Cristina Nita-Rotaru

CS6740: Network security

Anonymity.

Sources

1. Crowds: http://avirubin.com/crowds.pdf
2. Chaum mix: http://www.ovmj.org/GNUnet/papers/p84-chaum.pdf
3. Tor: https://svn.torproject.org/svn/projects/design-paper/tor-design.pdf
4. Predecessors attack: http://prisms.cs.umass.edu/brian/pubs/wright-tissec.pdf
5. Also based on slides prepared by Chi-Cun Chan.

1: Motivation

Anonymity

Anonymity (``without name”) means that a person is not identifiable within a set of subjects

- Unlinkability of action and identity
  - For example, sender and his email are no more related after adversary’s observations than they were before
  - Who talks to whom
- Unobservability
  - Adversary cannot tell whether someone is using a particular system and/or protocol

There is no anonymity on the Internet

- Your IP address can be linked directly to you
  - ISPs store communications records
  - Usually for several years (Data Retention Laws)
  - Law enforcement can subpoena these records
- Your browser is being tracked
  - Cookies, Flash cookies, E-Tags, HTML5 Storage
  - Browser fingerprinting
- Your activities can be used to identify you
  - Unique websites and apps that you use
  - Types of links that you click

Wiretapping is ubiquitous

- Wireless traffic can be trivially intercepted
  - Airsnort, Firesheep, etc.
  - Wifi and Cellular traffic!
  - Encryption helps, if it's strong
    - WEP and WPA are both vulnerable!
- Tier 1 ASs and IXPs are compromised
  - NSA, GCHQ, "5 Eyes"
  - ~1% of all Internet traffic
  - Focus on encrypted traffic

Who uses anonymity systems?

- "If you're not doing anything wrong, you shouldn't have anything to hide."
  - Implies that anonymous communication is for criminals
- The truth: who uses Tor?
  - Journalists
  - Law enforcement
  - Human rights activists
  - Normal people
  - **Business executives**
  - **Military/intelligence personnel**
  - **Abuse victims**
- In fact, the predecesor of Tor was developed by the U.S. Naval Research Laboratory.

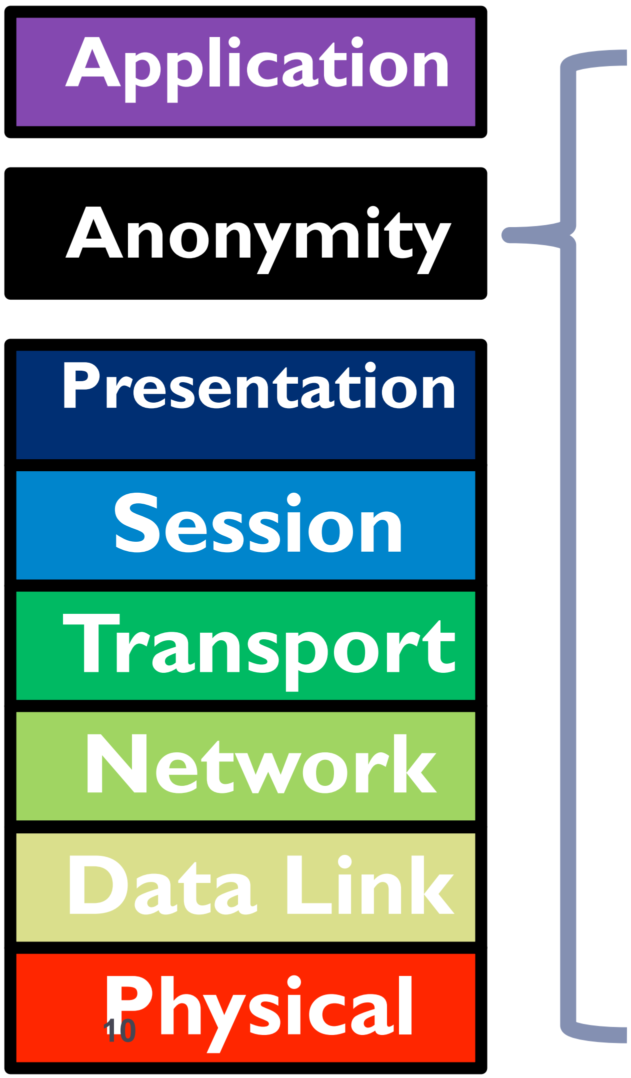
Why do we need anonymity?

- To protect privacy
  - Avoid tracking by advertising companies
  - Viewing sensitive content
    - Information on medical conditions
    - Advice on bankruptcy
- Protection from prosecution
  - Not every country guarantees free speech
  - Downloading copyrighted material
- To prevent chilling-effects
  - It's easier to voice unpopular or controversial opinions if you are anonymous

Relevant applications

- Anonymous communication
- Anonymizing bulletin board and email
- Electronic voting
- Incident reporting
- Anonymous e-commerce
- Private information retrieval

Anonymity layer

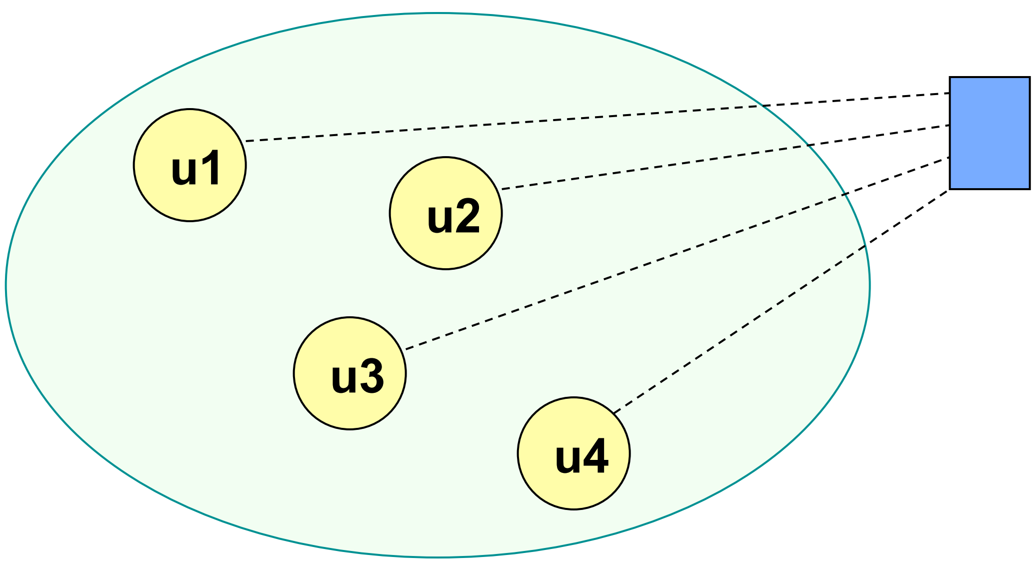


- Function:
  - Hide the source, destination, and content of Internet flows from eavesdroppers
- Key challenge:
  - Defining and quantifying anonymity
  - Building systems that are resilient to deanonymization
  - Maintaining performance

2: Terminology

Quantifying anonymity: Anonymity set

- Hiding one's action in many others' actions
- Anonymity set: a group of users in which every one is equally-probable to be associated with a given action $\Rightarrow$ every one has certain degree of innocence or deniability to an action



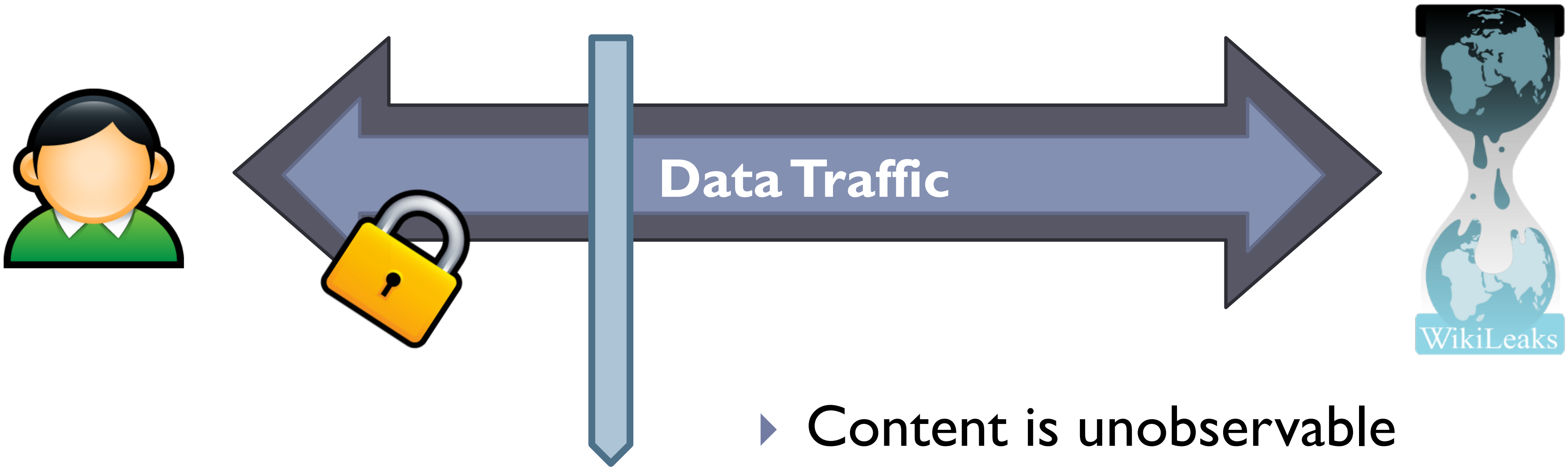
More definitions

- Unlinkability
  - From the adversaries perspective, the inability the link two or more items of interest; E.g. packets, events, people, actions, etc.
  - Three parts:
    - Sender anonymity (who sent this?)
    - Receiver anonymity (who is the destination?)
    - Relationship anonymity (are sender A and receiver B linked?)
- Unobservability
  - From the adversaries perspective, items of interest are indistinguishable from all other items

Types of adversary

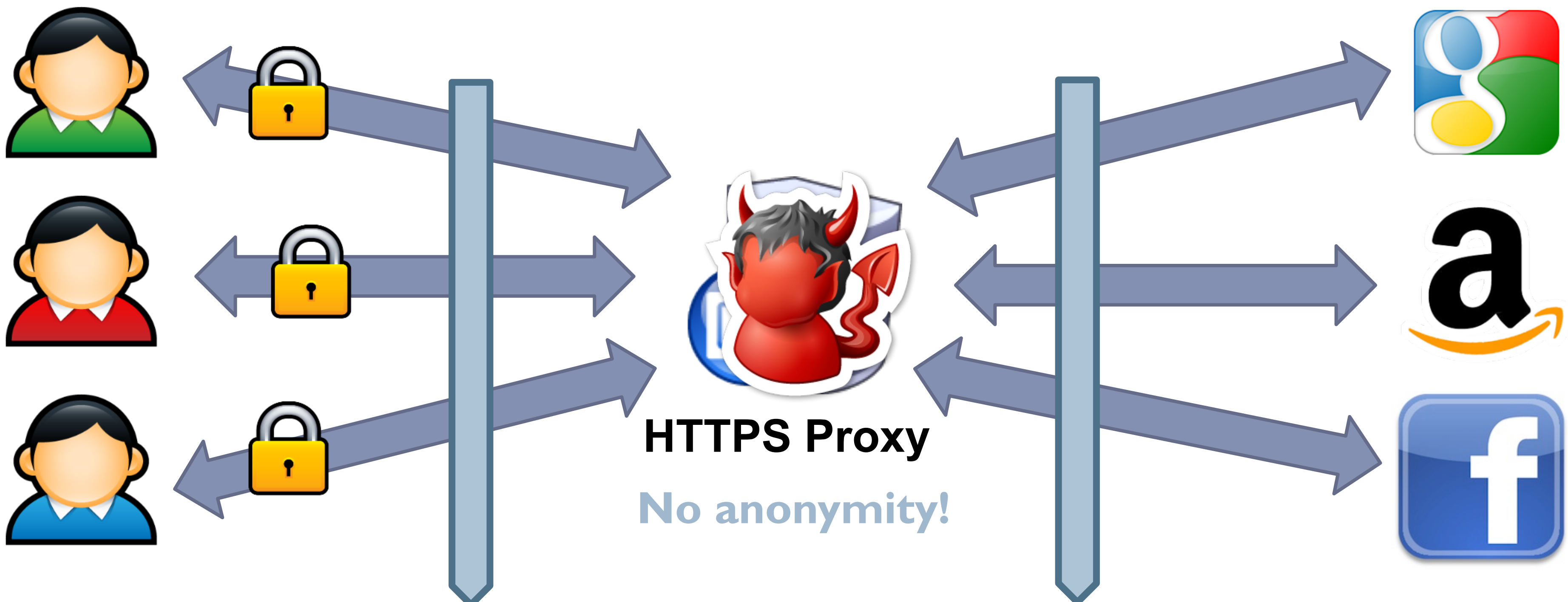
- **Passive/Active**
  - **Passive**: eavesdrop traffic
  - **Active**: able to observe, delay, alter and drop messages in the system
- **Local/Global**
  - **Local**: able to observe traffic to/form user's network link, within LAN
  - **Global**: able to observe effectively large amount or all network links, across LAN boundaries
- **Internal/External**
  - **Internal**: participants in the anonymity system, adversary-operated nodes
  - **External**: not participate in the protocol but may be able to observe, inject or modify traffic in the system

TLS does not provide anonymity



- Content is unobservable
  - Due to encryption
- Source and destination are trivially linkable
  - No anonymity!

