Administrivia

- No office hours tomorrow (were yesterday)
- Final Exam May 10th
  - Open book, open note
  - “Resurrection” final, can be whole grade if you do well
- May 3rd reading day (no class)
- Review sessions
  - I will dedicate a portion of next lecture to review
  - We will have review sections on May 4th and 5th
  - I will schedule extra office hours that week, too
Crowds: Anonymity for web browsing

- Why not just use mix-nets or dc-nets?
  - Need low-latency responses
  - Too much computational overhead
  - Can get away with weaker adversarial model

- Idea: Introduce new crowds paradigm
Anonymity goals

- **Sender anonymity**
  - Don’t know who sent a message
  - Might see servers receiving messages

- **Receiver anonymity**
  - Don’t know who is receiving messages

- **Unlinkability of sender and receiver**
  - Might know which clients are talking and which servers
  - Don’t know which client is talking to which server
Levels of security

- **Absolute privacy**
- **Beyond suspicion**
  - Sender/originator completely uncorrelated
- **Probable innocence**
  - Originator is not sender with at least 50% probability
- **Possible innocence**
  - Sender has non-trivial probability of not being originator
- **Exposed / Provably exposed**
Adversarial model

• Local eavesdropper
  - Observes all traffic to/from a user’s machine

• Collaborating crowd members
  - Can deviate from protocol and share info to expose users

• The end server
  - A web server trying to figure out identity of users
Crowds architecture

Crowd

Web Servers

1

2

3

4

5

6

3

5

1

2

4
Implementation

- Each client machine runs a *jondo* process
  - Acts as a web proxy
  - All jondos know about each other

- Jondos forward requests to each other
  - First request from browser gets forwarded to random jondo
  - At subsequent hops, gets forwarded with prob $p_f$
Paths

• System uses static paths between clients and servers
  - Chosen on first access to server by client
  - Same for all subsequent requests
  - Changes only when new jondos join, or jondo dies
  - Why static paths?

• What happens if bad jondo dies to force new path creation?

• Last jondo parses HTML, prefetches images...why?
## Anonymity properties

<table>
<thead>
<tr>
<th>Attacker</th>
<th>Sender anonymity</th>
<th>Receiver anonymity</th>
</tr>
</thead>
<tbody>
<tr>
<td>local eavesdrp</td>
<td>exposed</td>
<td>$(n \to \infty)$ beyond susp</td>
</tr>
<tr>
<td>$c$ bad jondos</td>
<td>prob innocence</td>
<td>$(n \to \infty)$ abs priv</td>
</tr>
<tr>
<td>server</td>
<td>beyond susp</td>
<td>N/A</td>
</tr>
<tr>
<td></td>
<td>$(n \to \infty)$ abs priv</td>
<td></td>
</tr>
</tbody>
</table>

- $(n \to \infty)$: As $n$ approaches infinity.
- abs priv: Absolute privacy.
- N/A: Not applicable.
Stretch break
Nym.alias.net pseudonym service

- People establish email pseudonyms or *nyms*
  - e.g., incognito@nym.alias.net

- *Nyms function like regular email addresses*
  - People can send and receive mail under a nym

- *Nym owners are anonymous*
  - Identities unknown even to the nym.alias.net administrators
In real use outside of CS research

- In public use since June, ’96
- Requires only PGP to use
- Software clients for DOS/Windows/Unix
- At peak, had 2,000–3,000 active nyms
- Provides auxiliary services:
  - Anonymous remailer, mail-to-news gateway
Nym.alias.net design

• Built on existing anonymous remailer network
  - Independently operated remailers span several countries
  - Remailers used as a watered-down *mix-net*
  - Multiple nodes must be compromised to expose a nym user

• Deployment favored over privacy
  - Use of PGP, existing remailers hurt security
  - Much subsequent work since ’81 could not be used
  - Our experience nonetheless relevant to future systems
    - No known technical attacks have occurred on privacy
    - Privacy still unbreakable by us, the administrators
Implementation details

