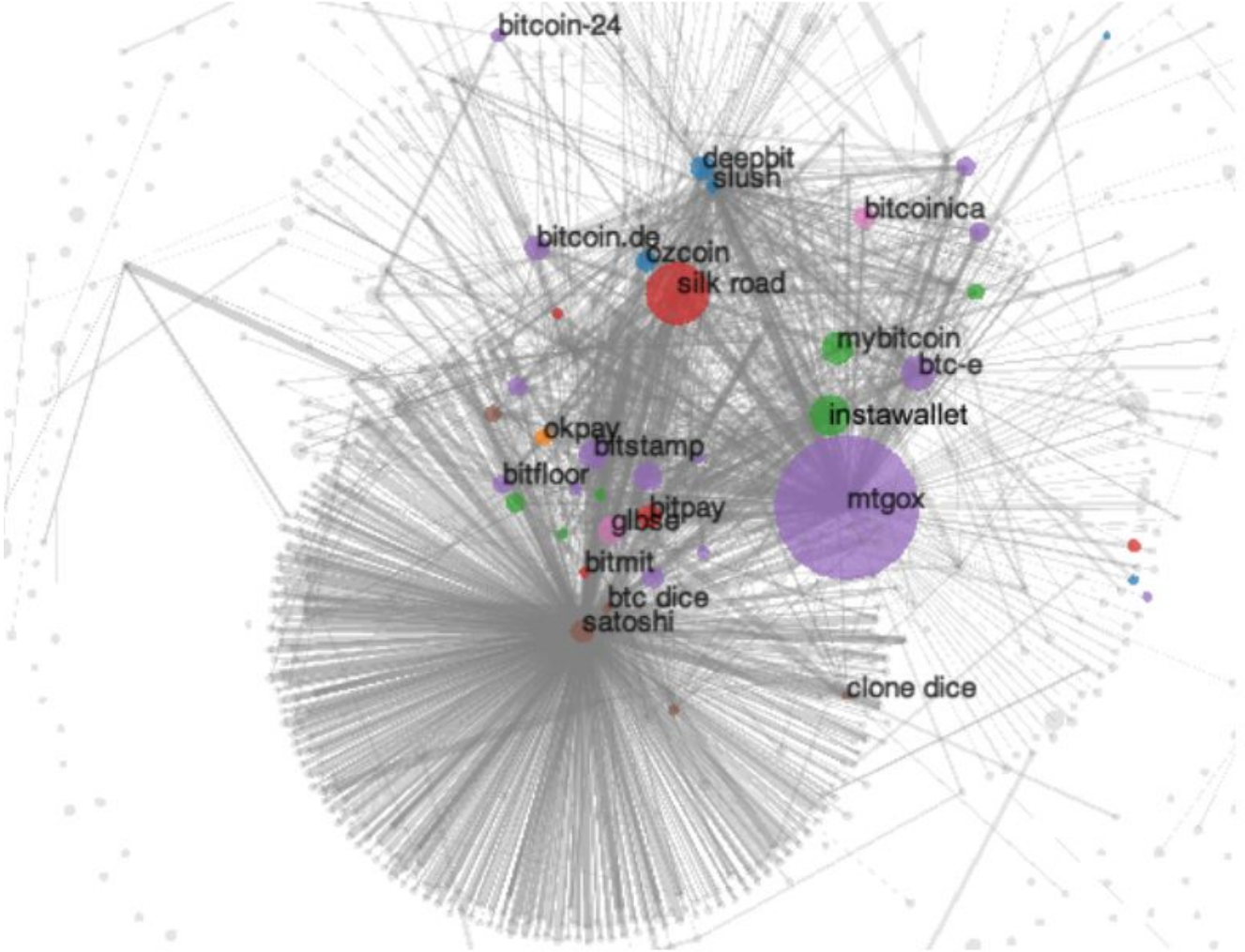
Account model vs transaction model

- Account model: account balances

- Transaction model: UTXO (unspent output of transactions)

