TOPIC: 6.857

DATE: 2/22/16

FILE UNDER:

PAGE: L06.1

Admin: Pset #1 due today. Pset #2 out.

Today: E-cash & bitcoin

- Money - basics
- Electronic checks
- Signed coin ID
- Identities
- (Public) Ledgers
- Bitcoin

"Electronic Money"

- What properties should it have?
- " " can " " ?

Atoms vs Bits

- What can "possessing value" (money) mean?
- How can we transfer value?

Easy to answer if we use (gold) atoms to represent value:

- gold atoms are hard to make
- only one person at a time can "own" an atom

Things get complicated if we want to use bits:

- easy to generate bits
- bits can be copied ⇒ double-spending becomes a problem!

(Token-based) Possession-based vs Account-based methods

- In a possession-based method, owning the representation ≡ owning the value
- In an account-based method, there is usually some or ledger → TTP who "maintains accounts" (e.g. a "bank"); xactions cause value to be shifted from one acct to another.
- Most "bit-based" methods are account-based.

Simple example: Electronic checks

- Account-based: Bank has PK_B, SK_B

  User has PK_u, SK_u, cert on (U, PK_u) by bank

- Check = $\left[\begin{array}{l}\text{cert (on } PK_u\text{, signed by } SK_B\text{)}\\ \text{sign(}SK_u\text{, "Pay Bob \$100, date, serial \#")}\end{array}\right.$

- Bank deposits check just once (using ser #)

- Usual problem of overdrawn acct (bad check)

- Bank knows xact details: payer, payee, amt, date
- Merchant " " "

This works.

What else is possible?

Can we make payments more like cash?

Desirable (?) Properties

- Non-forgeable (prevent fraud, inflation)
- Not <u>double-spendable</u>
- Reliability: can "back up" your $
- Exclusive ownership
- Transferability: A can pay B
- Transitivity: B can use A's payment to pay C
- Variable-denominations
- Divisibility & combinability
- Efficiency (esp. for small amts)
- On-line <u>versus</u> off-line transactions
- scalability
- anonymity
- security
- Conversion to "ordinary" money

traditional:

- medium of exchange
- store of value
- unit of account
  "unit of measure"

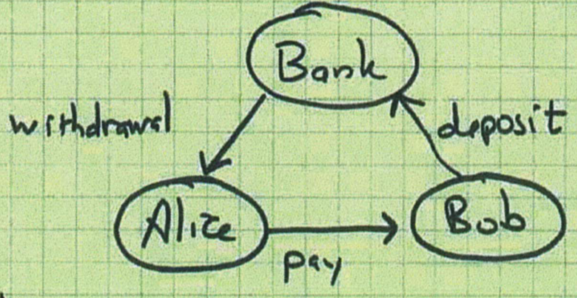
Double-spending

- essentially a "replay attack"
- if you can backup your $\$$, then "restore" gives you your spent money back !?
- <u>prevention</u> seems really tough (unless you use atoms)
- <u>detection</u> requires convergence of spending records (e.g. at bank) and large databases (?) ledger
- even if you can detect double-spending – what do you do?
  - roll back / deny transaction (2nd merchant to get same electronic coin can't deposit it)
  - punishing perpetrator may be impossible if we have (true) anonymity: payer is not identifiable
  - Furthermore: is <u>payer</u> or <u>payee</u> the culprit? (can merchant "frame" consumer?)
  - deterrence may be hard... how to punish (pay fine from account?)

Some approaches:

Signed coin ID

- Bank (TTP)
- Alice (payer)
- Bob (payee)



3 protocols to support:

① withdrawal / authorization
Alice becomes "able to pay"
(e.g. cert issuance in check scheme)

② payment

③ deposit

life of a "coin"

① withdrawal:

- Bank gives Alice $\underbrace{R, \text{sign}(SK_B, R)}_{= \text{coin}}$ ← unforgeable object!

