E-Cash & Bitcoin

Intro: What properties can/should electronic money have?

- What does "possessing val" mean?
- How can we transfer val?

With physical money:

- Hard to generate
- Only one person at a time "owns" the money.

With bits:

- Easy to generate
- Bits can be copied $\Rightarrow$ double spending.

Simple Example: Electronic Checks

Bank has PK_B, SK_B

User has PK_u, SK_u, certificate on pk_u by bank.

$$\text{Check} = \left[\begin{array}{l} \text{cert on } PK_u \text{ signed by bank} \\ \text{Sign}(SK_u, \text{"pay Bob \$100, date, ser \#"}) \end{array}\right]$$

- Bank deposits check just once (using ser #).

This works!

Goal: Make payments more like cash.

Without central authority!

Desirable properties:

- Non forgeable
- Not double-spendable.
- Transferability: A can pay B.
- Transitivity: B can use A's payment to pay C.
- Divisibility & combinability.
- Efficiency
- Scalability
- Anonymity
- Decentralized (no trusted party)

Double spending: Key problem, especially in scheme that offers transitivity.

- Alice can give "same" coin to Chuck & Donald!
- Prevention seems tough (using only bits).
- Detection requires DB with all spending records (public ledger).

Many approaches :

Micromint (Rivest & Shamir 1996)

Peppercorn (Micali & Rivest 2002)

BitCoin [From online lectures by Felten et. al] Introduced by Nakamoto

Launched 2009, worth over 170 billion dollars, more than 375,000 transactions per day

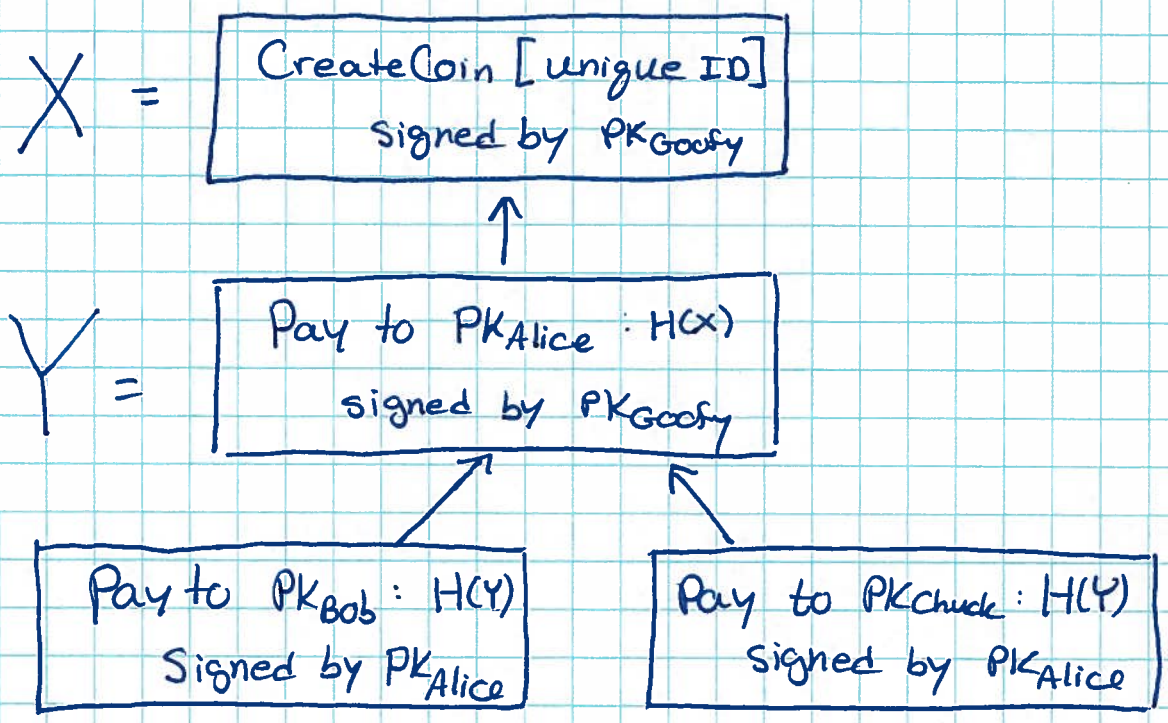
Decentralized identity management

- Identity = PK
- Each PK is an actor in the system.
- To speak for PK you must know corresponding SK.
- Each users can create many identities (as often as they want) simply by creating random key pairs (PK, SK).
- No central place needed to register (no central control).
- Is this anonymous ?