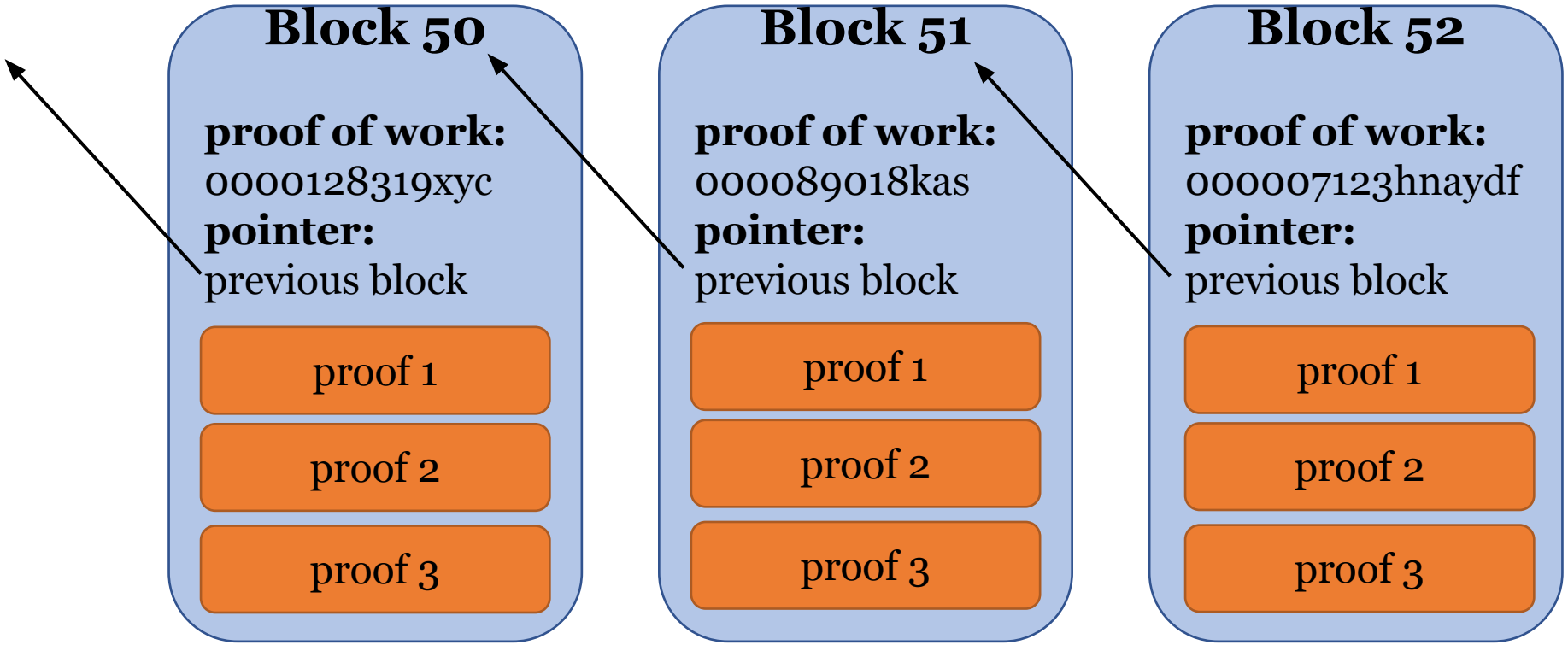
  - A set of unspent transactions
  - Each new transaction destroys 1 (or several) elements in the set, and insert 2 elements into the set