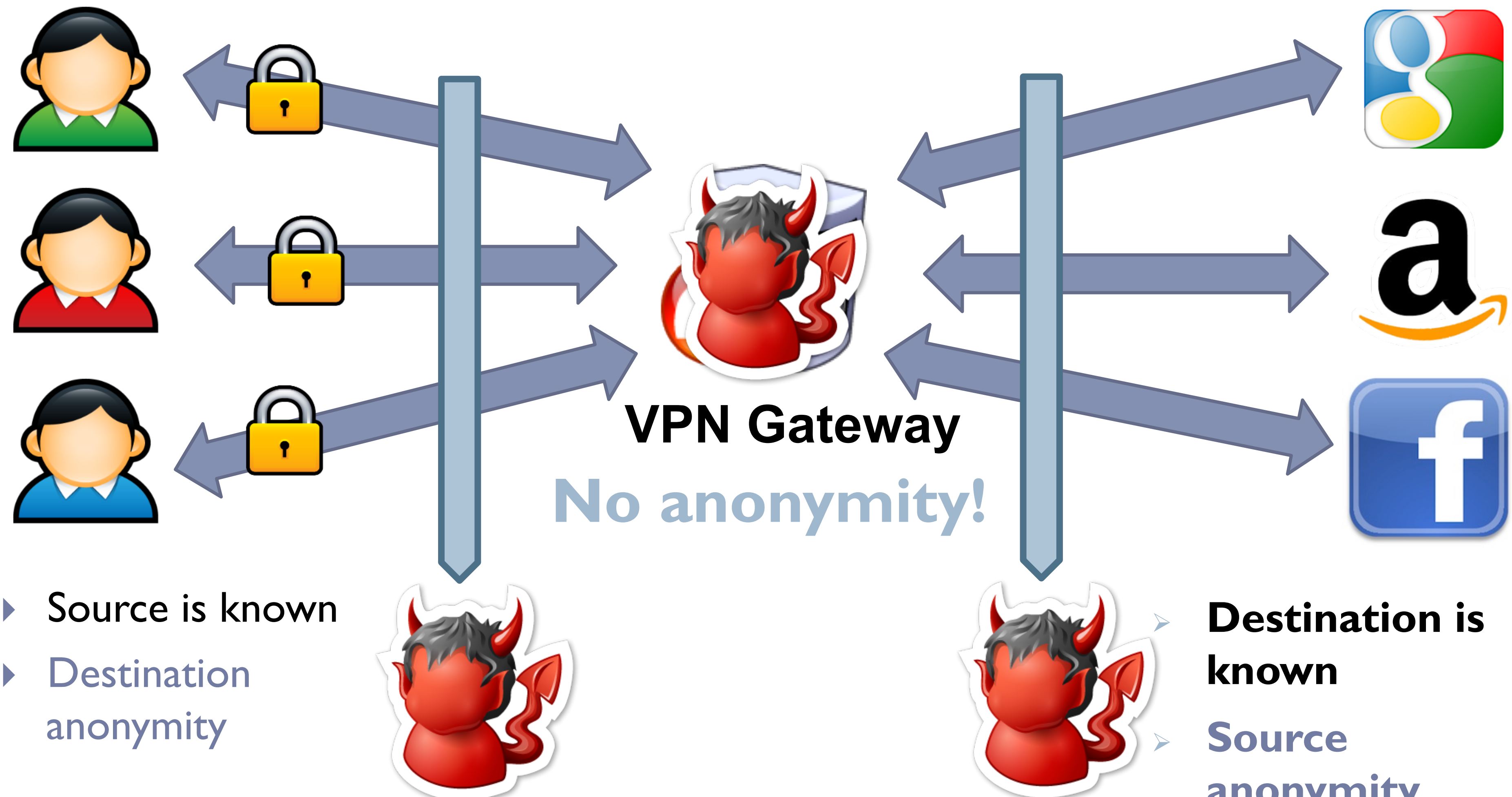
Anonymizing proxies



- Source is known
- Destination anonymity

- Destination is known
- Source anonymity

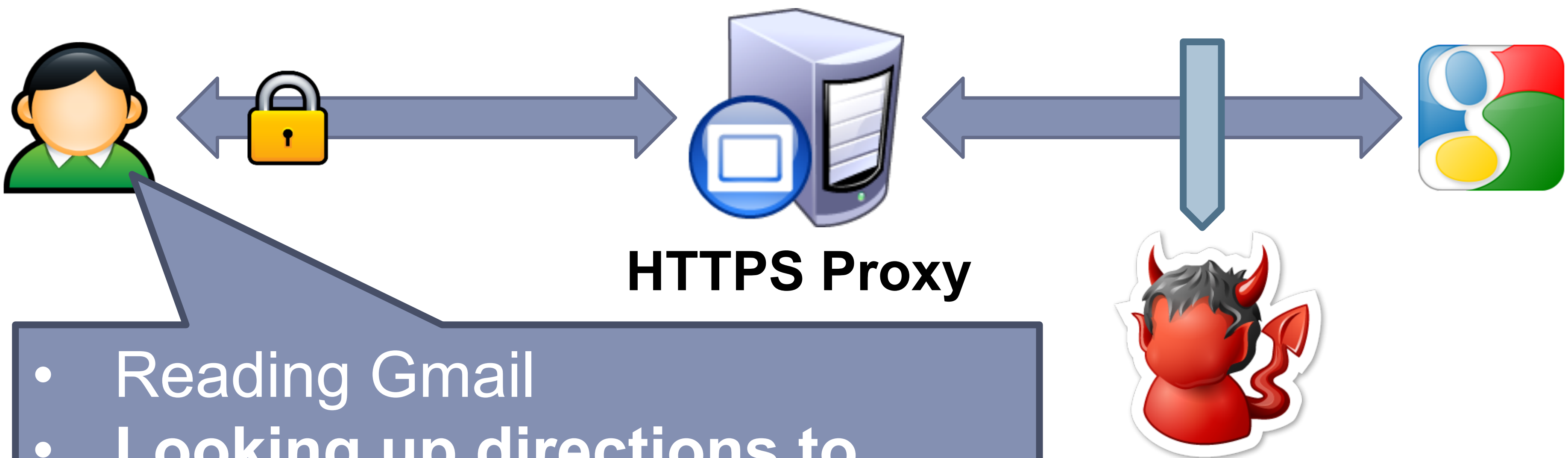
Anonymizing VPNs



- Source is known
- Destination anonymity

- Destination is known
- Source anonymity

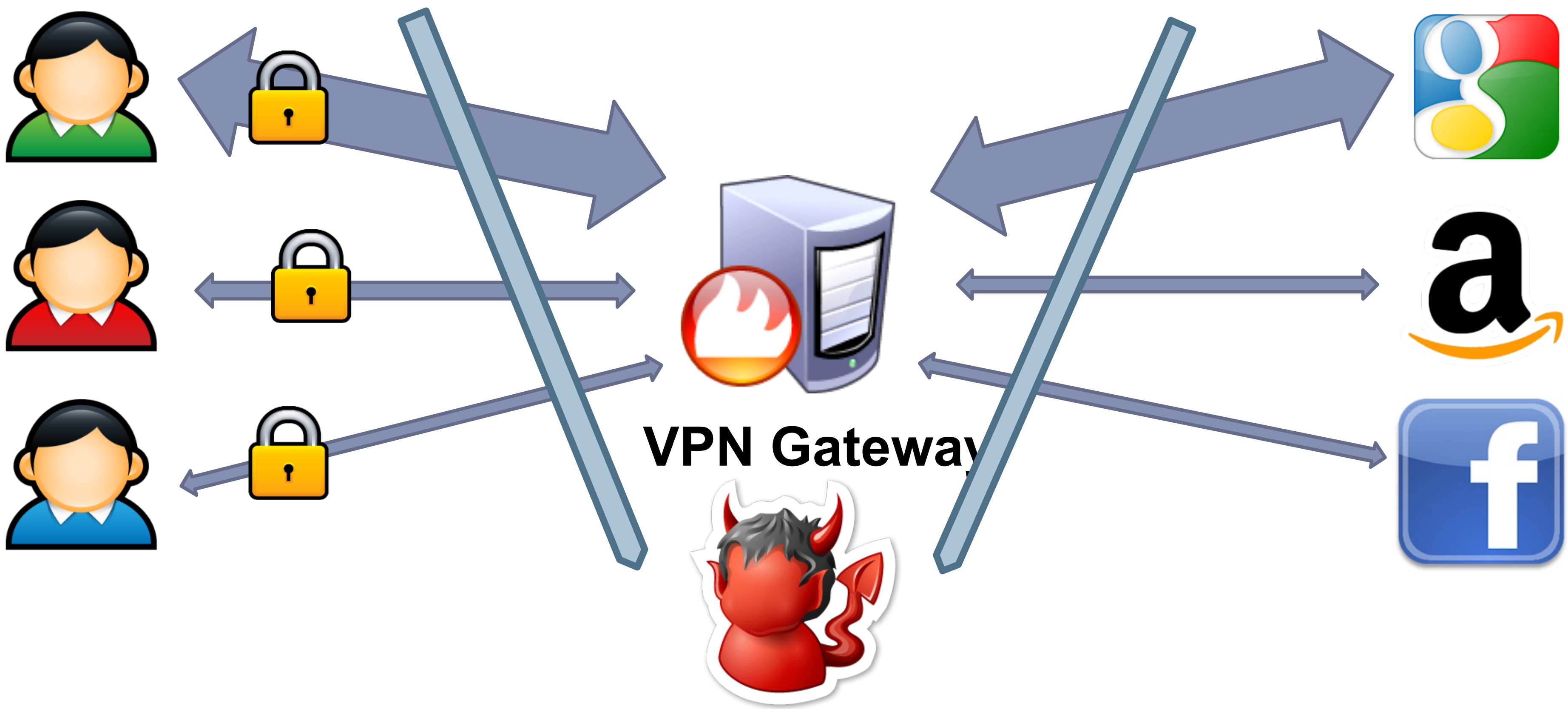
Using content to Deanonymize



- Reading Gmail
- **Looking up directions to home**
- Updating your **Facebook** profile

No anonymity!

Statistical inference attacks



- Statistical analysis of traffic patterns can compromise anonymity, i.e. the **timing** and/or **volume** of packets

Data to protect

- Personally Identifiable Information (PII)
  - Name, address, phone number, etc.
- OS and browser information
  - Cookies, etc.
- Language information
- IP address
- Amount of data sent and received
- Traffic timing

Key systems/concepts

- Mixes and mixnets
- Crowds
- Onion routing

3: Mixnets.

