

BITCOIN, TRANSACTIONS UTXO MODEL

Blockchain technologies, lecture 2



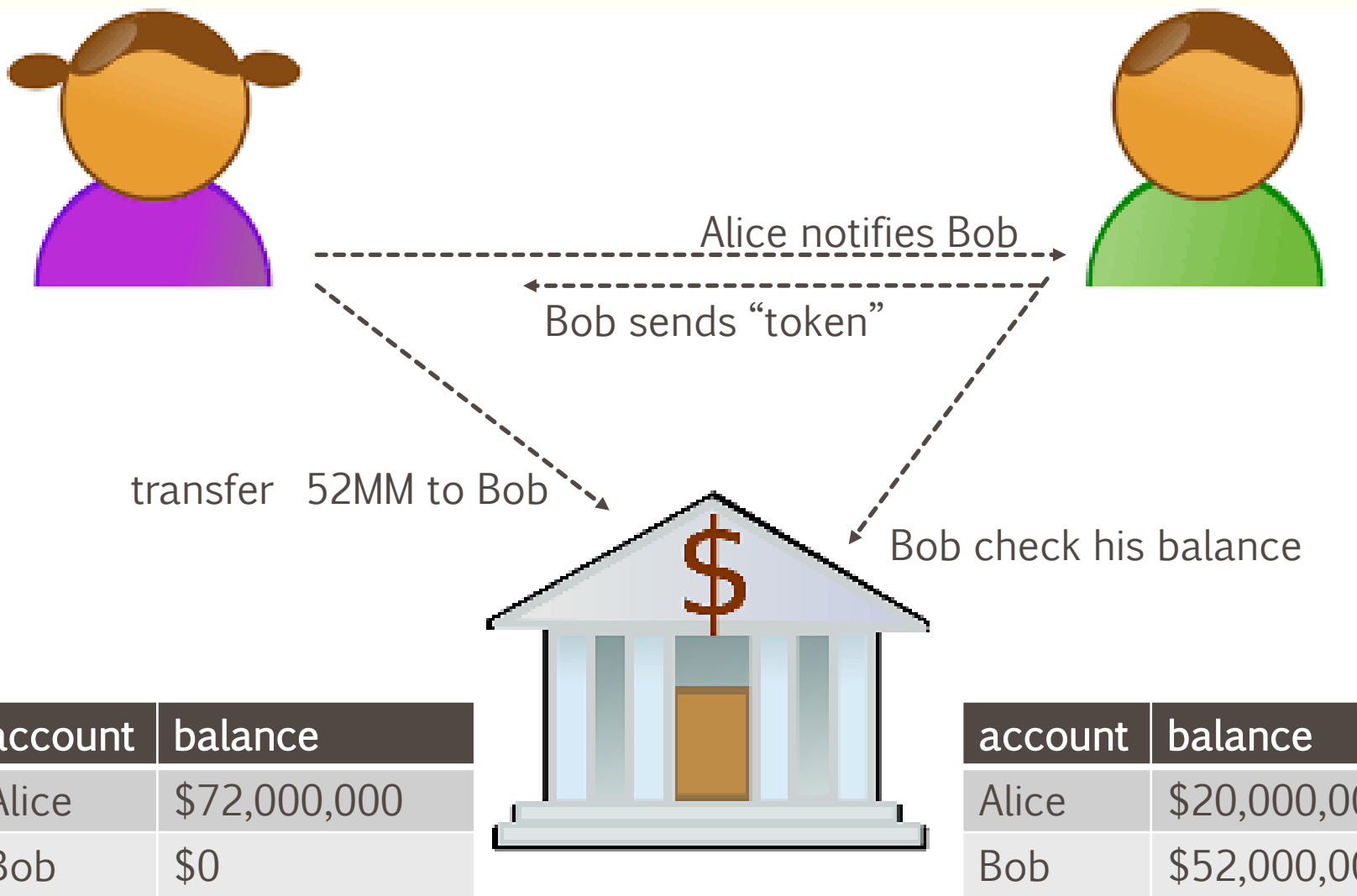
Course overview

- Blind signatures/zero-knowledge proofs
- UTXO model
- Transactions



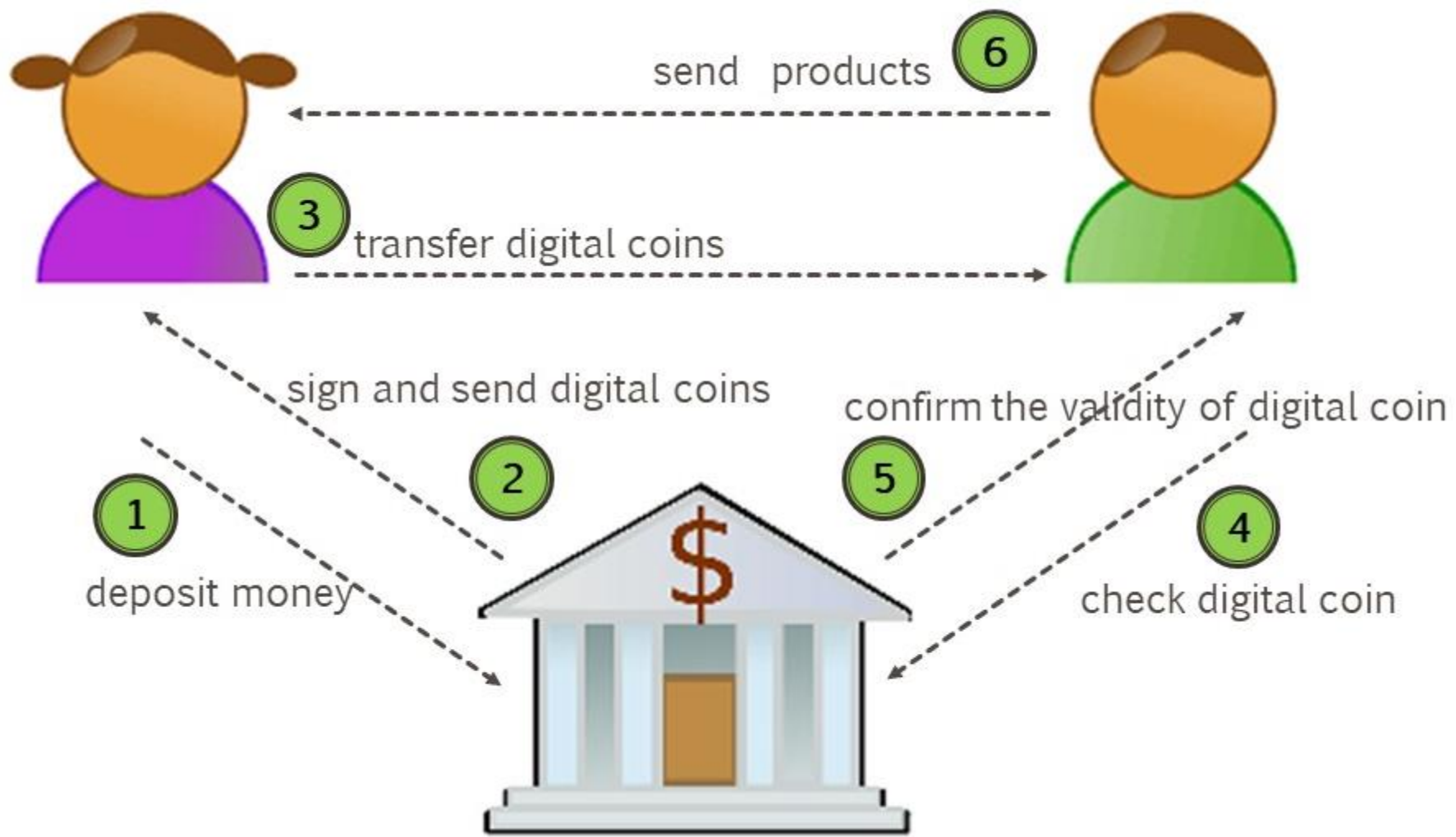
BLIND SIGNATURES

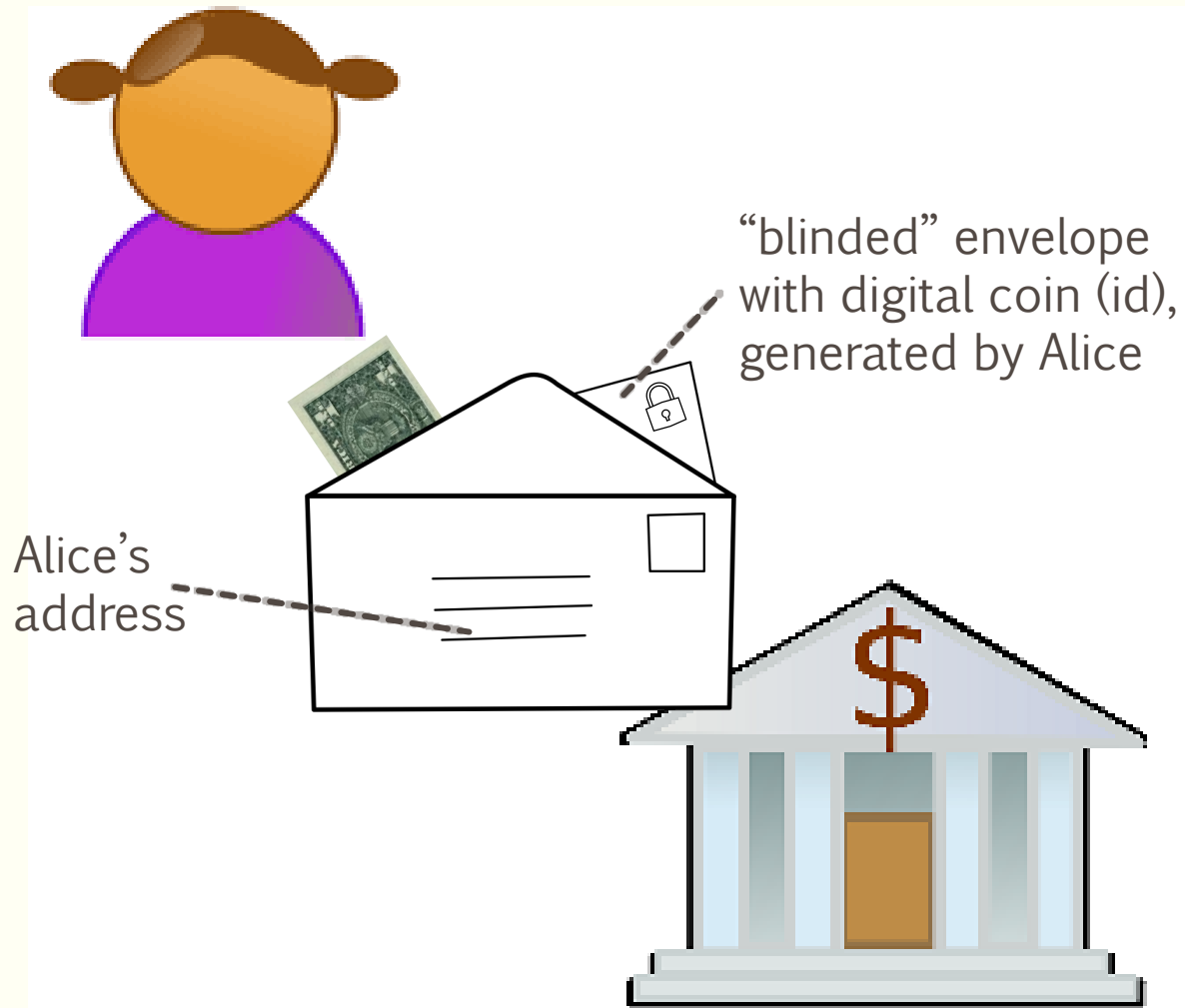
