Applied Cryptography and Network Security
CS 1653

Summer 2023

Sherif Khattab

ksm73@pitt.edu

(Slides are adapted from Prof. Adam Lee's CS1653 slides.)

Announcements

- Project Phase 4 Due tonight @ 11:59 pm
  - Please schedule Project Phase 4 Demo with Pratik as soon as possible
- Project Phase 5 Due on Monday 8/7 @ 11:59 pm
  - No demo meetings for this phase
- Programming Assignment 3 canceled
  - due to delays in certificate issuance for PA 2
- Things due this Friday
  - Phase 4 Peer Evaluation Survey
  - Homework 10
  - Midterm reattempts
- OMETs Bonus
  - Entire class gets 1% bonus when response rate is 80% or more (currently at 6%)
  - OMETs due on 8/5
- Office hours updated for this week
  - Please check https://khattab.youcanbook.me/
  - You can still request (by email) office hours outside those times

Final Exam

- Take home: download from GradeScope, (print), solve, (scan), and upload to GradeScope
  - open-book and open-notes
  - some overlap with midterm topics
- Please check study guide on Canvas
  - Review video will be posted soon
- Exam will be posted tomorrow
  - You will have until Sunday 11:59 pm to finish and upload the exam
- No class on Wednesday

Private Routing

(Very quick) overview of routing

Discussion: What are desirable security properties in routing?

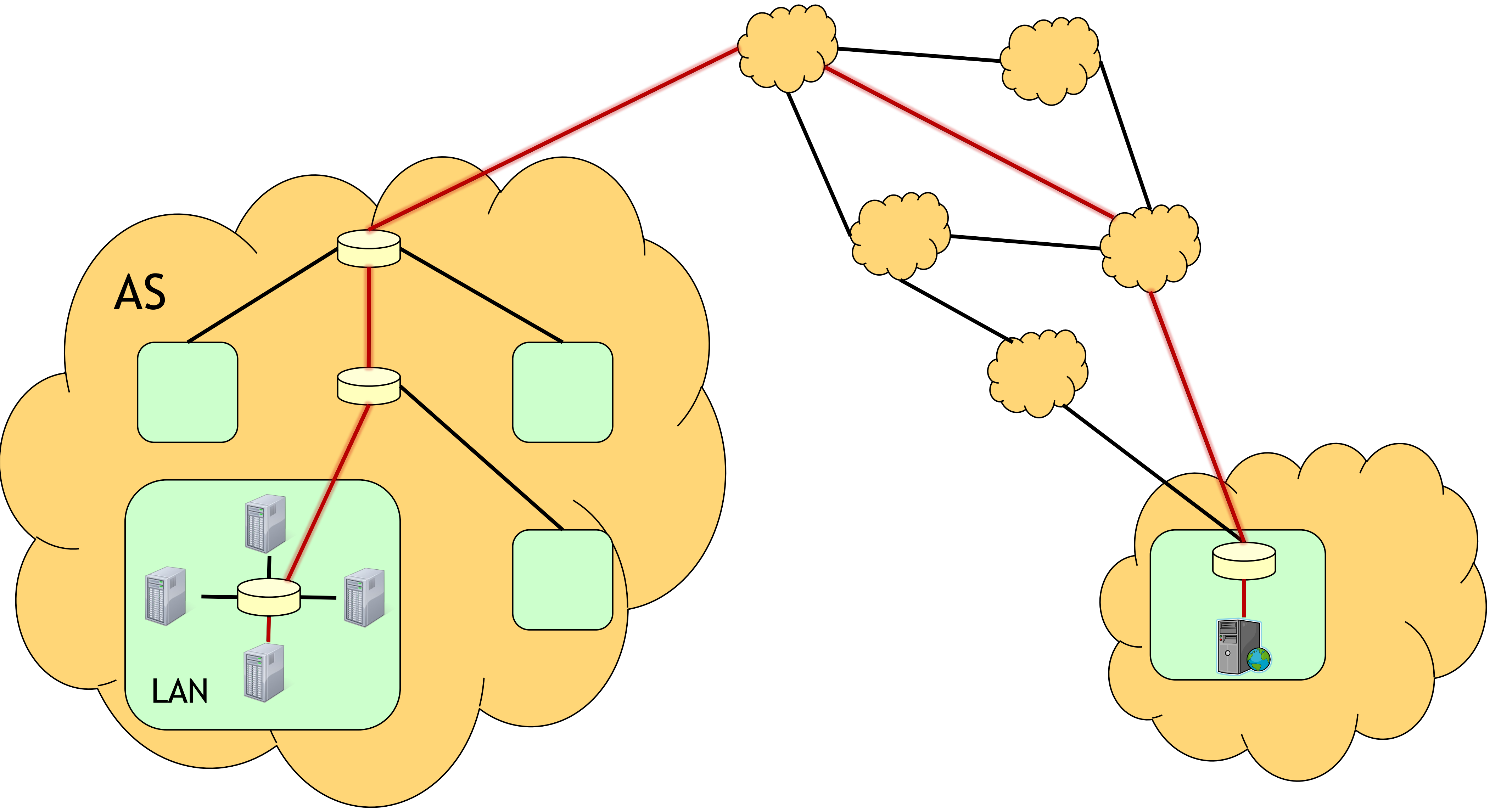
Confidentiality & Firewalls

Integrity & DNSSEC

Privacy & Crowds, Tor

Network overview and routing

The Internet is a network of networks



Networked systems represent a change in paradigm from the "old days"

What makes networked systems vulnerable to attack?

1. All traffic is routed in public
   - Early days: Principals largely trusted, plaintext traffic → IP simplified
   - The result is that communicating on the Internet is like shouting in a crowded room, but users think they're sending private messages

2. Traffic flows across many administrative domains
   - High degree of exposure across potentially untrusted territory

3. There may be multiple routes between two points

4. Principals have some degree of anonymity in the system
   - Users don't need physical access to resources
   - Simplified IP means that addresses are not bound to identities
   - Worse yet, stepping-stone attacks!

Discussion Question

Given these vulnerabilities, what are some threats against network systems?

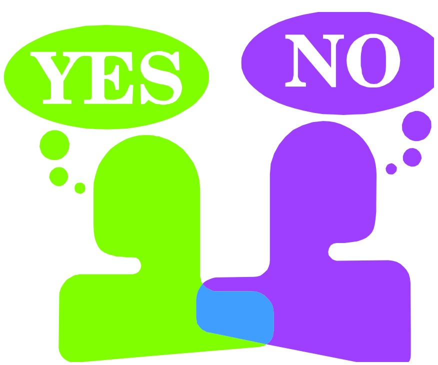
A sampling of threats for networked systems…

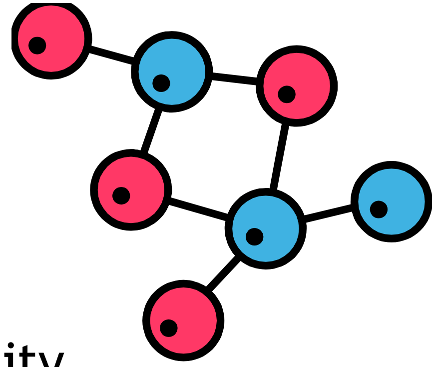
Encryption

Authentication

Confidentiality
- Exposure via passive wiretapping
- Inadvertent disclosure to impersonators
- Inference via traffic analysis

Tunnels





Authentication

Integrity
- Modifications by active wiretappers
- Information accepted from dubious sources

DNSSEC

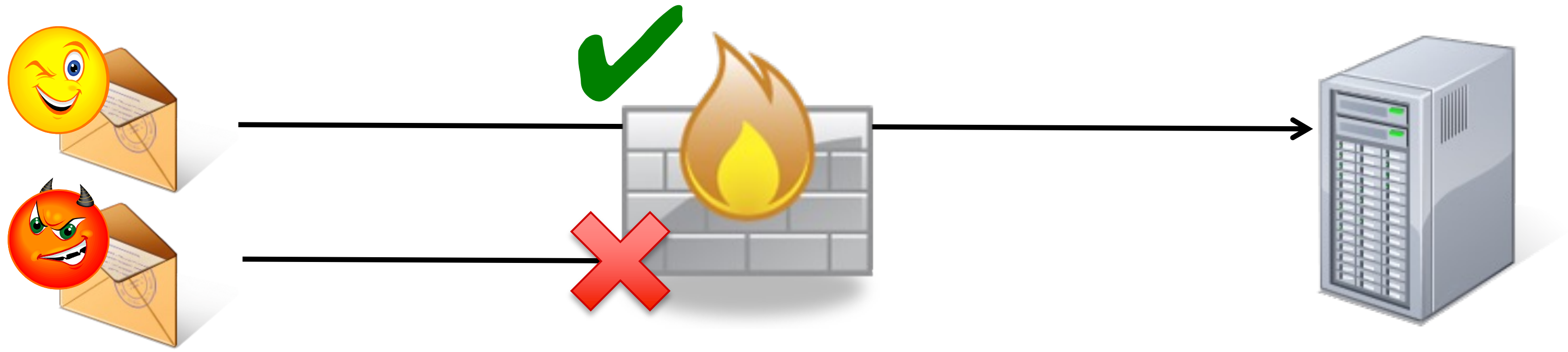
Availability
- Link failures
- DoS/DDoS attacks
- Accidental block-listing

Firewalls

FIREWALLS

Firewalls are *the* ubiquitous network security mechanism

First proposed in 1988, firewalls act like networked reference monitors



Why are firewalls necessary?

- The early days were a more optimistic time!
- Allowing all traffic to be routed between all points greatly simplified the design of TCP/IP

Several types of firewalls are in widespread use

- “1G”: Packet filtering firewalls
- “2G”: Stateful packet inspection firewalls
- “3G”: Application-layer firewalls
- Machine-learning powered firewalls

Packet filtering firewalls are the simplest type of firewall

As packets arrive, a pass/drop decision is made by inspecting headers:

- IP protocol (e.g., TCP, UDP, ICMP, …)
- Source and destination IP addresses
- Source and destination ports (if applicable)

Most firewalls allow rules to either grant or deny access

- allow TCP * * 123.4.5.6 80
- deny * 67.8.9.10 * * *

Everyone should have access to the web server

This might be a compromised server attempting to infect us with a worm

Packet filtering firewalls usually run on dedicated devices with very simple operating systems

- No linkable libraries, compilers, etc.
- Minimal system tools
- Sometimes, no system accounts!
- Why? Minimal TCB

If compromised, make it hard for the attacker to stage more attacks

Make it hard to change the system configuration

No accounts means no privilege escalation!

Simplicity gives rise to complexity

Packet filters often have enormous numbers of rules

- In large corporations, firewalls with 10k rules are not uncommon
- Anecdotally, some firewalls have up to 50k rules!

Worse, often the order of rules matters

- allow TCP * * 123.4.5.6 80
  deny TCP 67.8.9.10 * 123.4.5.6 80
- deny TCP 67.8.9.10 * 123.4.5.6 80
  allow TCP * * 123.4.5.6 80

Request from 67.8.9.10 permitted!

Request from 67.8.9.10 denied!

What happens if a packet does not match any defined rule?

- Security administrators like default deny policies
- Users like default permit policies

This complexity has sparked many interesting research directions

- What high level policy is encoded by a set of firewall rules?
- What are the effects of changes to firewall policies?
- How can rules be more efficiently organized?
- What is the net effect of multiple layered firewalls?

More complex firewalls allow more expressive policies, at a cost

"2G" Stateful firewalls keep track of connection state

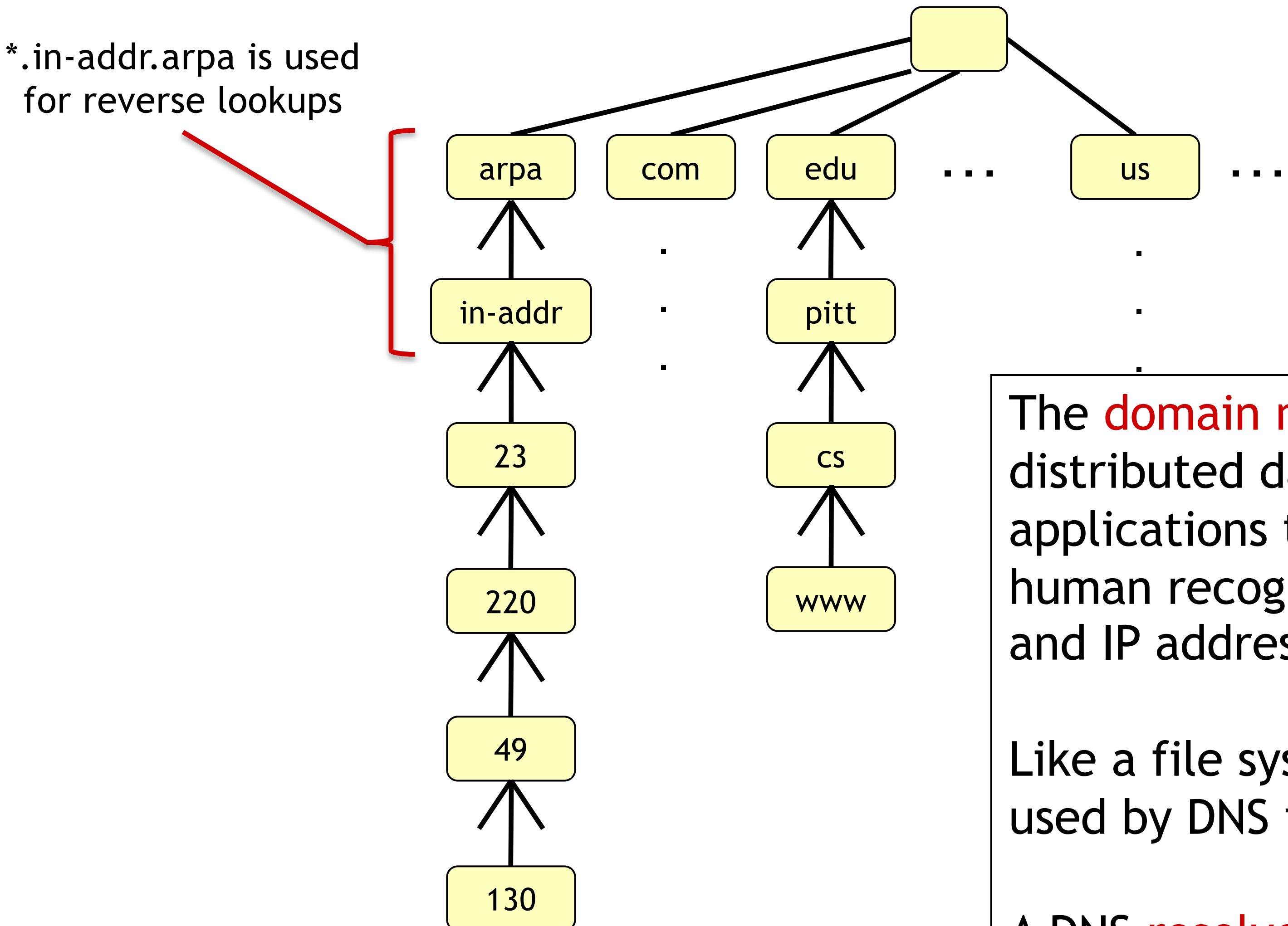
- Distinguishes new connections from existing connections
- Example: FTP
  - Commands over port 21, transfers over arbitrary high ports
  - Consider single packets → deny
  - Associate with existing connection → allow
- Vulnerable to DoS via filling connection state memory

"3G" Application-layer firewalls inspect application data in packets

- Ability to interpret application layer allows tighter control
- Detect forbidden protocols over allowed ports, misuse of allowed protocols
- Bind application-layer identity (e.g., FTP username) to originating IP
- Scan for viruses and worms
- Much higher overhead than lower-layer firewalls

