Blockchain

Digital Cash is not New

- DigiCash Inc., founded in 1989
  - Based on Chaum's "blind signatures"
  - Strong crypto that ensures anonymity
- Back then, the "killer app" was thought to be ***micropayments***
  - Users on the Internet pay only a tiny amount (fraction of cent) for something
- DigiCash declared bankruptcy in 1998

Digital Currency

- We want create an all-digital currency
  - Like \$ or ¥ or € or …., but "better"
- Real cash is (relatively) anonymous
  - So digital currency should be too
- Digital currency is "better" since…
  - No central authority (i.e., banks)
  - No government to issue currency, etc.

Preliminaries: Work

- How to measure (digital) ***work*** ?
- Our unit of work will be 1 hash
- Suppose that we have a hash function $h(x)$ that generates an N-bit output
- Then randomly chosen input generates one of 2^N equally likely outputs
  - For any input R, have, $0 \leq h(R) < 2^N$
  - Different R yield uncorrelated hashes

Hashing to Prove Work

- *Suppose we have a* 16*-bit hash function*
- *For any* R*, we have* $0 \leq h(R) < 65{,}536$
- *If we want* R *so that* $h(R) < 64$*, then how many* R *values do we need to hash?*
- *Since*

$$h(R) = y = (y_{15}y_{14}y_{13}y_{12}y_{11}y_{10}y_9y_8y_7y_6y_5y_4y_3y_2y_1y_0)$$

we want output like (10 leading 0s)…

$$h(R) = y = (0000000000y_5y_4y_3y_2y_1y_0)$$

Work and Hashing

- *For* 16 *-bit hash, how many hashes until*

  $h(R) = y = (0000000000y_5y_4y_3y_2y_1y_0)$ *?*

- *For random* R, *we have a* 1/2 *chance that*

  $y = (0y_{14}y_{13}y_{12}y_{11}y_{10}y_9y_8y_7y_6y_5y_4y_3y_2y_1y_0)$

- *And* 1/4 *chance that*

  $y = (00y_{13}y_{12}y_{11}y_{10}y_9y_8y_7y_6y_5y_4y_3y_2y_1y_0)$

- *And* 1/8 *chance that*

  $y = (000y_{12}y_{11}y_{10}y_9y_8y_7y_6y_5y_4y_3y_2y_1y_0)$

- *And so on…*

Work and Hashing

- For 16-bit hash, if someone gives us an R such that h(R) < 64
  - Then expected number of hashes computed is 2^{10} ("expected" means average case)
  - That is, they have done 1,000 units of work
- We use hashing to show work was done
- Why this obsession with work?
  - That will become clear later…

Work and Hashing

- We can adjust parameter so more work (or less) is required
  - For N-bit hash, if we require $h(R) < 2^n$ then expected work is 2^{N-n} hashes
- **Note**: We can easily verify that the expected amount of work was done
  - Only requires a single hash
  - No matter how much work to find R

Preliminaries: Ledgers

- **Ledger** is a book of financial accounts
- Suppose Alice, Bob, Charlie, Trudy play weekly poker game online
- They all insert ledger entries such as,

  "Bob owes Alice $10", "Charlie owes Trudy $30", "Trudy owes Alice $25", and so on
- Once a month, they meet and settle up
- Any possible problems here?

Signed Ledger Entries

- How to prevent Trudy from inserting, say, "Bob owes Trudy $1M" ?
- Let's require **digital signatures**
  - For ledger entry to be valid, Bob must sign "Bob owes Alice $10", Trudy must sign "Trudy owes Alice $25", and so on …
- Then we know ledger entries are valid
  - That is, the payer agrees to pay

Signed Ledger

- Ledger now looks like
  - $[\text{Bob owes Alice \$10}]_{\text{Bob}}$
  - $[\text{Charlie owes Trudy \$30}]_{\text{Charlie}}$
  - $[\text{Trudy owes Alice \$25}]_{\text{Trudy}}$
  - and so on …
- And we know ledger entries are valid
- But, still some problems here…