Bitcoin transactions' history



traditional payments

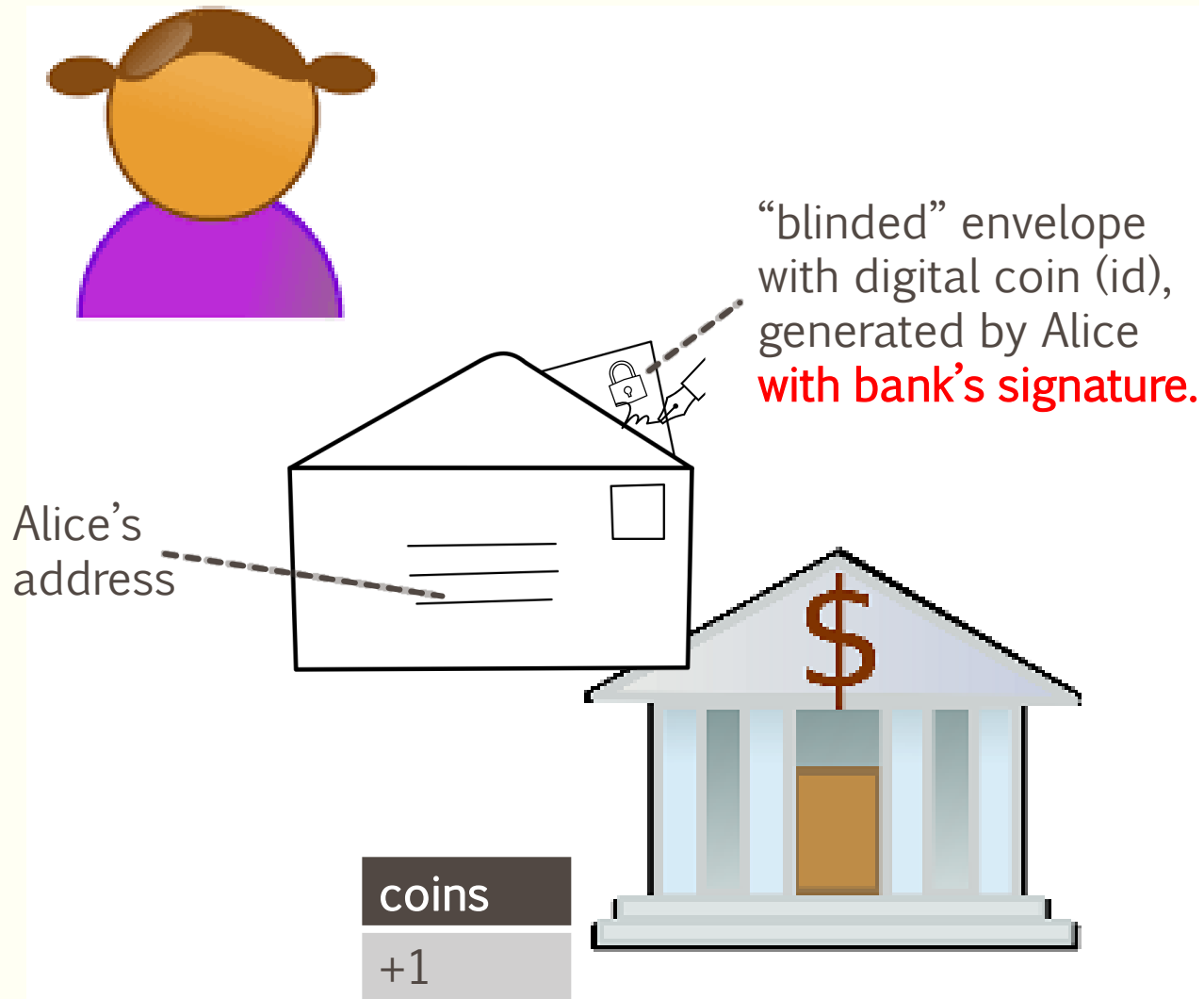
- Single point of failure, not peer-to-peer.
- Delayed or refused transactions.
- Security and privacy issues.
- Digital payments





Chaum e-cash Blind signatures (1983)

