Bitcoin

Tom Anderson

Outline

Last time: SpecPaxos

Today: Bitcoin

Bitcoin Goal

Electronic money without trust

Why Not Cash?

+ portable
+ cannot spend twice
+ cannot repudiate after payment
+ no need for trusted 3rd party
+ anonymous (serial #s?)
- doesn't work online
- easy to steal

+/- hard to tax / monitor
+/- government can print more as economy expands

Why Not Credit Cards/PayPal?

+ works online
+ somewhat hard to steal
+/- can repudiate
- requires trusted 3rd party
- tracks all your purchases
- can prohibit some transactions (e.g. wikileaks donations)
+/- easy for government to monitor/tax/control

Bitcoin

Suppose we had a system where a penny was just a string of bits

What's hard technically?

- Forgery: what's to keep someone creating many copies?
- Double spending: what's to keep someone from using the bits twice?
- Theft: what's to keep someone from learning the bits and then spending them?

Bitcoin

What's hard socially/economically?

- Why does the string of bits have value?
- How do you convert it to cash?
- How to pay for infrastructure?
- Monetary policy (intentional inflation, …)
- Laws (taxes, money laundering, drugs, terrorists)

Crossing the Chasm

Theory of technology adoption (Geoffrey Moore)

Early adopters

- Tech that solves a compelling problem
- Worth hassle of a partially working system

Early majority

- Pragmatists: need whole product solution

Late majority

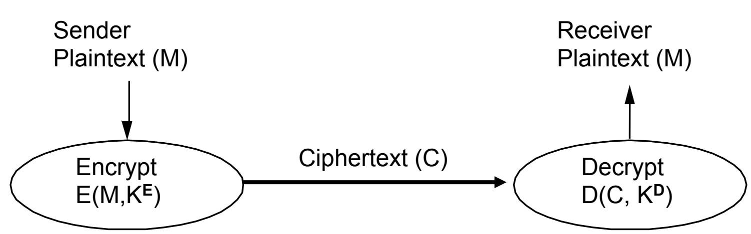
- Tech needs to be cheap, reliable, widely used

Laggards

Examples

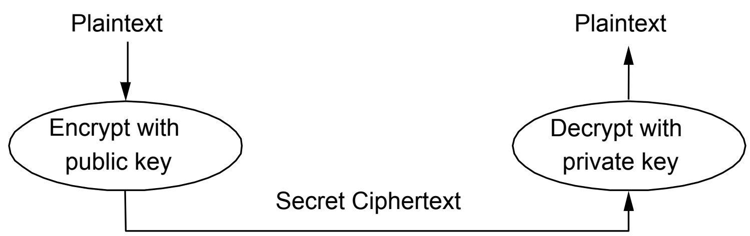
- Cellphones
  - Early users: drug dealers, international business travellers
- Email and the web
  - Early users: scientists, pornographers
- Cloud computing
  - Early users: Internet search, high-speed traders
- Bitcoin
  - Early users: drug dealers, money launderers

Encryption



- Cryptographer chooses functions E, D and keys K^E, K^D
  - Suppose everything is known (E, D, M and C), should not be able to determine keys K^E, K^D and/or modify msg
  - provides basis for authentication, privacy and integrity

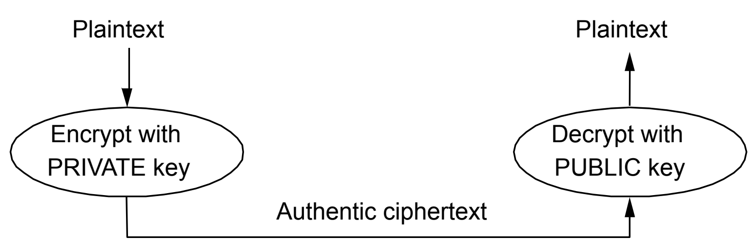
Public Key (RSA, PGP)



Keys come in pairs: public and private

- Each principal gets its own pair
- Public key can be published; private is secret to entity
  - can't derive K-private from K-public, even given M, (M)^K-priv

Public Key: Authentication



Keys come in pairs: public and private

- M = ((M)^K-private)^K-public
- Ensures authentication: can only be sent by sender

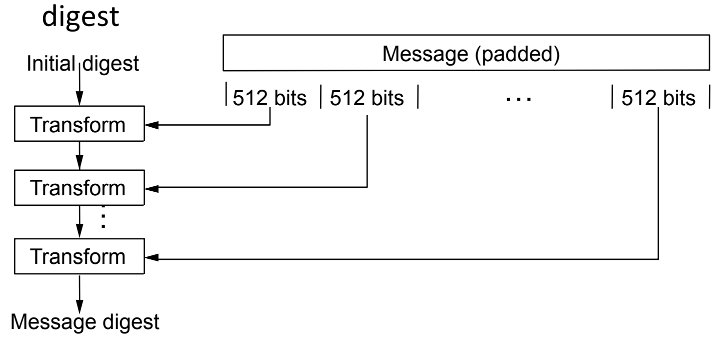
Public Key: Secrecy

Keys come in pairs: public and private

- M = ((M)^K-public)^K-private
- Ensures secrecy: can only be read by receiver

Message Digests (MD5, SHA)

