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 $\Rightarrow$ double-spending becomes a problem!

(Token-based)

Possession-based vs Account-based methods

- In a possession-based method, owning the representation $\equiv$ owning the value

- In an account-based method, there is usually some or ledger $\Rightarrow$ TTP who "maintains accounts" (e.g. a "bank");
  transactions 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 = \[\text{cert (on } PK_u, \text{ signed by } SK_B) \xrightarrow{\text{sign}(SK_u, "Pay Bob $100, date, ser#")}}\]

- Bank deposits check just once (using ser #)

- Usual problem of overdrawn acct (bad check)

- Bank knows exact details: payer, payee, amount, date

- Merchant " " " "

This works.

What else is possible?

Can we make payments more like cash?
Desirable (?) Properties

- Non-forgeable (prevent fraud, inflation)
- Not double-spendable
- 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 ants)
- On-line vs. 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 !?
- prevention seems really tough (unless you use atoms)
- detection 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 cash can't
     deposit it)
- punishing perpetrator may be impossible if we
  have (true) anonymity? payer is not
  identifiable
- furthermore: is payer or payee the culprit?
  (can merchant "frame" consumer ?)
- deterrence may be hard... how to punish
  (pay fine from account ?)
Some approaches:

Signed coin ID

1. Bank (TTP)  
   Alice (payer)  
   Bob (payee)

3 protocols to support:
1. withdrawal/authorization  
   Alice becomes "able to pay"  
   (e.g., can disable in check scheme)
2. payment  
3. deposit

1. withdrawal:
   - Bank gives Alice $R, \text{Sign}(SK_B, R) \leftarrow \text{unforgeable object!}$
   - $R$ is coin ID  
   - Bank keeps $R$ in database of unspent coins  
   - Bank debits Alice's account for withdrawal

2. payment:
   Alice gives coin to Bob; Bob checks Bank's sig

3. deposit:
   Bob gives coin to Bank; Bank checks sig & $R$ in DB  
   Flags $R$ as "spent"
Micromint (Rivest & Shamir 1996)

Let \( h: \Sigma^* \rightarrow \Sigma^d \) be a hash fn, where \( d \) is modest.

To find a \( k \)-way collision, find

\[ x_1, x_2, \ldots, x_k \quad \text{(all distinct)} \]

s.t.

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

By generalized "birthday paradox" arguments
need about \( 2^{(d-1)/k} \) 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's 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!

Peppercorn (Mizalk & Rivest)
- "Probabilistic payments":
  - paying $10 if $10 with probability 1/100 (micropayment)
  - paying $10 with probability 1/100 (micropayment, 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" when \(E\) is false, don't get deposited)
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
  (discuss: discuss need for loans!)
- Transaction detail:
  \[ \text{from acct: } -; -; - \quad \text{to acct: } -; -; - \quad \text{(& amt)} \]
  (other info: accs have enough value)
- Ledger detail: "block chain"

\[ \text{Block} \rightarrow \text{Prev hash} \rightarrow \text{Nonce} \]

\[ \text{Block} \rightarrow \text{Prev hash} \rightarrow \text{Nonce} \rightarrow \]

Block valid if hash (block) begins with enough zeros & its valid

need to search nonce to find one that works (hash hash chain)
solved blocks are "published"

Longest chain wins

difficulty level adapts to yield 1 solution every 10 minutes

Wait for enough (6) confirmations
View blockchain info

ledger is transaction log
implies account balances

Verifying a transaction
involves checking account balances

Issues:

• Scalability?
  • can it do e.g. 4000 tps (likewise?)
  • blocks may reach 1/3 GB, block now is only 306B or so
  • ECDSA signature verification main amplifier, 2x

• Phasing out of miners' fee?
  (in 2140 or so ... 21M BTC)
  • exact fees
  • 25 BTC/block now

• Fiat-backed but no loans!
  (Can't create money by giving a loan, as you can with current monetary systems.)
  • Like "gold standard" in terms of "in evetion"

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