- Independent remailers each have a public key
- Every nym has a *reply block*
- Reply blocks route mail through remailers

\[
\text{reply block 1, message 1} \xrightarrow{\text{remailer}} \text{reply block 2, message 2}
\]

- Example reply block: \(\{K, \text{user}\}_{K_2}\)
  - Encrypted with \(K_2\), only remailer2 can decrypt
  - Remailer2 encrypts message with \(K\), forwards it to user
  - Without \(K_2^{-1}\), reply block hides identity of user
Path of mail received by pseudonyms

To: nym
message

To: reemailer1
{K, reemailer2, {K, user}}
{message}

To: reemailer2
{K, user}
{message}

To: user
{message}

• How much privacy against various attackers?
People use nym.alis.net…

- Under oppressive governments
- When risking embarrassment, harassment, job loss
  - discussing alcoholism, depression, being a sexual minority
  - whistle-blowing, fighting harmful cults
- To keep correspondents out of mail log files
- To prevent every public statement from staying with them for life
  - For admittedly marginal purposes
    - marijuana cultivation, virus development, piracy
Anonymous services draw attacks

• Anonymous speech can upset people
  - Used to express unpopular opinions
  - Used to criticize people who respond vindictively
  - Used to denounce powerful organizations/governments
  - Abused to harass people, escaping responsibility

• Anger redirected against anonymity provider
Two threats to anonymous services

• Attempts to expose users’ identities
  - Eavesdropping, flooding, traffic analysis
  - Corrupt/compromised servers

• Attempts to silence users or shut down service
  - Attack anonymous server
  - Attack someone else with anonymous server
  - Marginalize service so everyone ignores it
  - Make life intolerable for someone who can close server
Defense poses unusual challenges

- Privacy concerns preclude usage logs
- Service designed to hide identities (abusers too)
- Attackers cannot be banned even when known
- Filtering or content-based censorship impractical
  - Human adversaries adapt to filtering
  - Content filters may block legitimate use (goal of attack)
  - Too much human effort to review all messages
  - Filtering exposes service providers to liability
Types of attack

- **Conventional attacks (as with non-anon. servers)**
  - Lack of logs may complicate defense
  - Find alternate places to record information
  - Short-term records may be permissible where logs not

- **Content-based attacks**
  - Help recipients ignore unwanted anonymous traffic
  - Keep what’s ignored secret from attacker

- **Overloading attacks**
  - Ignoring not sufficient when many resources at stake
  - Overloading can cost attacker his anonymity
  - Push cost back onto attacker
Harassment

- Unwanted offensive/threatening anonymous mail
- **Solution: destination blocking**
  - Drop messages to those who don’t want anonymous mail

- **Automatic destination blocking process**
  - User sends mail to dstblk-request@nym.alias.net
  - System requests confirmation (nonce in return address)
  - Mailing list blocking requires consent of list owner
    (check that owner-address, etc. bounce, first)

- Keep destination block list secret from senders
Mail-bomb

- One person can overload the system with mail
- **Solution: custom mail server throttles attack**
  - Short-term history sufficient to detect attack
  - Temporary SMTP error codes, SYN filtering delay mail
  - Exploited mail relays fill up, can use logs to deal with it
  - Direct clients use many PCBs, but no connections
Reverse mail-bomb

- *help@nym.alias.net* replies to mail with help file
- Attacker flooded *help* with forged mail
- Hard to track down perpetrator without mail logs
- Any logs might seriously hurt privacy
  - Many help requests presumably not anonymous
  - Likely correlation between help requests & new nyms
- Solution: return sender information with help file
Spam-baiting 1

- Software allowed poor news forgeries
- Posts to some newsgroups precipitate spam mail
  - *misc.entrepreneurs, biz.mlm, alt.sex.erotica.marketplace, …*
- Attacker forged articles to these newsgroups
  - Forgery victims flooded with spam
- Anon. mail incited victims to attack remailers
- Solution: prevent forgeries
Spam-baiting 2