  Its complicated... Depends how often you change your ID.

Version I : Goofy Coin (Centralized & insecure)

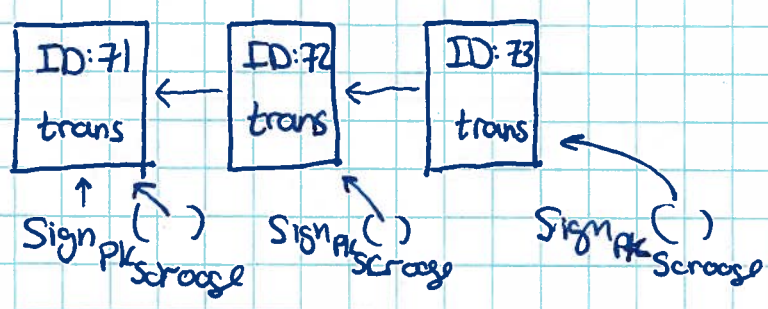
- Goofy can create new coins



Problem: Double spending!

Version II : ScroogeCoin (Centralized & Secure)

Idea: Scrooge will publish history of all transactions that happened



This will have the form of a block chain, digitally signed by Scrooge.

Scrooge publishes this history (public ledger).

- This allows to detect double spending.

Two types of transactions:

① Create Coins:

transID: 73		type: CreateCoin
#	value	recipient
0	3.2	PK_1
1	1.4	PK_2
2	7.1	PK_3

CoinID 73(0) → 0
CoinID 73(1) → 1
CoinID 73(2) → 2

Valid by def.

② PayCoins:

transID: 73		type: PayCoin
Consumed Coins 15(1), 42(0), 72(3)		
#	val	recipient
0	3.2	PK_1
1	1.4	PK_2
2	7.1	PK_3
Signed by all owners of consumed coins		

Valid if:
- All consumed coins are valid & not already consumed.
- Total value out = total value in.
- Signed by owners of all consumed coins.

If valid Scrooge will acknowledge & add to the public ledger & sign.

Note: Coins never subdivided or combined, only created & consumed.
But we get the same affect as if we are able to combine & subdivide, by making a Payfoin transaction to that affect.

Scrooge Coin works!

Core Problem: Scrooge!

This is a centralized solution.
What if Scrooge becomes malicious?

Bitcoin: De Scroogify the system!

Key Challenge: Distributed Consensus.

How can we provide the services that Scrooge provides in a decentralized way, where no specific party is trusted?
How can everyone agree on a single public block chain,

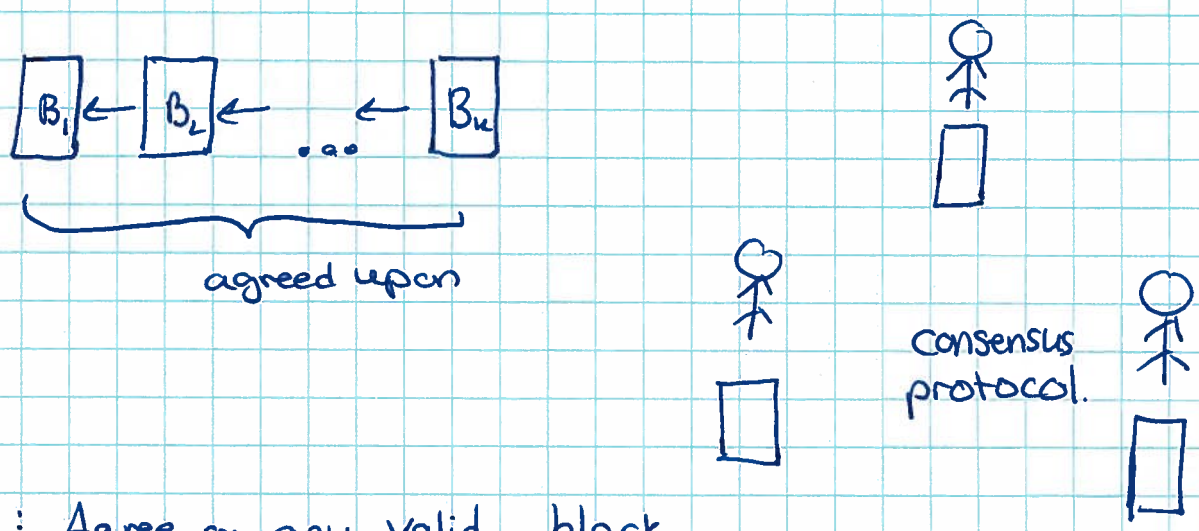
which is the agreed upon history of which transaction happened?

