Anonymity on the Internet

Matt Edman
edmanm2@cs.rpi.edu

CSCI-4220 Network Programming
Rensselaer Polytechnic Institute

1 December 2006
Outline

- Introduction
  - What is anonymity and why should I care?
- High-latency Designs
  - e-mail
- Low-latency Designs
  - www, ssh, irc, etc.
- Research Problems
  - Hard problems that still need to be solved
What is Anonymity?

**Definition**
Anonymity is the state of being not identifiable within a set of subjects, the anonymity set. – Pfitzmann and Hansen
Who Needs Anonymity?

Political Dissidents

- Bloggers in China and Iran
- Critical ex-KGB agents in Russia

Intelligence Agencies

- Intelligence gathering
- Agents in hostile territory
Who Needs Anonymity?

**Law Enforcement**
- Tip lines
- Whistleblowers

**Corporations**
- Competitive analysis
- Protection for employees
Who Needs Anonymity?

You

- Private web browsing and e-mail

Criminals and Spammers

- They already have their anonymity!
Types of Anonymity

Channel Anonymity
Communicant identification should not be a property of the channel

Data Anonymity
Message contents shouldn’t give away your identity
  ▶ Requires application-level scrubbing
Anonymous from Whom?

Outside observer/Network insider
Requires channel anonymity

Sender/Recipient Anonymity
Requires channel & data anonymity
Distributed Trust

“Anonymity Loves Company”

- You can’t be anonymous by yourself
- You can’t trust your anonymity to a single entity

Distributed trust

- Diversity and dispersal of security requirements
- Everyone whose anonymity is at stake should run a piece of the infrastructure.
Adversaries

Properties of Adversaries

- Internal or External
- Passive or Active
- Local or Global
- Static or Adaptive
High-latency Anonymity

“Message-based”
Mainly used for e-mail
Basic building block is a “mix” (Chaum ’81)
A Simple Mix

\[ C_i = E_{KU_{Mix}}[M_j] \quad M_j = D_{KR_{Mix}}[C_i] \]

Mix decrypts and randomly permutes messages.
A Simple Mix

\[ C_i = E_{KU_{Mix}}[M_j] \quad \quad M_j = D_{KR_{Mix}}[C_i] \]

An observer does not know the true mapping of \( C_i \rightarrow M_j \).
Mix Networks

Cascade Topology

\[ C_i = E_{KU_X}[E_{KU_Y}[E_{KU_Z}[M_j]]] \]

Only need 1 non-compromised/honest mix to be secure
Mix Networks

An Alternate Topology

C1 → X → Y → M1
C2 → X → Z → M2
C3 → Z → M3

Matt Edman edmanm2@cs.rpi.edu

Anonymity on the Internet
Mixing Algorithms

Timed mix
Flush the mix every $t$ seconds

Threshold mix
Flush the mix every $n$ messages

Threshold or timed mix
Flush the mix every $t$ seconds or when $n$ messages accumulate in the mix

Threshold and timed mix
Flush the mix every $t$ seconds but only when at least $n$ messages have accumulated in the mix
Mixing Algorithms

Pool mix
Every time the mix fires, randomly select some messages to send and retain $f_{min}$ in the mix

- Threshold pool mix
- Timed pool mix

Timed dynamic pool mix
Every time the mix fires, send $\text{max}(1, \lfloor m \times \frac{1}{m} \rfloor)$ messages, where $0 \leq \frac{1}{m} \leq 1$, and retain the rest in the mix
Active Attacks

Trickle attack (Threshold mixes)

“n – 1 attack”

1. Block all incoming messages to a mix, including one particular message of interest, \( M \)
2. Send dummy messages into the mix until it fires
3. Allow \( M \) into the mix
4. Send dummy messages into the mix until it fires again
5. Observe the outputs to identify \( M \)
Active Attacks

Flooding attack (Timed mixes)

1. Block all incoming messages to a mix, including one particular message of interest, $M$

2. Wait until the mix fires

3. Allow $M$ into the mix

4. Wait until the mix fires again

5. Observe the single output to identify $M$
Passive Attacks

Intersection Attack
Learn over time which Bobs are more likely to receive after certain Alices send

Partitioning Attack
Split large groups of Alices and Bobs into smaller groups (i.e., reduce the size of the anonymity set)
Deployed Remailers

**anom penet fi (Type 0)**

- Mapped pseudonym to identity
- Legal attacks by the Church of Scientology (1995)

**Cypherpunks remailer (Type I)**

- Little or no delaying
- Doesn’t hide message length
- Vulnerable to replay attacks
Deployed Remailers

Mixmaster (Type II)
- Added pools and message padding
- Keeps hashes of the headers for replay prevention

Mixminion (Type III)
- Added reply blocks
- Key rotation to counter replay attacks
Low-latency Anonymity

- Connection-oriented
- More suited for real-time applications (www, ssh, irc, etc.)
- Basic building block is a proxy
One-hop Proxies

- Single point of failure and attack
- Single point of trust
- Example: anonymizer.com
Early Onion Routing

Onion Routing

- Initially developed at NRL
- Uses slower asymmetric crypto to set up “circuits”
- Uses faster symmetric crypto for moving data
Building a Circuit

- Alice picks $X$, $Y$, $Z$ nodes from the network
Building a Circuit

Construct an Onion

- Alice generates symmetric keys for each hop
- Each layer is encrypted with the node’s public key

(D. Goldschlag et al., “Hiding Routing Information”, IH '96)
Building a Circuit

- Alice sends the onion to $X$
- $X$ decrypts the outermost layer, learns its symmetric keys and the next hop
Building a Circuit

- X sends the onion to Y
- Y decrypts the outermost layer, learns its symmetric keys and the next hop
Building a Circuit

- \( Y \) sends the onion to \( Z \)
- \( Z \) decrypts the outermost layer, learns its symmetric keys and the next hop
Building a Circuit

- Z establishes a connection to rpi.edu
Sending a Message

Alice sends $C = E_{K_{fx}} [E_{K_{fy}} [E_{K_{fz}} [M]]]$. 