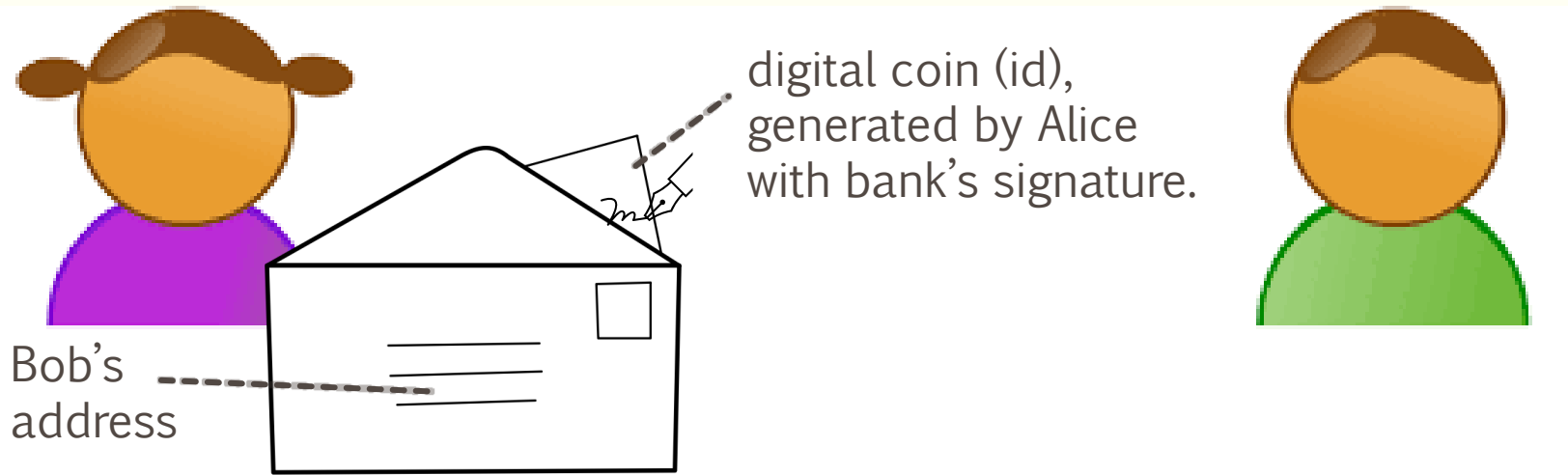
- Alice sends a dollar to the bank and a blinded id, representing the digital coin she will be authorized to use in further transactions.
- Imagine an encrypted envelope or carbon paper nested in an envelope with Alice's address on it.