MIX-based systems

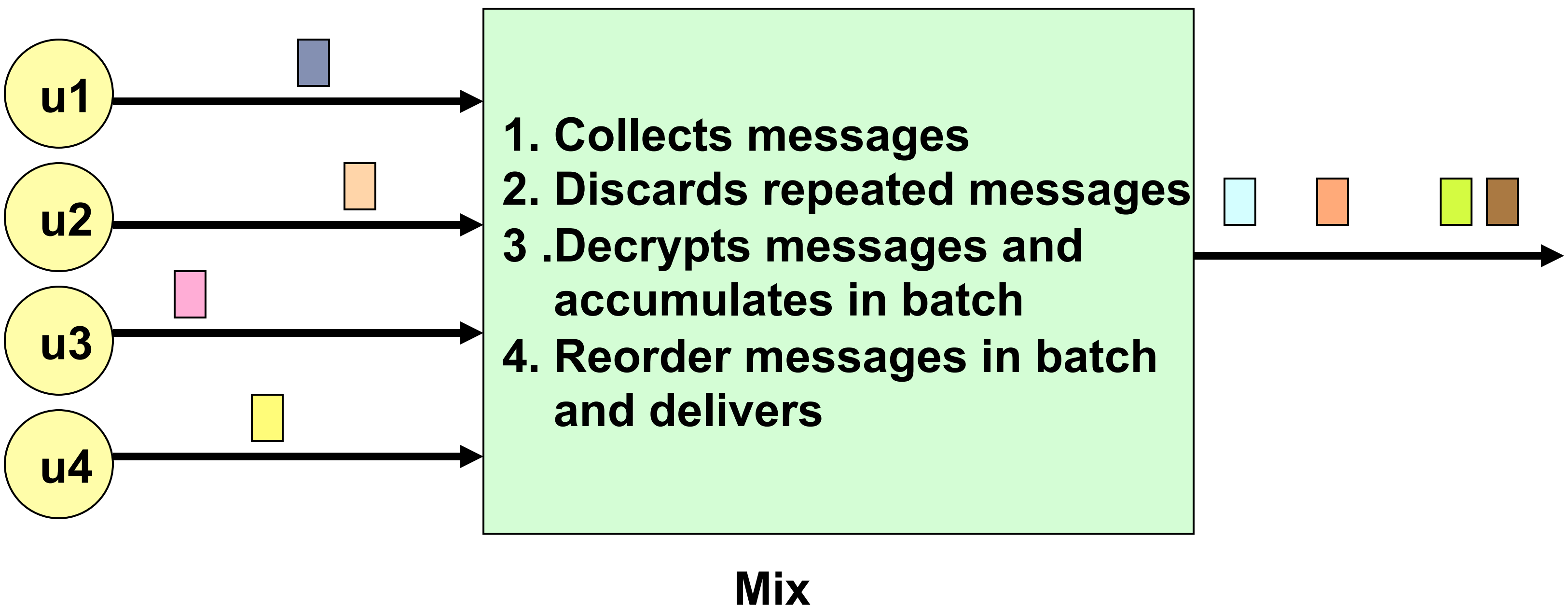
- Introduced by David Chaum (1981) for anonymous email; has been generalized to TCP traffic
- Uses relay servers (MIXes) for anonymous communication
- Goals
  - Sender anonymity
  - Unlinkability against global eavesdroppers
- Idea: Messages from sender **"look"** (contents, time) differently than messages to recipient
- Had impact on other ideas such as: onion routing, traffic mixing, dummy traffic (a.k.a. cover traffic)

MIX – basic operations

- A mix is a store-and-forward relay
- Batching
  - collect fixed-length messages from different sources
  - accumulate a batch of n messages
- Mixing
  - cryptographically transform collected messages
  - forwarding messages to their recipients in random order

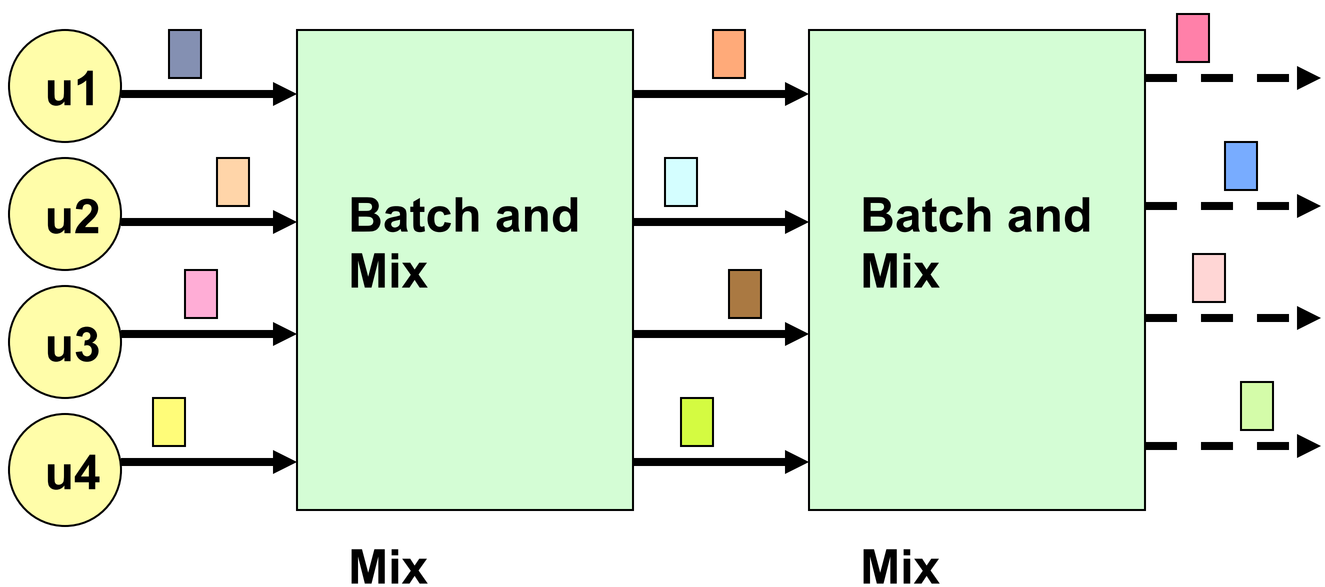
MIX - example

- Each mix has a public key
- Each sender encrypts its message (with randomness) using public key of mix



MIX - variants

- Single mix (also single point of trust, attack and failure)
- Mix cascade
- Mix network
- Different ways of batch and mix operations

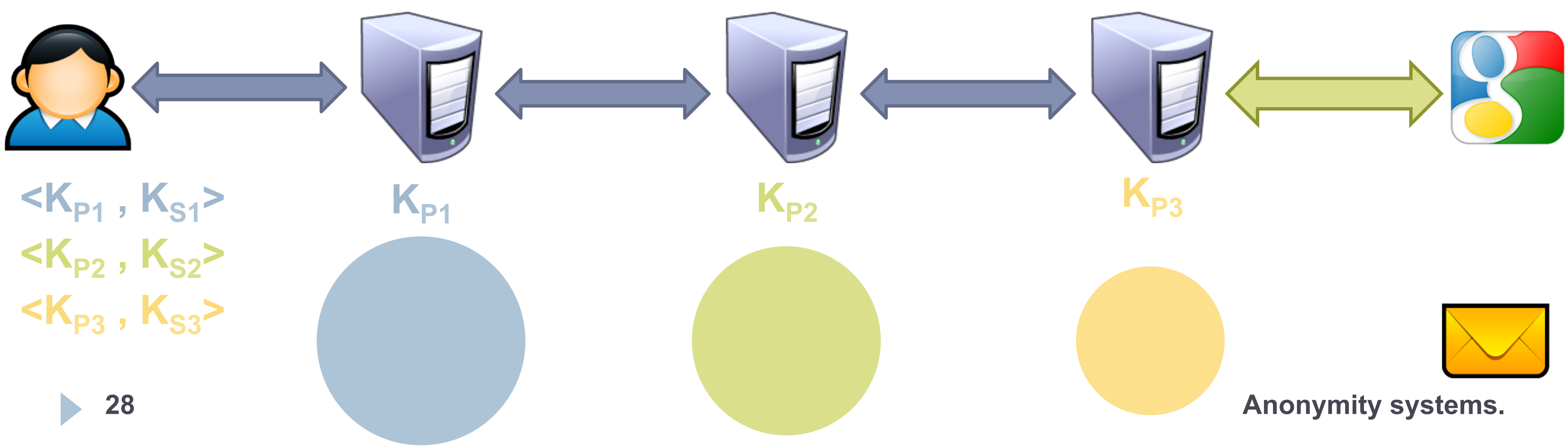


MIX (cont.)

- Traditional designs are message-based
- Advantage: Hinders timing attacks
  - Messages may be artificially delayed
  - Temporal correlation is warped
- Disadvantage: high latency and asynchronous due to batch and mix operations
  - may be acceptable for applications like email
  - frustrating user experience in low latency or interactive applications: web browsing, instant messaging, SSH
- Alternatives: circuit-based designs

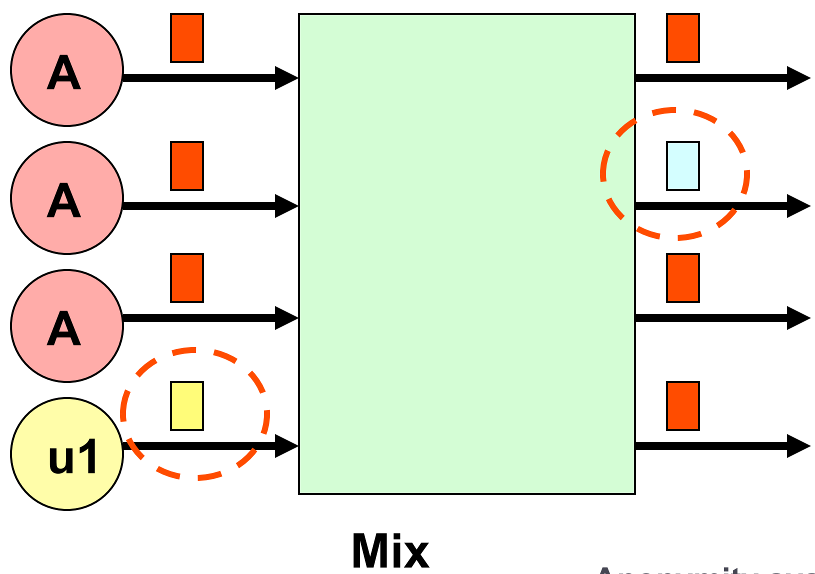
Return Traffic

- In a mix network, how can the destination respond to the sender?
- During path establishment, the sender places keys at each mix along the path
  - Data is re-encrypted as it travels the reverse path



Node flushing attack

- Intended to defeat MIX-based systems
- Flooding attack, (n-1) attack
- Flood a node with identifiable fake messages but leave a room for a single message to be traced
- Link user's input message with messages leaving the node

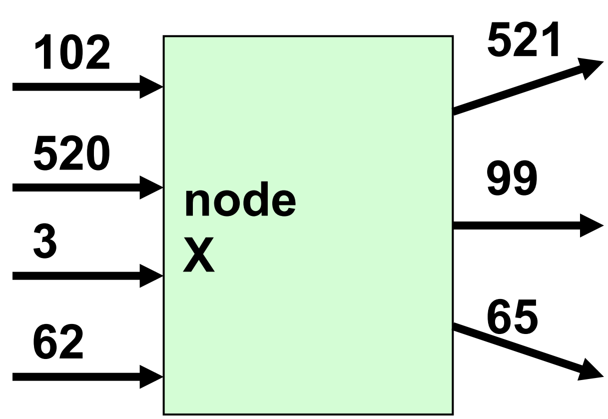


Trickle attack

- Trickle, flushing attack - referred as blending attack
- Suppose a MIX accumulates and emits messages in rounds
- An active attacker holds a target message until the mix emits a batch of messages
- He then submits target message to mix while blocking other incoming messages
- Only the target message is emitted in the next round
- Requires control over traffic flow

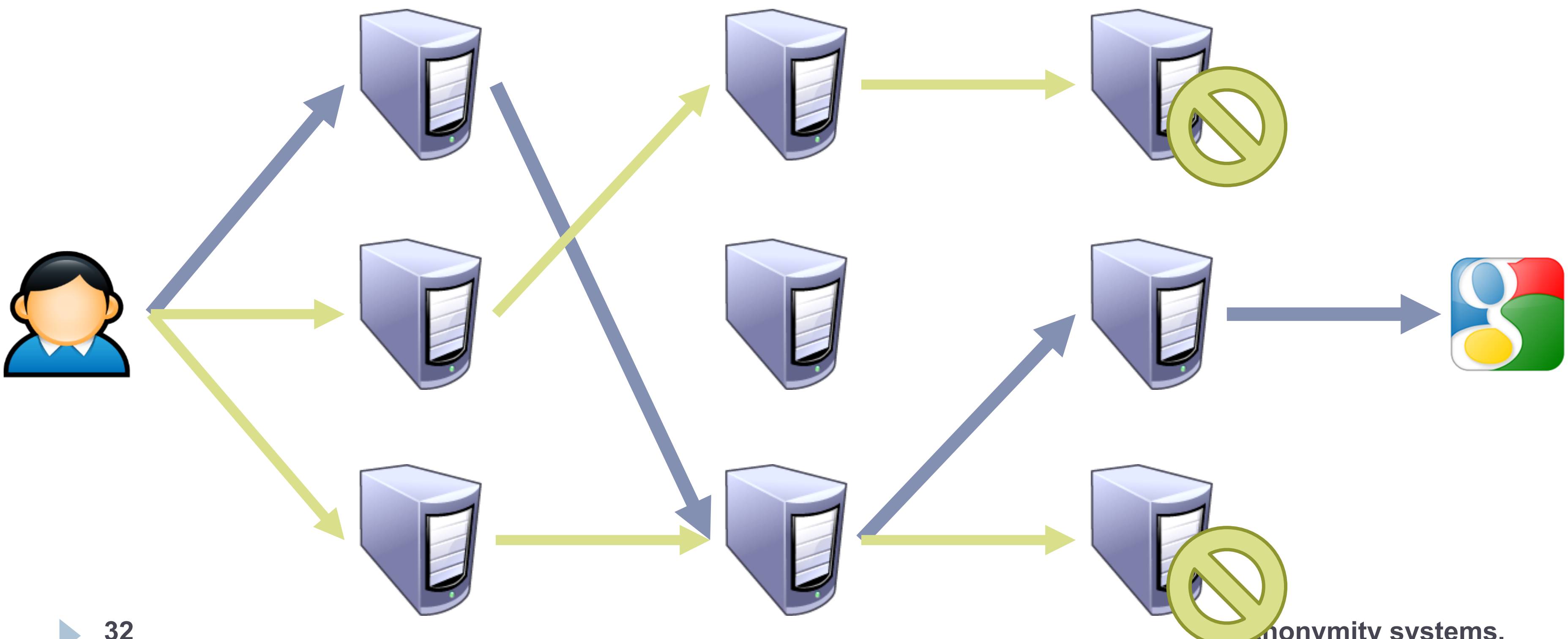
Packet counting attack

- Count the number of messages entering a node and leaving an anonymous tunnel
- Constant link padding may help:
  - Two nodes exchange a constant number of same-sized packets per time unit
  - Generate dummy traffic on idle or lightly loaded links
  - Costly