DNS SECURITY

DNS is the mechanism through which host names are resolved into IP addresses

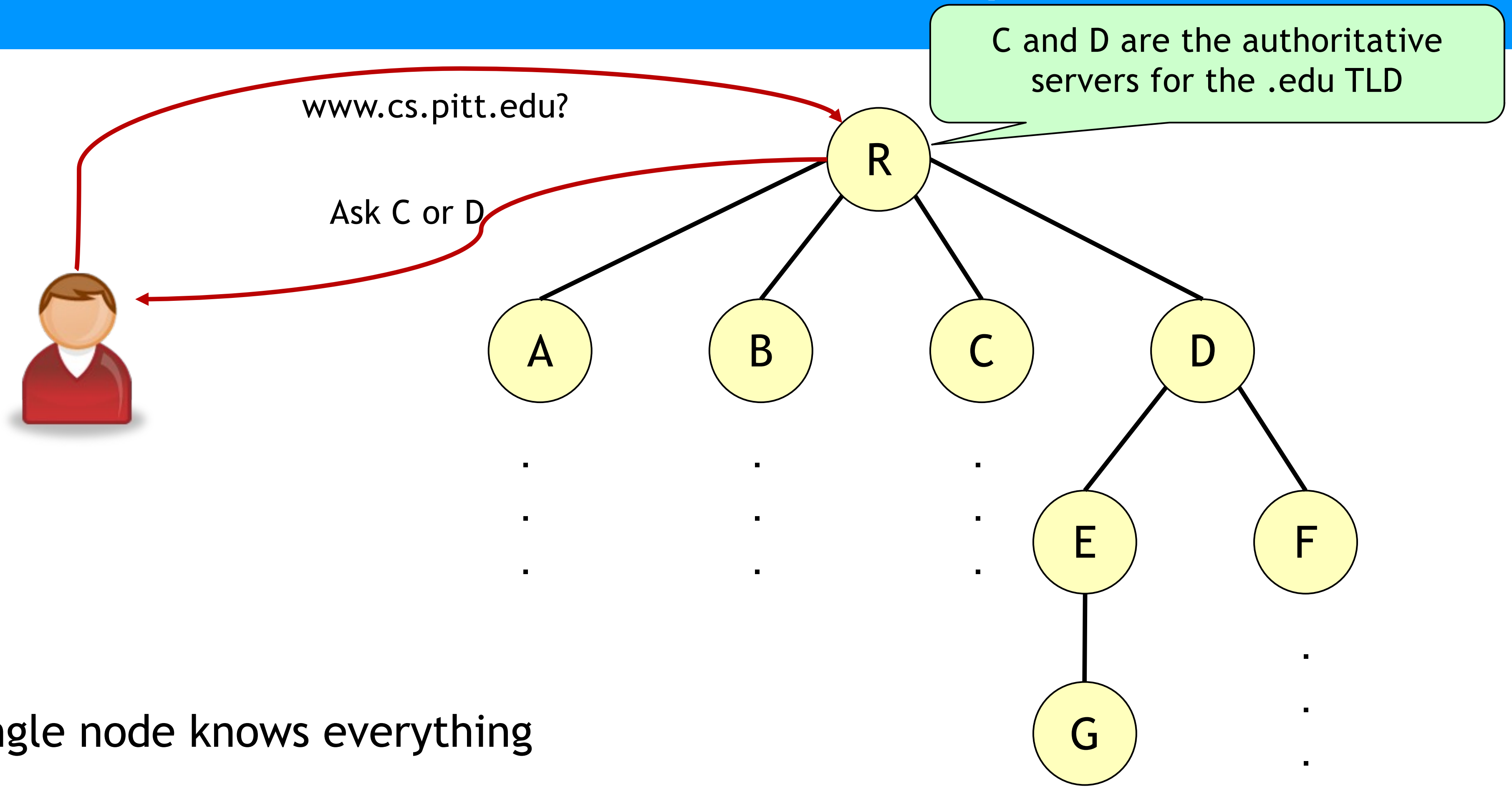


The domain name system (DNS) is a distributed database that allows applications to map between human recognizable domain names and IP addresses

Like a file system, the namespace used by DNS is hierarchical

A DNS resolver can start at a known "root" node and issue recursive queries until it finds the address that it is interested in

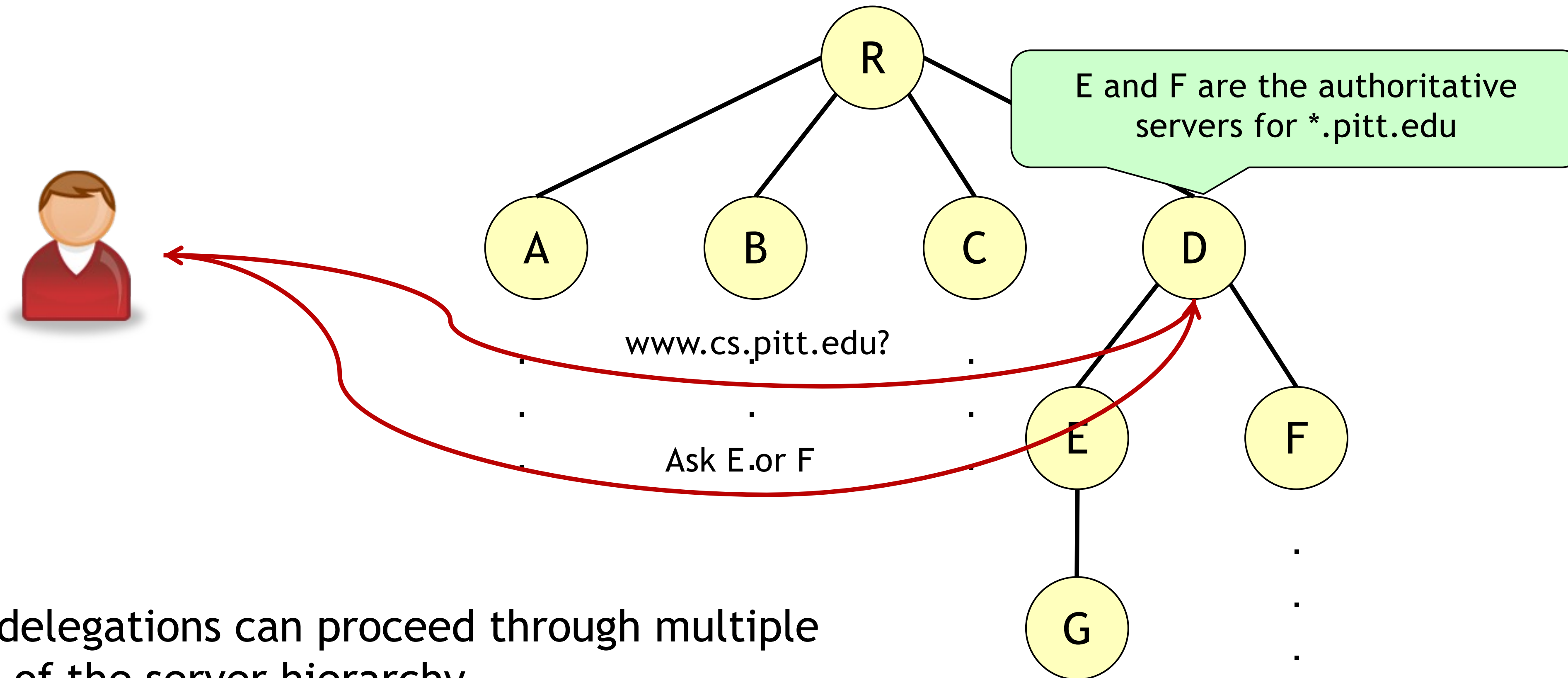
DNS Resolution Example



No single node knows everything

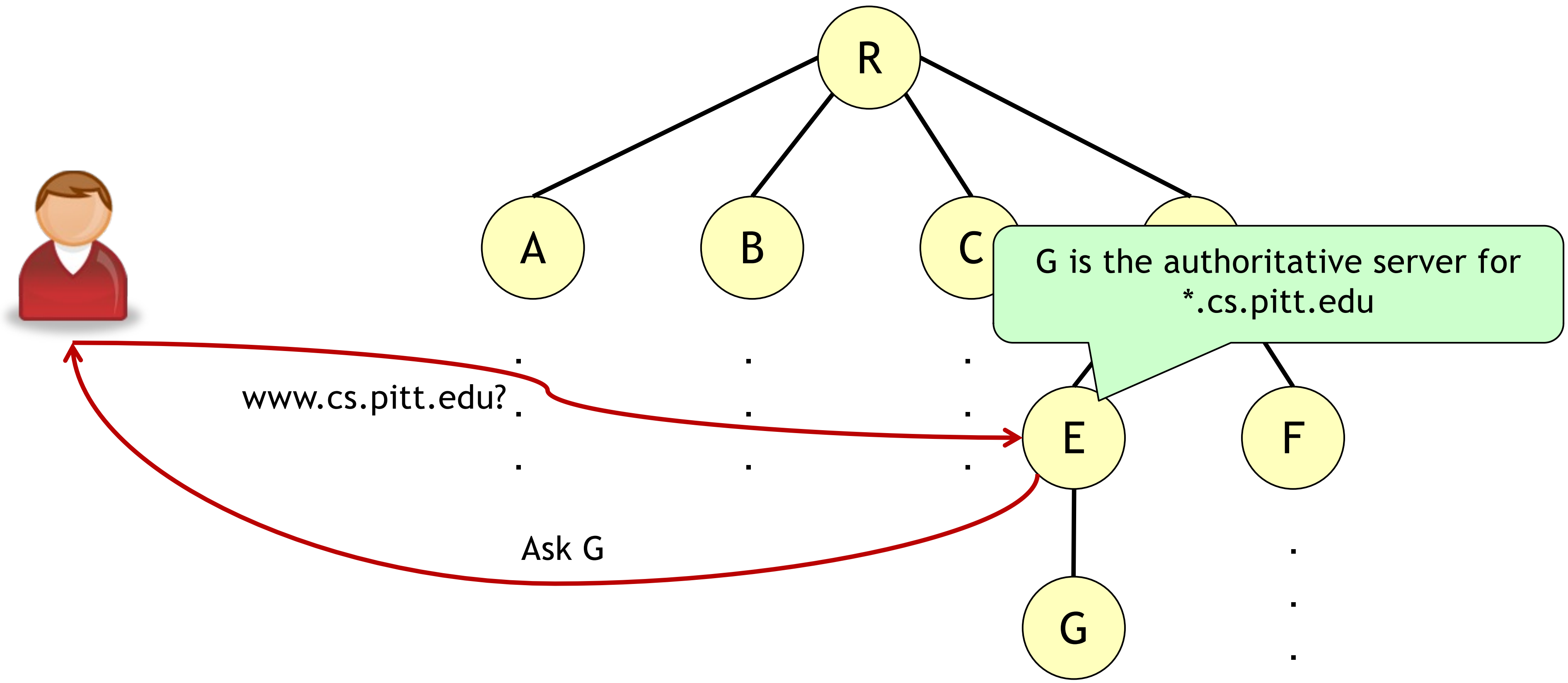
NS records identify the authoritative servers for a particular zone

DNS Resolution Example

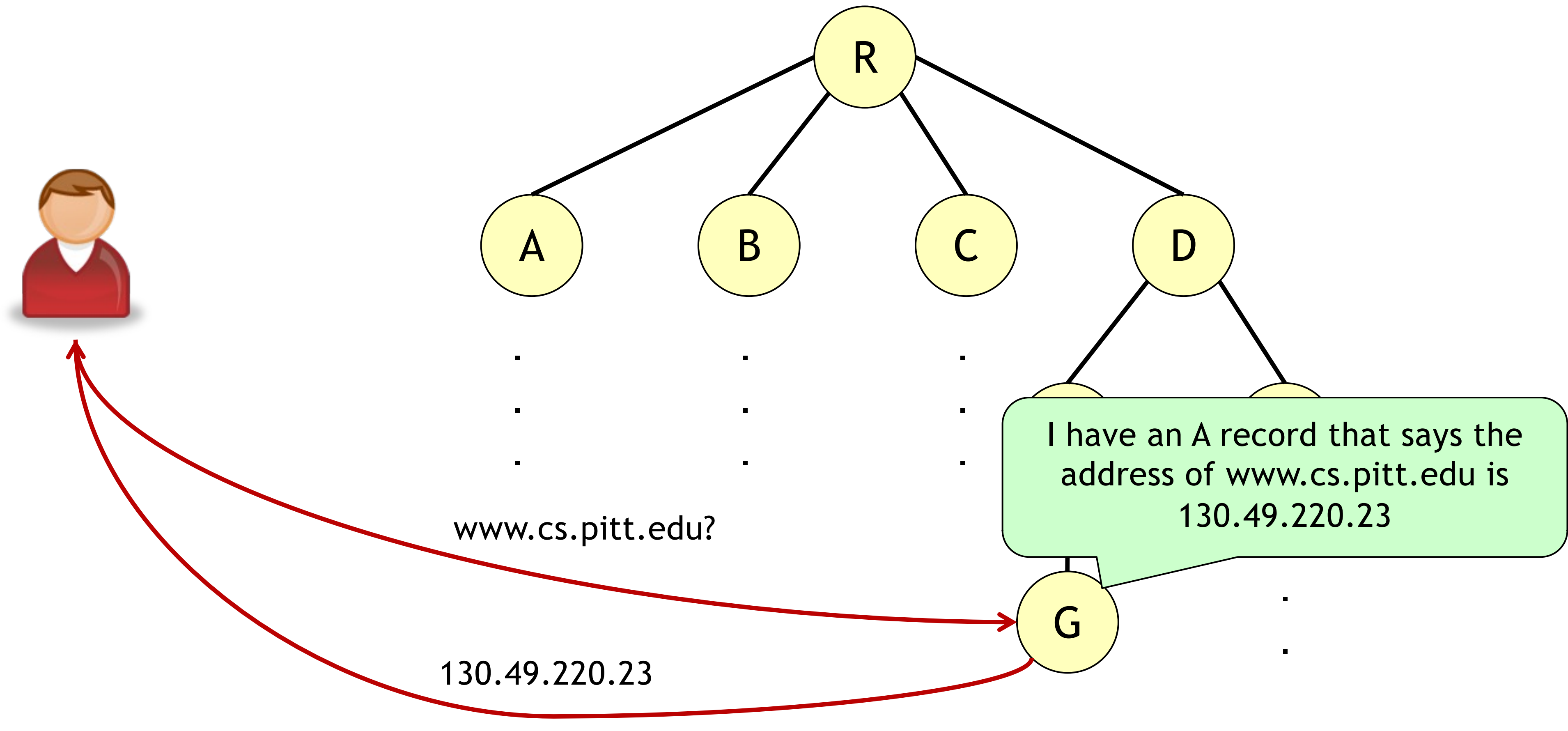


Zone delegations can proceed through multiple levels of the server hierarchy

DNS Resolution Example



DNS Resolution Example

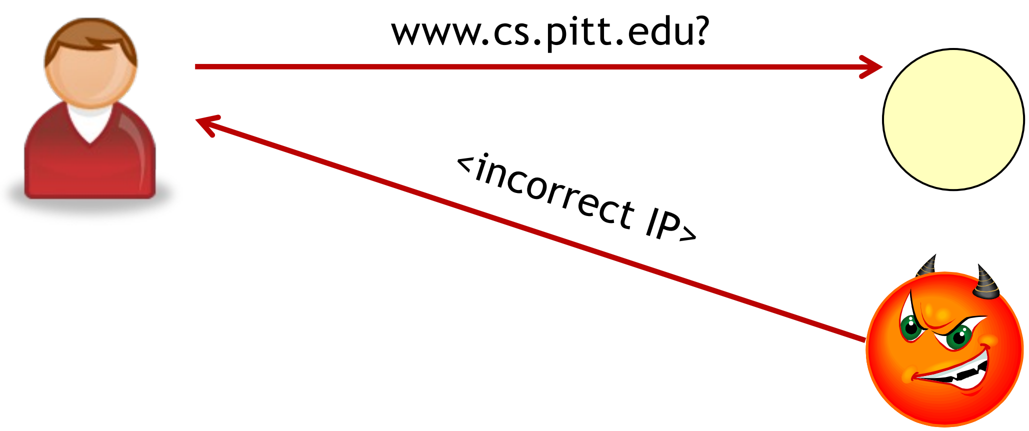


A records are used to manage the bindings between a domain name and an IP address

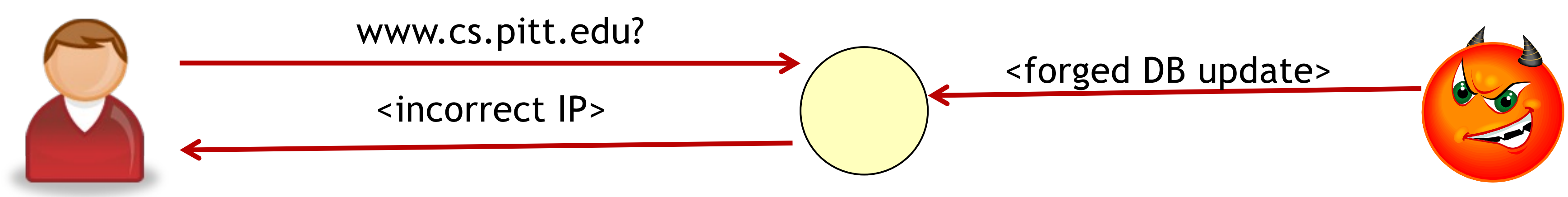
How could such a simple system have problems?