Chaum e-cash (1983)

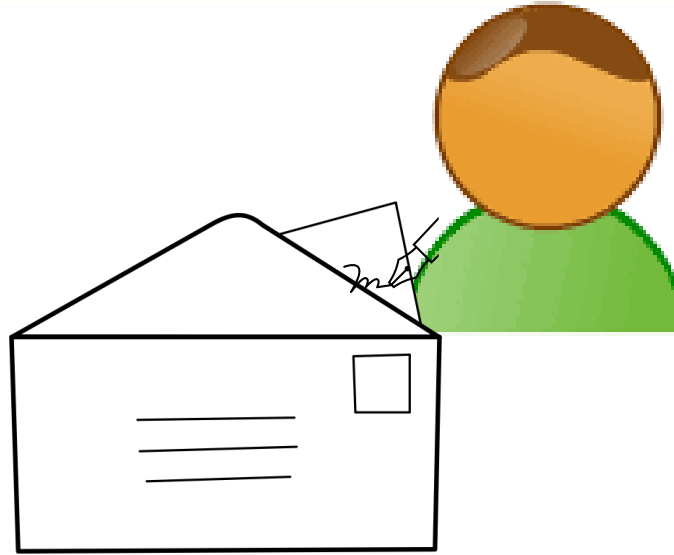
Blind signatures

- Bank is not able to discover the id, it can only register how many coins are signed and sign Alice's blinded id.
- Bank has no record of Alice's balance or transactions.
- Alice can recognize/recover the id she generated.
- Anyone can recognize the signature of the bank.



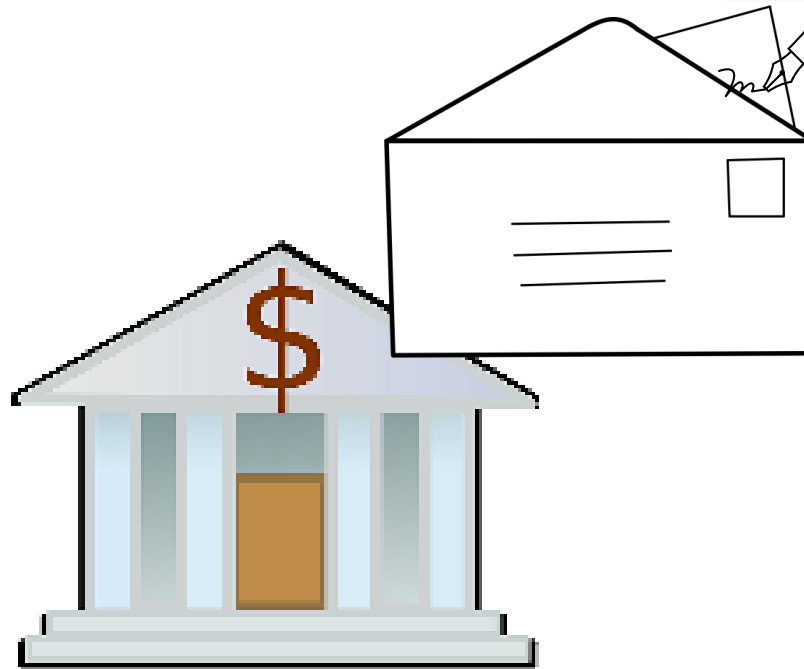
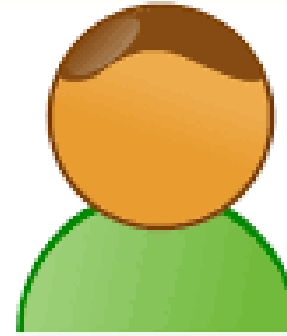
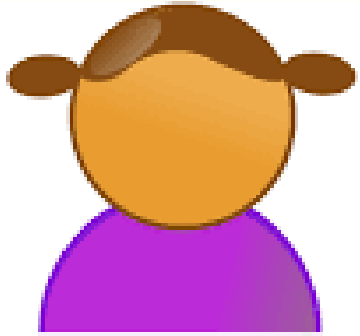
Chaum e-cash Blind signatures

- Alice sends the signed, not blinded digital coin to Bob.



Chaum e-cash Blind signatures

- Bob receives from Alice the id signed by the bank.
- Bob recognize the signature of the bank.