Linkage attack on Bitcoin network



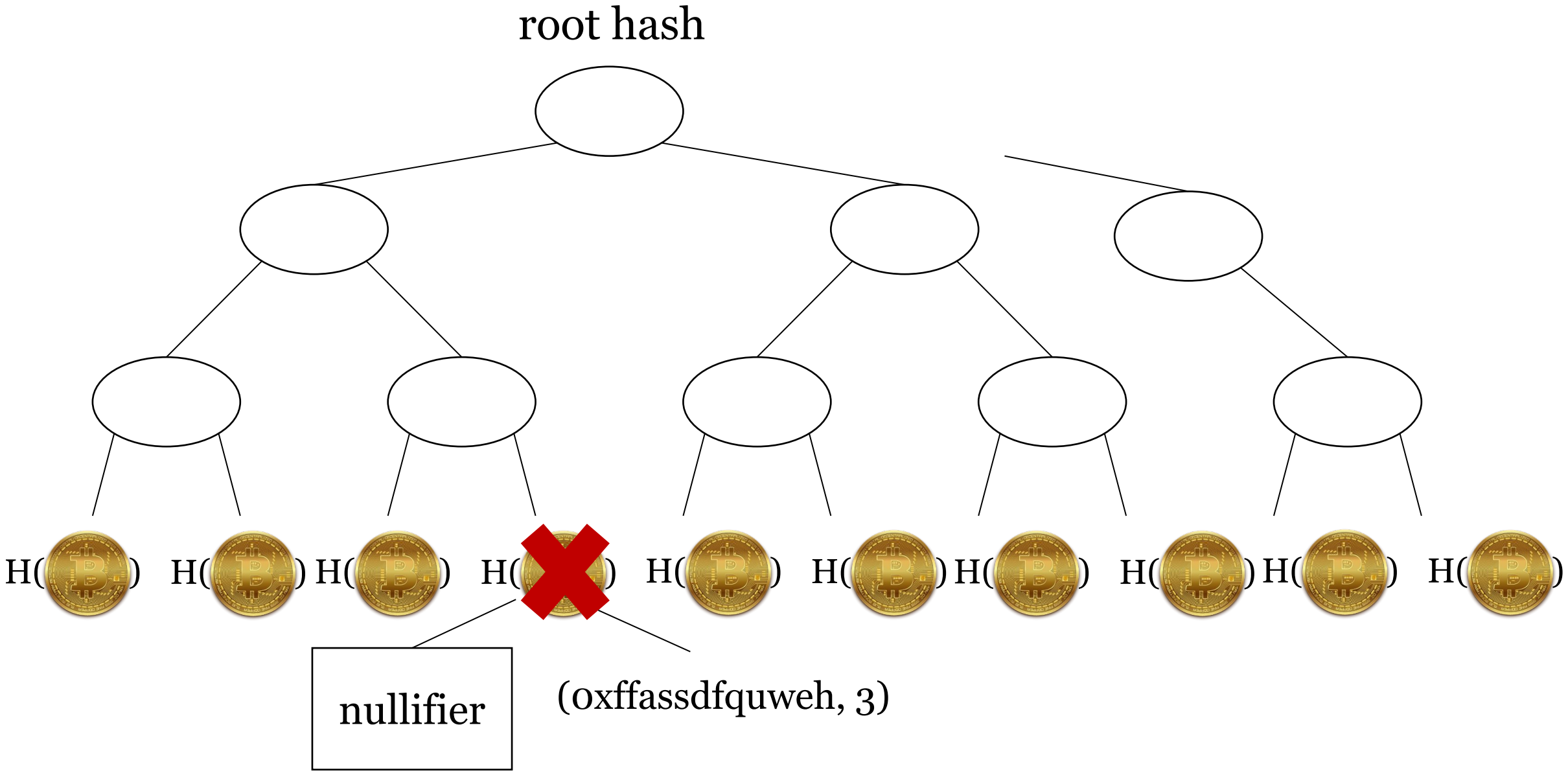
A fistful of bitcoins: characterizing payments among men with no names, Meiklejohn et al. 2013

Solution: zero knowledge proof



Publish **zero knowledge proofs** of data validity on blockchain

Zcash



Zcash

- Uses zero knowledge proof
- Avoid linear scan: Merkle tree
- Double spending: nullifier
- Send money: encryption

Zero knowledge proof:

1. There is a Merkle tree path for the coin/commitment/hash
2. I know the pre-image of the commitment/hash
3. I know the secret key of the public key
4. The amount of two new coins are less than the old coin
5. The commitments are computed from the new coins
6. The encryption is the new coin, etc.

