Day 7: UTXOs

(and payments via ledgers)

Today

1. History of Money
2. Public Ledger
3. Goofy Coin
4. Transactions for SPCS-Coin

1 A history of money

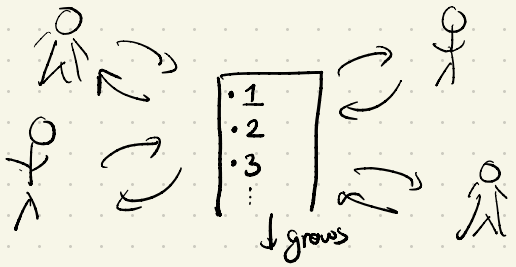
- Before computers:
  - Money is a scarce object
    - naturally scarce: gold, seashells, ...
    - artificially scarce: government-issued notes
  - Payments are physical transfers of the object
  - Properties:
    - Payments are irrevocable
    - Scarcity is natural or from unforgeability
    - Very private
- After computers
  - Money is data in a bank's computer
    - Redundantly stored
    - Backed by 'real' money
    - Highly regulated
  - Payments are data updates
  - Properties
    - Fast & convenient
    - relies on integrity of banks (and laws)
    - somewhat private: bank sees all

Key Question: Does a financial system require trust? → trusting banks to follow rules
→ trusting regulators to enforce laws
→ trusting The Fed / The US treasury to have 'good' monetary policy

We'll see that the answer is NO!

2 The Public Ledger

Key tool: a distributed public ledger



- Rules
  - Anyone can post
  - Everyone can see
  - Everyone always sees the same ledger (consensus)
  - Entries can't be removed ("append-only")

Example:

Initially: ["A"]

After Alice posts "B": ["A", "B"]

After Bob posts "C": ["A", "B", "C"]

Today: How to build a financial system from ledger

Tommorow: How to build the ledger & secure it!

3 Goofy Coin

- Ledger has two kinds of entries:
  - "CreateCoin ID NAME": mints (creates from nothing) a coin ID to NAME
  - "Pay ID FROM TO": gives a coin ID from FROM to TO
- Rules: in order to "Pay", you must have the coin

Examples:

Create 0 A	Create 0 A	Create 0 A
Pay 0 A B	Create 1 B	Pay 0 A B
Pay 0 B C	Pay 0 B C X	Pay 0 A C X
✓	unauthorized OR not owned	double-spend

Problems:
- Anyone can spend anyone's money !!
- Anyone can mint money !!
- Can't make change !

Ideas:
← signatures
← ?? (tomorrow)
← merging/splitting coins

4 SPCS transactions:

- One type of ledge entry: a transaction (tx)
- A tx has lists of inputs and outputs
- An output creates a coin and has
  - id: a unique name for this coin
  - pk: public key of who owns it
  - amt: how much it is worth
- An input spends a coin an has
  - id: which coin is spent
  - sig: a signature on id by sk for pk.
- Validation
  - Transactions are validated from first to last

- As txs are validated, we maintain a set of unspent tx outputs (UTXOs).
- Tx rules ← in python: {id: output}
  - All input ids must be in UTXO set
  - All output ids must not be in UTXO set
  - All input signatures must be valid
  - Sum of input amounts must ≥ sum of of output amounts.
    - Q: how to get input amounts: from UTXO set!

Examples:

Start with {0: Out(id=0, pk=A, amt=3)}

Tx([In(id=0, sig=...)], [Out(id=1, pk=B, amt=2), Out(id=2, pk=A, amt=1)])

✓ Valid.

If the 2nd output has amount 2? ✗
If the 1st output has amount 1? ✓
If the signature is invalid? ✗
If the 2nd output has id = 0? ✗
If the 1st output has pk=A? ✓

Small Example:

Alice has coins

id:	0	1	2	3	4	5
amt:	7	3	4	17	5	1

Alice wants to build a tx to pay Bob $7.

One way to do this: Use $17 coin, get $10 in change:

Ins	Outs
id=3	id=6, pk = pk_{Bob}, amt=7
	id=7, pk = pk_{Alice}, amt=10

- How many ways can Alice get $0 in change? Two $7 or $3+$4
- What if Alice wants $2 in change? $4+$5
- Many ways to build txs...

- Revisiting Problems:
  - Anyone can spend anyone's money
    → Signatures prevent this!
  - Anyone can mint money
    → Unclear: we haven't discussed how $ is created
  - Can't make change
    → Multiple inputs & outputs!

Problems: Implementing Tx validation!