DNS lacks a strong authentication mechanism that can be used to verify the integrity of records returned

Example: Forgeries injected at the last hop



Example: Incorrect zone transfers



DNSSEC has been proposed as a solution to these types of problems

Insight: DNS servers form a hierarchy, PKIs also form a hierarchy!

To hybridize DNS and PKI, new DNS resource records were introduced

- DNSKEY records are used for each node to store its public key
- DS records are used to delegate authority to other nodes
  - Kind of like an NS record
  - A signed binding between a node IP, zone, and public key
- Furthermore, all DNS resource records can be signed

To use DNSSEC, nodes need the keys of trusted root DNS servers

The result is authenticated responses!

- DNSKEY→[DS→DNSKEY]*→Signed RRset

Trust root

Zero or more signed delegations
through the zone hierarchy
(like PKI certificates)

Signed A records

So, why haven't we seen widespread deployment of DNSSEC yet?

Reason 1: Some people want to see the protocol more thoroughly vetted

- Earlier versions of DNSSEC had problems
- Complex 6-message protocol to update node keys
- Every record on the child server had to be signed by parent
- Result: Couldn't scale to Internet sizes

Reason 2: US representation on the Internet

- Many root servers are located in the US
- Other nations perhaps uneasy about giving so much control over cryptographic keys to one political entity

Reason 3: Privacy

- DNS zone data is usually kept private (phone book vs. reverse lookup)
- DNSSEC also needs signed failure messages
- Combination of signed failures and signed successes allows adversaries to look up all data regarding a zone

PRIVATE ROUTING

Application-level encryption cannot fully protect privacy

Open System Interconnection Reference Model

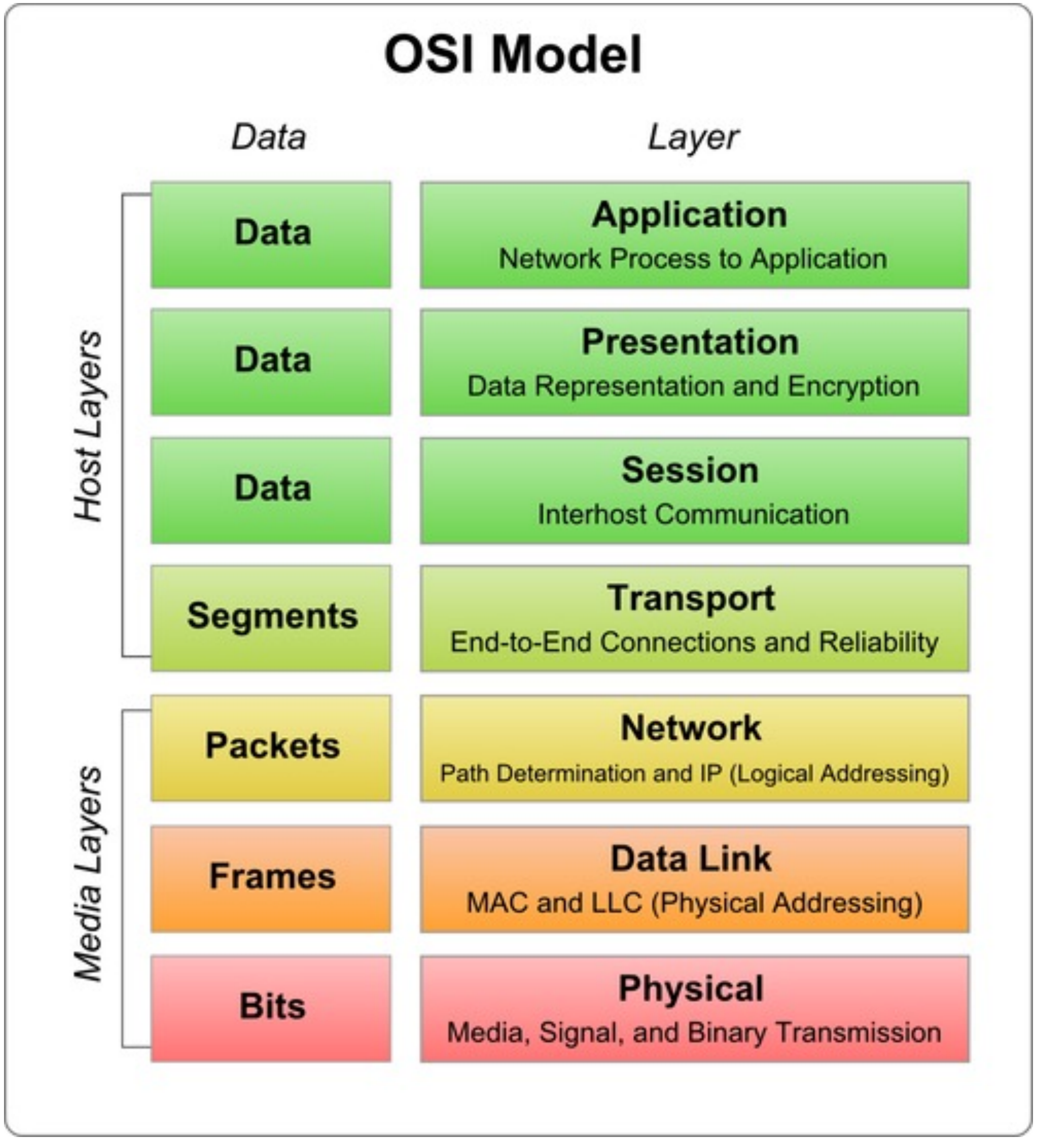
Host layers can easily be encrypted

- e.g., SSH, TLS

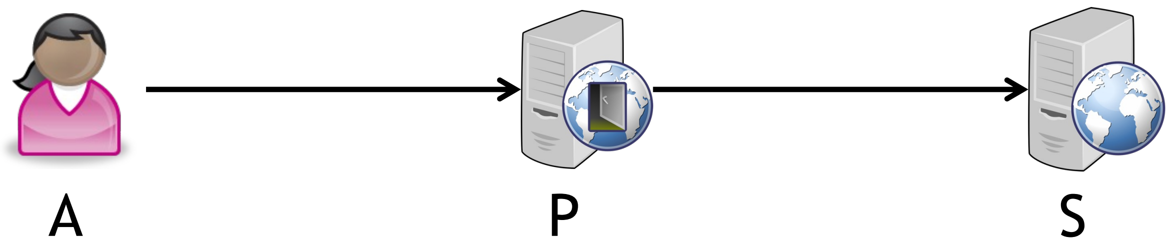
What about info at the network layer?

- e.g., IP addresses?

Passive adversary can infer a lot from this information!



A single trusted proxy server can mitigate some of these problems



By using a proxy, Alice obscures the fact that she is communicating with S

Passive eavesdropper (e.g., ISP) never sees a packet identifying both Alice and S

S sees incoming traffic from P, not Alice

Question: What does the proxy see?

Crowds can reduce the amount of trust placed in a single proxy

Intuitively: Hide in a *crowd* of other users

Two types of entities

- *Jondos:* clients wishing to hide in the crowd
- *Blenders:* trackers maintaining a list of active jondos

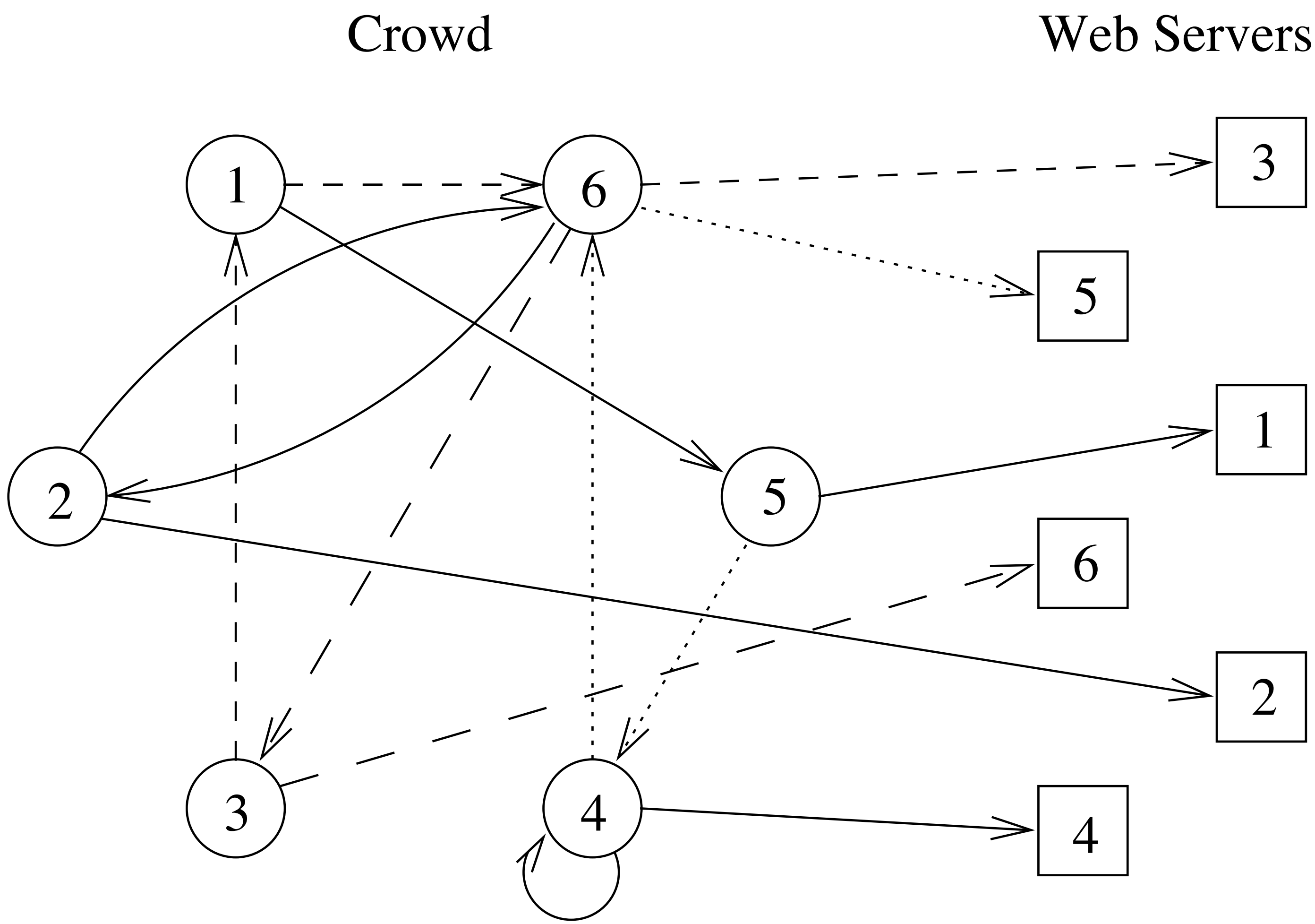
New Jondo registration with a blender

- Blender assigns new jondo a symmetric key
- Blender notifies other jondos and distributes key

Path construction when sending messages

- Each message is sent via a random path through the crowd
- When a jondo receives a message, it either:
  - Forwards it to another jondo (probability $p_f > ½$)
  - Sends it to its destination (probability $1 - p_f$)

Paths through the crowd



How much privacy can crowds give Alice?

Passive eavesdropper:

- Sees an outgoing packet to another jondo
- Knows a packet originated with Alice if there is no corresponding incoming packet
- With 1 - p_f chance, sees Alice submit message to endpoint

Endpoint:

- Sees an incoming packet, could be from anyone in the crowd

Other jondos:

- Cannot tell whether packet originated with Alice or if she is passing it along
- Colluding jondos? Alice is safe if:

Probability of forwarding

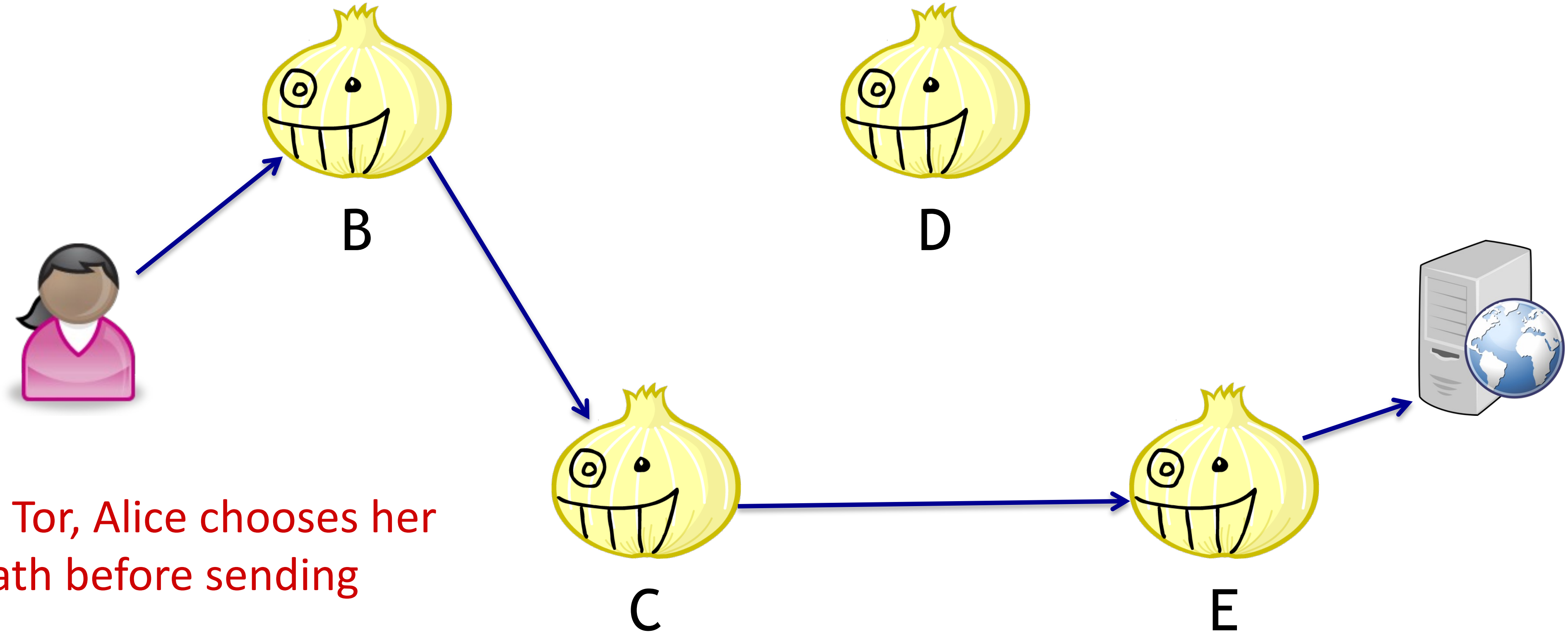
Number of jondos in crowd

$$n \geq \frac{p_f}{p_f - \frac{1}{2}}(c+1)$$

Number of colluding jondos

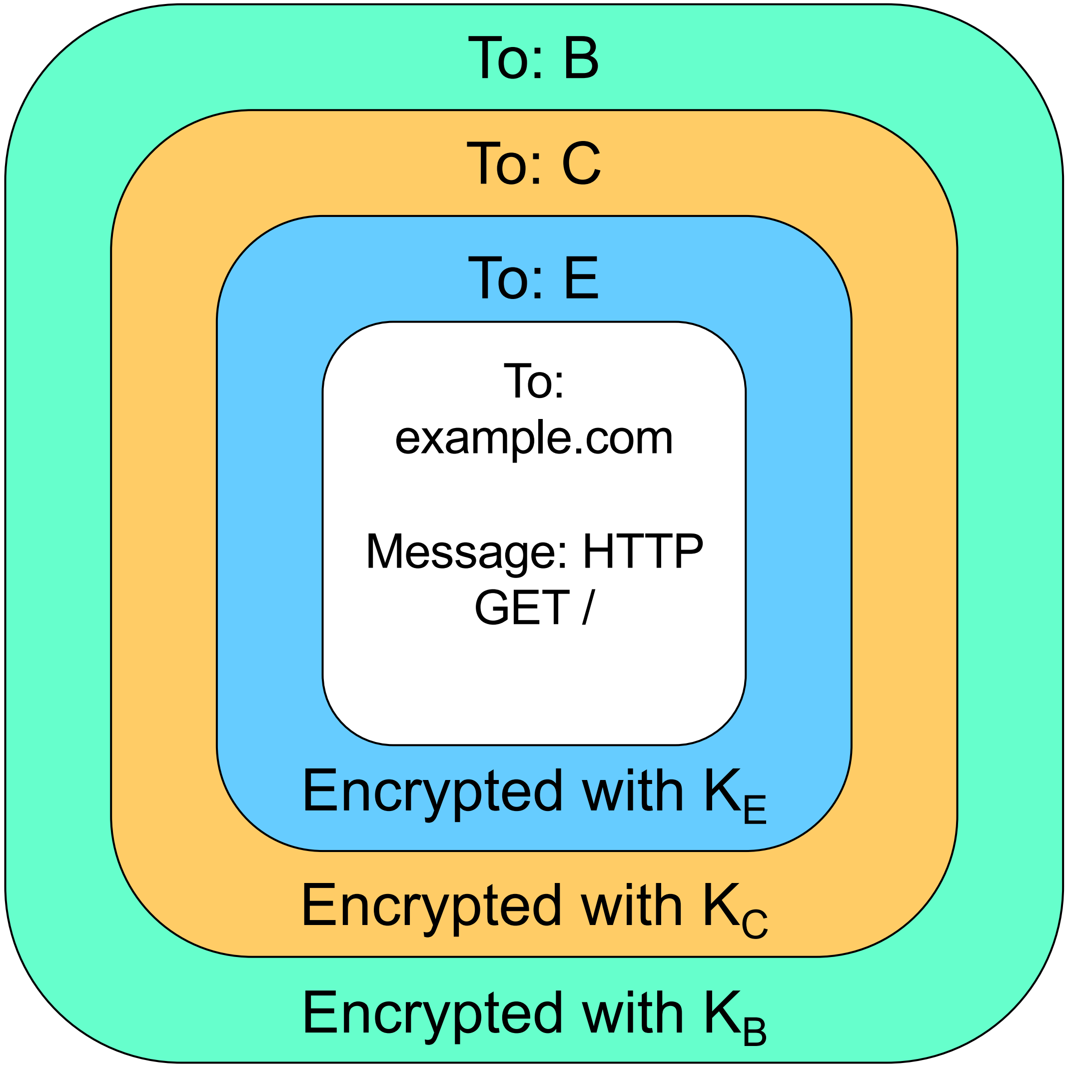
Can we introduce encryption to increase privacy guarantees?