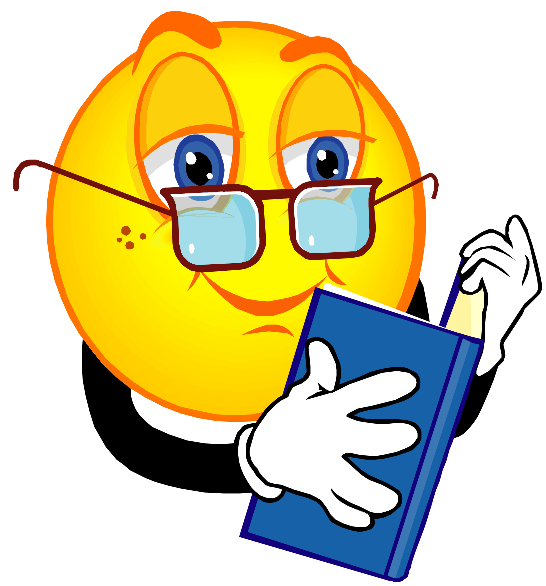
Dummy / Cover Traffic

- Simple idea:
  - Send useless traffic to help obfuscate real traffic



Summary for Mixes

- Key idea is to gather a bunch of messages, then mix them and output in random order
- Can be used as a network
- Resilient to timing attacks but possible attacks include packet counting, flushing, etc
- Disadvantage is that it is slow



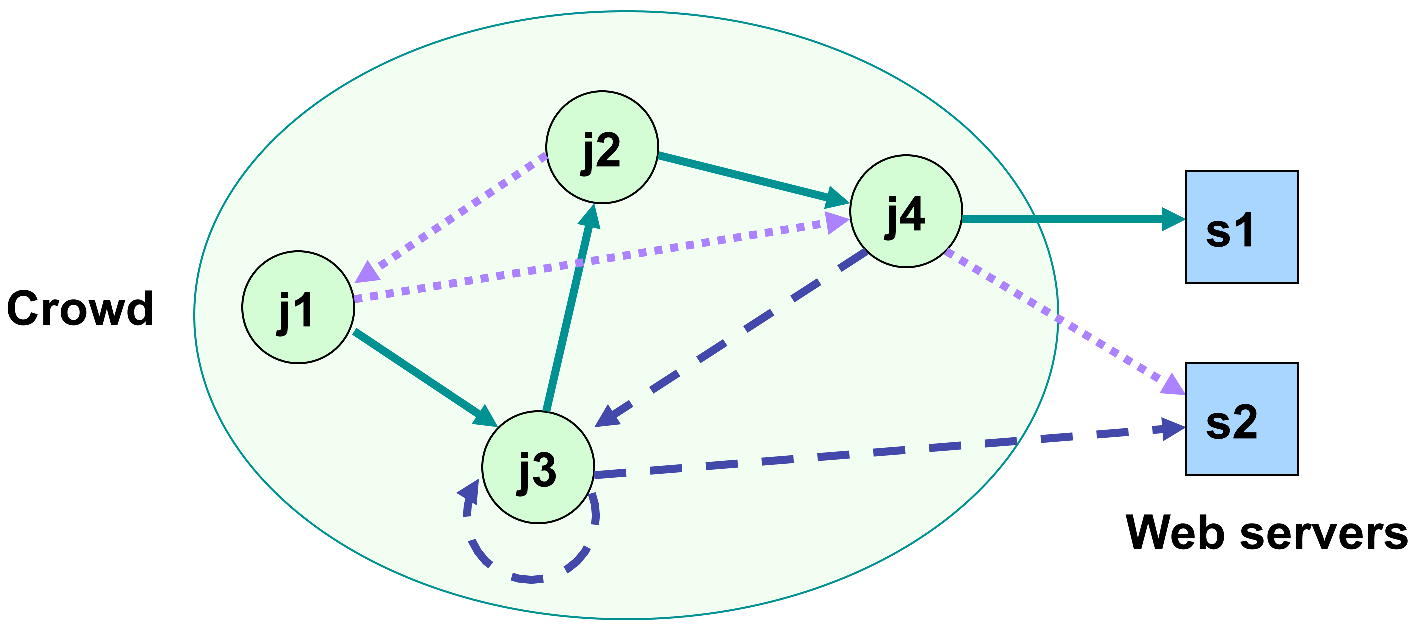
4: Crowds

Crowds

- Key idea
  - Users' traffic blends into a crowd of users
  - Eavesdroppers and end-hosts don't know which user originated what traffic
- High-level implementation
  - Every user runs a proxy on their system
  - Proxy is called a jondo
    - From "John Doe," i.e. an unknown person
  - When a message is received, select $x \in [0, 1]$
    - If $x > p_f$: forward the message to a random jondo
    - Else: deliver the message to the actual receiver

Crowds

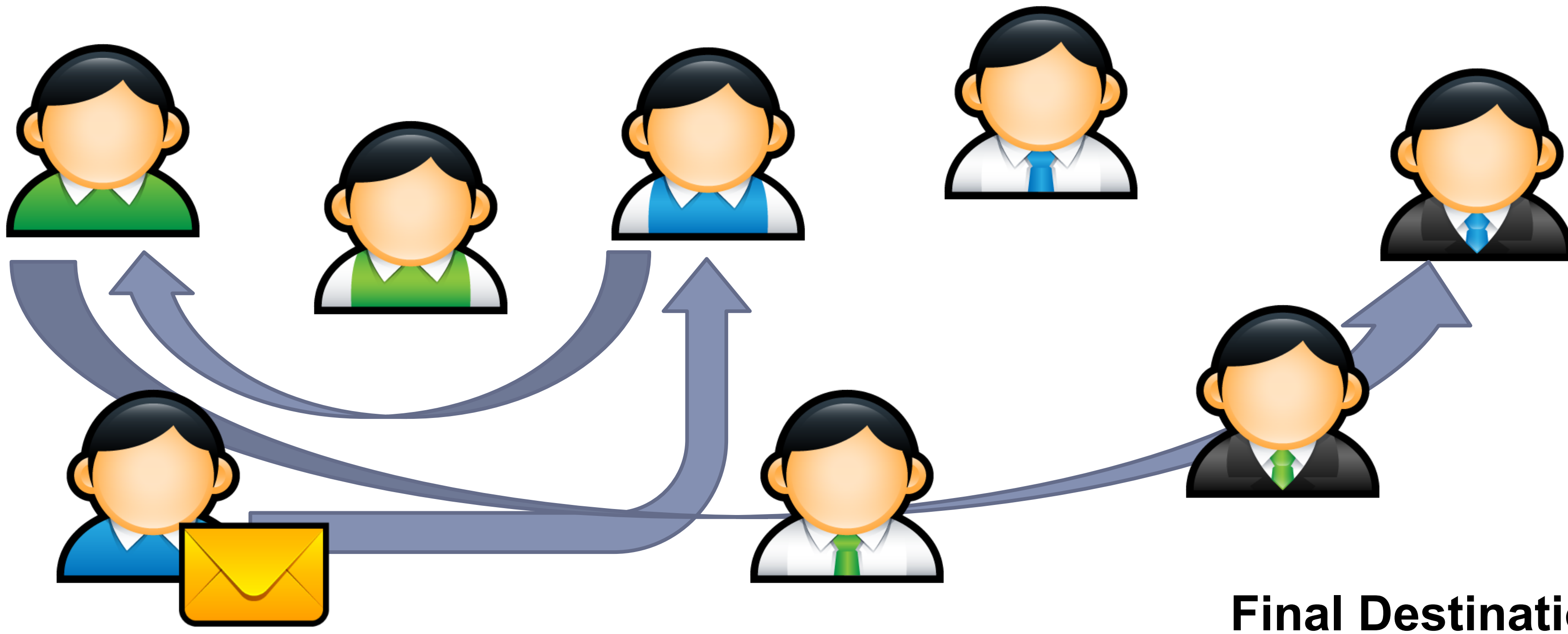
- Anonymous web browsing
- Dynamic collecting users (jondo) in a group (crowd)
- Member list maintained in a central server (blender)



Crowds (cont.)

- Initiator submits request to a random member
- Upon receiving a request, a member either:
  - forwards to another random member (p = pf)
  - submits to end server (p = 1 - pf)
- A random path is created during the first request, subsequent requests use the same path; server replies using the same path but in reverse order
- Link encryption of messages with a shared key known to all members

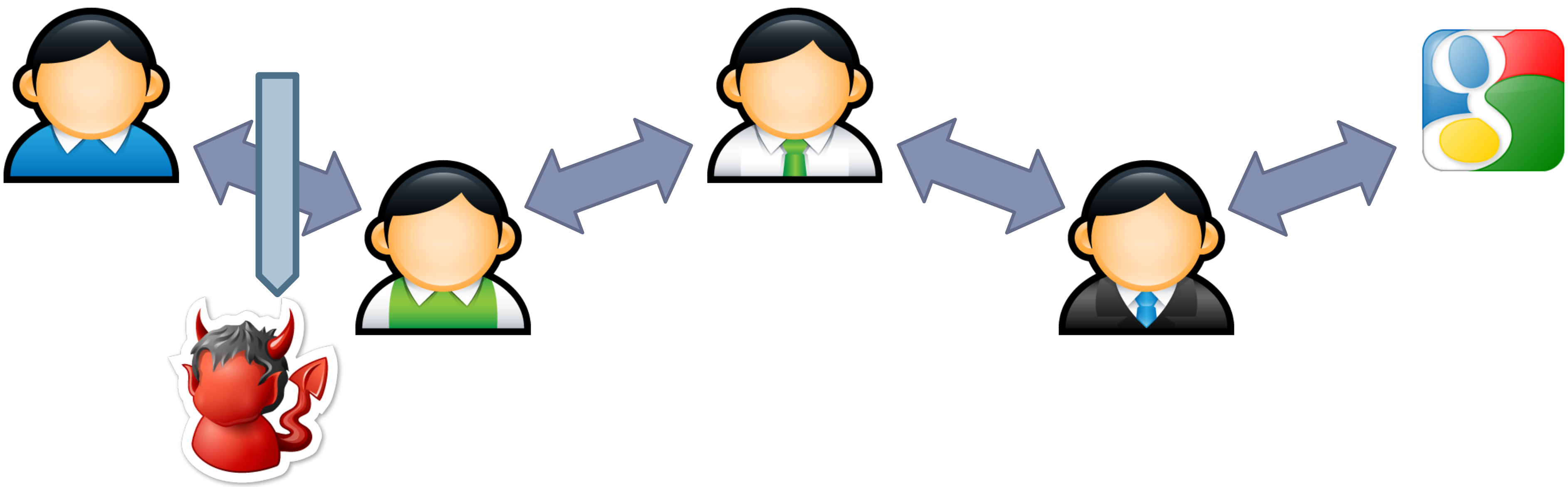
Crowds example



Final Destination

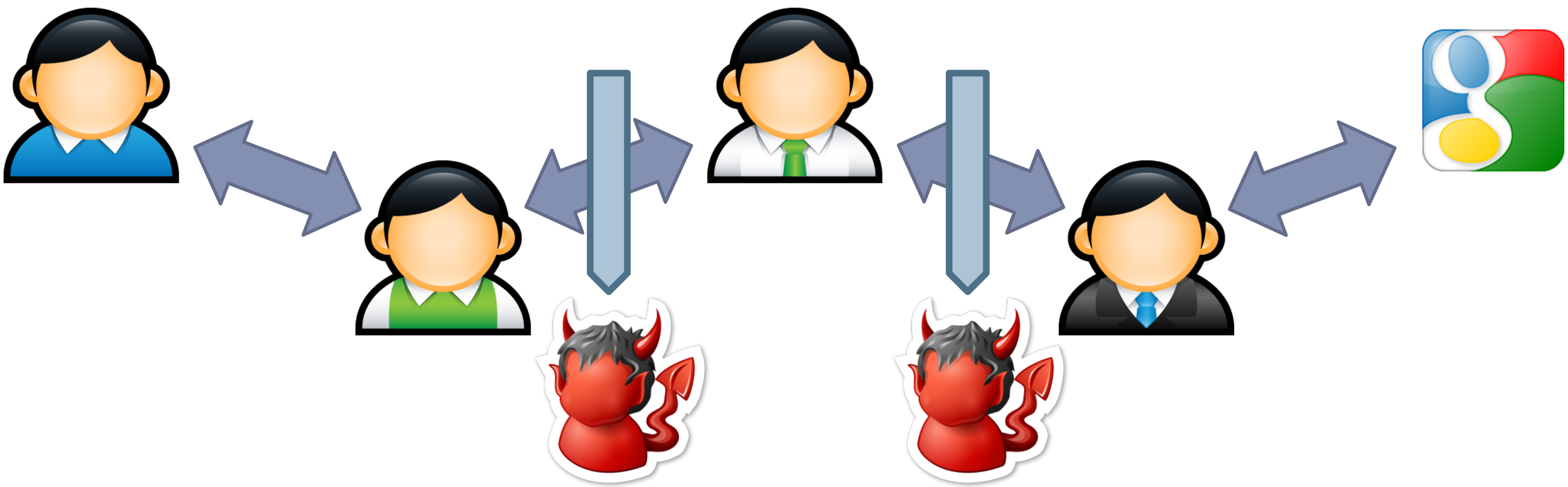
- Links between users use public key crypto
- Users may appear on the path multiple times