Matt Edman edmanm2@cs.rpi.edu

Anonymity on the Internet
Sending a Message

\[ X \text{ sends } C' = D_{K_{fx}}[C] = E_{K_{fy}}[E_{K_{fz}}[M]] \]
Y sends $C'' = D_{K_{fy}}[C'] = E_{K_{fz}}[M]$
Sending a Message

Z sends $M = D_{K_{fz}}[C'']$
Problems

- Nodes have to remember what onions they have seen to prevent replay attacks
- Was never really widely used
- Patented
Tor (not TOR)

The Onion Router

- “Second Generation Onion Routing”
- Open source (3-clause BSD)
- Currently 800+ volunteer servers
- Tens/hundreds of thousands of users?
Threat Model

- Assume end-to-end attacks are trivial
  - No mixing, no padding
- Assume an adversary can monitor or manipulate traffic on part of the network
- Assume an adversary owns some of the nodes

Design and deploy for usability
Building a Circuit

**Alice** (link is TLS-encrypted)  **OR 1** (link is TLS-encrypted)  **OR 2** (unencrypted)  **website**

- Create c1, E(g^x1)
- Created c1, g^y1, H(K1)
- Relay c1{Extend, OR2, E(g^x2)}
- Relay c1{Extended, g^y2, H(K2)}
- Relay c1{[Begin <website>:80]}
- Relay c1{[Connected]}
- Relay c1{[Data, "HTTP GET..."]}
- Relay c1{[Data, (response)]}

- Create c2, E(g^x2)
- Created c2, g^y2, H(K2)
- Relay c2{Begin <website>:80}
- Relay c2{Connected}
- Relay c2{Data, "HTTP GET..."}
- Relay c2{Data, (response)}

- Legend:
  - E(x)--RSA encryption
  - {X}--AES encryption
  - cN--a circID

- (TCP handshake)
- "HTTP GET..."
- (response)

Negotiate a session key at each hop, ephemeral Diffie-Hellman w/ RSA (Tor design, USENIX Security '04)
Directory Servers

- Lets clients learn about servers
- Directories signed by authoritative directory servers, mirrored by other servers
- Fingerprints of authoritative dirserver public keys hard-coded into source
- Trust bottleneck
Hidden Services

- Provides a service (e.g., web server) without revealing its IP address
- Uses a special .onion TLD, accessible only with a Tor client
- Addresses are based on a the hash of the hidden service’s public key (e.g., 6sxoyfb3h2nvok2d.onion)
- Client (server) doesn’t know who/where the server (client) is
- Allows you to offer a service behind a firewall with no open ports
Hidden Services

The hidden service picks *introduction points* and publishes them
Hidden Services

Alice learns about the hidden service's intro. points.
Alice randomly chooses a *rendezvous point*, \( RP \), and gives it a “rendezvous cookie”, \( RC \).
Hidden Services

Alice connects to one of the hidden service’s intro. points and gives it $E_{KU_{HS}}[RP|RC|g^x]$
The hidden service connects to $RP$, presents $RC$, $RP$ pairs up the circuits, gives Alice $g^y$ and $SHA1[g^{xy}]$. 
Alice and the hidden service now share an encrypted tunnel.
An Attack on Hidden Services

(L. Øverlier and P. Syverson, “Locating Hidden Servers”, IEEE Symposium on Security and Privacy ’06)
Other Designs

Tarzan/Morphmix

- Fully p2p
- You lose if you pick a bad first node
- Morphmix added “collusion resistance”

JAP

- Java Anonymous Proxy
- Uses cascades instead of free-routes
- Backdoored by court order (2003)
  - Successfully fought in court
Other Designs

- **Freedom Network**
  - Zero-Knowledge Systems
  - Commercial attempt at an anonymity network
  - Similar to onion routing
  - Bankrupt
Attacks on Low-Latency Anonymity

Packet counting/timing attacks
Watch inputs to and outputs from the network, looking for timing and volume correlations between inputs and outputs

Website fingerprinting
Build “fingerprints” of target websites through the anonymity system and compare to observed client traffic
Research Problems

Scalability

- Non-clique network topology
- De-centralized directory
- Avoiding a bad person running 10,000 servers (Sybil attack) without a human
End-to-end Attacks

- Avoiding some end-to-end attackers (e.g., ISP-level adversaries)
- How much mixing or padding do we need for it to be useful without killing performance? (Mid-latency?)
- Blending high-latency traffic with low-latency traffic
Research Problems

Blocking Resistance

- “The China Problem”
- Let the good guys know about nodes in the network without the bad guys finding out too
- Hide the fact that we’re speaking some anonymity network’s protocol

Usability and Incentives

- Make the software easier to use (securely)
- Convince people to run useful servers
Questions

Questions?

See http://freehaven.net/anonbib/ for lots of neat papers