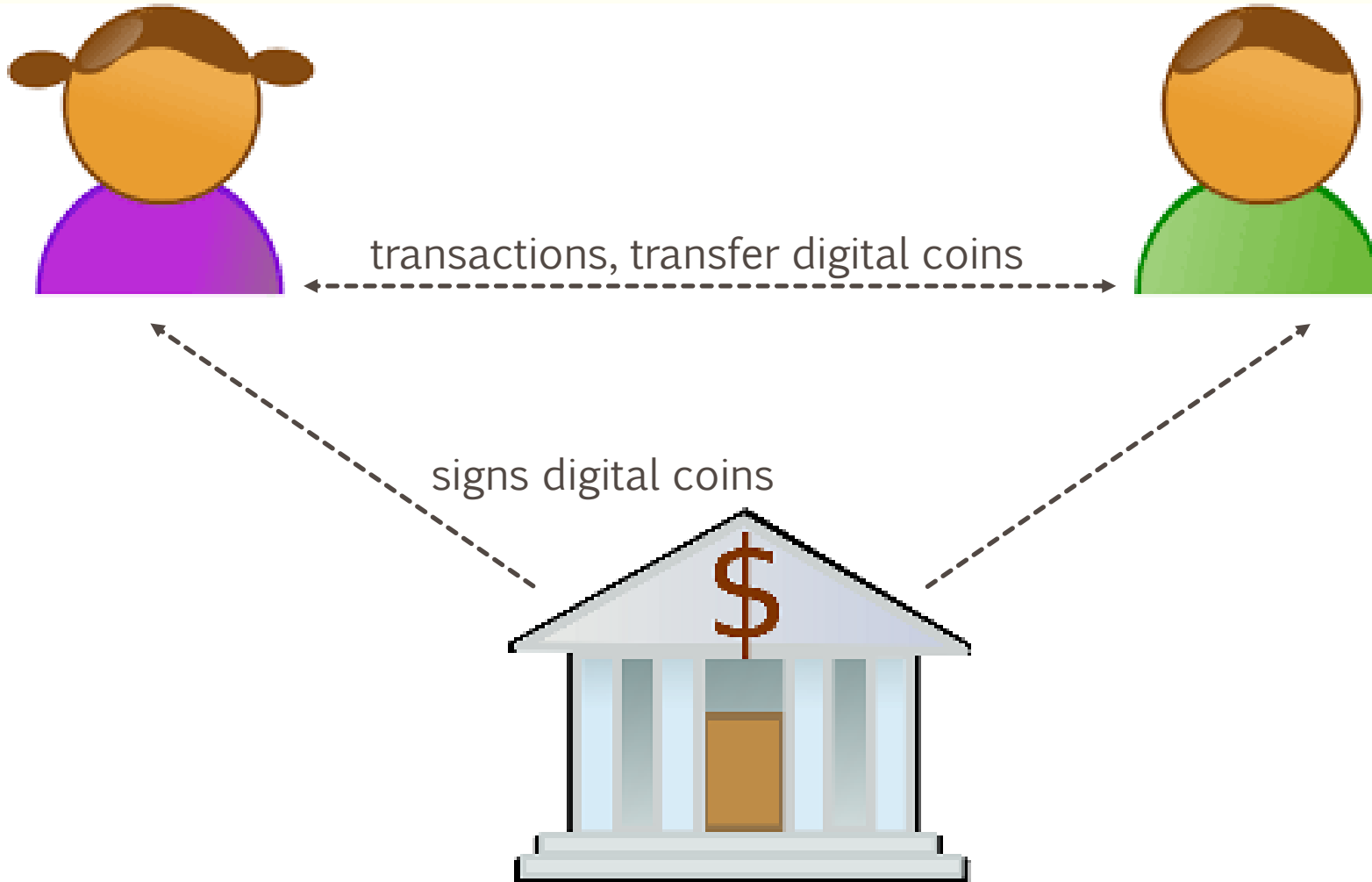
claimed ids

.....

.....

Chaum e-cash Blind signatures

- Bobs claims 1\$ in return for his digital coin.
- Bank registers the id.
- Bank doesn't know that the id came from Alice and it has no information about the transaction.
- Bank is able to detect **double spending**.



Chaum e-cash Blind signatures

- Peer-to-peer, but not decentralized.
- bank signs all digital coins.
- Advantages:
 - Privacy.
 - Digital payments.
 - Double spending detection.

RSA

- RSA: find three large positive integers e , d , and n , such that: knowing e and n (the public key), it can be computationally difficult to find d (the private key);
- Anyone knowing the public key may send encoded messages that can be decoded only by someone knowing the private key, provided that $(m^e)^d = m \pmod{n}$.

1. Choose p and q , two large prime numbers and calculate $n = pq$.
2. Calculate $\lambda(n) = \text{lcm}(p - 1, q - 1)$.
3. Choose e such that $\text{gcd}(e, \lambda(n)) = 1$ and $1 < e < \lambda(n)$
4. Find d such that $ed \equiv 1 \pmod{\lambda(n)}$
5. Set public key (e, n) . Set private key (d) .

Set **coding** function $c(m) = m^e \pmod{n}$

Set **decoding** function $d(c) = c^d \pmod{n}$

Blind signatures

1. a signing function s' known only by the signer, and the corresponding inverse s , such that $s(s'(x)) = x$ without revealing information about x ;

Signature of the Bank, Bank can't reveal any information about the id sent by Alice.

2. a commuting (or blinding) function c and its inverse c' , known only by the provider, such that $c'(s'(c(x))) = s'(x)$, and c and c' does not leak any information about x . Informally, function c' recovers the signed message from the blinding envelope;

c is the blinded envelope. Bank signs the envelope. Alice is able to recover from the envelope the id with a valid signature of the bank.

RSA - Blind signatures

Set **coding** function $c(m) = m^e \pmod n$

Set **decoding** function $d(c) = c^d \pmod n$

1. a signing function s' known only by the signer, and the corresponding inverse s , such that $s(s'(x)) = x$ without revealing information about x ;

$$s'(x) = (x)^d \pmod n.$$

2. a commuting (or blinding) function c and its inverse c' , known only by the provider, such that $c'(s'(c(x))) = s'(x)$, and c and c' does not leak any information about x . Informally, function c' recovers the signed message from the blinding envelope;

$$c(x) = xr^e \quad \gcd(r, n) = 1.$$

Signature of the Bank,
Bank can't reveal any
information about the id
sent by Alice.

c is the blinded envelope.
Bank signs the envelope.
Alice is able to recover
from the envelope the id
with a valid signature of
the bank.
 x is the token to be signed
by the bank.