Anonymity in crowds



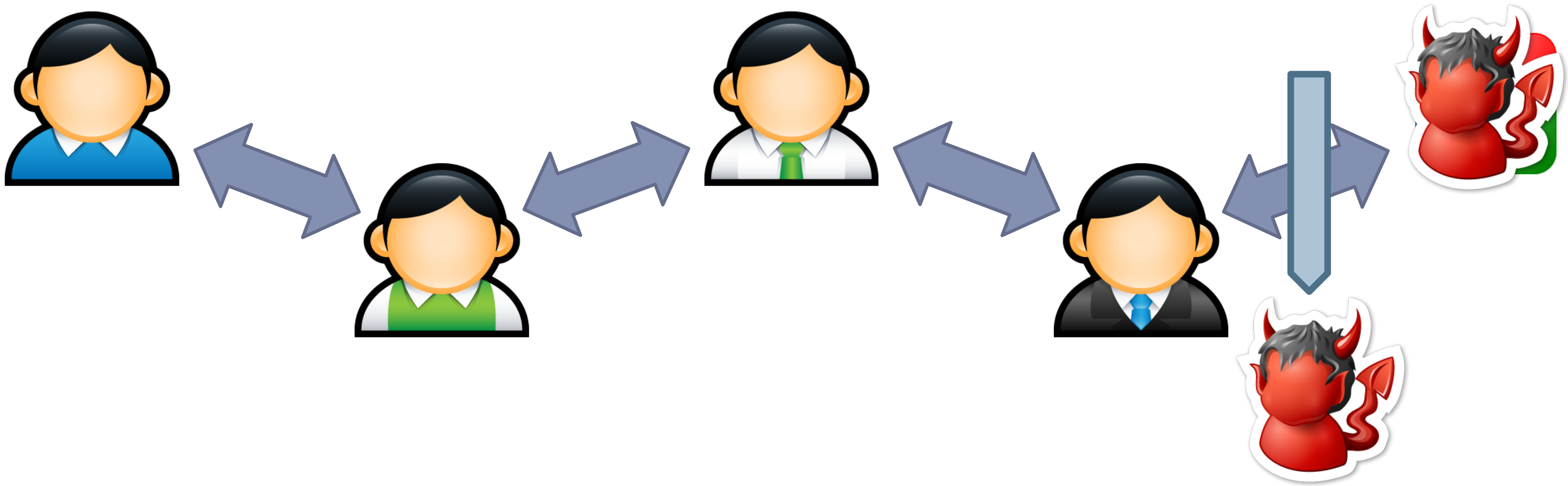
- No source anonymity
  - Target receives *m* incoming messages (*m* may = 0)
  - Target sends *m* + *1* outgoing messages
  - Thus, the target is sending something
- Destination anonymity is maintained
  - If the source isn't sending directly to the receiver

Anonymity in crowds



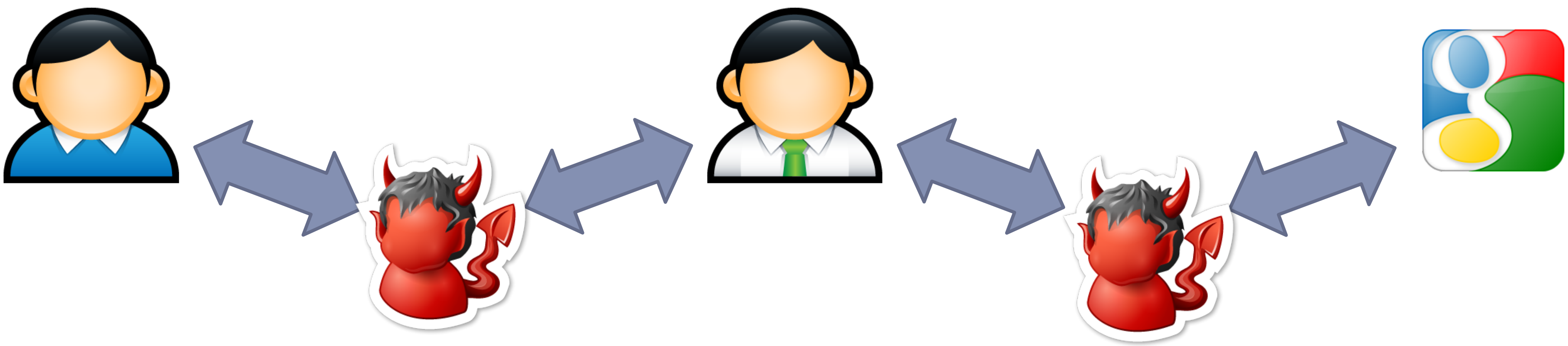
- Source and destination are anonymous
  - Source and destination are jondo proxies
  - Destination is hidden by encryption

Anonymity in crowds



- Destination is known
  - Obviously
- Source is anonymous
  - O(n) possible sources, where n is the number of jondos

Anonymity in crowds



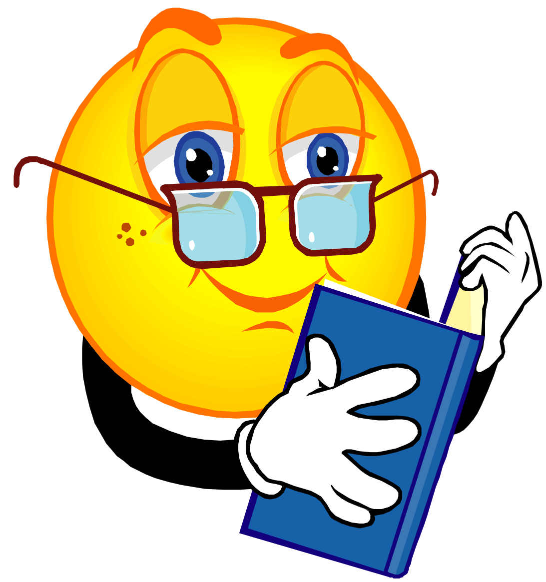
- Destination is known
  - Evil jondo is able to decrypt the message
- Source is somewhat anonymous
  - Suppose there are *c* evil jondos and *n* total jondos
  - If $p_f > 0.5$, and $n > 3(c + 1)$, then the source cannot be inferred with probability > 0.5

Other implementation details

- Crowds requires a central server called a Blender
  - Keep track of who is running jondos
    - Kind of like a BitTorrent tracker
  - Broadcasts new jondos to existing jondos
  - Facilitates exchanges of public keys

Summary for crowds

- Crowds has excellent scalability
  - Each user helps forward messages and handle load
  - More users = better anonymity for everyone
  - Strong source anonymity guarantees
- Very weak destination anonymity
  - Evil jondos can always see the destination
  - Weak unlinkability guarantees



5: Onion routing

Disadvantages of Basic Mixnets

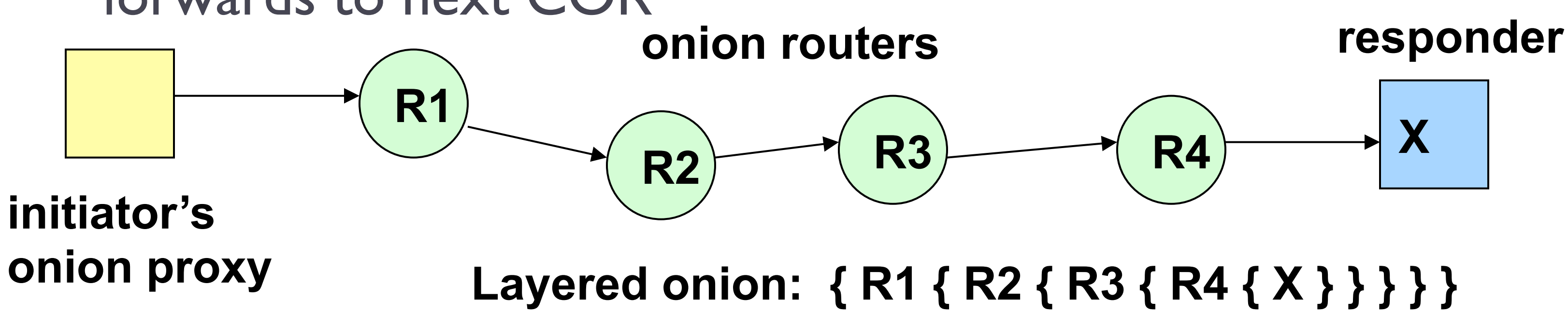
- Public-key encryption and decryption at each mix are computationally expensive
- Basic mixnets have high latency
  - Ok for email, not Ok for anonymous Web browsing
- Challenge: low-latency anonymity network
  - Use public-key cryptography to establish a "circuit" with pairwise symmetric keys between hops on the circuit
  - Then use symmetric decryption and re-encryption to move data messages along the established circuits
  - Each node behaves like a mix; anonymity is preserved even if some nodes are compromised

Onion routing

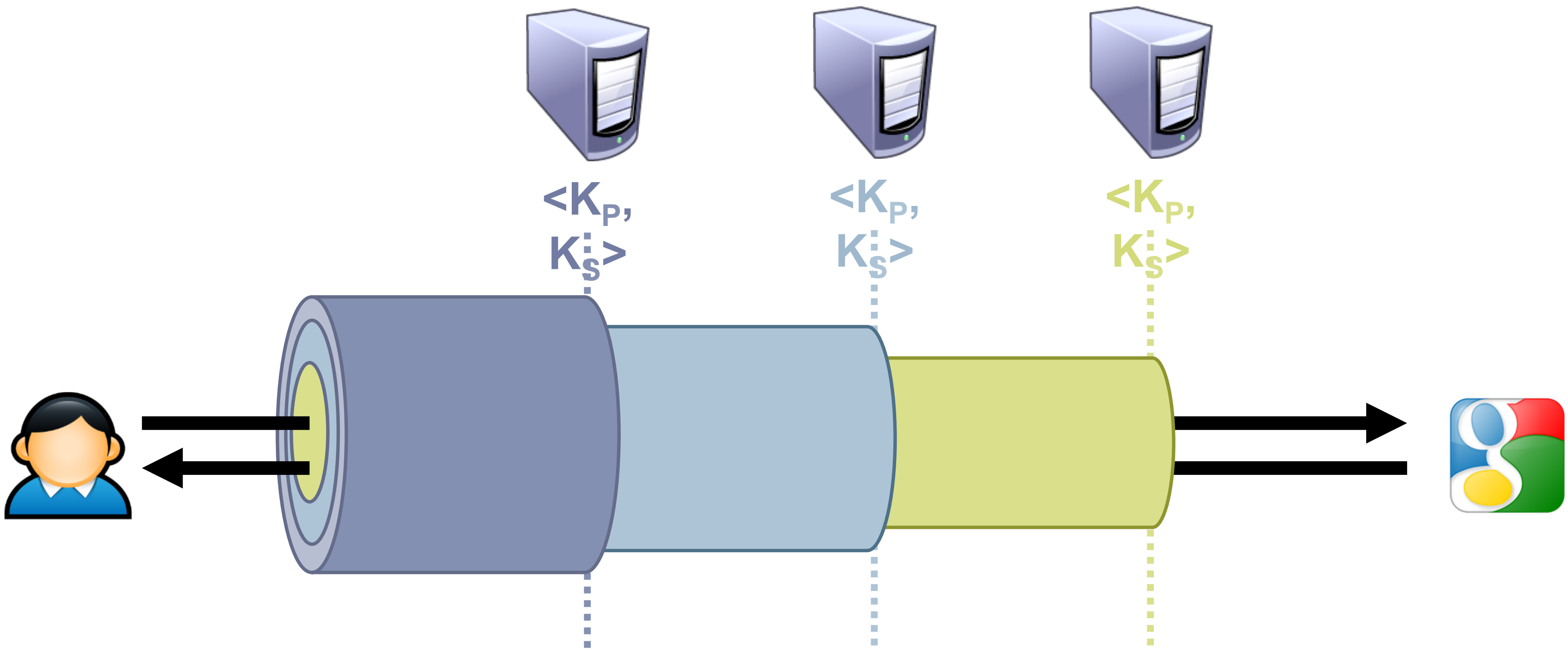
- A (small) fixed core set of relays
  - Core Onion Router (COR)
- Designed to support low-latency service
- Initiator defines an anonymous path for a connection through an "onion"
- An onion is a layered structure (recursively encrypted using public keys of CORs) that defines:
  - path of a connection through CORs
  - properties of the connection at each point, e.g. cryptographic algorithms, symmetric keys

Onion routing (cont.)

