CS 1699
Privacy in the Electronic Society

William Garrison
bill@cs.pitt.edu
6311 Sennott Square
http://cs.pitt.edu/~bill/1699

20: Anonymous networking
Today: Anonymous routing/networking

Briefly, what is revealed by knowing who is communicating with whom?
  • This is the idea behind flow confidentiality

As with location privacy, flow confidentiality can be addressed using mix nets
  • Mix zones were physical, mix nets are logical

We’ll discuss several techniques to provide anonymity in networking
  • Crowds: Obscure initiator by randomly passing packet between participants
  • Onion routing (Tor): Layers of encryption, one for each node that passes the packet
  • Hordes: Multicast to send response to many nodes at once, non-recipients ignore
  • GAP: Aimed at filesharing; individual nodes can make efficiency tradeoffs without negatively impacting other nodes (efficiency or anonymity)
What are the privacy implications of a lack of flow confidentiality?

Why might we care that an adversary learns who we’re communicating with?
Can we address this with a VPN/proxy?

Alice passes her traffic to the **proxy**
- Proxy passes on to server, reflects responses back to Alice

An eavesdropper will never see a packet identifying both Alice and S
- Instead, traffic $A \rightarrow P$ and $P \rightarrow S$

What does the proxy see?
- How much do we need to trust the proxy?
- Is Alice anonymous?
- Is Alice’s traffic “hiding” among other traffic?
How to blend in with the crowd

Crowds (Reiter and Rubin, 1998) helps us reduce the trust in a single proxy, and allows Alice’s traffic to hide among others’

Two types of nodes

- Jondos: Clients wishing to hide in the crowd
- Blenders: Trackers maintaining a list of active jondos
- When a new jondo registers, a blender assigns a symmetric key and notifies other jondos

When sending a message, a random path is constructed

- When receiving a message, either:
  - Forward it to a random other jondo (probability $p_f > 1/2$)
  - Send it to its destination (probability $1 - p_f$)
Paths through the crowd

Crowd

Web Servers

1

2

3

4

5

6

3

5

1

2

6

4
Can we quantify the privacy offered by Crowds?

What does a passive eavesdropper see?
- Packets between jondos
- Might observe Alice sending a message in absence of incoming packet
- Alice sends message to destination with probability $1/n$

What does the recipient see?
- Incoming packet from crowd, could equally be any member

What do other jondos learn?
- Cannot tell if Alice is the initiator or just another jondo
- What if they collude? Is Alice safe?
  - Yes, if:

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

*Number of jondos in crowd*  
*Probability of forwarding*  
*Number of colluding jondos*
The predecessor attack tracks nodes over the creation of many paths

Wright et al. (2004) discussed an attack to weaken anonymity
- Specifically for long-lived communications

Insight: When a message’s path is created, the initiator is necessarily along the path
- Attacker participates in many messages of the same nature (what might this mean?)
- Since initiator is along the path, predecessor is more likely to be initiator than any other
- By observing most common predecessors, attacker learns who is likely to have sent the messages

When will this attack succeed or fail?
- How can we mitigate?
Onion routing (Tor) takes advantage of encryption layers to increase privacy.

In Tor, Alice chooses her path before sending.
Tor message: Multiple layers of encryption

To: B
Encrypted with $K_B$

To: C
Encrypted with $K_C$

To: E
Encrypted with $K_E$

To: example.com
Message: HTTP
GET /

Diagram: 4 concentric circles, each encrypted with different keys.
How does Alice send an onion message?
How much privacy does this give to Alice?

Passive eavesdropper:
- Onion packet from Alice to an onion router
  - Cannot hide her participation in Tor (also true for Crowds)
  - May know that Alice is initiator, if Alice is not an onion router

Endpoint:
- Sees an incoming packet from a Tor exit node
  - Knows that the requester is using Tor

Onion routers:
- First sees that the message is from Alice, knows the next router
- Exit node sees the intended endpoint, and the previous router
- In between, nothing except previous/next routers
Additional threats to anonymity in Tor?

Global passive adversary
- Can observe all traffic between all nodes
- Who might this be?
- By observing packet timing, may be able to statistically determine who is talking to whom
- How can we prevent this?

Control of exit node
- Exit node sees the plaintext message (if not using encryption atop Tor)
- What if this packet is identifying?
- Intuition behind Bad Apple attack: Application sends real IP in onion message, can be identified by exit node (and endpoint)

Onion router collusion
- What if an attacker controls many onion routers?
Hordes attempts to obscure responses using multicast

Idea: Send response packet to many nodes at once
• Ignored as noise by nodes that aren’t the intended recipient

In this way, attacker may determine which multicast group the recipient is in, but not the exact node
• In general, it is even complex to determine who receives a multicast
• No single entity can reveal this, requires coordination between routers

Nonetheless, Hordes is still vulnerable to Predecessor attack
• This attack takes place on the forward path, not the response path
GAP is the protocol that provides anonymity in GNUnet

GNUnet: Decentralized, peer-to-peer networking
  • Primarily for file sharing

GAP has two message types
  • Query: A request for a file by hash value
    • Also specifies where the reply should be sent
    • Recipient may be the original requester or a node asking on her behalf
    • If no reply within a (randomized) time period, re-send (no guarantees)
  • Reply: The response containing the file content

Insight: GAP should allow nodes to make efficiency trade-offs without affecting other nodes’ privacy
GAP nodes have options when receiving a packet

When receiving a packet, first determine if resources are available to be a part of the relay chain
  • CPU, bandwidth, routing table...

If resource is cached by this node, enqueue a response
  • ... to whomever was listed as the recipient—may not be original requester

Otherwise, forward the query to other nodes
  • May decide to indirect: replace the requester identifier with their own
    • Save current requester in the routing table to forward response later
  • May decide to forward: leave requester alone, don’t be involved in response
    • For instance, if routing table is full or bandwidth is limited
  • When receiving a reply, check routing table for matching query
    • And cache commonly-requested files
What is the result of a node choosing not to relay the response?

Suppose that A sends a query to B, and B sends it to C without replacing A’s return address

- e.g., forwards it rather than indirecting it
- Is B limiting A’s privacy?

What does C learn?

- Cannot be certain that A is originator and hadn’t indicated it herself
- B’s anonymity depends on how many packets she indirects for others
- Note that C can assume that B was not the originator of the message
  - i.e., B has potentially harmed her own privacy by reducing the number of packets among which her sensitive data can hide!
- B’s efficiency trade-off has increased C’s chances of guessing which traffic is from B
So, which should we use?
Conclusions

Anonymity is complex and difficult to achieve in networking

Main idea: Hide in a crowd
- Crowds: Random paths
- Onion routing: Predetermined paths, encryption layers
- Hordes: Multicast for responses
- GAP: Allow nodes to choose to forward rather than indirect

Predecessor attack is one possible way participants can learn about each other
- Global passive adversary also hard to mitigate