R is coin ID

- Bank keeps R in data base of unspent coins
- Bank debits Alices acct for withdrawal

② payment:

Alice gives coin to Bob; Bob checks Bank sig

③ deposit:

Bob gives coin to Bank, Bank checks sig & R in DB
flags R as "spent"

TOPIC:

DATE:

FILE UNDER:

PAGE:

MicroMint (Rivest & Shamir 1996)

Let $h: \{0,1\}^* \rightarrow \{0,1\}^d$ be a hash fn, where d is modest.

To find a k-way collision, find

$$x_1, x_2, \ldots, x_k \qquad \text{(all distinct)}$$

s.t.

$$h(x_1) = h(x_2) = \cdots = h(x_k) \qquad [\text{Verification easy}]$$

By generalized "birthday paradox" arguments need about $2^{d(k-1)/k}$ hashes to find a k-way collision. (E.g. $n^{2/3}$ for a 3-way collision, where $n = 2^d$)

Looking at c times as many yields $\approx c^k$ collisions (great economy of scale).

Micromint: coin is a k-way collision.

Efficient generation seems to require some memory (e.g. $n^{1/3}$ memory).

- Not very efficient – bank has to sign each coin!
- Double-spending can be a problem!
- Check scheme better – merchant can't frame user!

~~Peppercoin (Micali & Rivest)~~

- ~~"probabilistic payments":~~

  ~~paying 10¢ $\cong$ paying \$10 with probability 1/100~~
  ~~(micropayment) (macropayment, sometimes)~~
- ~~based on electronic checks method~~
- ~~Alice pays Bob 10¢ as follows:~~

  ~~She gives Bob electronic check for \$10 that contains condition: "This check valid if and only if E is true" (where E holds with probability 1/100)~~
- ~~Bob must be able to test if E is true~~
  - ~~if so, he can deposit check~~
  - ~~if not, he throws check away (but gives Alice her purchase)~~
  - ~~he gets paid correctly on the average (law of large numbers)~~
- ~~Alice should not be able to tell if E is true when she writes check (else she can filter checks...)~~
- ~~Bank should be able to tell if E is true (so "bad checks" where E is false, don't get deposited.~~

TOPIC:

DATE:

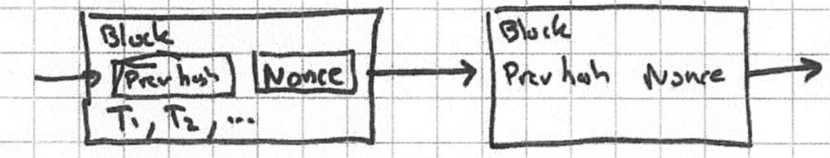
FILE UNDER:

PAGE: L07

Bitcoin:

- ID's are PK's (used for signing transactions) ECDSA
- public ledger records all transactions
  (account-based) PK ≡ acct name
- money created by/for those maintaining ledger ("miners")
  no other way to create money

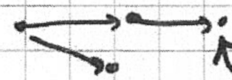
  (digress: discuss need for loans!)
- transaction detail:
  from accts: —, —,
  to accts: —, — (& amts)
  time
  valid if froms accts have enough value
- ledger detail: "block chain"



block valid if hash (block) begins with enough zeros & Tx's valid

no TTP!

need to search nonces to find one that works (like hash cash)
solved blocks are "published"
longest chain wins



difficulty level adapts to
yield ≈ one solution every 10 minutes

wait for enough (6) confirmations

TOPIC:

DATE:

FILE UNDER:

PAGE: L07

View blockchain.info

ledger is transaction log
implies <u>account</u> <u>balances</u>

verifying a transaction
involves checking account balances

<u>Issues:</u>

- Scalability?
  - can it do e.g. 4000 tps (like Visa?)
  - blocks may reach 1/2 GB, block now is only 30GB or so
  - ECDSA signature verification main computational need

- Phasing out of miner's fee? (in 2140 or so ... 21M BTC) (reward halves every 4 years)
  xact fees
  25 BTC/block now

- Acct-based but <u>no loans</u>!

  (Can't create money by giving a loan, as you can with current monetary system.)

  Like "gold standard" in terms of "coin creation"

- Protocol vulnerabilities:
  - large pool of miners (>50%)
  - "majority is not enough" (Eyal/Sirer)