Privacy on Public Networks

- Internet is designed as a public network
  - Wi-Fi access points, network routers see all traffic that passes through them
- Routing information is public
  - IP packet headers identify source and destination
  - Even a passive observer can easily figure out who is talking to whom
- Encryption does not hide identities
  - Encryption hides payload, but not routing information
  - Even IP-level encryption (tunnel-mode IPsec/ESP) reveals IP addresses of IPsec gateways

Anonymity

- Anonymity = the person is not identifiable within a set of subjects
  - You cannot be anonymous by yourself!
    - Big difference between anonymity and confidentiality
  - Hide your activities among others’ similar activities
- Unlinkability of action and identity
  - For example, sender and his email are no more related after adversary’s observations than they were before
- Unobservability (hard to achieve)
  - Adversary can’t even tell whether someone is using a particular system and/or protocol

Attacks on Anonymity

- Passive traffic analysis
  - Infer from network traffic who is talking to whom
- Active traffic analysis
  - Inject packets or put a timing signature on packet flow
- Compromise of network nodes
  - Attacker may compromise some routers
  - It is not obvious which nodes have been compromised
    - Attack may be passively logging traffic
  - Better not to trust any individual router
    - Can assume that some fraction of routers is good, but don’t know which
Chaum’s Mix

- Early proposal for anonymous email

- Public key crypto + trusted re-mailer (Mix)
  - Untrusted communication medium
  - Public keys used as persistent pseudonyms

- Many modern anonymity systems use Mix as the basic building block

Anonymous Return Addresses

- M includes \( \{K_1, A\}_{pk(mix)} \) \( K_2 \) where \( K_2 \) is a fresh public key

\[ \{r_1, \{r_0, M\}_{pk(B)}\}_{pk(mix)} \]

\[ \{r_0, M\}_{pk(B)} \]

\[ \{r_2, \{r_3, M'\}_{pk(E)}\}_{pk(mix)} \]

\[ \{r_3, M'\}_{pk(E)} \]

\[ \{r_4, \{r_5, M''\}_{pk(B)}\} \]

\[ \{r_5, M''\}_{pk(B)} \]

\[ \{r_6, \{r_7, M''\}_{pk(B)}\} \]

\[ \{r_7, M''\}_{pk(B)} \]

Secrecy without authentication (good for an online confession service ☻)

Basic Mix Design

Mix Cascade

- Messages are sent through a sequence of mixes
  - Can also form an arbitrary network of mixes (mixnet)

- Some of the mixes may be controlled by attacker, but even a single good mix guarantees anonymity

- Pad and buffer traffic to foil correlation attacks

Adversary knows all senders and all receivers, but cannot link a sent message with a received message ☻
Randomized Routing

- Hide message source by routing it randomly
  - Popular technique: Crowds, Freenet, Onion routing
- Routers don't know for sure if the apparent source of a message is the true sender or another router

Onion Routing

- Sender chooses a sequence of routers
  - Some may be honest, some controlled by attacker
  - Sender controls the length of the path

Route Establishment

- Routing info for each link encrypted with router’s public key
- Each router learns only the identity of the next router

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
Tor

- Deployed onion routing network
  - http://torproject.org
  - Specifically designed for low-latency anonymous Internet communications
- Running since October 2003
  - Thousands of relay nodes, 100K-500K of users
- Easy-to-use client proxy, integrated Web browser

Tor Circuit Setup (1)

- Client proxy establish a symmetric session key and circuit with relay node #1

Tor Circuit Setup (2)

- Client proxy extends the circuit by establishing a symmetric session key with relay node #2
  - Tunnel through relay node #1 - don't need 🙄!
Tor Circuit Setup (3)

- Client proxy extends the circuit by establishing a symmetric session key with relay node #3
  - Tunnel through relay nodes #1 and #2

Using a Tor Circuit

- Client applications connect and communicate over the established Tor circuit
  - Datagrams decrypted and re-encrypted at each link

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

Hidden Services

- Goal: deploy a server on the Internet that anyone can connect to without knowing where it is or who runs it
- Accessible from anywhere
- Resistant to censorship, denial of service, physical attack
  - Network address of the server is hidden, thus can’t find the physical server
Creating a Location Hidden Server

Server creates onion routes to "introduction points"

Server gives intro points' descriptors and addresses to service lookup directory

Client obtains service descriptor and intro point address from directory

Using a Location Hidden Server

Client creates a route to a "rendezvous point"

Client sends the address of the rendezvous point and any authorization, if needed, to the server through an intro point

If server chooses to talk to client, connect to rendezvous point

Server gives intro points' descriptors and addresses to service lookup directory

Rendezvous point mates the circuits from client & server