- Cryptographic checksum: message integrity
  - Typically small compared to message (MD5 128 bits)
  - “One-way”: infeasible to find two messages with same digest



Infocoin Straw Proposal

Suppose a transfer is a signed statement, in Alice's private key: "Alice gives Bob infocoin #57”

Issues?

- Who assigned the serial #? can Alice just mint money?
- Easy for Bob to copy Alice’s statement; why can't he use it twice?
- Easy for Alice to sign statement; why can’t she do that twice?

With a Trusted Intermediary (Bank)

- Alice withdraws a coin from the bank; gets a unique serial # (signed with Bank's private key)
- Alice signs certificate (with her private key)
- Bob checks certificate with bank to see that serial # is valid (belongs to Alice) and not double spent

Do we have to trust the bank?

Suppose bank keeps a visible log of operations
- Replicated public ledger (block chain) with all transfers in sequence
- Replicas could be run by volunteers!

Alice creates block, signed by A's private key
- B's public key
- Coin #

B creates block, signed by B's private key
- C's public key
- Coin #

Preventing Double Spending

Want each transfer to be unique, applied at a specific place in the sequence of operations, so:

B creates block, signed by B's private key
- hash of previous block
- C's public key
- coin #

Any recipient can check coin # against an (up to date) replica, to prevent double spending

Managing the Public Log

- Need updates to be applied in the same order at each replica
- Different replicas receive updates at different times
  - How do readers know replica is up to date?
- Use Paxos?
  - What if replicas aren't trusted?
- Use Byzantine Paxos?
  - Still need to trust 2f + 1 replicas

Use Metasync?

- Dropbox, Baidu, … have append-only logs
  - allow anyone to read from log
- With Metasync, no need to trust any single replica, but ok to trust the aggregate?
- However, Dropbox permissions are too soft
  - anyone who can write log, can also delete log

Bitcoin

Protocol for managing replicated log

- Replicas run by volunteers
- Allow double spending to be detected
- Provided a majority of replicas are well-intentioned
- Make it hard for anyone to control a majority of replicas

Log Management Straw Proposal

- Assume large number of replicas
- Every new op sent to one replica, rebroadcast to all
- Slow system down to reduce the chance of a conflicting updates
  - Every node picks a random delay before applying update
  - For 1M nodes, 1/600M => 1 update every 10 minutes
  - Might still conflict!
  - For higher throughput, batch transactions
- Still requires some trust
  - to pick the random # correctly, etc.

Sybil Attack

- If anyone can be a replica, then:
  - Alice run a billion replicas, convinces Bob to accept transfer as legitimate
  - Bob will only be able to check a subset
  - How does Bob know the subset isn't colluding?
  - how can he know
- Proof of work: force replicas to do work
- But that will discourage volunteers, make it easier for Alice to acquire a majority of replicas
- Bitcoin solution: reward replicas for doing work

Proof of Work

- Replicas perform a puzzle
  - Puzzle is public: whoever completes the puzzle first determines the next (batch of) ops in log
  - and gets a reward
- Bitcoin uses a simple computational puzzle, find a nonce such that:
  - SHA256(msg!nonce) = 0...
- SHA is a cryptographic hash: no easier way to find a match except to guess

Proof of Work

Match on first zero? Too easy; two tries on average

Match on first two zeroes? Too easy; four tries on average

Bitcoin (currently) requires 69 leading zeroes

- 1,210,954,923 GHash/sec
- $10K reward per solution, 10 minutes
- Difficulty adjusted to keep solutions at fixed rate

Some Details

Hash difficulty is not binary

- SHA256(msg|nonce) < value
- Allows fine-grained adjustment of proof of work

Prevent solving ahead

- SHA256(previous hash|msg|nonce) < target

Transactions batched

- Roughly 2000 ops per batch, so ~ 3/second

Reward

- Solution is broadcast to every replica; what keeps replicas from stealing the solution?
  – Every replica works on a slightly different puzzle
- X works on:
  – SHA(previous hash|mint coin and give it to X|msg|nonce) < target
- Y works on:
  – SHA(previous hash|mint coin and give it to Y|msg|nonce) < target

When Nonce is Found

Replicas have a choice:

– Ignore the answer and continue to try to find another one

– Take the answer as a given and work on the next puzzle.