Notice that $c'(s'(c(x))) \equiv s'(x) \pmod{n}$ holds, since
 $c'(s'(c(x))) \equiv (xr^e)^d r^{-1} \equiv x^d r^{ed} r^{-1} \equiv s'(x) \pmod{n}$

1. a signing function s' known only by the signer, and the corresponding inverse s , such that $s(s'(x)) = x$ without revealing information about x ;

Signature of the Bank,
Bank can't reveal any
information about the id
sent by Alice.

$$s'(x) = (x)^d \pmod{n}.$$

2. a commuting (or blinding) function c and its inverse c' , known only by the provider, such that $c'(s'(c(x))) = s'(x)$, and c and c' does not leak any information about x . Informally, function c' recovers the signed message from the blinding envelope;

c is the blinded envelope.
Bank signs the envelope.
Alice is able to recover
from the envelope the id
with a valid signature of
the bank.

x is the token to be signed
by the bank.

$$c(x) = xr^e \quad \gcd(r, n) = 1.$$



UTXO UNSPENT TRANSACTIONS OUTPUT

Bitcoin transactions

UTXO

- **Public key – private key:** Alice and Bob both keep a set of public/private key pairs. Alice can send coins that only Bob will be able to reuse, without the intervention of a third party.
- **Wallet:** A wallet is associated with a pair of private/public keys. Each user pseudonymously may store a set of pairs of key.
- **UTXO:** coins are placed in a wallet, being associated with a public key. Only the owner of the corresponding private key will be able to spend the UTXOs.

UTXO vs ACCOUNT model

- Bitcoin does not memorize users' balances, i.e. UTXO model does not memorize the sum of all unspent UTXOs owned by a wallet.
- **Ethereum** has a state transition model. In each state of the blockchain, an account has a balance. A transactions is a state transition, i.e after a transaction the account balance of the sender and the account balance of the receiver are changed.

