Using the Internet Pseudonymously

Ian Goldberg

Zero-Knowledge Systems

ian@zeroknowledge.com

19 January, 2000

RSA 2000 Conference

The Problem

- Protecting whistleblowers from companies
- Protecting dissidents from governments
- “Insider” trading based on traffic analysis
- Preventing the “dossier effect”
  - H.R. people using DejaNews
- Can’t use electronic cash over today’s Internet
  - So what if the ecash is anonymous? Your net connection isn’t.
- Echelon is scary

What to do?

- Try to avoid linking all of your actions to your **True Name**.
  - You only get one, and it's hard to clean when tarnished.
  - Regardless of your wishes (or local laws), everything that *can* be linked together, *will* be.
  - Credit records are a good example of this going wrong.
- What alternatives are there?
  - There are different levels of "nymity" (revelation of identity) that are used in different transactions.

The Nymity Slider

- Complete Nymity
  - e.g. real estate transactions
- Persistent Pseudonymity
  - e.g. noms de plume
- Linkable Anonymity
  - e.g. prepaid phone cards
- Complete Anonymity
  - e.g. paying with cash

$\Rightarrow$ Moving “up” the slider gives more persistance and more linkability.

The Nymity Slider

- An important property of the Nymity Slider is that it's **easy** to move *up* and **hard** to move *down*.
  - I can always tell you my name when I pay with cash, but I can't remove my name from my credit card.
  - Maybe the "Nymity Ratchet"?
- Choose where we want it to end up, but design it *below* that point.
  - This lets us easily change our minds later.
  - The Internet (IP) got this *wrong*.

Goal

- Build a Pseudonymous IP (PIP) Network
  - Persistent Peudonymity is a nice place to be on the Nymity Slider:
    - You can have multiple pseudonyms, for separating the different parts of your life.
    - But each pseudonym can gain *reputation capital*.
  - Allows the use of the Internet without revealing True Names (or IP or email addresses, which are also high on the Nymity Slider).

PIP Network Design Criteria

- Deployable over the existing Internet
  - It would be easier if we could start from scratch.
    - It's hard to move down the Nymity Slider.
  - But we'd like to be able to "get there from here".
  - Also, we don't want to "compete" with the Internet.
    - Don't force a user to choose between signing up with an ISP or a "PIPSP".
    - The user should connect to the PIP Network via the regular Internet.

PIP Network Design Criteria

- Applicable and Low-latency
  - We want people to use it
    - “Anonymity Loves Company”
  - Must support at least common Internet services
    - WWW, email, Usenet, chat
    - Should also be easily extensible to new services
  - Must have minimal impact on end-to-end performance
    - bandwidth, latency, “view of the net”

PIP Network Design Criteria

- Resistance to Attack
  - Attacking Pseudonyms
    * It should be “hard” to trace origins of a packet or to find the user behind a pseudonym.
  - Attacking the PIP Network
    * It also should be “hard” to attack the infrastructure itself (no single point of failure).

PIP Network Design Methodology

- Remember the rule of the Nymity Slider.
  - Design the nymity of the system *below* where we want it to end up.
- We will actually design an Anonymous IP Infrastructure (AIPI).
- Then we'll build a PIP Network on top of that.
  - The Nymity Slider says this should be easy.

How to Build an AIPI

- The primary (interesting) piece is the "IP Wormhole".
- Other pieces include:
  - Network Information Database (NIDB)
    - Keeps track of the network state
  - Application-level proxies
    - For removing identifying information from HTTP, SMTP, etc.

The IP Wormhole

- The IP Wormhole is an Internet service.
- A client anywhere on the Internet can “inject” an IP packet into the IP Wormhole.
- The client can specify where on the Internet that IP packet (perhaps slightly modified) should reappear.
- The original source address of the IP packet will have been removed.
  - But we still need a way to allow packets to come *back*, of course.

Building an IP Wormhole: Attempt 1

- Turn on IP-in-IP tunneling on a number of hosts around the Internet.
- A client would just encapsulate IP packets (destined for a web server, say) in other packets, destined for one of the tunneling hosts.
- This is basically Mobile IP
  – The “true” client can be anywhere, but always appears to have the IP address of his “home base”.

Building an IP Wormhole: Attempt 1

- We additionally allow clients to select from any of a number of bases.
- These bases we call **Anonymous Internet Proxies** (AIPs).

$\Rightarrow$ Anyone who can read packets coming into the AIP can easily tell the real identity of the client.

Building an IP Wormhole: Attempt 2

- Add cryptography to encrypt and authenticate the connection between the client and the AIP.

$\Rightarrow$ The attacker can still see the encrypted packets arriving at the AIP, and the corresponding decrypted packets leaving the AIP.

$\Rightarrow$ By using correlations of *size*, it is easy to figure out which client is talking to which server.

  – Even if many different clients are using the same AIP to talk to many different servers.

Building an IP Wormhole: Attempt 3

- Add *packet padding*.
  - Each packet is padded to a fixed length before encryption.
  - Remove the correlation between the size of the encrypted and decrypted packets.

$\Rightarrow$ This helps very little; now the attacker uses correlation of *times*.

$\Rightarrow$ The decrypted packet will leave the AIP shortly after the encrypted packet arrives.

  - It doesn't take much data to deduce which clients are talking to which servers.

Building an IP Wormhole: Attempt 4

- Add *link padding*.
  - Send the same *number* of (fixed-size) packets between the client and the AIP in each unit time period.
- This effectively solves the eavesdropper problem.