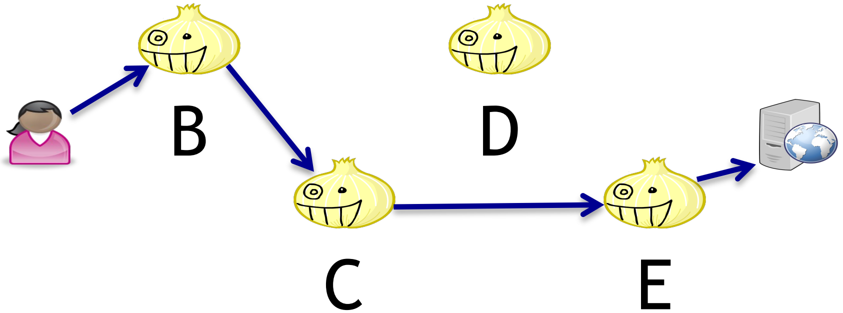
This is the idea behind onion routing (Tor)!



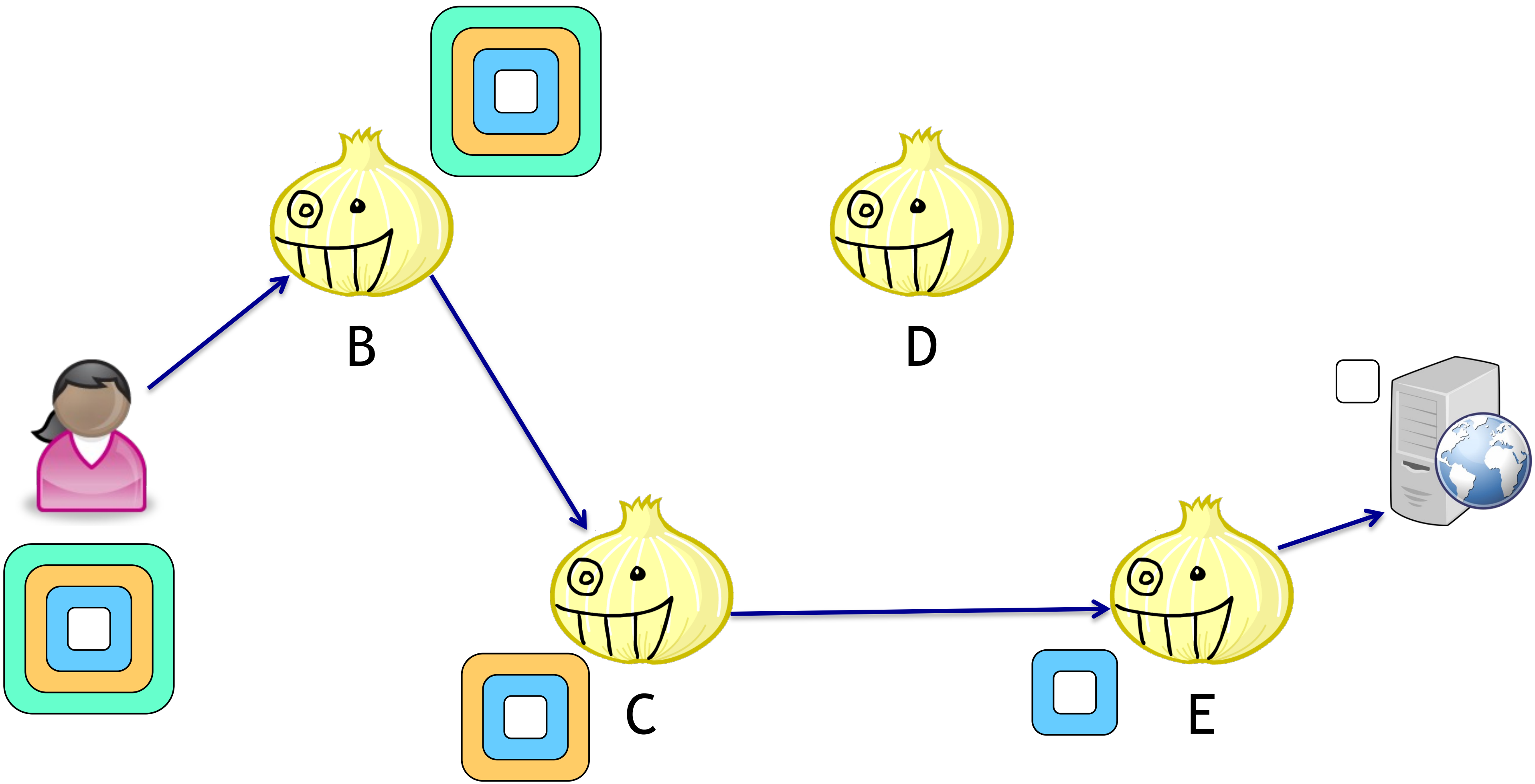
In Tor, Alice chooses her path before sending

A message in Tor is protected by multiple layers of encryption, starting at the end of the path





How does Alice send an onion message?



How much privacy can Tor give Alice?

Passive eavesdropper:

- Sees an onion packet outbound to an onion router
- Might know the packet originated with Alice, if Alice is not an onion router

Endpoint:

- Sees an incoming packet from the Tor exit node

Onion routers:

- First router sees that the message is from Alice (and the next onion router)
- Exit node sees that the message is intended for the endpoint (and the onion router that sent it)
- In between, onion routers see nothing except previous/next routers

What threats still exist against Tor?

Global passive adversary

- Can observe all traffic between all nodes
- By observing timing of packets, might be able to statistically determine who is talking to whom
- Question: How can this be prevented?

Control of exit node

- Exit node sees plaintext message and destination
- What if this packet contains any identifying information?
- Intuition behind Bad apple attack: Application sends real IP in onion message, then exit node can identify the client

Onion router collusion

- What if an attacker controls many onion routers?

Summary

Network security is a huge topic; one size rarely fits all

Firewalls are used repel intruders

- Different classes of firewalls trade off efficiency for policy expressiveness

DNSSEC is a proposal to maintain lookup integrity

- However, concern over its implications for privacy and political power have inhibited its wide-scale deployment

Crowds and onion routing protect message flow privacy

- Privacy is not a boolean property—it is more like a continuum

Data Privacy

What is data privacy? Why should I care?

Models for data privacy

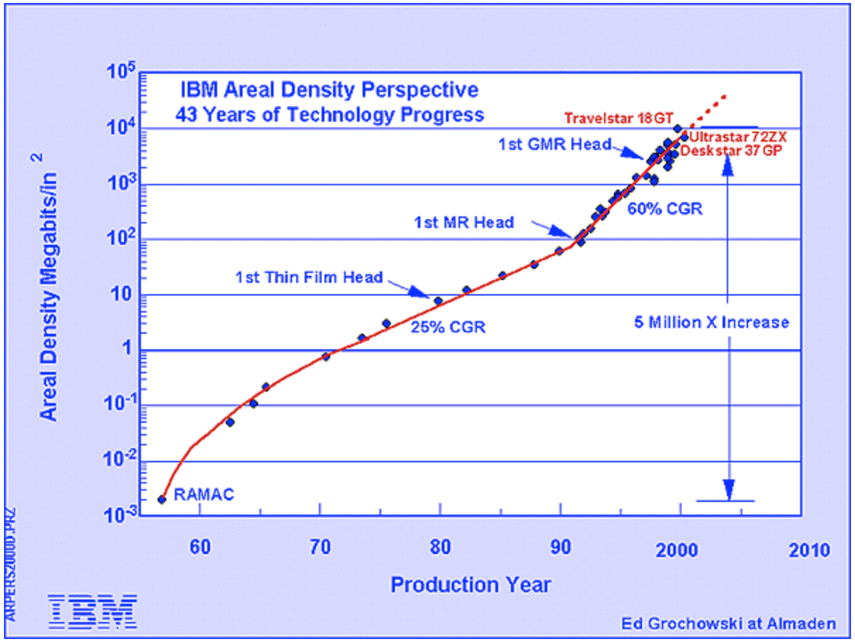
- Anonymize and release
- Mediated query processing
- Outsourced data management

Case study: k-Anonymity

- How does it work?
- Why doesn’t it work?

Future directions

Data, data everywhere!



Hard drive sizes are absurd!

- Capacity increasing
- Cost decreasing
- Example: 1TB backup drive costs < $10

Thought: Why delete anything? Just as easy to keep it all…

Result: Our whole lives are on disk!

These days, "data" means more than "documents"

- Electronic health records
- Pay as you go car insurance
- Browsing/shopping histories
- Location-based services
- Social networking blunders
- …





Result: Compromise can hurt more than productivity

We can learn a lot from this data

Google: Advertising and search

- Why are Google's services free?
- Because they use your information to intelligently place ads!
- Portions of this data is also available to you (cf. Analytics)

Walmart: Marketing experts

- Over 580 TB in 2006, hosted on 1000 processor system
- Data used to predict/control inventory, coordinate with suppliers, and adjust to local trends

Medical data and imaging

- Medical data mining
- Google flu trends (stopped due to privacy concerns)
- Drug and prosthesis design
- …

Widespread data availability is not always a good thing, though…

August 2006: AOL releases search data

- 20,000,000 search keywords
- Over 650,000 users
- 3 months worth of records

Intended use: Learning about search patterns

Result: Records for individual users were recovered!

October 2006: Netflix releases movie rating data

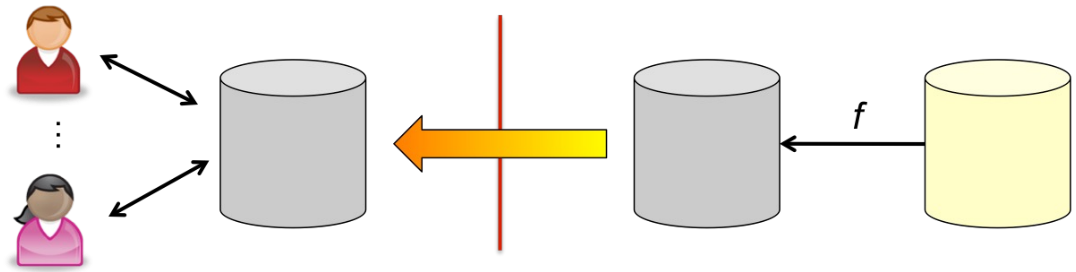
- 100,480,507 ratings that 480,189 users gave to 17,770 movies
- 〈User, Movie, Date, Rating〉 tuples

Intended use: Developing and testing new collaborative filtering algorithms

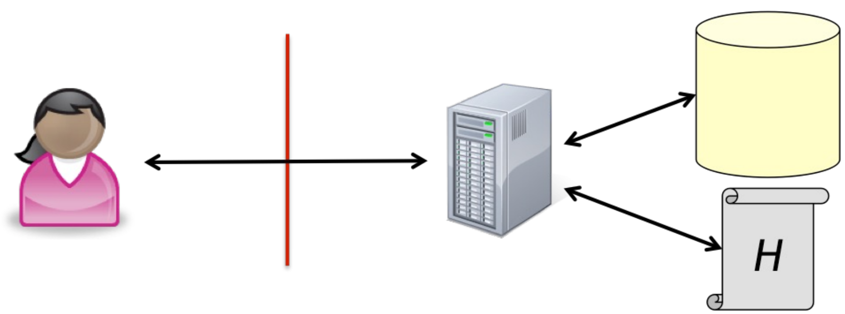
Result: Records for individual users were recovered!

There is a need to balance privacy and availability when releasing data

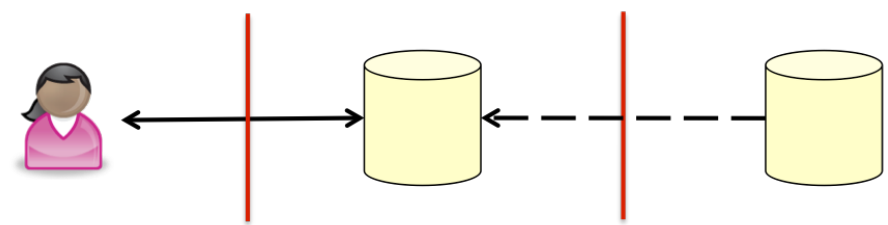
Today, we'll talk about three privacy models for data



Anonymize and Release

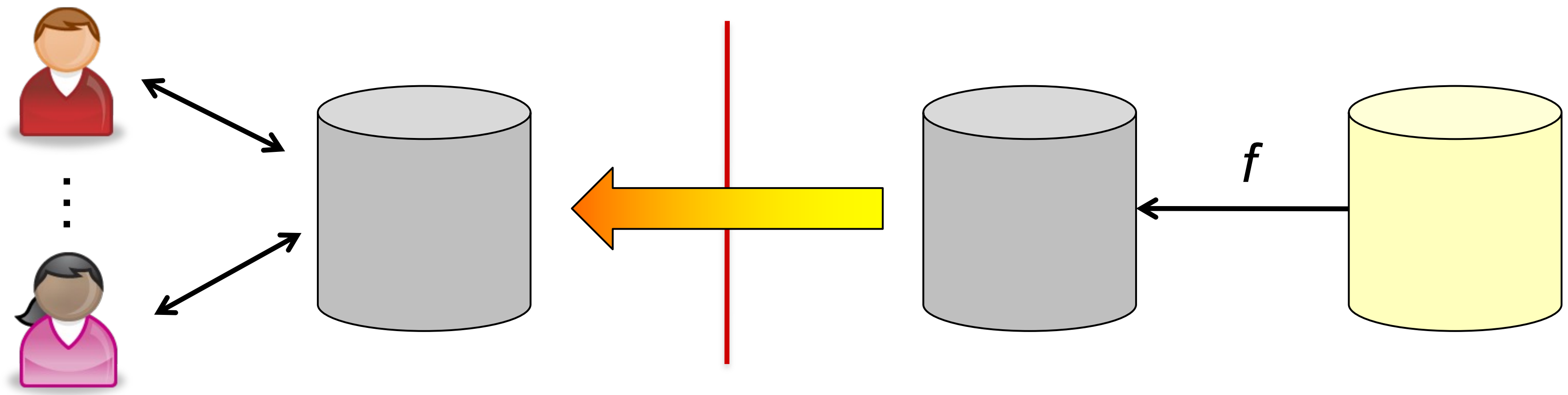


Mediated Query Processing



Outsourced Data Hosting

Anonymize and Release



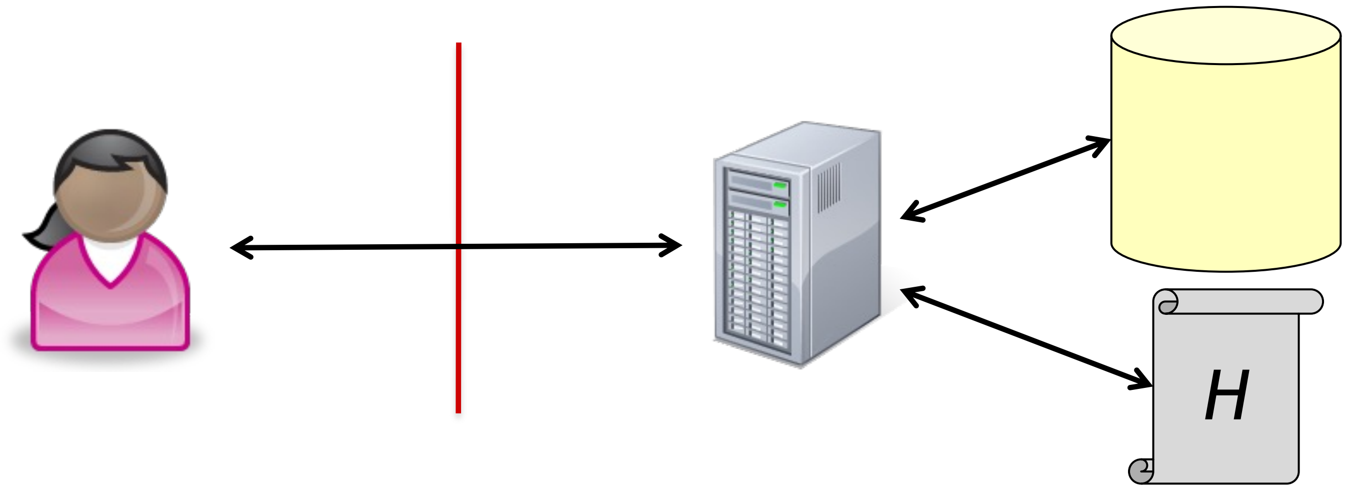
Rather than releasing the original dataset, data providers release a modified version of the dataset to the public/analysts

Data analysts often prefer this model of data release (Why?)