Secret/witness: secret key, public key, amount, Merkle tree path, new coins (receiver public key, amount)

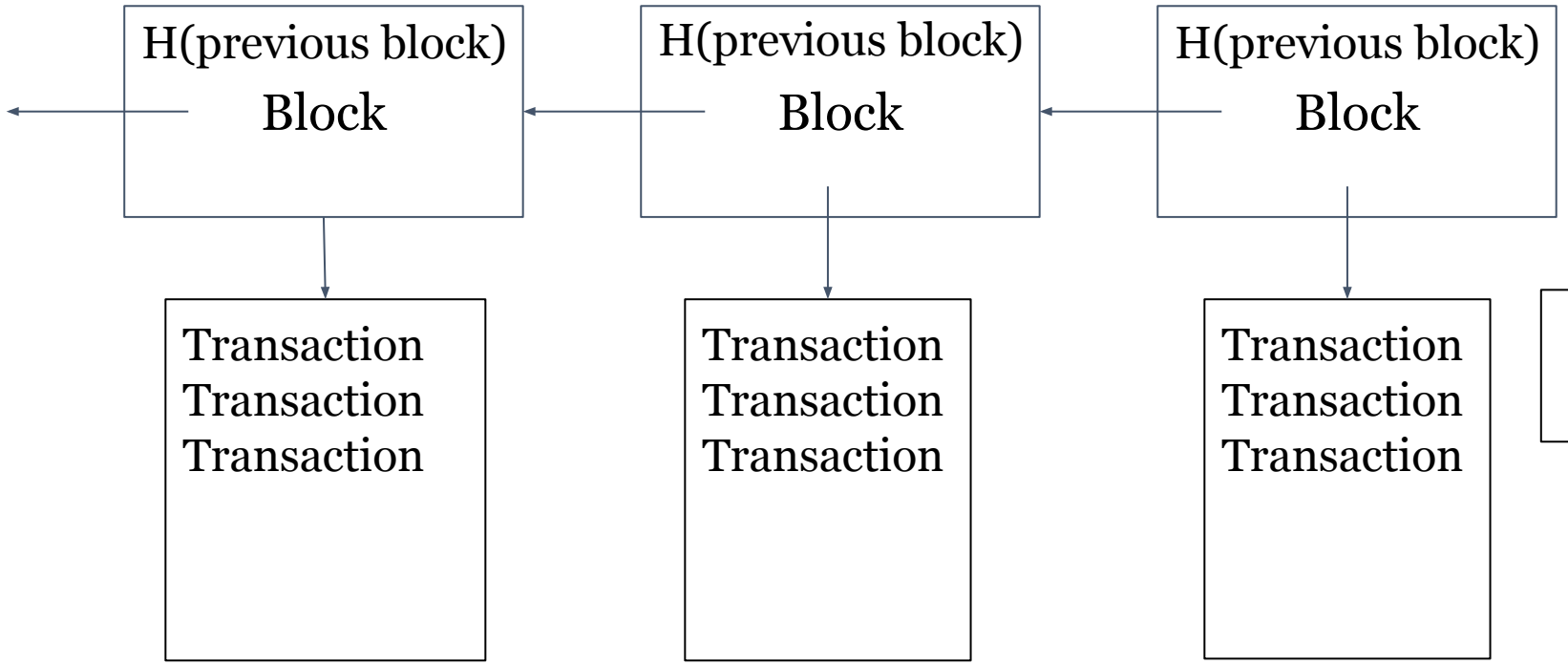
Public input: root hash of Merkle tree, nullifier, commitments of new coins, encryption of new coins

Zero knowledge proof scheme: zkSNARK

- ✔ Supports all functions (modeled as arithmetic circuit)
- ✔ Constant proof size < 200 bytes
- ✔ Fast verification time 3ms