- Attacker posted lists of email addresses
- Victims demanded we filter against their addresses
  - Would conveniently block anonymous followups
- Censorship failed
- Solution: post more bad addresses than good ones
Problem: creating many accounts

- Someone started creating many accounts
  - In the extreme, could run the server out of files
  - Could circumvent per-account traffic limits for bulk email

- First solution: require account confirmation

- If problem recurs...
  - PGP key generation requires both CPU and human time
  - All accounts created had same PGP key (prohibit this)

- If attacker hacks PGP key generator...
  - Increase CPU cost (e.g. hashcash [Back])
  - Increase human cost (e.g. GIF to ASCII challenge)
Defending anonymous systems

- **Conventional attacks (as with non-anon. servers)**
  - Lack of logs may complicate defense—keep info elsewhere
  - Short-term records may be permissible where logs not

- **Overloading attacks**
  - Make overloading cost the attacker his anonymity
  - Push cost back onto attacker
  - Put the human in the loop

- **Content-based attacks**
  - Let people easily ignore anonymously published content
  - Never store and serve objectionable content
Lessons learned

• **Factor abuse into design of anonymous services**
  - People will get angry at existence of service
  - People will exploit the system to attack itself

• **The most precious resource is often human time**
  - Attackers can win by forcing human service operator to “clean up the mess”

• **Avoid storing and serving objectionable content**
  - But *don’t* have humans categorizing published data
Putting the lessons to use

- **Tangler [Waldman]:** A censorship-resistant publishing system

- **Goals of a censorship-resistant publishing system**
  - Let anyone pseudonymously publish, update documents
  - Make it impractical to suppress published information
  - Harden the system against attackers who abuse it
Tangler overview

• **Architecture:** World-wide server network
  - Assume ~ 20 Tangler servers around the world
  - Run by volunteers, as with anonymous remailers
  - System highly tolerant of server failures

• **Published documents broken into blocks**
  - Agree upon mapping from each block to several servers
  - Publisher stores blocks on appropriate servers
  - Clients fetch blocks to reconstruct documents
Challenges

- Providing anonymity to publishers
- Flooding attacks – consume all storage on system
- Malicious servers
  - Drop blocks from a particular document
  - Replace blocks with “redacted” versions
- Publisher must be able to update documents
- Objectionable/illegal content
  - Anthrax recipes, libel, decss, “abortionist” home addresses, …
  - Didn’t I just say not to store and serve this stuff?
  - But if some mechanism allowed us to suppress it, cults would compel us also to suppress documents that expose them
Dealing with objectionable content

- **Dissociate blocks served from documents published**
  - No server should serve blocks of objectionable documents

- **Dissociate servers from blocks served**
  - Blocks should migrate regularly between servers
  - By the time someone takes action against a server, its blocks should have moved somewhere else

- **Dissociate block→server assignment from server**
  - A server cannot chose which blocks to store and serve
  - Misbehaving servers should be automatically ejected
Published data blocks broken into 4 server blocks
- 2/4 server blocks from previously published documents
- Any server block information-theoretically unrelated to data
- Any server block may be part of multiple data blocks
- No single server block necessary to reconstruct data
Document collections

- Server blocks named by SHA-1 hash

- Data blocks named by 4 server block hashes
  - Can reconstruct data from any 3 matching server blocks

- Collection data structure maps file names to data blocks
  - Metadata much like in SFSRO
  - Collection broken into blocks, themselves entangled
  - Root block of collection digitally signed w. version number
  - Only owner of collection key can update contents
Block-to-server mapping

• Every server is assigned a number of IDs
  - Each server has known pub. key, $K$, and capacity $N$ GB
  - If $d$ is number of days since Jan 1, 1970, server’s IDs are:
    \[ H(K, \lfloor dN/14 \rfloor - N), \ldots, H(K, \lfloor dN/14 \rfloor - 1), H(K, \lfloor dN/14 \rfloor) \]
  - $1/14$ of IDs change every day, all points change in 2 weeks