Bitcoin UTXO model

Bob is selling a product
price 27

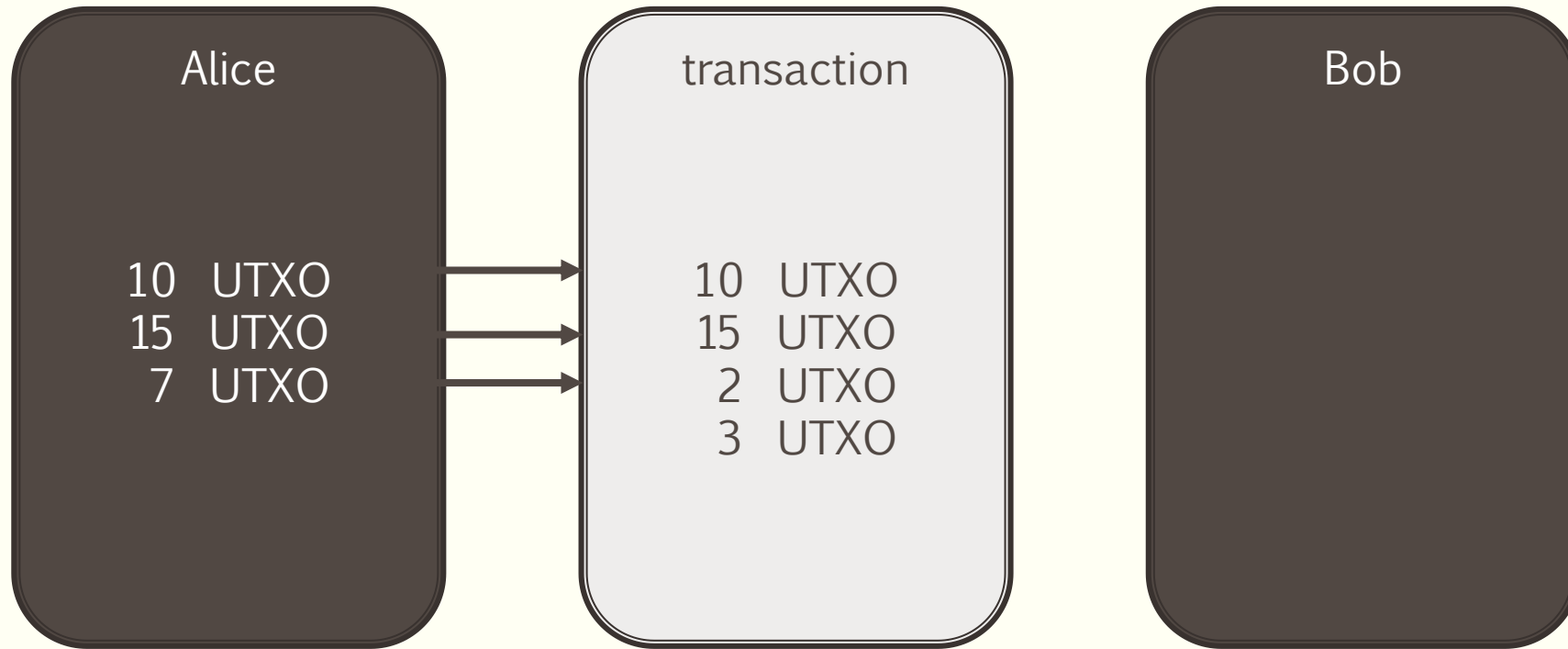
Alice

10 UTXO
15 UTXO
7 UTXO

Bob

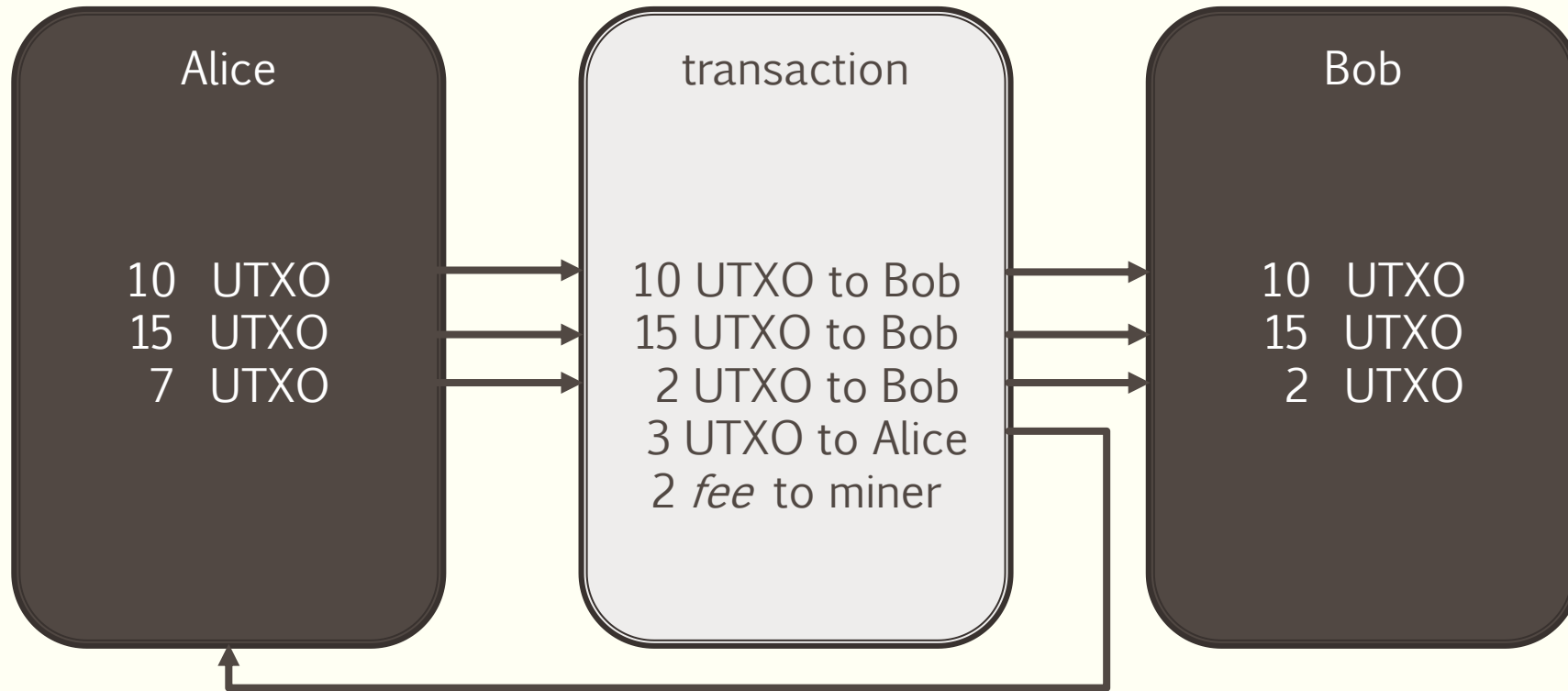
Bitcoin UTXO model

- multiple inputs
- coins have different values
- a coin may be spent only once



Bitcoin UTXO model

- multiple outputs
- some outputs are placed in the sender's wallet as change
- some outputs represent fees



Bitcoin UTXO model

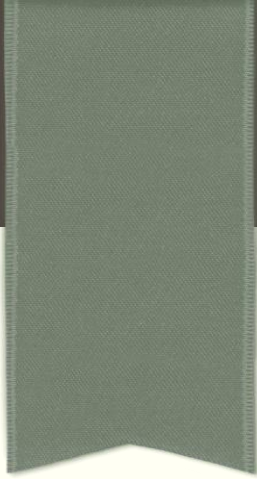
- Bitcoin Balance = sum of the unspent transaction outputs “owned”, not memorized!
pay with previous unspent transaction outputs
“change” is considered unspent transaction output locked to payer himself
- no double spending:
total input of a transaction = equal total output + fees
in a transaction coins are consumed and replaced with new ones.

Bitcoin UTXO model

- New coins are also created by mining
- transactions without input: **Coinbase transaction** -- miners reward!
- Coinbase transaction is the first transaction in a block.
- The reward halves each four year (after 210,000). Since 2020 block reward is 6.25BTC

UTXO set

- **Set of UTXO:** the set of all UTXOs is replicated on all full nodes but only the rightful owners are able to use them in transactions.
- UTXO are memorize in a Level DB database: **Chainstate**.
- Chainstate is the set of all unspent transactions as they appear up to the last block added in the blockchain.
- Chainstate is redundant as it can be reconstructed from the data stored in the blockchain, but this would be extremely slow and inefficient



TRANSACTIONS

Bitcoin transactions

Field	Description	Size
Version	currently 1	4 bytes
In-counter	Number of inputs	1 – 9 bytes
List of inputs	List of inputs	<in-counter>-many inputs
Out-counter	Number of outputs	1 – 9 bytes
List of outputs	List of outputs	<out-counter>-many outputs
lock_time	block height or timestamp when transaction is final	4 bytes

lock_time field -- transaction will not be confirmed until a specific timestamp
or
transaction will be confirmed after 6 confirmations

transaction ID=ac4be...

value:
scriptPubkey

value: 10
scriptPubkey

transaction ID=589e0...

value:
scriptPubkey

value: 15
scriptPubkey

transaction ID=14e9f

value: 7
scriptPubkey

transaction ID=c84be...

TRANSACTION INPUTS

prev_txID: ac4be...
Index: 1
scriptSig

prev_txID: 589e0...
Index: 1
scriptSig

prev_txnID: 14e9f...
Index: 0
scriptSig

lock_time

TRANSACTION OUTPUTS

value: 10
scriptPubkey

value: 15
scriptPubkey

value: 2
scriptPubkey

value: 3
scriptPubkey

Transactions – Locking script

- ECDSA signature => public/private key
- public key => bitcoin address
- Bitcoin address => **scriptPubKey**



- Challenge to users that want to claim a specific output. Only the owner of the UTXO should be able to unlock the UTXO providing the unlocking script **scriptSig**.
- Alice signs the transaction and sends the output to public key.