- Initiator’s onion proxy (OP)
  - connects to COR
  - initiates a random circuit using an onion
  - converts data to fixed size cells
  - performs layered encryption, one per router
- Circuit-based multi-hop forward
  - Each COR decrypts and removes a layer of received cells, then forwards to next COR



"The onion"



Onion creation

- To create and transmit an onion, the originator selects a set of nodes from a list provided by a *directory node*
- Chosen nodes are arranged into a path, called a **circuit**, through which the message will be transmitted
- To preserve the anonymity of the sender, no node in the circuit should be able to tell whether the node before it is the originator or another intermediary like itself
- No node in the circuit should be able to tell how many other nodes are in the circuit
- Note: the final node, the "exit node", is able to determine its own location in the chain

Tarzan & MorphMix

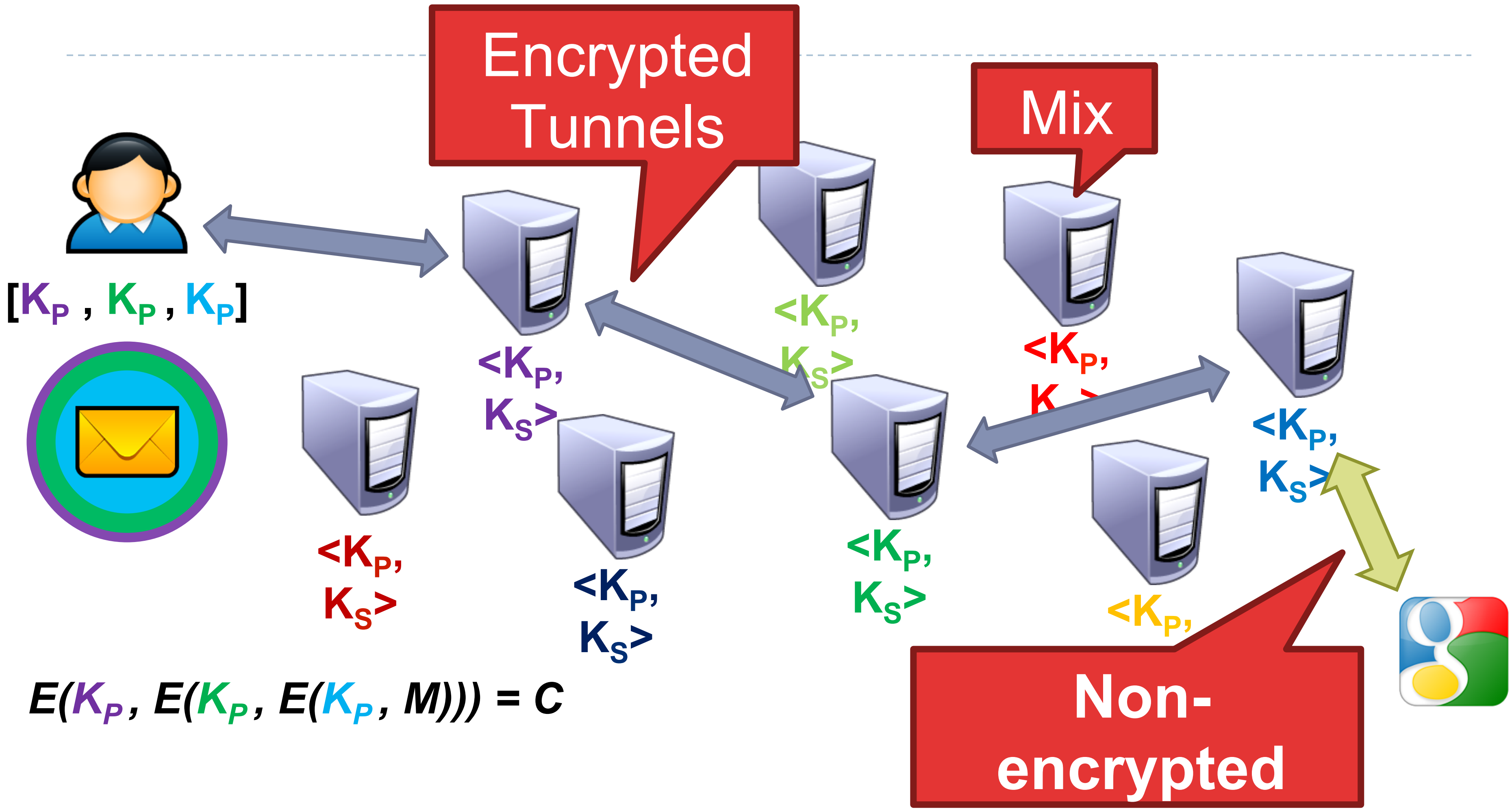
- Similar to Onion routing, Mix-net approach but extended to peer-to-peer environment
  - Layered/nested encryption with multi-hop forwarding
- All peers are potential message originators and relays
  - More potential relays than a small fixed core set
  - More scalable
  - Hide one's action in a large dynamic set of users
- Tarzan targets at network layer while MorphMix runs at application layer

Tarzan & MorphMix (cont.)

- Larger dynamic set of unreliable nodes

- More efforts to defense against colluding nodes (dishonest or adversary controlled)
  - Restricted peer-selection in Tarzan
  - Collusion detection mechanism in MorphMix

Mix Proxies and Onion Routing



- Mixes form a cascade of anonymous proxies
- All traffic is protected with layers of encryption

6: Tor:The Second-Generation Onion Router

Tor: The 2nd Generation Onion Router

- Basic design: a mix network with improvements
  - Perfect forward secrecy
  - Introduces guards to improve source anonymity
  - Takes bandwidth into account when selecting relays
    - Mixes in Tor are called relays
  - Introduces hidden services
    - Servers that are only accessible via the Tor overlay

Deployment and statistics

- Largest, most well deployed anonymity preserving service on the Internet http://torproject.org
  - Publicly available since 2002
  - Continues to be developed and improved
- Currently, ~5000 Tor relays around the world
  - All relays are run by volunteers
  - It is suspected that some are controlled by intelligence agencies
- 500K – 900K daily users, probably larger
- Easy-to-use client proxy,
  - integrated Web browser

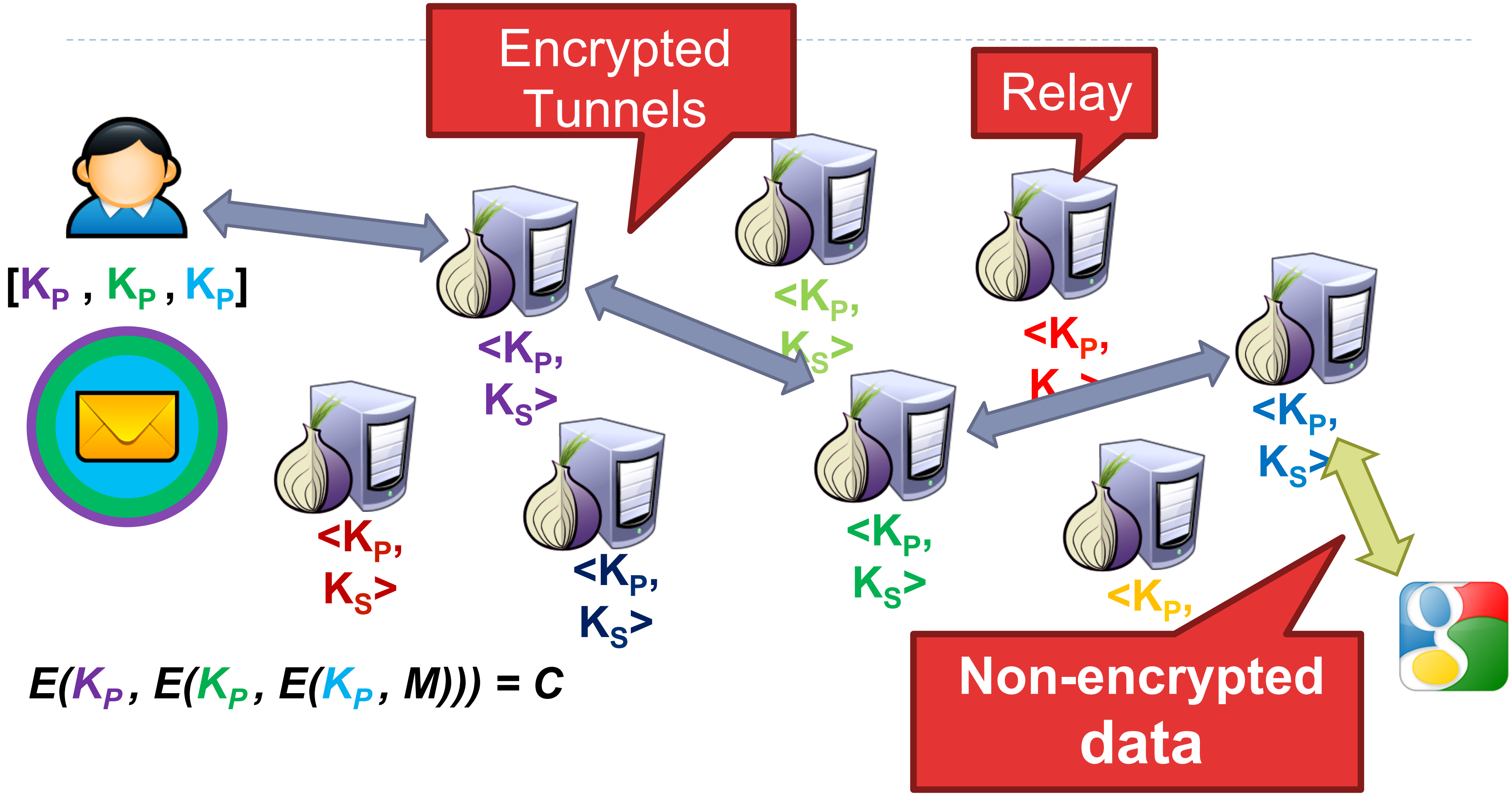
How to use Tor?