Common operations performed on data include:

- Stripping out names and other identifiers (Suppression)
- Grouping data values into less precise buckets (Generalization)
- Adding noise to records or groups of records (Perturbation)

Mediated Query Processing



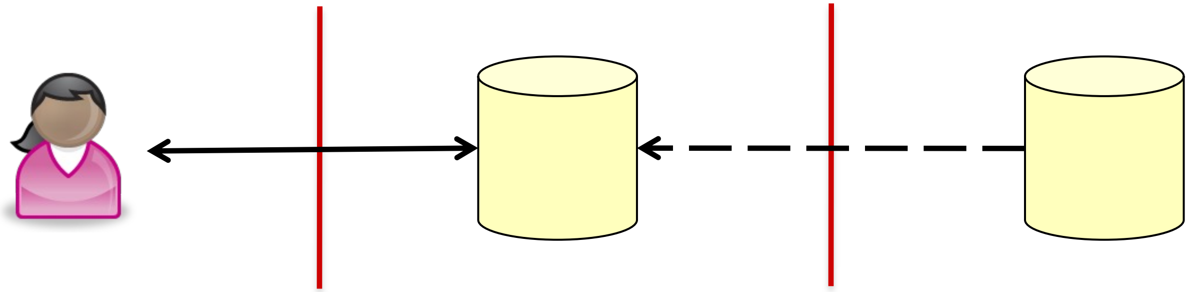
Critical point: Data is not released by the data owner

Since data is retained by the owner, they also retain control

- Do I think that this query is safe to answer?
- What other questions has this querier asked? Should this affect my answer?
- What type of perturbation is needed to make answering this question safe?

This data model is used by the US Census Bureau

Outsourced Data Hosting



Scenario: Let's pay someone else to host our data

- Became popular with the increased prevalence of the web
- Increasingly interesting as cloud computing becomes a reality

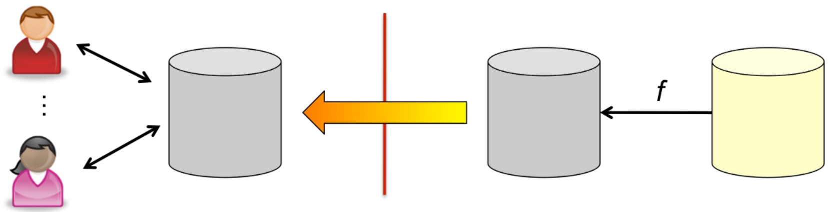
Potential uses include offsite backups and outsourced DB management

Depending on the reasons behind outsourcing, a variety of questions deserve some attention:

- Should the data host be able to read the data?
- Should the data host be able to learn about the organization of the data?
- Should queries be revealed to the data host?
- Is the data that is claimed to be hosted actually available?

This is an active area of academic research

Each of these data models has various pros and cons

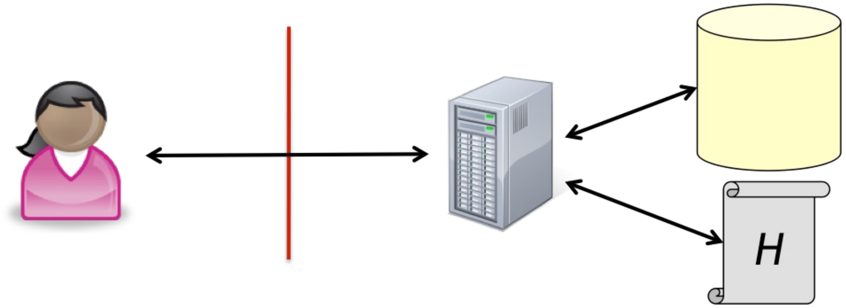


Strengths

- Analysts get (nearly) complete access to data
  - Can explore data in novel/unpredicted ways
  - Can ask any questions they want
- Providers do not need to host data locally

Weaknesses

- When is data “safe” to release?
- Balancing privacy versus utility?
- Quantifying anonymization?



Strengths

- Analysts can ask many types of queries to the data store
- Providers can see all access to data and can adjust as needed

Weaknesses

- Potentially need to store LOTS of query history
- How should data be perturbed?

Case Study: k-Anonymity

L. Sweeney, “k-anonymity: A Model for Protecting Privacy,” International Journal on Uncertainty, Fuzziness and Knowledge-based Systems, 10 (5), 2002; 557-570.

The state of the art for protecting privacy in the early 1990s was simply removing "identifiers"

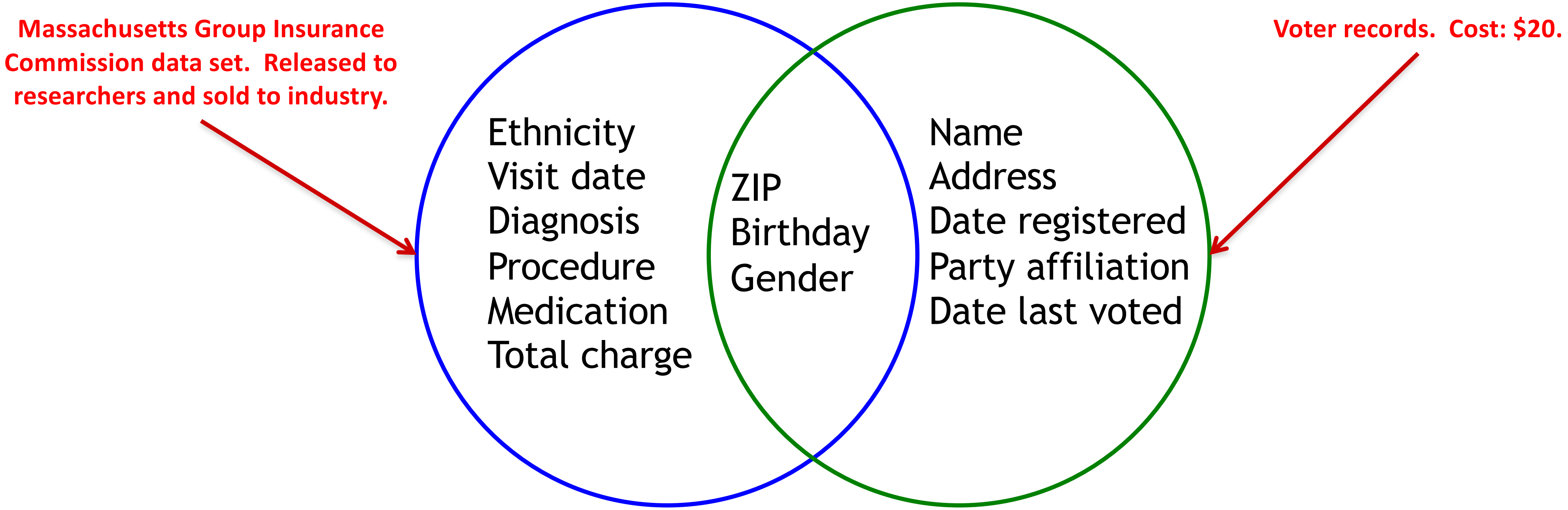
Name	*Race*	*Birth*	*Gender*	*ZIP*	*Problem*
Aaron	Black	1965	M	02145	Short breath
Bob	Black	1965	M	02143	Chest pain
Christina	Black	1965	F	02133	Hypertension
Danielle	Black	1965	F	02137	Hypertension
Eve	Black	1964	F	02137	Obesity
Francine	Black	1964	F	02134	Chest pain
George	White	1964	M	02138	Chest pain
Harry	White	1964	M	02138	Obesity
Ian	White	1964	M	02134	Short breath
James	White	1967	M	02133	Chest pain
Kevin	White	1967	M	02133	Chest pain

Question: Who can see a problem with this?

Answer: Your name is not your only unique identifier!

One example: The triple (City, Birthday, Gender) is a unique identifier for 53% of the population, while (County, Birthday, Gender) identifies 18%

Interesting attack: Reidentifying medical records



After joining these two datasets, Sweeny was able to recover the medical records of William Weld, the (then) governor of Massachusetts!

Some terminology…

Quasi-identifiers

Explicit identifier

Sensitive attribute(s)

Name	*Race*	*Birth*	*Gender*	*ZIP*	*Problem*
Aaron	Black	1965	M	02145	Short breath
Bob	Black	1965	M	02143	Chest pain

…

Steps for “anonymize and release” data processing:

1. Remove all explicit identifiers
2. Manipulate rows to ensure that quasi-identifiers cannot be used to map specific individuals to sensitive attributes

Seems easy, right?

k-Anonymity was one of the first rigorously studied anonymization methods

High-level goal: Each unique quasi-identifier should appear at least *k* times in the released data set

This provides a sort of plausible deniability…

How can we accomplish this goal?

- Attribute generalization
- Attribute suppression
- Attribute perturbation

Race	*Birth*	*Gender*	*ZIP*	*Problem*
Black	1965	M	02145	Short breath

Black	1967	*	021**	Short breath

Let's see an example…

This is a 2-anonymous version of our hospital data table example

Race	*Birth*	*Gender*	*ZIP*	*Problem*
Black	1965	M	0214*	Short breath
Black	1965	M	0214*	Chest pain
Black	1965	F	0213*	Hypertension
Black	1965	F	0213*	Hypertension
Black	1964	F	0213*	Obesity
Black	1964	F	0213*	Chest pain
White	1964	M	0213*	Chest pain
White	1964	M	0213*	Obesity
White	1964	M	0213*	Short breath
White	1967	M	0213*	Chest pain
White	1967	M	0213*	Chest pain

Question: Why is this table 2-anonymous?

- Each quasi-identifier appears (at least) 2 times

Question: Is the following table 3-anonymous?

Race	*Birth*	*Gender*	*ZIP*	*Problem*
Black	1965	M	0214*	Short breath
Black	1965	M	0214*	Chest pain
Black	1965	M	0214*	Hypertension
Black	1965	F	0213*	Hypertension
Black	1964	F	0213*	Obesity
Black	1964	F	0213*	Chest pain
White	1964	M	0213*	Chest pain
White	1964	M	0213*	Obesity
White	1964	M	0213*	Short breath

k-Anonymity sounds great! So the data anonymization problem is solved, right?

Problem 1: Solving the anonymization quality/efficiency trade-off

This table is 5-anonymous, but useless!

Race	*Birth*	*Gender*	*ZIP*	*Problem*
*	19**	*	*	Short breath
*	19**	*	*	Chest pain
*	19**	*	*	Hypertension
*	19**	*	*	Hypertension
*	19**	*	*	Obesity

Question: How can we define the "goodness" of a dataset?

- Less suppression/generalization/perturbation → better quality

Fact: Finding an optimal k-anonymization is an NP-Hard problem

Fortunately, heuristic methods do a pretty good job of this with fairly low overheads (see work by LeFevre et al.)

Efficiency is solved, but what other problems are there?

Problem 2: How do we choose the value of k to use?

Essentially, there is no good answer to this question…

- How much better is 3-anonymity than 2-anonymity?
- Is the same value of k reasonable for all individuals in the dataset?
- How much does adjusting k impact the quality of the released data?

More problems still…

Scenario: Bob has a record in the dataset, was born in the 1960s, and lives in the 15260 ZIP code

Race	*Birth*	*Gender*	*ZIP*	*Problem*
Black	196*	M	15260	Brain cancer
Black	196*	M	15260	Brain cancer
Black	196*	M	15260	Brain cancer
Black	196*	M	15260	Brain cancer

This is a 4-anonymous table, but Bob has brain cancer…

Race	*Birth*	*Gender*	*ZIP*	*Problem*
Black	196*	M	15260	Brain cancer
Black	196*	M	15260	Lung cancer
Black	196*	M	15260	Leukemia
Black	196*	M	15260	Bone cancer

This is a 4-anonymous table, but Bob has cancer…

A generalization of this problem…

These problems emerge because the data set was not diverse

- All entries for a quasi-identifier map to the same sensitive attribute
- All entries for a quasi-identifier map to related sensitive attributes

Follow-on work addresses this but is subject to attacks of its own!

The bigger problem is that this class of solutions does not adequately model the knowledge of the attacker

- I know that Bob visited the hospital and should be in this data set
- I know that Bob has some type of cancer
- …

Recent work on differential privacy works for any attacker, but uses the mediated query model

In short, this is still a very active research area

Data Privacy Summary

So we talked about three types of models for managing private data

- Anonymize and release
- Mediated query processing
- Outsourced data hosting

k-Anonymity is one solution in the “anonymize and release” model

- Strip out explicit identifiers
- Be sure that each quasi-identifier appears at least k times

How do we manage diversity and model attacker knowledge?

- Recent work does this with limited success

Take away point: Data anonymization is hard. “Anonymization” is probably a flawed term, as it is hard to quantify…

Next: Blockchains

Blockchains

What *is* a blockchain?

Building a blockchain

- Important properties
- Crypto crash course
- Basic construction

Cryptocurrency: Bitcoin as an example

Other application areas

Blockchains

What *is* a blockchain?