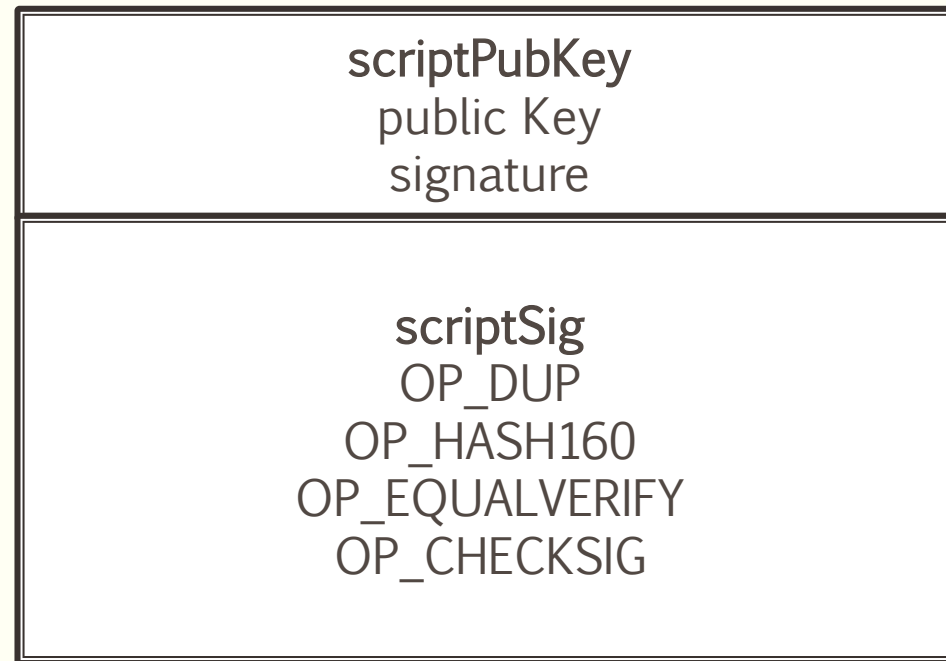
Transactions -- Unlocking Script

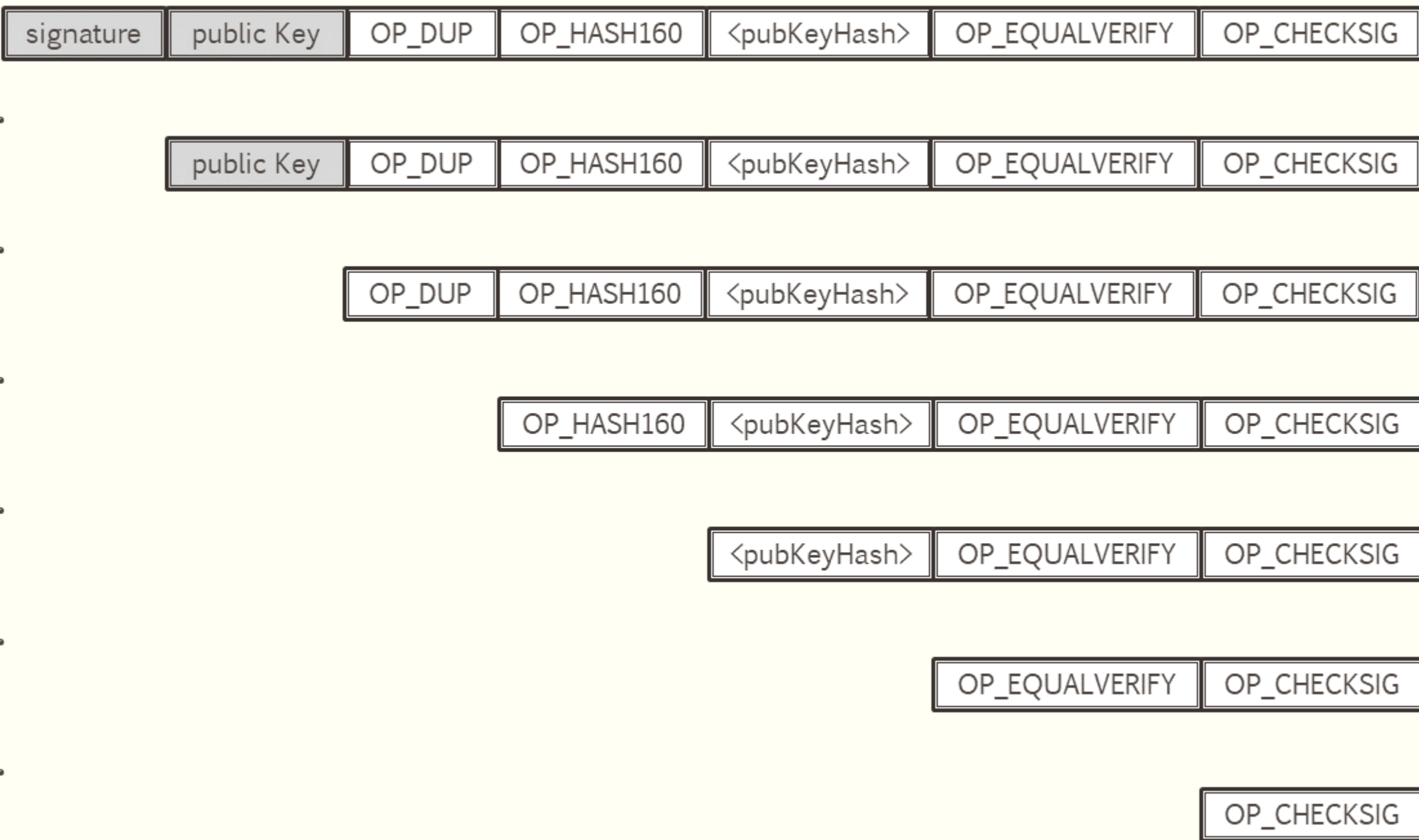
- **scriptSig**
 - include **receiver public key hash**
 - instructions (op_codes) to verify public key

```
scriptSig
OP_DUP
OP_HASH160
OP_EQUALVERIFY
OP_CHECKSIG
```

Bitcoin UTXO model

- P2PKH pay to public key hash





signature

public Key
signature

public Key
public Key
signature

<pubKeyHash>
public Key
signature

<pubKeyHash>
<pubKeyHash>
public Key
signature

public Key
signature

TRUE

UTXO model

- Simple
- Verify transactions in parallel,
 - a UTXO affected by only one transaction
- Privacy (wallets)
- Limited smart contract capabilities