1. Download, install, and execute the Tor client
   - The client acts as a SOCKS proxy
   - The client builds and maintains circuits of relays
2. Configure your browser to use the Tor client as a proxy
   - Any app that supports SOCKS proxies will work with Tor
3. All traffic from the browser will now be routed through the Tor overlay

Using Tor

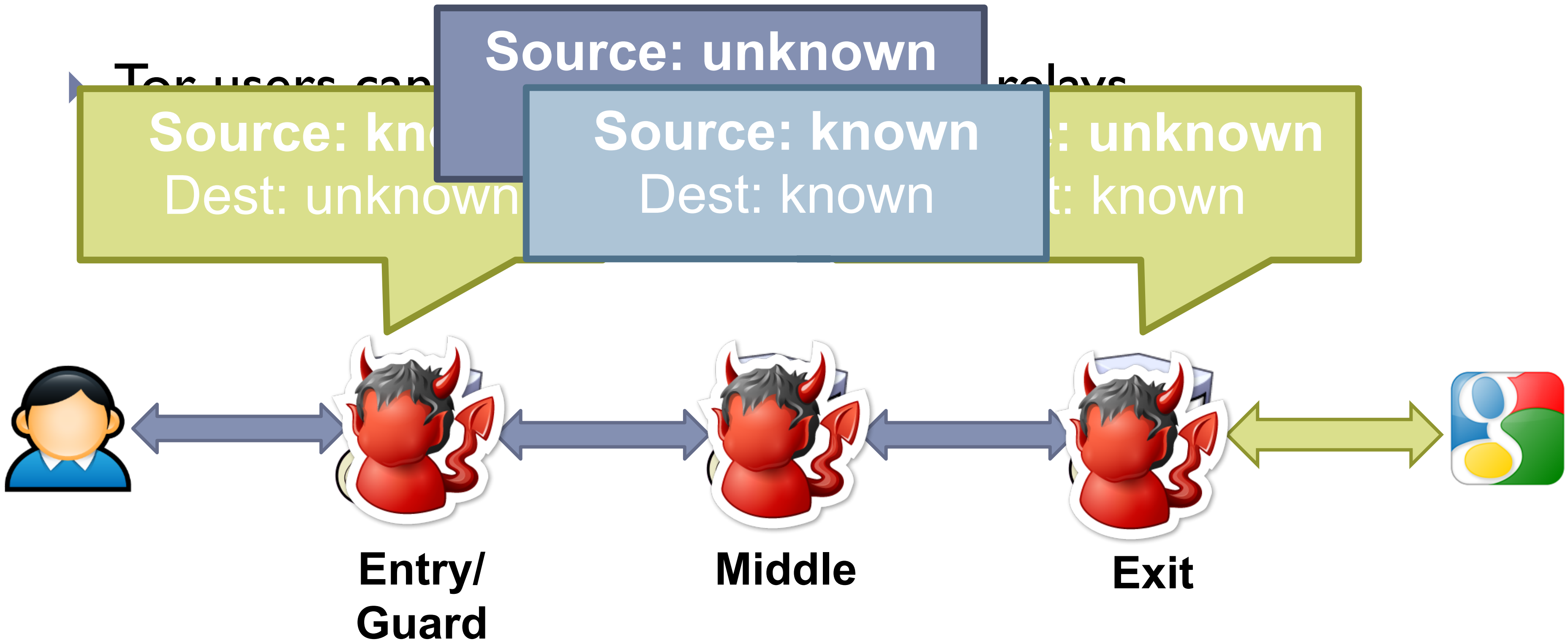
- Many applications can share one circuit
  - Multiple TCP streams over one anonymous connection
- Tor router doesn't need root privileges
  - Encourages people to set up their own routers
  - More participants = better anonymity for everyone
- Directory servers
  - Maintain lists of active relay nodes, their locations, current public keys, etc.
  - Control how new nodes join the network
    - "Sybil attack": attacker creates a large number of relays
  - Directory servers' keys ship with Tor code

Tor Example



- Relays form an anonymous circuit
- All traffic is protected with layers of encryption

Attacks Against Tor Circuits



Predecessor Attack

- Assumptions:
  - *N* total relays

- This is the predecessor attack
- **Attacker controls the first and last relay**
- Probability of being in the right positions increases over time

- However, client periodically builds new circuits
  - Over time, the chances for the attacker to be in the correct positions improves!

Circuit Lifetime

- One possible mitigation against the predecessor attack is to increase the circuit lifetime
  - E.g. suppose your circuit was persistent for 30 days
  - Attacker has 1 chance of being selected as guard and exit
- Problems?
  - If you happen to choose the attacker as guard and exit, you are screwed
  - A single attacker in the circuit (as guard or exit) can still perform statistical inference attacks
  - Tor relays are not 100% stable, long lived circuits will die
- Bottom line: long lived circuits are not a solution
  - Tor's default circuit lifetime is 10 minutes

Selecting Relays

- How do clients locate the Tor relays?
- Tor Consensus File
  - Hosted by trusted directory servers
  - Lists all known relays
    - IP address, uptime, measured bandwidth, etc.
- Not all relays are created equal
  - Entry/guard and exit relays are specially labelled
- Tor does not select relays randomly
  - Chance of selection is proportional to bandwidth

Guard Relays

- Guard relays help prevent attackers from becoming the first relay
  - Tor selects 3 guard relays and uses them for 3 months
  - After 3 months, 3 new guards are selected
- Only certain relays may become guards:
  - Have long and consistent uptimes…
  - Have high bandwidth…
  - Are manually vetted by the Tor community
- Problem: what happens if you choose an evil guard?
  - M/N chance of full compromise (i.e. source and destination)

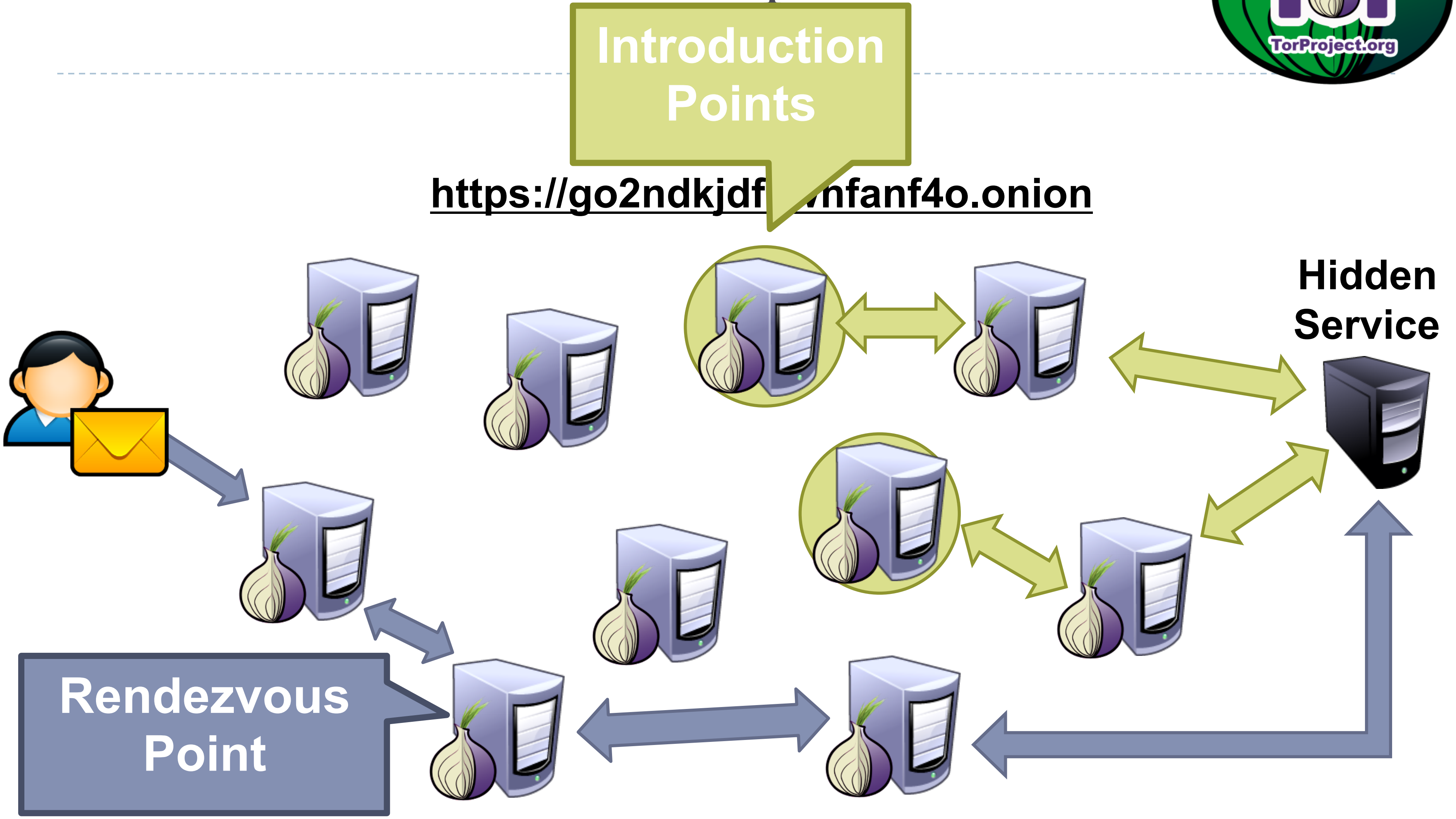
Exit Relays

- Relays must self-elect to be exit nodes
- Why?
  - Legal problems.
  - If someone does something malicious or illegal using Tor and the police trace the traffic, the trace leads to the exit node
- Running a Tor exit is not for the faint of heart

Hidden Services

- Tor is very good at hiding the source of traffic
  - But the destination is often an exposed website
- What if we want to run an anonymous service?
  - i.e. a website, where nobody knows the IP address?
- Tor supports Hidden Services
  - Allows you to run a server and have people connect
  - … without disclosing the IP or DNS name
- Many hidden services
  - Tor Mail, Tor Char
  - DuckDuckGo
  - Wikileaks

Hidden Service Example



- Onion URL is a hash, allows any Tor user to find the introduction points

Perfect Forward Secrecy

- In
  p
- P

- An attacker who compromises a private key can still eavesdrop on future traffic
- **… but past traffic is encrypted with ephemeral keypairs that are not stored**

- Tor implements Perfect Forward Secrecy (PFS)
  - The client negotiates a new public key pair with each relay
  - Original keypairs are only used for signatures
    - i.e. to verify the authenticity of messages

Tor Bridges

- Anyone can look up the IP addresses of Tor relays
  - Public information in the consensus file
- Many countries block traffic to these IPs
  - Essentially a denial-of-service against Tor
- Solution: Tor Bridges
  - Essentially, Tor proxies that are not publicly known
  - Used to connect clients in censored areas to the rest of the Tor network
- Tor maintains bridges in many countries

Obfuscating Tor Traffic

- Bridges alone may be insufficient to get around all types of censorship
  - DPI can be used to locate and drop Tor frames
  - Some countries blocked all encrypted packets for some time
- Tor adopts a pluggable transport design
  - Tor traffic is forwarded to an obfuscation program
  - Obfuscator transforms the Tor traffic to look like some other protocol
    - BitTorrent, HTTP, streaming audio, etc.
  - Deobfuscator on the receiver side extracts the Tor data from the encoding

Passive attacks

- Observe Traffic Patterns
  - Multiplexing minimizes damage
- Observe User Content
  - Use of Privoxy
- Option Distinguishability
  - Leads to tracing due to distinct pattern behavior
- End-to-end Timing Correlation
  - Tor does not hide timing (low-latency requirement)
- End-to-end Size Correlation
  - Leaky-Pipe Topology
- Website Fingerprinting
  - New attack as of 2004, semi-defended by mitigation

Active attacks

- Compromise Keys
  - Mitigated by key rotation and redundant multiple layer encryption. Replacing a node via identity key could theoretically avoid this defense.
- Iterated Compromise
  - Short lifetimes for circuits
- Run Recipient
  - Adversary controls end server, which allows him to use Tor to attack the other end. Privoxy would help minimize chance of revealing initiator
- Run Onion Proxy
  - Compromised OPs compromise all information sent through OP
- DoS non-observed nodes
  - Only real defense is robustness
- Run hostile OR
  - Requires nodes at both ends of a circuit to obtain information
- Introduce Timing
  - Similar to timing discussed in passive version

Active attacks (cont.)

- Tag Attacks
  - Integrity check mitigates this
- Replay Attacks
  - Session key changes if replay used
- Replace End Server
  - No real solution, verify that server is actually server with authentication. Similar to Recipient attack
- Smear Attacks
  - Good press and exit policies
- Hostile Code Distribution
  - All Tor releases signed

Directory subversion

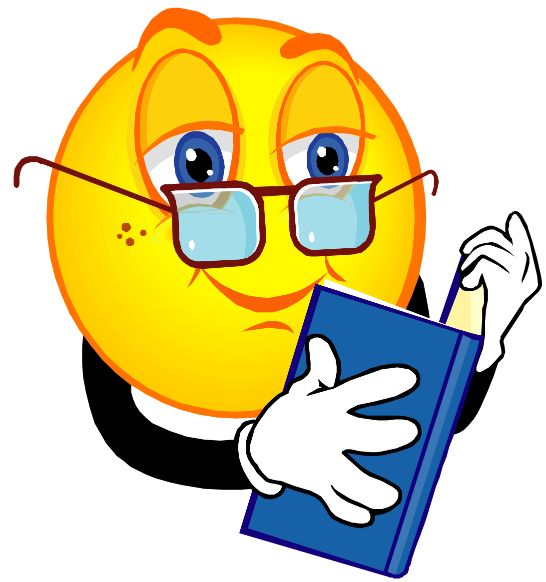
- Destroy Servers
  - Directories require majority rule, or human intervention if more than half destroyed.
- Subvert Server
  - At worst, cast tie-breaker vote
- Subvert Majority of Servers
  - Ensure Directories are independent and resistant to attacks
- Encourage Dissent in Directory Operators
  - People problem, not Tor problem.
- Trick Directories
  - Server Operators should be able to filter out hostile nodes.
- Convince Directories that OR is Functional
  - Directory servers should test by building circuit and streams to OR.