BitCoin is a peer-to-peer system:

When Alice wants to pay Bob she broadcasts the transaction (PayCoin) to all bitcoin nodes.

At any given time:

- All nodes have a sequence of blocks (for efficiency) of transactions they have reached a consensus on.
- Each node has a set of outstanding transactions it has heard about (different nodes may have different versions)



Goal: Agree on any valid block (even if proposed by only one node)

<u>Idea</u>: Select a random node to add the next block.

<u>How</u>? ← should be decentralized!

Proof of Work (PoW)

"Approximates" picking a random node.

<u>Idea</u>: Select nodes in proportion to a resource (we hope) no one can monopolize: Computational power!

Hash puzzles

To add a block, one must find nonce s.t.

$$H(\text{nonce} \| \text{prev\_hash} \| \text{this block}) < \underset{(0^k, *)}{\overset{\text{target}}{\shortparallel}}$$

- If the hash is secure then the only way to succeed is to try enough nonces, until you get lucky.

- The lucky node that found a good nonce gets to propose the next block.

Completely decentralized!

PoW Property 1: Difficult to compute

$k \approx \cancel{10^{20}}\ 76$

PoW Property 2: Parameterizable cost

(not fixed throughout time)

Goal: Any time between any two succ. blocks $\approx$ 10 min.

[Not too long (for efficiency reasons), and not to short to maintain agreement]

PoW Property 3: trivial to verify

H(nonce || prev_hash || this block) < target

In reality:

- Only a few nodes bother to compete to solve the puzzle ← miners

⟹ A lot of concentration power in the mining echosystem (even though technically anyone can be a miner) ← undesirable!

Why would anyone want to mine?

Incentive 1 Block reward

- Creator of a block gets to include a special coin-creation transaction in the block & choose the recipient address (PK).

- Value is fixed:
  - Currently 12.5 BTC
  - Halves every 4 years
  - Run out in 2040

[Finite supply of 21 million BTCs]

Why does this incentivize honesty?

- Block creator gets to "collect" the reward only if block ends up on long term consensus branch.

- This incentivizes nodes to behave in a way that other nodes agree with.

Key Assumption: Majority is honest!

Incentive 2: Transaction fees

A creator of a transaction can choose to make

output val < input val

The remainder is a transaction fee that goes to the block creator

Security

Secure if:

* Majority of miners weighted by hash power are honest (i.e. follow the protocol)

* Majority is honest:

Ownership of a bitcoin ≡ other nodes think I own this Bitcoin

What can a malicious node do?

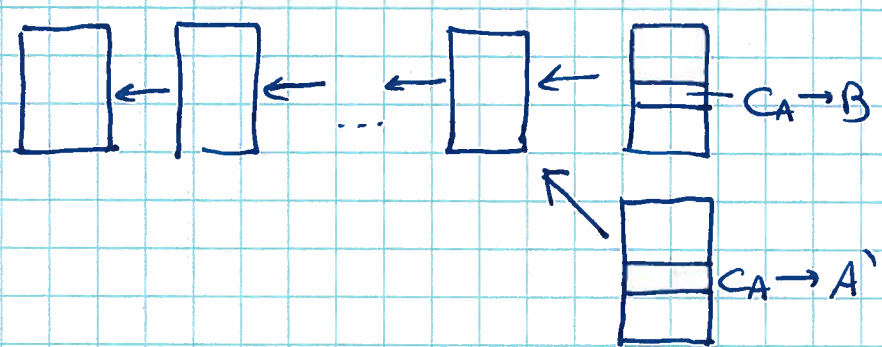
① Denial of service: Adv can decide not to include transactions to Bob.

$\Rightarrow$ Bob will need to wait until next block.

② Double spending:

Alice can give Bob a coin by generating a (signed) transaction $C_A \rightarrow B$ but can also generate & broadcast a transaction $C_A \rightarrow A'$ – where $PK_{A'}$ belongs to Alice.

How can Bob protect himself?



Honest nodes always extend longest valid chain.

Bob can wait until a few more blocks are added

(recommendation: wait 6 confirmations, until 6 additional blocks were added)