Which should it choose?

– If more than half of the computational power chooses (b), replica should choose (b)

Who Wins?

- If two nodes find the nonce at about the same time, who wins?
- Depends on solution to the next puzzle!
- Everyone has an incentive to work on chain that others will work on
  - If next solution uses A's solution, A wins
  - If next solution uses B's solution, B wins

Mining Groups

- Reward is sporadic: if 1M replicas search for hash, each will win once every few decades.
- Can we pool resources so group of replicas win more regularly?
  - Pay nodes to look for solutions
- Suppose Y is a coordinator. Ask replicas to do:
  - SHA(previous hash|mint coin and give it to Y|msg|nonce)
- Hand out small reward for anything with 50 leading zeros

Mining Incentives

- Do replicas have an incentive to announce a solution as soon as it is found, or keep it secret?
- Release and get reward, if standalone solver
- Keep secret, if control > 50% of compute power
  - Solve puzzle
  - Start solving next puzzle
  - Release first solution if competing solution is announced
- Bitcoin creator performed first k entries in block chain, taking first k rewards

Mining Incentives

- Do replicas have an incentive to include a proposed transaction in hash computation?
  - Hash is valid even if the miner ignores all requested transfers
- Each transaction transfers fee to whoever computes the hash
  - Currently $0.10/transaction
- How does that compare to a debit card transaction fee?

Serial Numbers Revisited

- Proof of work solves how we create new coins
- Every 10 minutes, another reward
- What about inflation?
  - Reward decreases by 2x every few years
  - Increasing number of coins in circulation
  - Fixed total number of coins (today, 93% of total)

Bitcoin

- Network of bitcoin peers (servers) run by volunteers
- Peers are not trusted: many may be corrupt
- Each peer knows about all bitcoins and transactions
- Transaction (sender -> receiver):
  - sender sends transaction info to some peers
  - peers flood transaction to all other peers
  - receiver checks that lots of peers have seen transaction
  - receiver checks that bitcoin hasn't already been spent

Transactions

- Mined coins aggregated into transaction record
- Each transaction record has a public key
  - Only owner can transfer funds onward
  - Multi-output: to receiver, to miner
  - Check remaining balance > transfer
  - Prevents double spending
- Bitcoin servers maintain the complete chain
- Miners only accept valid transactions

What's in a Transaction Record?

- Hash pointer to source of funds (unspent transaction)
- Amount to be transferred
- Amount to be paid to miner
- Public key of new owner
- Signed by private key of previous owner

Block Chain

- Transactions aggregated into blocks
- Each block includes hash of previous block
- Miners receive transactions
  - Validate before include
  - Compute hash on set of transactions in block
- Block valid only if solve puzzle
- And next solved block includes hash, …

Example

- Bitcoin owned by user Y (who received it in payment from X)
- T7: pub(Y), hash(T6), sig(X)
- Y buys a hamburger from Z and pays with this bitcoin
- Z needs to tell Y Z's public key (bitcoin "address")
  - Perhaps create a new address just for Y's purchase
- Y creates a new transaction and signs it
- T8: pub(Z), hash(T7), sig(Y)

Example

- T8: pub(Z), hash(T7), sig(Y)
- Y sends T8 to bitcoin peers, which flood it
- honest peers verify that
  - no other transaction mentions hash(T7),
  - T8's sig() corresponds to T7's pub()
- Z waits until lots of peers have seen/verified T8
- verifies that T8's pub() is Z's public key,
- then Z gives hamburger to Y

Questions

Where is Z's resulting bitcoin value "stored"?
- bitcoin balance = unspent transaction
- Z "owns" the bitcoin: has private key that allows Z to make next transaction

Does transaction chain prevent stealing?
- current owner's private key needed to sign next transaction
- Attacker can steal Z's private key
- Z uses private key a lot, so probably on his PC, easy to steal?
- a significant problem for bitcoin in practice

Double Spending

- Suppose Y creates two transactions: Y->Z, Y->Q
- Z and Q probably don't check all the peers
  - Y has a chance to tell diff peers diff transactions
- Maybe some peers are corrupt and cooperating with Y
  - hide Y->Q from Z, hide Y->Z from Q
- Only need to play tricks briefly
  - just until Z gives the hamburger to Y

Double Spending

How long should Z wait before giving Y the hamburger?

Until Z sees Y flood the transaction to many peers?
- not in the chain, Y might flood conflicting xaction

Until Z sees one peer with chain ...<-BZ (containing Y->Z)?
- maybe that peer is corrupt, in league with Y

Until Z sees lots of peers with chain ...<-BZ?
- risky -- some other chain may win
- perhaps that chain won't have Y->Z

Until Z sees chain with multiple blocks after BZ?
- slim chance attacker can catch up