Rendezvous point attacks

- Many Introduction Point Requests
  - IP can block requests with authorization tokens, or require certain amounts of computation per request.
- Attack Introduction Point
  - Server re-advertises on different IP, or advertise secretly. Attacker must disable all IPs.
- Compromise Introduction Point
  - Servers should occasionally verify their IPs, and close circuits that flood them.
- Compromise Rendezvous Point
  - Similar to active attacks against ORs

Summary for Tor

- Most popular anonymous communication systems
- Not perfect, several attacks (and mitigation solutions) exist
- Hidden services are also provided
- Very well studied and continues to be studied



7: More about attacks against anonymous systems.

Attacks on anonymity systems

- Degrading the quality of anonymity service
  - Break sender/receiver anonymity, unlinkability
  - Control anonymity to certain level
  - Traffic analysis, traffic confirmation

- Degrading the utilization of anonymity system
  - Decrease the performance, reliability and availability of system, so as to drive users not using the service
  - Denial-of-Service attacks

Traffic analysis

- If one is interested in breaking the anonymity …
- Based on features in communication traffic, one may infer
  - who's the initiator $\Rightarrow$ NO sender anonymity
  - who's the responder $\Rightarrow$ NO receiver anonymity
  - an initiator-responder mapping $\Rightarrow$ NO unlinkability

Common vulnerabilities

- Message features
  - distinguishable contents, size
- Communication patterns
  - user online/offline period
  - send-receive sequence
  - message frequencies, e.g. burst stream
- Properties/constraints in anonymity systems
  - low-latency requirement
  - link capacity and traffic shaping

Attacks on message features

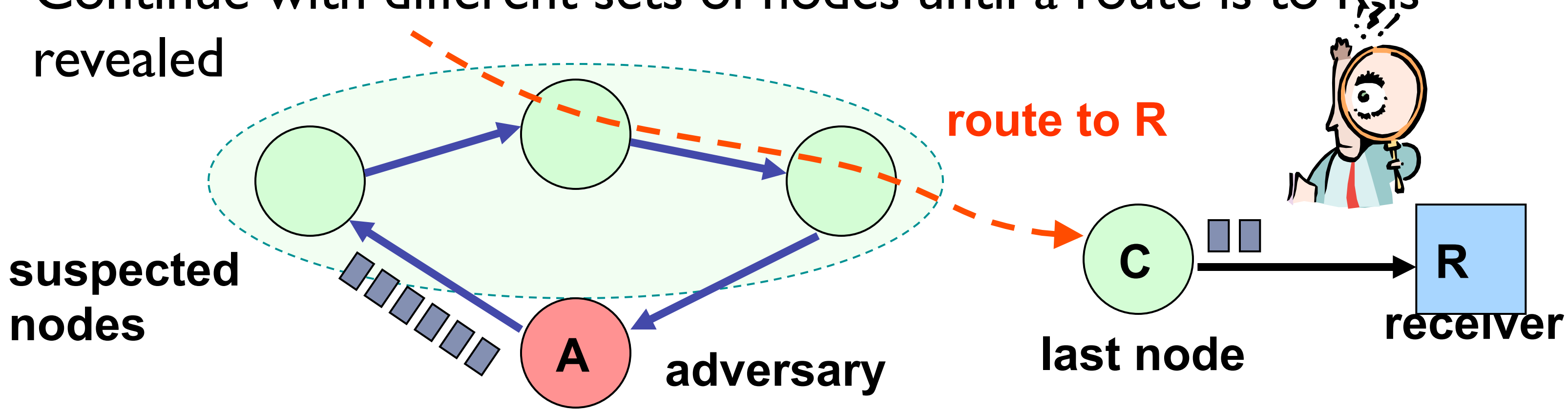
- If a message itself reveals one's identity or more, anonymity is defeated regardless of the strength of an anonymity system!
- Message features
  - size, format, writing style ..., etc
- Message size
  - Varieties of message sizes may help linking a message to some application or sender
  - Fixed by constant-size message padding

Distinguishable message contents

- Message contents
  - may expose user information or the route of a message
  - e.g. host information, Referer, User-Agent fields in HTTP header
- Active adversary can perform message tagging attack
  - Alter bits in message header/payload
  - Recognize altered messages to exploit the route
- Solutions
  - Proper message transformation: e.g. encryption
  - Removal of distinguishable information: e.g. Privoxy (privacy enhancing proxy)

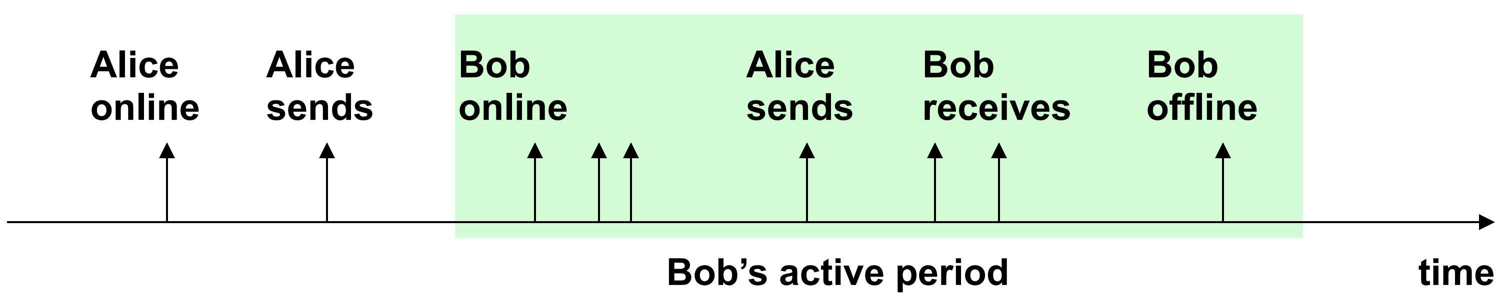
Clogging attack

- Observe traffic between a certain last node C and end receiver R
- Create a route through a set of suspected nodes
- Clog the route with high volume of traffic
- Decrease in throughput from C to R may indicate at least one node in the suspected route belongs to a route containing C
- Continue with different sets of nodes until a route is to R is revealed



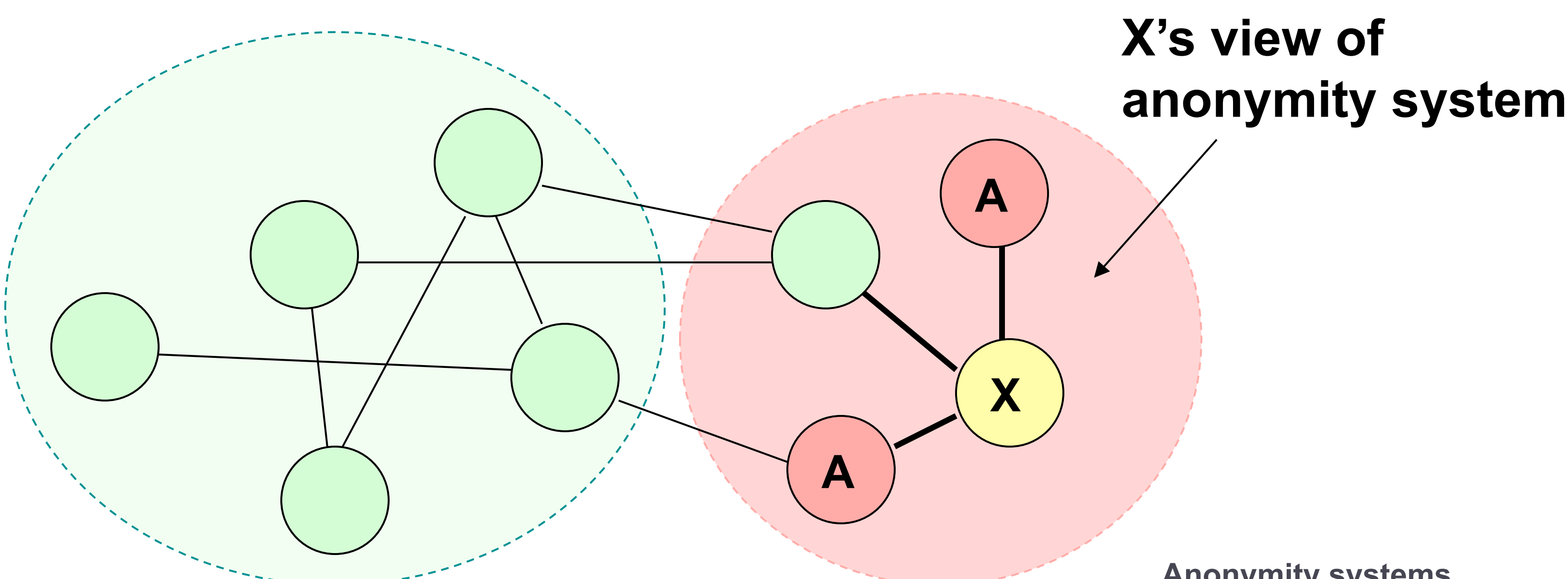
Intersection attacks

- Communication pattern
  - Users join and leave the system from time to time
  - Users are not active in communication all the time
  - Some receivers receive messages after some senders transmit messages
- Intersecting sets of possible senders over different time periods → anonymity set shrinks
- Short term vs Long term



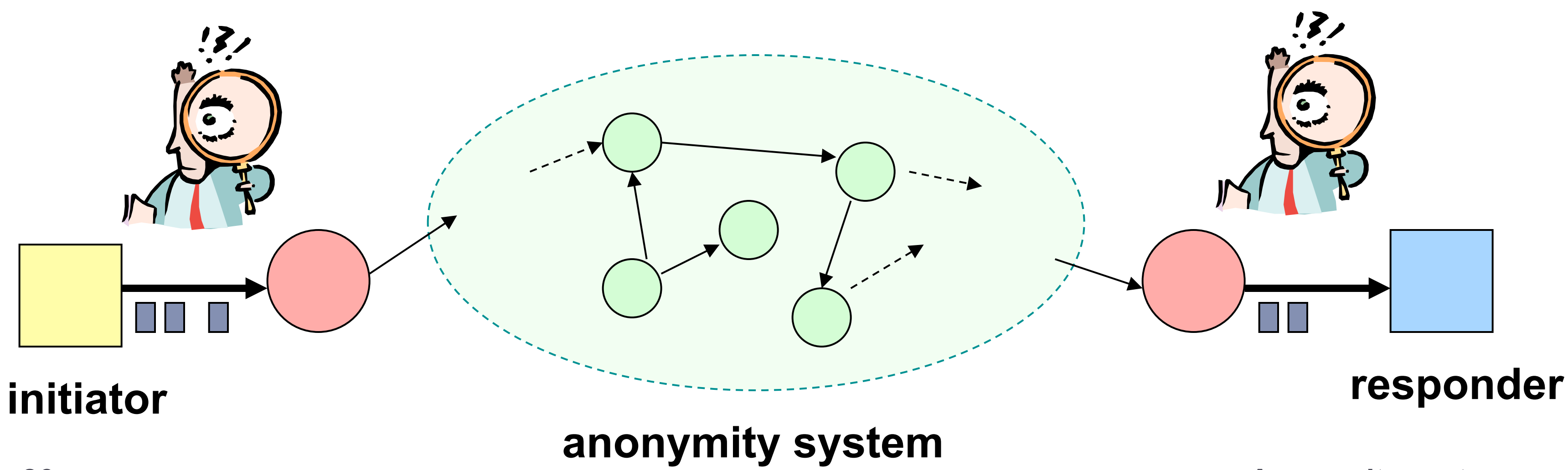
Partition attack on client knowledge

- Render inconsistent views of anonymity system on clients
  - e.g. member list on directory server
- Identify clients who always choose a particular subset of neighbors



Attacks on endpoints

- Sometimes referred as traffic confirmation rather than traffic analysis
- Suppose an adversary controls the first and the last node of a route
- Observe the traffic entering the first node and leaving the last node



Attacks on endpoints (cont.)

- Correlate the timings of a message entering the first node with those coming out of the last node
  - Packet counting attack, Timing attacks, Message frequency attack
- An adversary may be able to:
  - figure out some input message to output message mappings
  - rule out some potential senders or receivers from the anonymity sets
  - link a particular pair of sender and receiver
- An active adversary may increase the chance of success and speedup the analysis by delaying and dropping messages, flooding several nodes and links

More attacks …

- The "Sting" Attack
- The "Send n' Seek" Attack
- Active Attacks Exploiting User Reactions
- Denial of Service Attack
- Social Engineering

- Alternative attack goal:
  - Drive users to less secure anonymity systems or not using anonymity service at all