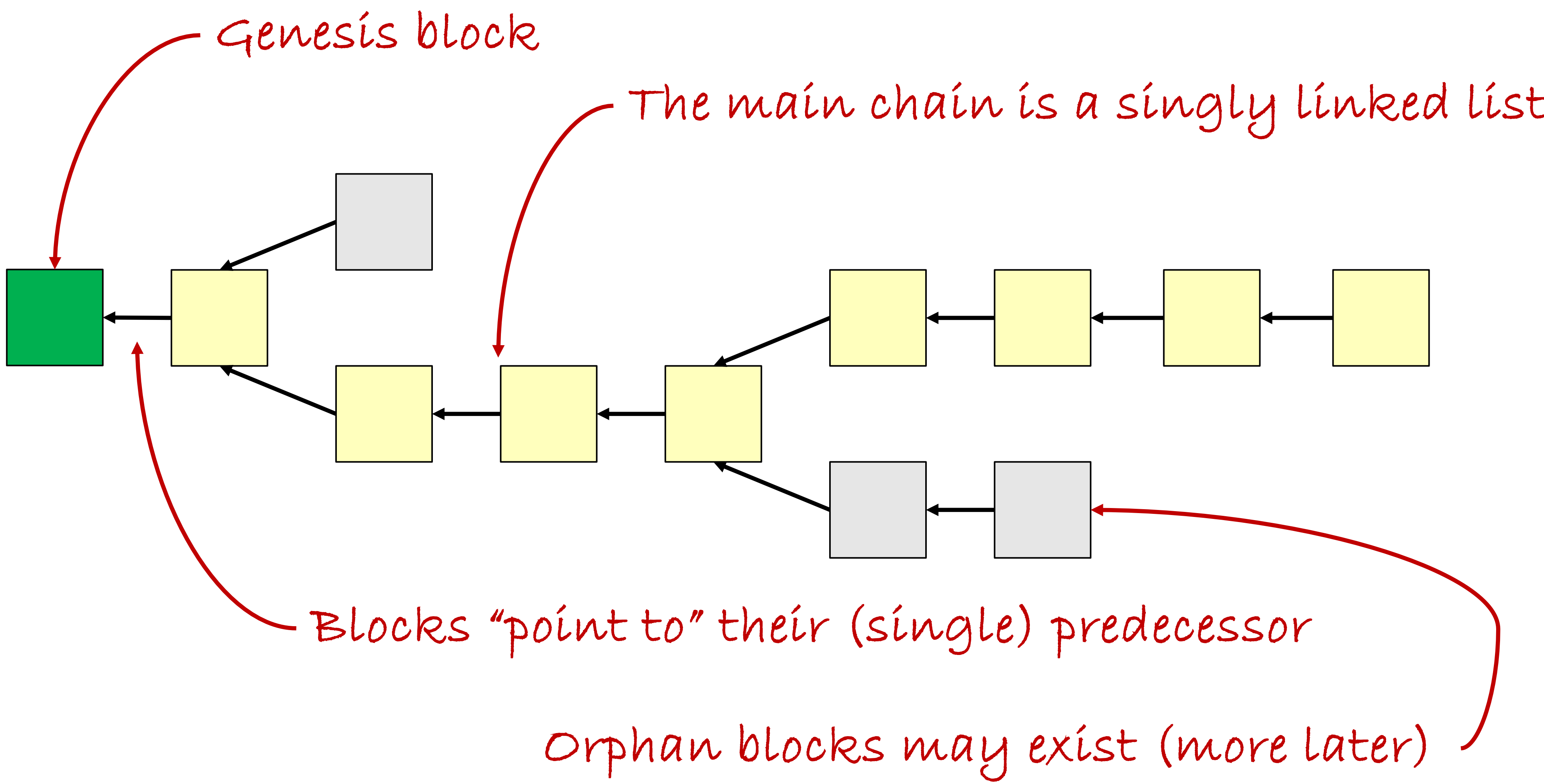
Building a blockchain

- Important properties
- Crypto crash course
- Basic construction

Cryptocurrency: Bitcoin as an example

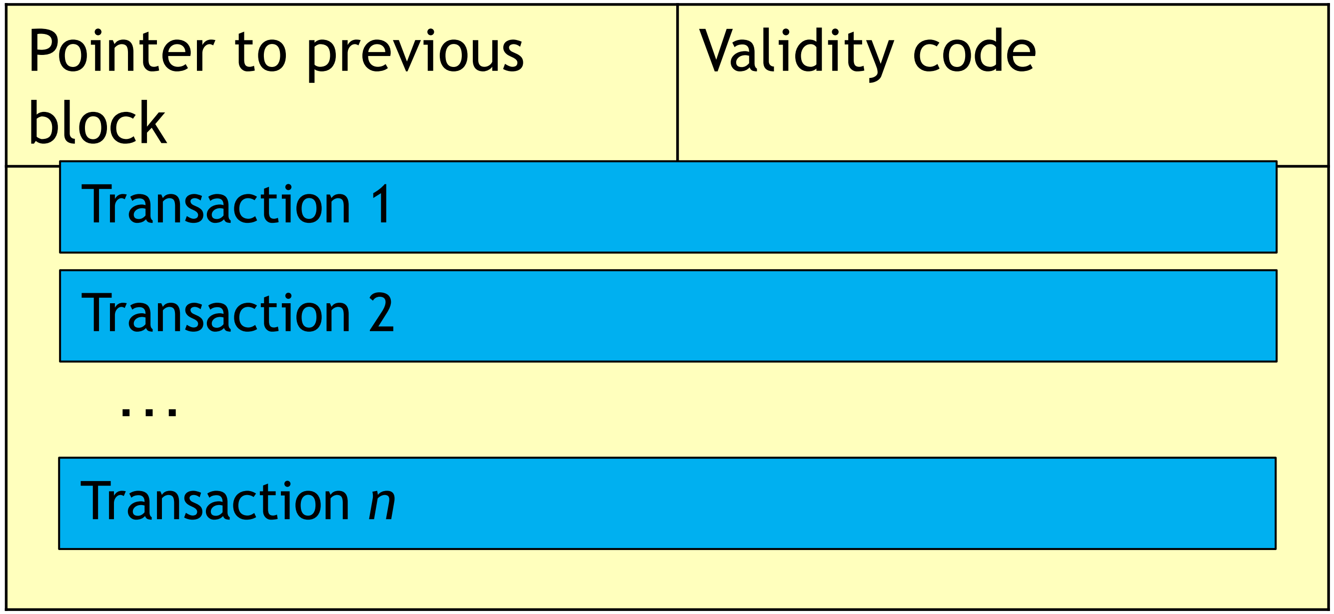
Other application areas

A blockchain is just a distributed data structure



What's in a block? How is the blockchain extended?

Each block can be thought of as a page in a (decentralized) transaction registry



What's a transaction? Depends upon the application…

- A transfer of currency (e.g., Bitcoin)
- Events in distributed monitoring application
- Claims filed against an insurance policy
- …

Yes, I'm waving my hands here…

The validity code is used for obtaining consensus and making the block (and blockchain) tamper evident

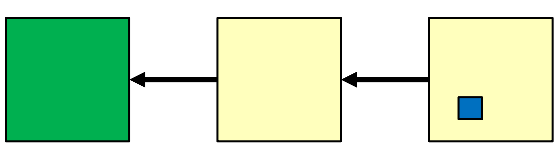
How is the blockchain extended?

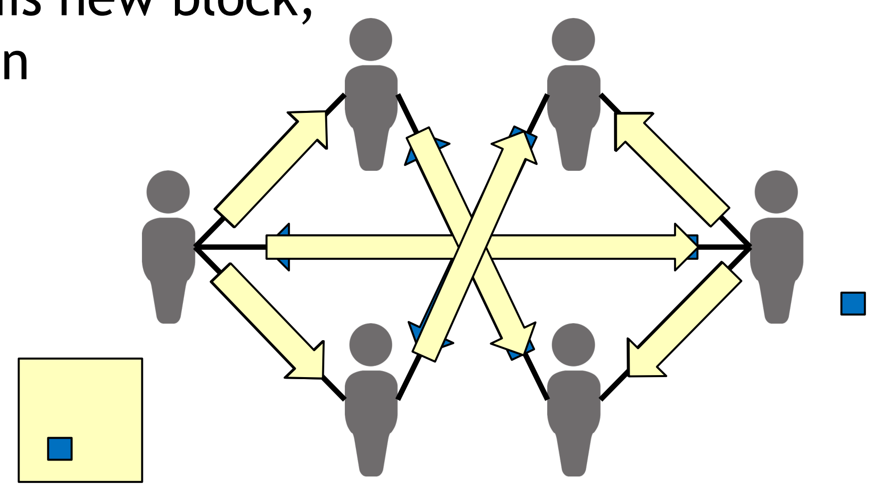
Blockchain protocols are based on decentralized (P2P) networks

Extending the blockchain

“Mining”

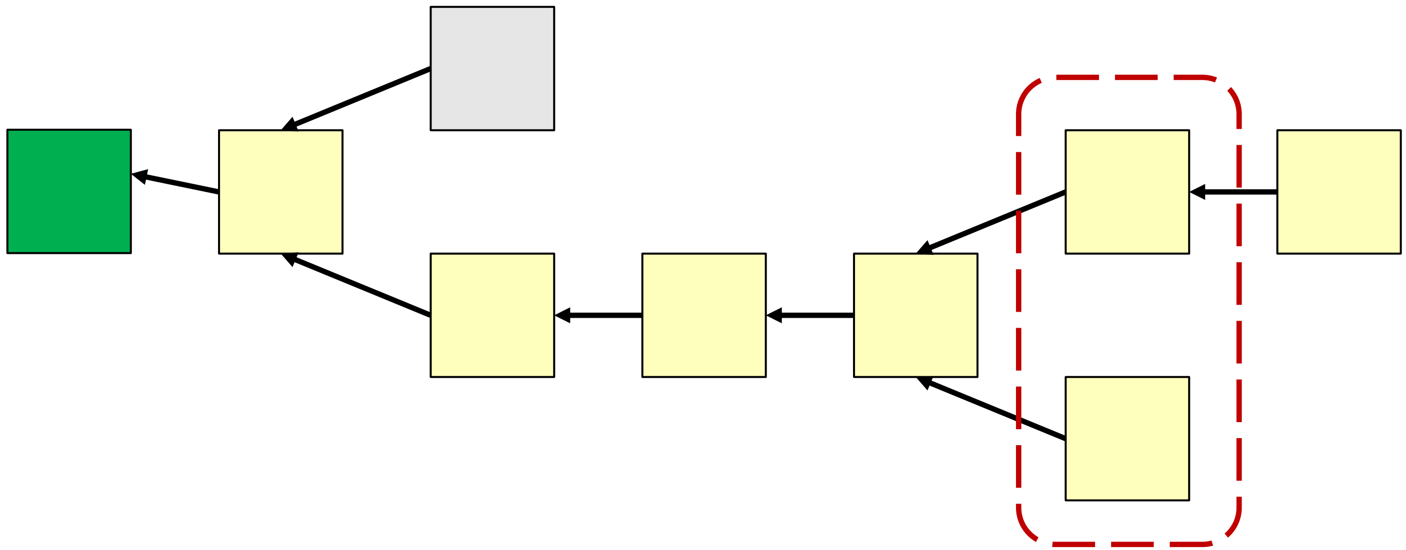
1. Individual transactions are flooded through the network
2. If valid*, transactions are added to in-progress blocks
3. Participants solve a puzzle to compute the block validity code
4. New blocks are flooded through the network
5. If valid*, the peers build upon this new block, thereby extending the blockchain





Remember orphan blocks?

Nodes build upon the longest chain that they know



What happens if blocks are created concurrently?

- Build upon the first received

The result? Blocks can become orphaned.

- What happens to transactions in orphaned blocks? Flood again!
- When can a transaction be considered committed?

Further considerations

If properly constructed, a blockchain can provide a highly-available, publicly-verifiable, immutable, consensus-based distributed transaction register

Technical and application-specific questions:

- How do we manage identities in a decentralized system?
- How can we efficiently verify transaction validity? Block validity?
- How do we ensure that the blockchain remains immutable?
- How can we balance throughput and consistency?
- How do we incentivize participation and mining?

Let's dig deeper and find out…

Blockchains

What *is* a blockchain?

Building a blockchain

- Important properties
- Crypto crash course
- Basic construction

Cryptocurrency: Bitcoin as an example

Other application areas

What properties should a minimally-interesting blockchain offer us?

A minimal set of important properties includes:

- Availability of the blockchain
- Append-only and immutable transaction log
- Tamper-proof and authenticated transaction records
- Verifiability of transactions and structural invariants
- Consensus on the transactions to be included in the blockchain

Let's build a blockchain structure that provides these properties

Note: Assume an open system in which anyone can participate

Proof-of-work systems are computational puzzles that can be used to rate limit behavior

A computational puzzle is a problem that is *computationally-difficult* to solve, but *easy* to verify

We can build a simple computational puzzle with tunable hardness using hash functions:

- Given: a hash function H(.), a message *m*, and a hardness *h*
- Compute: a value *v*, such that H(*m* || *v*) ends with *h* 0-bits

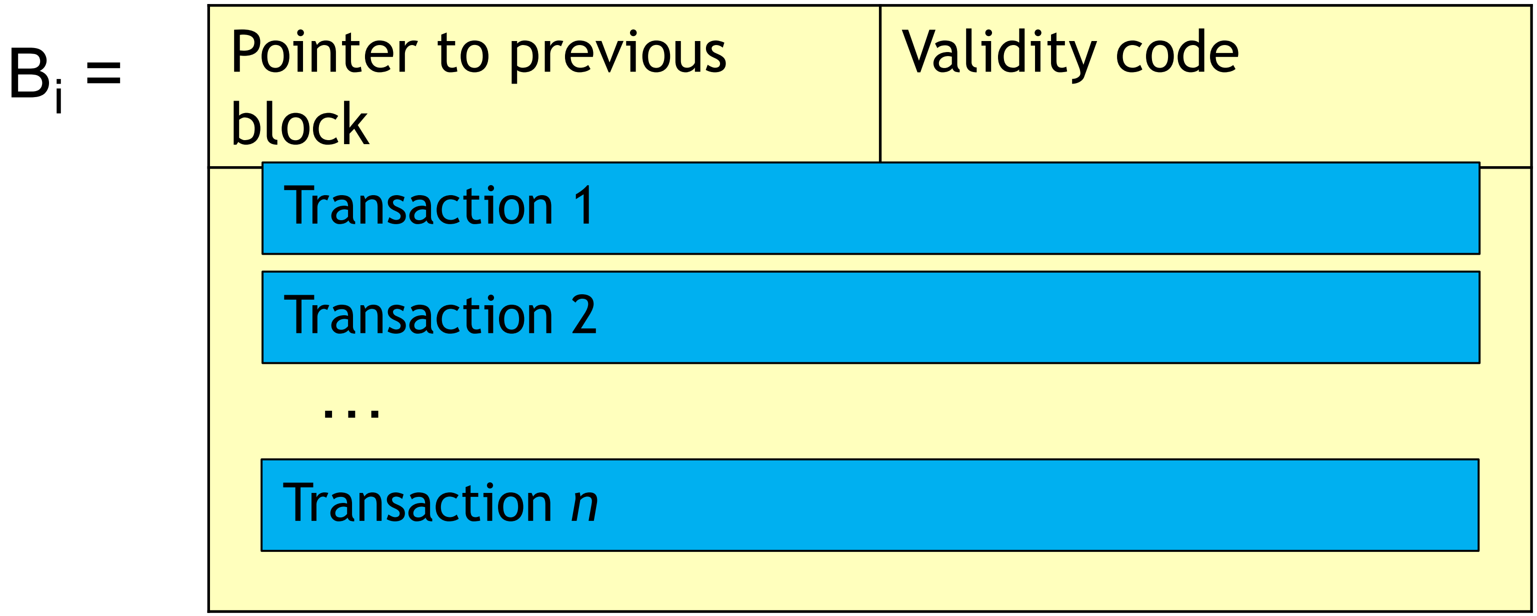
Preimage resistance

Why is this hard to solve?