• Map server IDs and block hashes onto circle
  - Store blocks at successor servers around circle [KLLLLLP97]
  - Minimizes incorrect placement when servers join/leave
Block lookup

servers:
- A
- B
- C

block 011001
Ingress control

• Each server can consume space proportional to its capacity
  - If server $A$ has capacity $C_A$, $B$ has $C_B$, and total capacity is $C$, then $A$ can store $C_A C_B / C$ blocks on $B$.
  - Servers themselves can use a variety of ingress-control techniques (as with remailers)

• Space is committed through blind storage credits
  - $A$ constructs message $m =$ “I agree to store block $H(x)$”
  - $B$ digitally signs $m$ blindly (cannot see message signed)
  - $A$ later anonymously ships $\langle m, \text{sig}, x \rangle$ to $B$
  - $B$ digitally signs receipt: “I have received $H(x)$”
  - At end of day, $B$ signs commitment of all blocks accepted
Malicious servers

• **Goal:** Easily prove misbehavior to eject servers

• **Some attacks cause server to contradict itself**
  - Server signs receipt for \( x \), but \( x \) not in commitment
  - Server requests too many credits

• **Otherwise can relay any request through a witness**
  - Server signed commitment but cannot produce block
  - Server signed credit but won’t issue receipt

• **Entanglement maximizes chances of catching bad servers**
  - Retrieves random blocks from commitment—audit
  - Causes people to care about each other’s server blocks
Infranet: Circumventing web censorship

- **Want to circumvent censors**
  - E.g., Great firewall of China, Singapore, etc.

- **Could just use SSL to special anonymizing proxy**
  - But censor would know you are up to something

- **Goal: Censor shouldn’t be able to detect use**
  - Normal web server and client cooperate to create covert channel
  - Client doesn’t get in trouble
  - Server doesn’t face (legal, economic) sanctions increases participation & robustness
Idea: Hide data in legitimate traffic

- Censors allow certain traffic
- Exploit allowed channels to construct a covert channel
  - Talk to normal servers
  - Embed requests for censored content in normal-seeming requests
  - Receive censored content hidden in normal-seeming responses
Infranet architecture

- Talk to Infranet Requester on local machine
- E.g., Requester gets request for cnn.com
- Gets data for cnn.com by sending innocent-looking requests to www.nyu.edu
Downloading contents surreptitiously

- **Use steganography**
  - E.g., embed cnn.com web page into other content

- **Example: Manipulate images**
  - E.g., spread message over least-significant bits of pixels
  - Encrypt message first, so not detectable
  - Shouldn’t visibly impact image

- **Problem: Image will change each download**
  - Solution: Make it look like a web cam!
Steganography Example

- One of these images has embedded content
Sending requests upstream

- HTTP requests generally small (just ask for URL)
  - Same steganography trick won’t work upstream

- Idea: Embed desired URL (cnn.com) in sequence of requests
  - E.g., Say each page of www.nyu.edu has $k$ links
  - Encode cnn.com with an alphabet of $k$ symbols
  - For each symbol, request the $k$th link on each page

- Also encrypt requested URL, so censor can’t detect
Issues

- Encoding encrypted URL will look like random browsing

- Most browsing not actually random
  - Expect some URLs to be more popular than others
  - Browsing also depends on history (e.g., more likely to click on unread page)

- Moreover, encoding URL will take a lot of symbols
  - Means large number of cover HTTP requests for one real request
Idea: Have server send back dictionary

- **Server knows which destinations are most popular**
  - Could say:
    - “link 1 means http://www.cnn.com”
    - “link 2 means http://bbc.co.uk”

- **But can’t list all URLs, so play “20 questions”**
  - Server offers lexicographical ranges or URLs
  - Skews probabilities so that popular URLs selected faster
  - Also ensures web requests don’t look random