Signed Ledger in Detail

- *As an aside, note that signatures on previous slide really look like*
  - $(M_1,[h(M_1)]_{Bob})$, *where* M_1=“Bob owes Alice \$10”
  - $(M_2,[h(M_2)]_{Charlie})$, M_2=“Charlie owes Trudy \$30”
  - $(M_3,[h(M_3)]_{Trudy})$, M_3=“Trudy owes Alice \$25”
  - *And so on …*
- *We'll use the shorthand on previous slide*

Ledger Duplication

- Still, nothing to prevent Trudy from duplicating a line…
  - [Bob owes Alice \$10]$_{Bob}$
  - [Charlie owes Trudy \$30]$_{Charlie}$
  - [Trudy owes Alice \$25]$_{Trudy}$
  - [Charlie owes Trudy \$30]$_{Charlie}$
- Signatures are still all valid
- How to prevent this attack?

Unique Ledger Entries

- Include unique transaction numbers
  - [1, Bob owes Alice $10]$_{\text{Bob}}$
  - [2, Charlie owes Trudy $30]$_{\text{Charlie}}$
  - [3, Trudy owes Alice $25]$_{\text{Trudy}}$
  - And so on…
- Why does this help?
- We will never have an exact duplicate
  - So any duplicate is invalid

Ledger Prepayment

- How to be sure participants pay up?
- Can start with Alice, Bob, Charlie, and Trudy all putting money into the pot
- And don't allow any transaction that would result in negative balance
- Transaction must still be signed and …
- … now, nobody can "overdraw" account

Ledger Prepayment Example

- Ledger example…
  - Alice has $100 // Alice's initial stake
  - Bob has $100 // Bob's intial stake
  - Charlie has $100 // Charlie's initial stake
  - Trudy has $100 // Trudy's initial stake
  - $[\text{Bob owes Alice \$10}]_{\text{Bob}}$ // **valid**
  - $[\text{Charlie owes Trudy \$30}]_{\text{Charlie}}$ // **valid**
  - $[\text{Trudy owes Alice \$25}]_{\text{Trudy}}$ // **valid**
  - $[\text{Trudy owes Bob \$120}]_{\text{Trudy}}$ // **invalid**

Ledger Prepayment

- Note that we must know the ***entire*** transaction history
  - So that we can know current balances
  - Then we can be sure a given transaction does not cause user to be overdrawn
- This seems like kind of a hassle, but some big benefits come from it
  - As we will soon see…

Eternal Ledger?

- Alice, Bob, Charlie, and Trudy could continue to settle accounts each month
- But, as the ledger currently stands, settling accounts is not necessary!
- We know the current balances, and no risk of anyone being "overdrawn"
- So, could play poker for months, years, or forever, without settling accounts

Ledger as Currency

- This ledger can act as its own currency!
  - Need a cool symbol, let's use "§"
- Transactions **within** ledger are all in terms of § currency protocol
- Anyone can exchanged ledger currency (i.e., §) for $ or ¥ or € or …
  - But, such exchanges occur **outside** the ledger currency protocol

Ledger Currency

- For example, Alice could pay Bob $10 in real world dollars for, say, §5 of currency in the ledger system
- Comparable to exchanging, say, $ for ¥
- But, ledger is a history of transactions within the ledger currency system
- In fact, **the ledger is the currency**
  - This is **the** key insight for cryptocurrency

Distributed Ledger

- The ledger is the currency
  - So who is in charge of the ledger?
  - A govt? The UN? A bank? An individual?
- We don't trust them, so let's make **everybody** in charge of the ledger
  - Anybody can have copy of ledger, anyone can add entries (there is a protocol…)
  - Protocol without a central authority
- What problem(s) do you foresee?