- × Function dependent **trusted setup**
- × Slow prover time (modular exponentiations for every gate)

Bitcoin's scripting language



pk_A pk_B

Transfer 10 Bitcoins from Alice to Bob

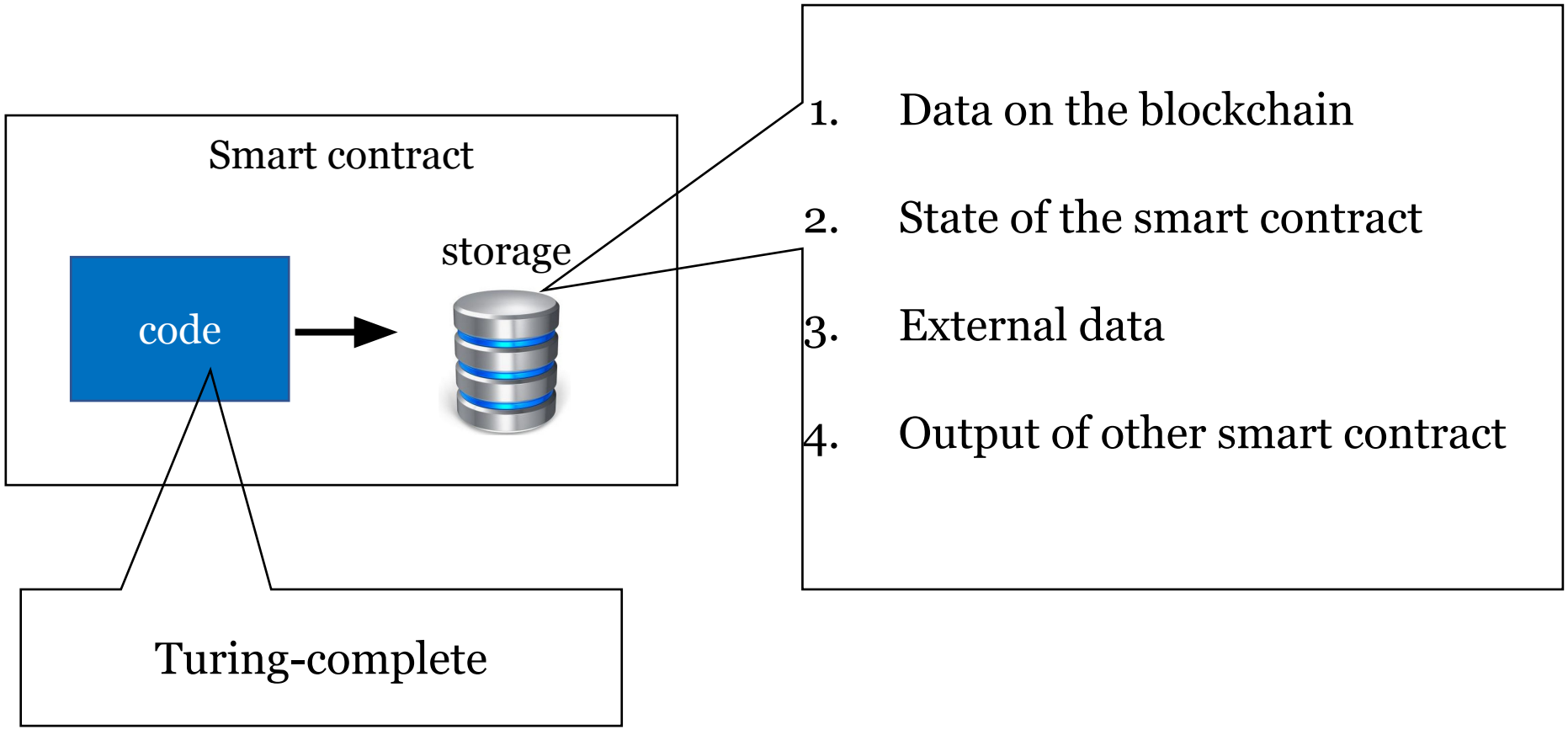
Signed with Alice's private key

pk_A

Transfer 10 Bitcoins from Alice to whoever provides x s.t. H(x) = y

Signed with Alice's private key

Smart contract



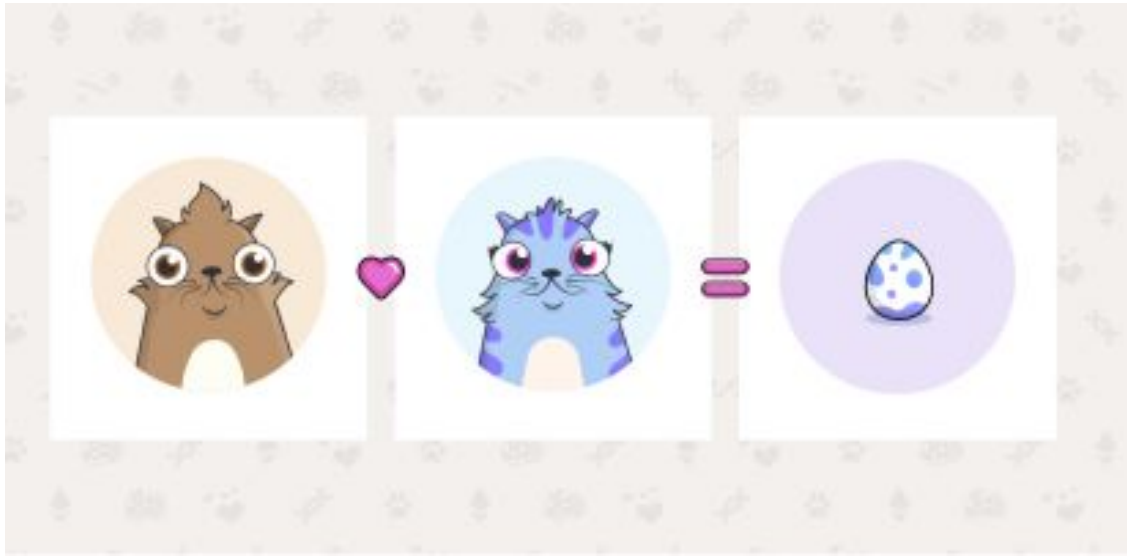
- A smart contract is similar to a transaction on a block
- User/miner computes the result of a smart contract
- Others validates the result by re-executing the smart contract

Examples: bidding

- Everyone can send their bids during a bidding period
- The bids send money to the contract
- Refund if higher bids are provided
- After the end of the bidding period, beneficiary calls the contract manually to receive money

https://solidity.readthedocs.io/en/v0.4.24/solidity-by-example.html

Examples: CryptoKitties



Gas limit

- Validation is the same as computing the result
- Limit the running time of a smart contract