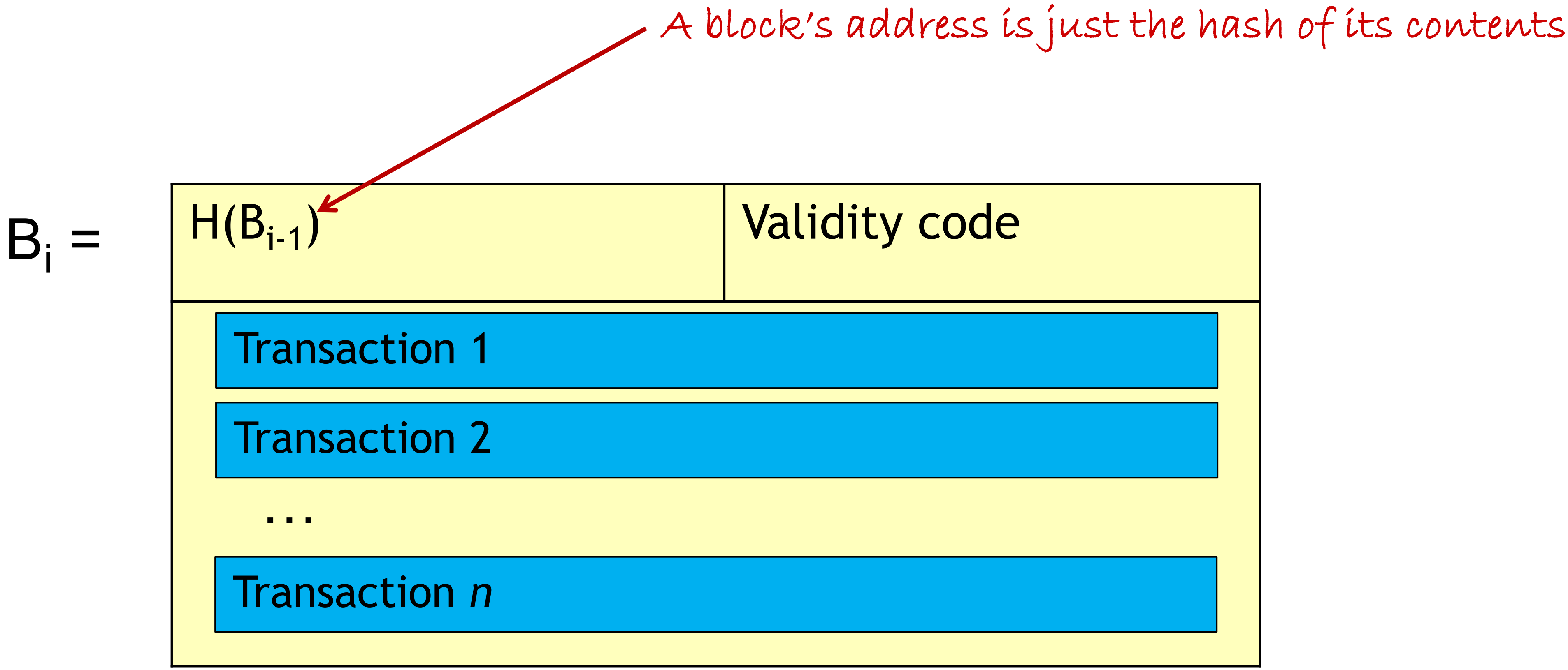
- It is computationally-difficult to invert a hash function
- Must brute-force search for a *v* that satisfies the puzzle constraints
- I.e., $O(2^h)$ tries to solve the puzzle

Let’s detail the structure of a basic blockchain

We won't change our basic block structure, we'll just fill in the details…



We won't change our basic block structure, we'll just fill in the details…



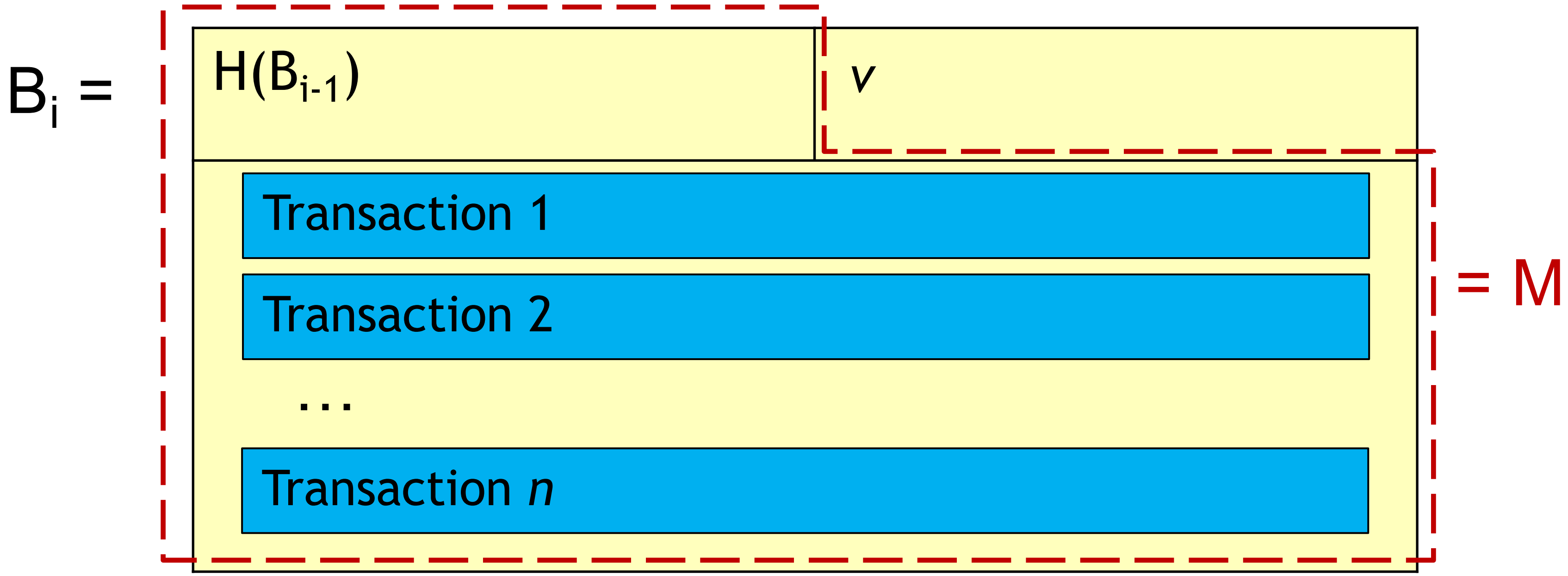
$H(B_{i-1})$ is a unique value identifying B_{i-1}

- Why? Collision resistance and 2nd preimage resistance of H(.)

Side-effect: Changing block *contents* changes the block *address*

- This will be useful to us later…

We won't change our basic block structure, we'll just fill in the details…



We'll use a hash puzzle to compute the validity code

- i.e., find v s.t. $H(M \| v)$ ends with h 0-bits

How do we represent individual transactions?

Main idea: *Public* keys represent user identities

B_i =

$H(B_{i-1})$	v

$< k_A,\ tx_1,\ [tx_1]k_A^{-1} >$

$< k_B,\ tx_2,\ [tx_2]k_B^{-1} >$

…

$< k_C,\ tx_n,\ [tx_n]k_C^{-1} >$

Each transaction record consists of three parts:

Who registered this transaction?

- A public key (e.g., k_A)
- The transaction contents (e.g., tx_1) Application specific…
- A digital signature over tx (i.e., $[tx_1]k_A^{-1}$) Ensures accountability

That's it! What about our properties?

What properties should a minimally-interesting blockchain offer us?

A minimal set of important properties includes:

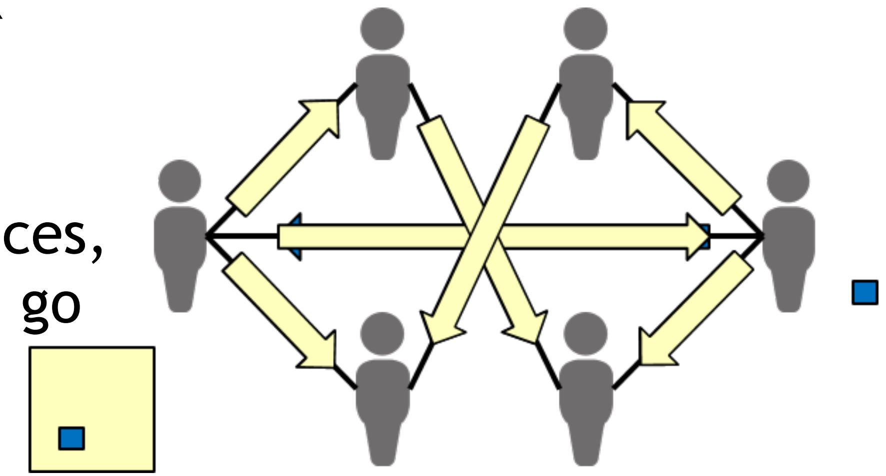
- ✓ Availability of the blockchain
- Verifiability of transactions and structural invariants
- Tamper-proof and authenticated transaction records
- Append-only and immutable transaction log
- Consensus on the transactions to be included in the blockchain

---

Gossip -> Availability

- Each transaction floods the network
- Each block floods the network

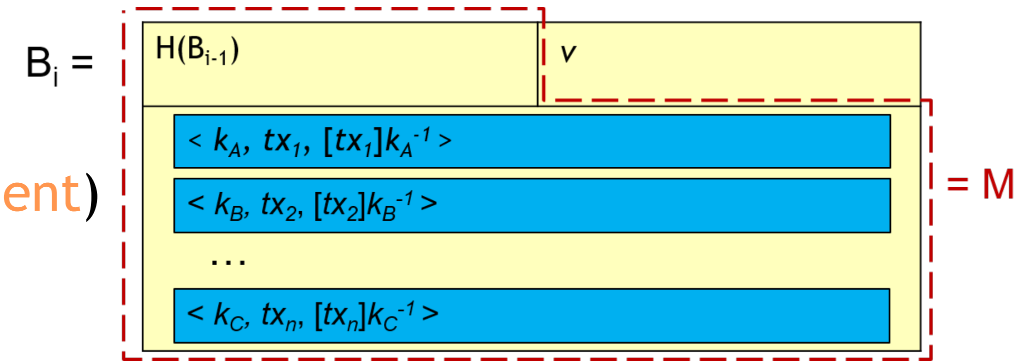
Copies of txns/blocks exist in many places, remaining available as nodes come and go



Verifiability is more nuanced

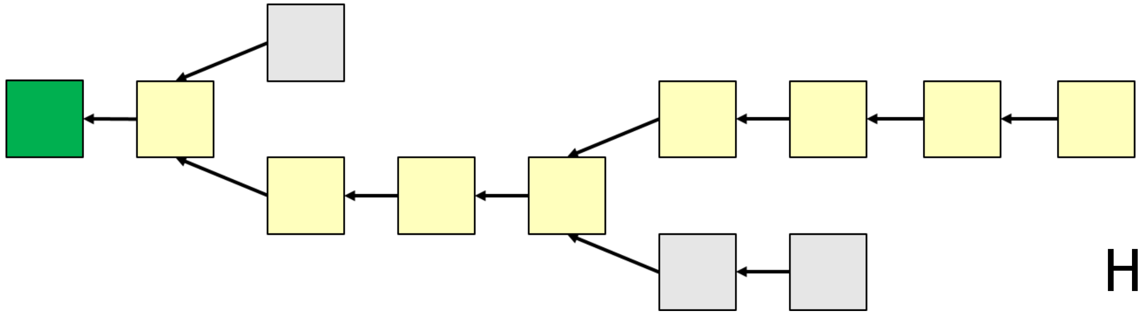
Two steps to validate each transaction

- Validate the txn signature (easy)
- Validate the txn (application dependent)



To validate the block

- Validate each transaction (easy)
- Check the validity code (easy: compute H(M || v) and check structure)



Validating structural invariants

- Is this new block valid? (easy)
- Does it build upon the most recent block I know of? (easy)

Handling forks isn't hard, either

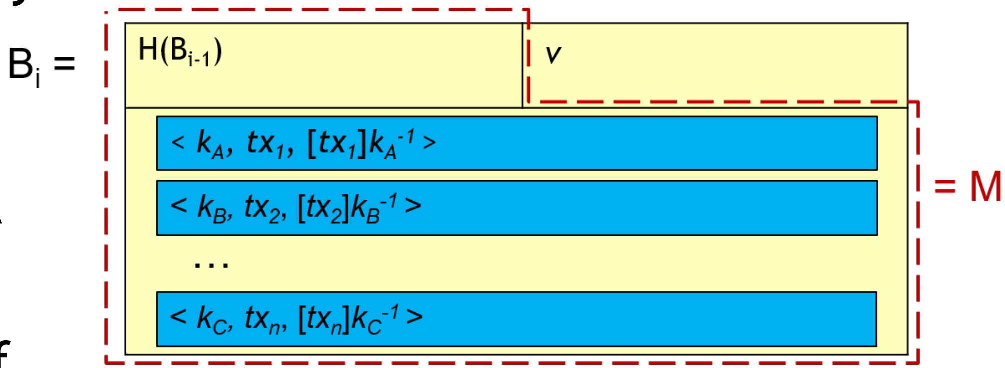
Digital signatures ensure that transactions are authenticated and tamper-proof

A minimal set of important properties includes:

- ✓ Availability of the blockchain
- ✓ Verifiability of transactions and structural invariants
- ✓ Tamper-proof and authenticated transaction records
- Append-only and immutable transaction log
- Consensus on the transactions to be included in the blockchain

Authentication via non-repudiability

- k_A identifies user A
- Only k_A^{-1} can produce $[tx_1]k_A^{-1}$
- Result: tx_1 was registered by A



Transactions are also tamper-proof

- Changing a transaction invalidates its signature
- "Updating" a record will invalidate the proof-of-work v
  - Why? 2nd preimage resistance of H(.)

Modifying a block creates a fork in the blockchain

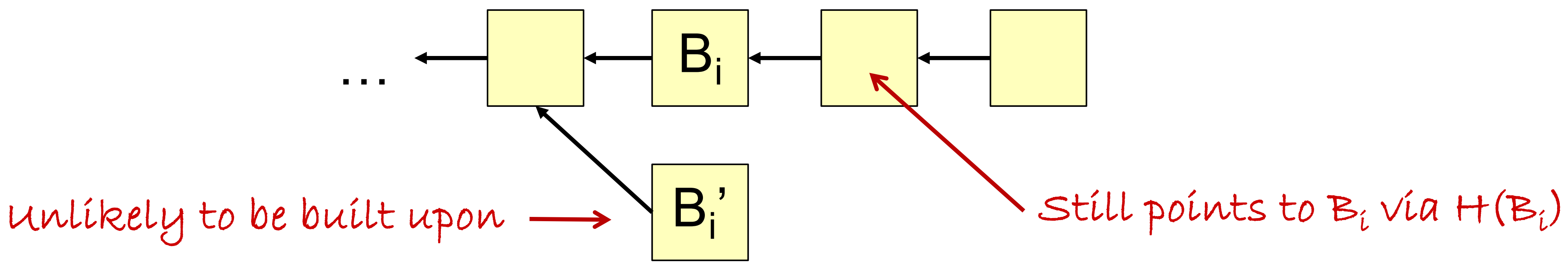
A minimal set of important properties includes:

- ✓ Availability of the blockchain
- ✓ Verifiability of transactions and structural invariants
- ✓ Tamper-proof and authenticated transaction records
- ✓ Append-only and immutable transaction log
- Consensus on the transactions to be included in the blockchain

---

Recall: *Deleting* a block is a non-starter

Modifying a block creates a fork in the chain since $H(B_i)$ =\= $H(B_i')$



The blockchain is advanced through distributed work and community validation

A minimal set of important properties includes:

- ✓ Availability of the blockchain
- ✓ Verifiability of transactions and structural invariants
- ✓ Tamper-proof and authenticated transaction records
- ✓ Append-only and immutable transaction log
- ✓ Consensus on the transactions to be included in the blockchain

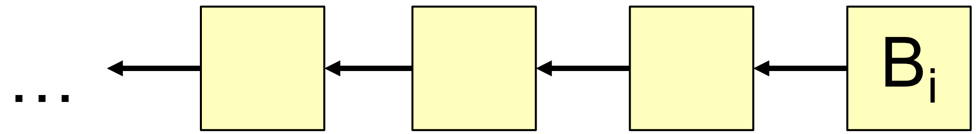
---

No single node is in charge of advancing the blockchain

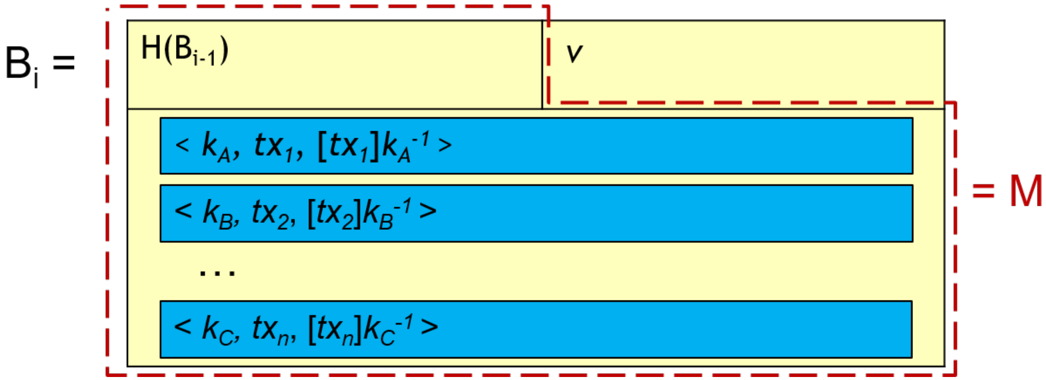
- Must solve proof of work to finalize new block, B_i
- This is a random process governed by compute power

A majority of nodes must accept each new block

- Think of building off of a new block as "voting" for it



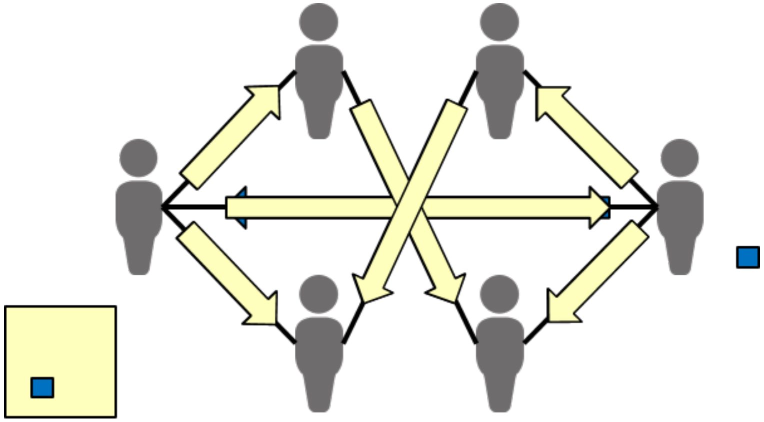
This technical foundation is helpful, but there are still application-specific considerations



How do we incentivize participation (e.g., mining?)