Distributed Ledger

1. Transactions must be signed
2. Nobody can be overdrawn
3. Transactions broadcast to everybody
   - How to have a consistent view of this distributed ledger?
   - Multiple ledgers exist at any time
   - This is the heart of the issue for a distributed cryptocurrency (e.g. Bitcoin)

Distributed Ledger and Work

- Every ledger will have some amount of work associated with it
- And, ledger with most work "wins"
  - That is, everyone accepts ledger that has the most work put into it
- Recall, work is measured in hashes
- So, more hashes is "more better"

Blocks and Hashes

- Each transaction is signed
- Transactions grouped into **blocks**
  - Let B be one such block
- Find (nonce) R so that $h(B,R) < 2^n$
  - Just fancy way of saying $h(B,R)$ starts with a specified number of 0s
- Work required to find R?
  - On average 2^{N-n} hashes for N-bit hash

Chain

- *Don't want to revalidate each block, want to order blocks, and so on*
- *We'll **chain** blocks together*
  - *Put hash of previous block in header of current block before computing hash*
- *So, must find* R *so that* $h(Y,B,R) < 2^n$
  - *Where* Y *is hash of previous block*

Blockchain

- We now have

$$Y_{i+1} = h(Y_i,B_i,R_i) < 2^n$$

$$Y_{i+2} = h(Y_{i+1},B_{i+1},R_{i+1}) < 2^n$$

$$Y_{i+3} = h(Y_{i+2},B_{i+2},R_{i+2}) < 2^n$$

- Each B is a block
  - Block is a group of signed transactions
- Each R is chosen so inequality holds
  - Much work to find R, easy to verify $Y < 2^n$

Mining?

- Anyone can create a new block
- But lots of work to find a valid hash
- So what is the incentive to do work?
- "Free" money!
  - Get (new) money for doing work, say, §1
  - Put this info at start of block, does not need to be signed (since new money)

One Block

- *Block* B_i *looks like…*

$Y_i = h(B_{i-1})$
Miner_x gets §1 $[1, \text{Bob owes Alice §10}]_{\text{Bob}}$ $[2, \text{Charlie owes Trudy §30}]_{\text{Charlie}}$ $[3, \text{Trudy owes Alice §25}]_{\text{Trudy}}$ $\vdots$
R_i

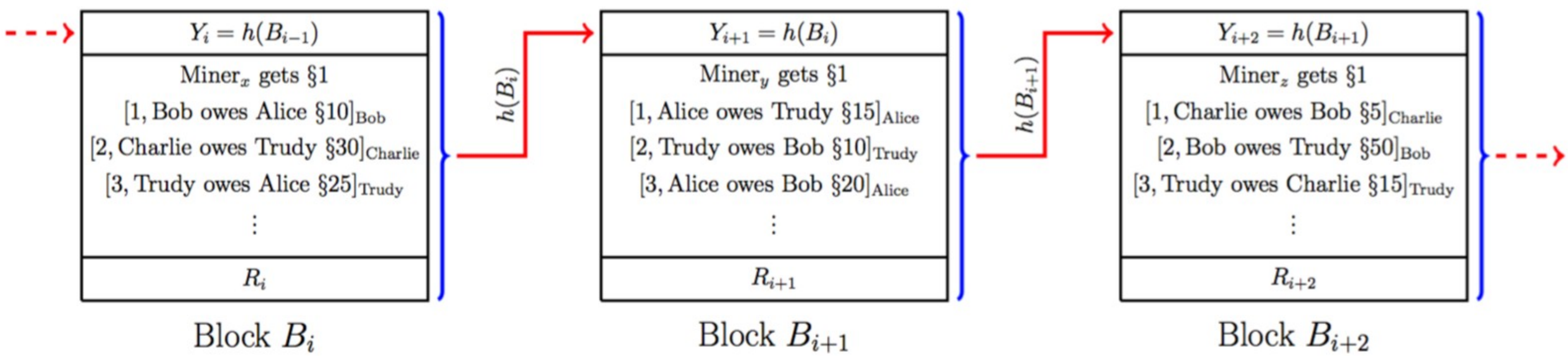
Block B_i

Mining

- Free money, so miners are in a race to find hashes that yield valid blocks
- The more computing power a miner has, the better chance to win race
- Once a valid hash is found, miner sends the block out to everybody
- Again, easy to verify hash is correct