Privacy problems in smart contract

Everything is public on the blockchain

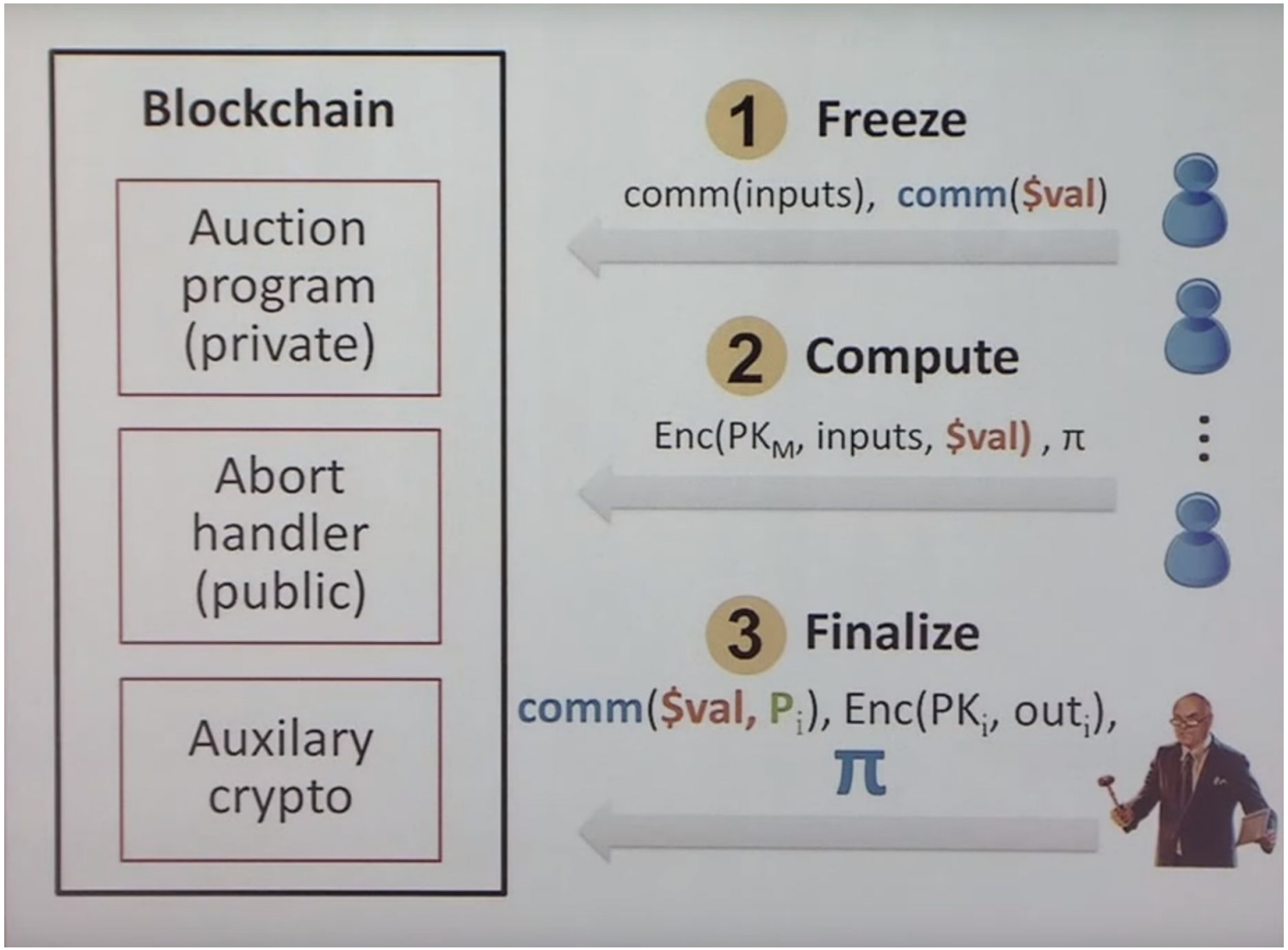
- Transaction sender and receiver
- Transaction amount
- Smart contract code
- Smart contract state

Hawk: privacy-preserving smart contract

- Privacy-preserving transactions in the UTXO model similar to Zerocash

- Freeze money to smart contracts
  - Commitment of a new coin
  - Zero knowledge proof that it is from a coin in the pool/UTXO

Contract manager



Problem: trust on the manager

- Not for correctness
- Not for input independence
- Not for security of the currency / consensus
- Trusted for privacy

Solution

- MPC with ZKP

Problem: zkSNARK

- Function dependent trusted setup (different from Zerocash)

Solution

- Transparent zero knowledge proof without trusted setup

- Proof size
  - Cannot be O(1) without knowledge assumption
  - O(log n)

- Verification time
  - Linear to the circuit size (worst case for random circuits)
  - Sublinear for structured circuits
  - Sublinear for RAM programs

Scalability

- Verification is faster than computing (theoretically)
  - 1 pairing is 0.3ms