How should transactions be structured to make them efficiently verifiable?

How can we balance the efficiency of arriving at consensus with the scalability and/or openness of the network?



Blockchains

What *is* a blockchain?

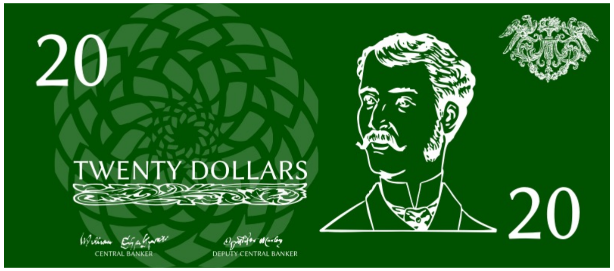
Building a blockchain

- Important properties
- Crypto crash course
- Basic construction

Cryptocurrency: Bitcoin as an example

Other application areas

The goals of any currency



- **Counterfeit detection:** How do I know this bill is authentic?

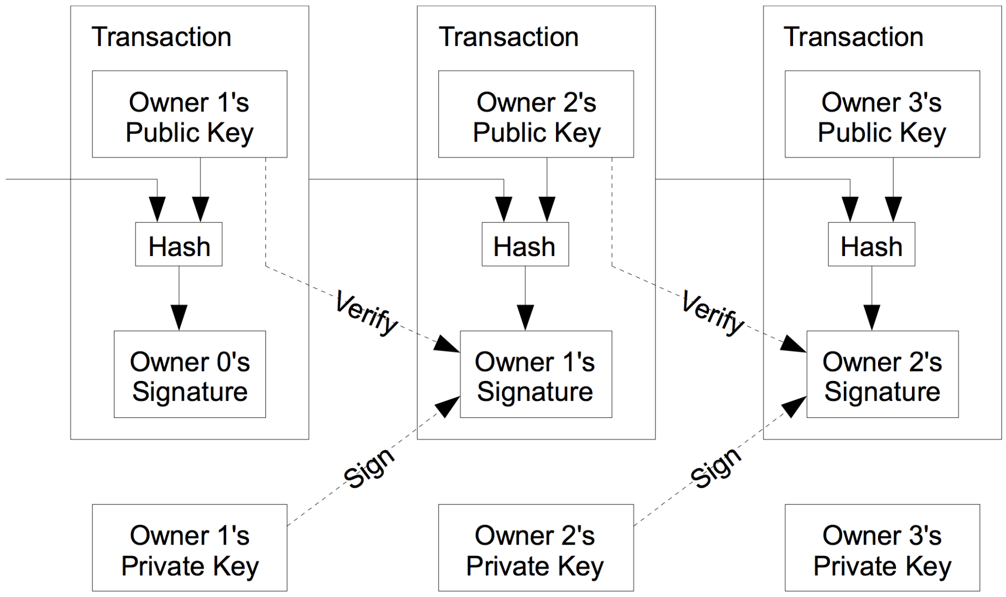
- **Double-spending protection:** How do I know that nobody can claim this money belongs to them and not to me?

- **Question:** Which is easier to accomplish with physical cash?
  - Counterfeits can be very sophisticated; evasion is a cat-and-mouse game
  - Anti-counterfeiting technologies aim to be expensive to duplicate
  - Double-spending is easy: whoever holds the bill is the owner!

What could we do in a distributed, digital setting to achieve these goals?

- In a distributed system, agreeing on whether a coin is valid is a form of the Byzantine Generals' problem
  - Need to reach agreement without central authority

- Spending can be tracked by using public-key crypto and keeping a public, distributed, signed log of transactions relating to a coin



- How do we prevent double-spending?

How can we reach a consensus about which transaction occurred first?

- Like in physical counterfeiting, we want it to be overwhelmingly expensive to cheat

- What about voting?
  - If a second transaction shows up before agreement, cheating is detected
  - How do we prevent duplicate votes?

- Bitcoin uses a proof-of-work system to rate-limit voting

- The version of history written by the group with the most computing power becomes the truth
  - As long as *most** users are honest, the log is correct!

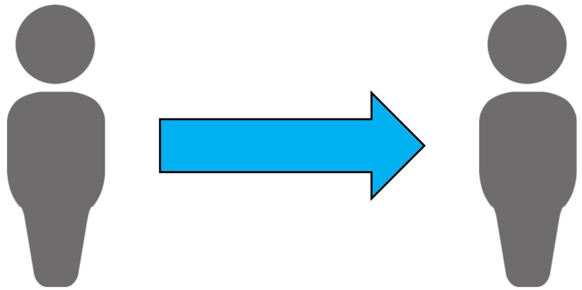
Bitcoin is a cryptocurrency based upon a blockchain like the one that we’ve been exploring today

Technical parameters:

- Digital signatures using ECDSA on the secp256k1 curve
  - ~128 bits of security
- Hashes computed using SHA-256
- Proof-of-work slightly different than the one we used
  - Can adjust with higher precision than 1 bit
  - Goal: ~10 minutes to produce a block

Two classes of transactions:

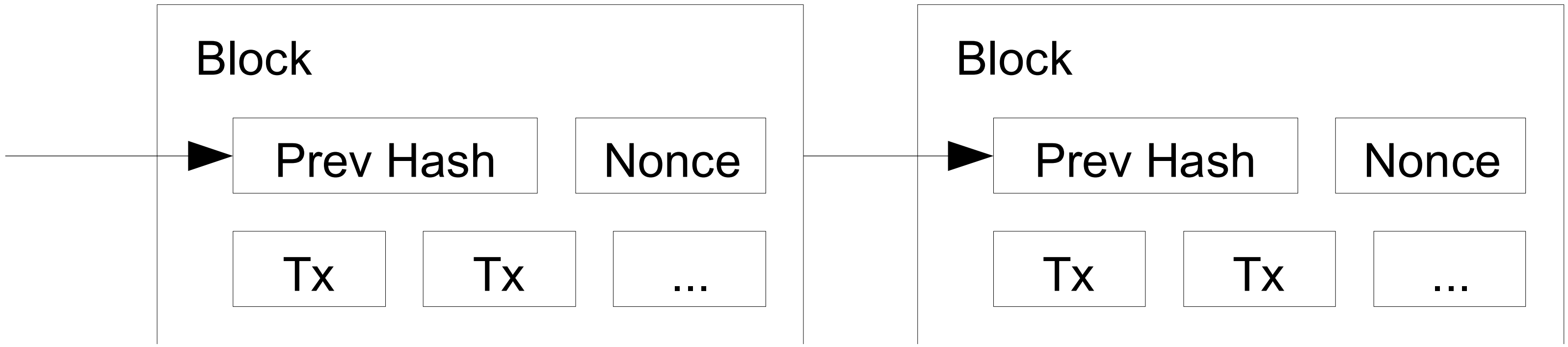
Coinbase transactions



*Transfers**

Bitcoin's Proof-of-work System

- Hashcash: Originally developed for email (why?)
- To send message M, Alice must compute nonce n such that $H(M||n)$ starts with b '0' bits
  - How long will this take if H is a cryptographic hash function?
- Bitcoin uses a block chain of transaction blocks, where each one requires a Hashcash nonce: longest chain wins!



One vote per CPU/FLOP

So how do I know I've received money for a product or service?

- Just because a transaction is in *some* block, doesn't mean that this block will be chosen by the majority
  - Blocks that aren't incorporated into the "canonical" chain are called orphaned blocks

- In order to accept Bitcoin, a vendor must wait until a transaction has sufficient blocks built on top of it
  - Otherwise, the spender may be able to double-spend

- Unfortunately, this can take a long time
  - Blocks created every ~10 minutes (tuned with b automatically)
  - Block restricted to 1 MB

- Bitcoin XT is a fork of the Bitcoin software
  - Allows clients to vote on whether to increase the block size
  - Once 75% of clients vote yes, and after a 4-week waiting period, 2 MB blocks will be accepted
  - This would create a fork in the block chain

The steps so far…

1. Broadcast new transactions to all nodes
2. Each node collects new transactions into a block
3. Each node searches for nonce to complete block
4. First successful node broadcasts complete block
5. All nodes verify all transactions in the new block
6. Nodes attest to a block's correctness by building upon it for the next block

How does bitcoin incentivize mining?

Incentive 1: Mining rewards

- The first transaction in a block is a coinbase transaction
- Initially, 50 BTC per block, halves every 210,000 blocks
  - Current reward: 6.25 BTC
  - Maximum currency volume: 21,000,000 BTC (currently ~19.5M)

Incentive 2: Transaction fees

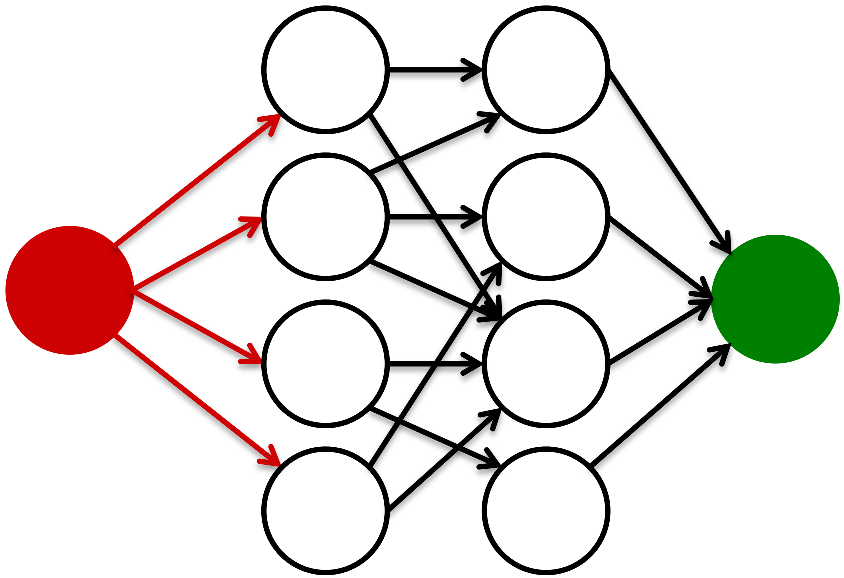
- Small fees attached to each transaction
- Fees claimable by the node that creates a block

Mining pools: groups of nodes to work together, split rewards

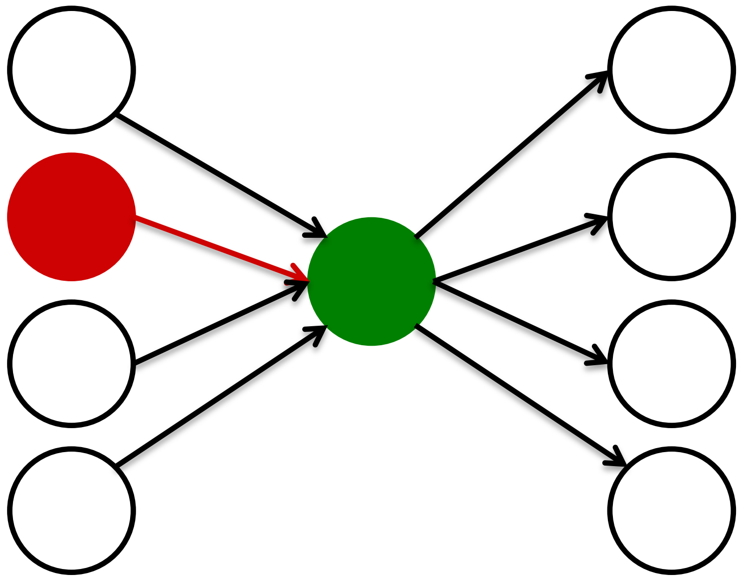
Is Bitcoin really as private as cash?

- Since all transactions are public, looking at the history can reveal trends

- Idea: Transfer coins between multiple addresses



Fork-merge



Mixing

BlockChain Concluding thoughts

At its core, a blockchain is simply a data structure

Blockchains have a number of useful features

- Distributed and highly-available transaction ledger
- Append-only and immutable record of activities
- Tamper-proof and authenticated transactions
- (Perhaps public) verifiability of transactions and structure
- Consensus required to extend the structure

Applications to many interesting classes of problems

Flexibility can be extended via the use of smart contracts

Cryptocurrency Conclusions

Bitcoin and other cryptocurrencies satisfy properties to be a usable currency

- Counterfeit detection
- Double-spending resistance

Although often thought of as “the digital equivalent of cash,” Bitcoin is not entirely private

- Public ledger, statistical analysis

Proof-of-work systems enable solution to Byzantine fault tolerance

Proposals such as Zerocash expand Bitcoin to strengthen its guarantees

Topic Area 1: The Basics

Objectives:

- Why study computer security? Is it really important?
- What are the CIA properties?
- How do these properties parameterize most computer security goals?
- Security is a relative (not absolute!) property

Topic Area 2: The Tools

Objectives:

- Understanding cryptography as a black box
  - When should I use public key versus secret key cryptography?
  - When are block ciphers good to use? Stream ciphers?
  - Which modes of operation are good for which types of task?
  - Why are hash functions the “duct tape” of security protocols?
- Understanding cryptography in depth
  - What is semantic security? Why is it useful?
  - How do ciphers like AES work? Why are they safe? Why are they fast?
  - Why is RSA secure?
  - How does the mathematics behind RSA influence attacks against implementations?

Topic Area 3: Assembling the Primitives

Objectives:

- I have a toolbox full of cryptography, how do I use it?
- Basic cryptographic protocols
  - Identifying/authenticating users
  - Mutually authentication with services
  - Safely exchanging secrets
  - Managing public keys in a realistic manner
- Subtleties galore…
  - Why doesn’t “just encrypt it” or “just HMAC it” work?
  - Protecting against replay/reflection attacks
  - Whose clock?
- Main point: Having tools is important. Using tools properly is more important.

Topic Area 4: Security Challenges in the Real World

Objectives:

- Understanding security in the abstract is good, but how do we put it into practice?
- What issues are involved with OS/application protection?
    - Integrity of mechanism
    - Storing, managing, accessing, and updating user permissions
    - Insiders vs. outsiders
- Many real-world attacks come from violated design assumptions
- How is privacy different from confidentiality? How is it the same?

So, what was the point of the term project?

Phase 1:

- Security is not an absolute property!
- How will we define “secure” for this application domain?
- What design principles do we need to respect?
- Writing: Informal writing can facilitate brainstorming

Phase 2:

- Build a non-trivial code base in a cooperative manner
  - Team work is the norm outside of classes
  - Learn about version control and useful tools
- Think about functionality, not security
- Get used to documenting the code that you produce

So, what was the point of the term project?

Phases 3 and 4:

- Adding security to a non-trivial codebase as an afterthought is a pain…
  - So think about security from the get-go!
- Given the tools that we've learned about in class, how can we apply them?
- Given good ideas, implementing them still involves lots of subtlety
- Writing:
  - Informal specifications can guide development and facilitate teamwork
  - Precision, precision, precision!

Phase 5:

- Threat models are just assumptions: what happens when they change?
- When securing a system, it is helpful to think like an attacker
- Writing: Formal writing
  - Good vulnerability documentation is a must
  - Good system designs help with formal verification and software engineering

What do I hope that you have learned?

Academic/Professional

- What does it mean for a system to be secure?
- Develop a working knowledge of cryptography
  - How to use properly, rather than how to design/prove/implement
  - I.e., build systems using tools, not craft new tools per se
- Understand the subtleties of protocol/system design
- Get comfortable assessing the security of non-trivial software systems

Everyday/Real World

- What types of security/privacy issues should you be aware of?
  - Private email
  - Exposed network traffic
  - Your data in public
- Realistically evaluating the security of services that you use
- Managing the complexities of non-trivial team work