Blockchain

- Blockchain looks like…



Mining

- Why is "mining" called mining ?
  - Really, just finding a valid block hash
- Miner is doing work, and creating new money that did not previously exist
  - In a sense, this is comparable to mining gold or silver (for example)
- This may be the most misunderstood part of cryptocurrency protocols

Non-Miners

- Users do not have to be miners
- Non-miner just wants blockchain
  - Needed to know how many §s others have
- Also, non-miner sends out transactions for others to make blocks (and mine)
- User might see conflicting blockchains
  - What to do in such cases???
- More work is "more better"!

More Work

- If conflicting blockchains, how to know which represents more work?
- Each block is a fixed amount of work
  - In terms of expected number of hashes
- So, longer block chain is more work
- Thus, **longer** block chain always wins
  - If it's a tie, wait until one is longer

Summary of Protocol

1. New transactions broadcast
2. Miners collect transactions into blocks
3. Miners race to find valid block hash
4. When miner finds hash, broadcast it
5. Block accepted if all transactions signed, no overdraft, & block hash valid
6. New block extends blockchain
   - Miners use hash of new block in next block

Attack Scenario

- Suppose Trudy makes a block B that includes transaction
  - [Trudy pays Alice §100]$_{\text{Trudy}}$
  - Trudy sends B to Alice **only**, nobody else
- Q: Why would Trudy do this?
- A: So she can spend that §100 again
  - Trudy likes double spending!
  - It's free money!

Double Spending

- For Trudy's double spending attack to work, she must compute valid hash
  - That is, find R, so that $h(Y,B,R) < 2^n$
- And send chain with block B to Alice
- But, nobody else knows about B, or the chain that contains it
  - All other miners working on other chains
  - Those other chains can (and will) grow
  - Trudy is in ongoing race with ***all miners***

Double Spending Attack

- Alice will reject Trudy's chain once a longer chain appears
- Trudy would need most of computing power in the network to win
  - Trudy needs to win a lot!
- Or, miners collude with Trudy
  - But is it in their interest to do so?

Blockchain

- From users perspective…
  - Transaction in last block might not be entirely trustworthy
  - Possibility of double spending attack
  - But, the more blocks that follow, the more certain that a transaction is valid
- Just wait until a few more blocks are added before accepting a transaction

Refinements

- Number of hashes can change so that winning hash takes constant time
  - Computing power in network can increase
  - In Bitcoin, new block every 10 minutes
- Can decrease mining reward so money supply does not grow forever
  - E.g., maximum of 21,000,000 bitcoins
  - Then what will be incentive for miners?

Refinements

- Merkle tree can be used to reduce storage requirements
  - Transactions in a block hashed in a tree, only the root is needed in block hash
- Simplified payment verification
  - In effect, rely on others to verify for you
- Combining and splitting value
  - Transaction can have multiple input/output

Privacy?

- Can use pseudonym in public key
- But, can still connect transactions to a specific public key
  - Might be able to tie public key to an individual based on transactions
  - We'll see examples like this later…
- Not a strong form of anonymity
- Bitcoin is said to be "pseudonymous"

Future of Blockchain?

- Blockchain can be viewed as a way to implement a distributed ledger
- Useful for cryptocurrency, but many other possible applications too
- Blockchain said to be a "foundational" and/or "disruptive" technology
- Perhaps, but I'm not convinced…

References

- Excellent video: https://www.youtube.com/watch?v=bBC-nXj3Ng4
- Original bitcoin paper (surprisingly easy to read): Bitcoin: A peer-to-peer electronic cash system, Satoshi Nakamoto, https://bitcoin.org/bitcoin.pdf