$\Rightarrow$ But the operator of the AIP still knows which clients are communicating with which servers.

  - He or the AIP being compromised is a single point of failure for the system.

Building an IP Wormhole: Attempt 5

- Use *chaining*.
  - Instead of a single AIP, the client can choose a *chain* of AIPs.
  - Client $\rightarrow$ AIP 1 $\rightarrow$ AIP 2 $\rightarrow \cdots \rightarrow$ AIP n $\rightarrow$ Server
  - Client $\leftarrow$ AIP 1 $\leftarrow$ AIP 2 $\leftarrow \cdots \leftarrow$ AIP n $\leftarrow$ Server
- Concatenate multiple instances of Attempt 4
  - Each AIP in the chain is the “client” for the next one.

$\Rightarrow$ Not quite

  - The IP packet is repeatedly encrypted and decrypted along the chain.

Building an IP Wormhole: Attempt 5

- Compromising the first and last AIPs suffices to defeat the chaining.
  - regardless of the number of “hops”
- Compromising *any one* AIP in the chain will reveal the data in the packet.
  - The plaintext of the data (an email message, for example) is often enough to identify the recipient.
    - So just compromise the first AIP.

Building an IP Wormhole: Attempt 6

- In Attempt 5, we have a number of AIPs in a chain, with secure links between them.
  - But packets are popping in and out of the secure links at each hop.
- Add more secure links: one between the client and each AIP in the chain.
  - These new secure links will be tunneled over the hop-by-hop secure links we had before.

Setting up the IP Wormhole

- We assume there exist AIPs deployed across the Internet, and that there are secure links between them, forming some (not necessarily complete) graph, called the AIP graph.
- A client will be activated somewhere on the Internet and will start up a secure link to its nearest AIP.

Setting up the IP Wormhole

- The client will pick an *exit AIP*; this will be the IP address he will appear to be when he communicates with servers.
  - Sometimes the physical jurisdiction of the exit AIP is important to the client; sometimes it can just be random.
- The client picks a random path through the AIP graph from himself (or his nearest AIP) to the exit AIP.

Setting up the IP Wormhole

- The client (using public-key cryptography) establishes shared secrets between himself and each AIP in the chain.
  - These secrets are then used to build the client-to-AIP secure links.

Sending Packets through the Wormhole

- The client first prepares the IP packet by multiply-encrypting it.
  - Pad the packet to a constant size.
  - Encrypt with the secret shared between the client and the *last* AIP in the chain (the exit AIP).
  - Encrypt with the secret shared between the client and the next-to-last AIP.
  - etc.
  - Encrypt with the secret shared between the client and the first AIP.

Sending Packets through the Wormhole

- The client drops the packet into the Wormhole.
  - The packet will travel through the secure link to the first AIP.
    - It will be encrypted, sent as one of the packets in a constant-rate stream, and decrypted by the first AIP.
  - The outermost layer of encryption will be removed.
- The first AIP will drop the result into the Wormhole.
  - To be delivered to the second AIP via the secure link, as above.

Sending Packets through the Wormhole

- The second AIP will then remove the outer layer of encryption on the packet, and drop it back into the Wormhole.
- etc.
- The last AIP will receive the packet, decrypt it, and (now that it is fully decrypted) send it to its intended destination.

The Security of the Wormhole

- The exit AIP knows:
  - the address of the server
  - the next-to-last AIP in the chain
  - the (unencrypted) data packet
    - so the client must be sure to remove identifying info from it

$\Rightarrow$ But *not*:
  - the address of the client

The Security of the Wormhole

- The first AIP knows:
  - the address of the client
  - the address of the second AIP

$\Rightarrow$ But *not*:

  - the address of the server
  - the contents of the data

- The middle AIPs know:
  - the addresses of the adjacent AIPs

$\Rightarrow$ But *not*:

  - the addresses of the client or server
  - the contents of the data

The Security of the Wormhole

- In order to link a client to a server, an attacker would need to compromise *every* AIP in the chain.
  - As opposed to *any* AIP in the chain (the case with Attempt 5).
- At least, if the attacker cannot delete arbitrary packets from the Internet.
  - The effect of this threat model is part of the research for this work.

Other Pieces of the AIPI

- Network Information Database
  - Keeps track of the AIP Graph.
  - Clients can query it in order to build chains.
- Application-level Proxies
  - Client-side proxies
    - Remove identifying information from application data.
  - Exit AIP proxies
    - Prevent the AIPI from being used maliciously.
    - Enforce the addition of PIP layer (below).

Adding Pseudonymity

- Accoring to the Nymity Slider, it should be easy to add a Pseudonymous IP (PIP) layer on top of the AIPI. It is:
  - Clients choose a unique “nym” (pseudonym) and generate a signature key pair.
  - The nym and the public key are submitted to a CA, who generates a certificate binding the key to the nym.
  - The exit AIP only accepts a client-to-AIP secure link if it has been signed by a valid pseudonym’s key.
    - Almost; there’s the bootstrapping problem of getting a nym in the first place.

Adding Pseudonymity

- The exit AIP also uses application-level proxies to force pseudonymity.
  - For example, all outgoing email must be properly (digitally) signed by the sending nym.
- The exit AIP now knows the *nym* of the client, but *not* his address (or real identity).
  - *No one* knows which nyms are associated with any client, except for the client himself.

Summary

- The Nymity Slider
  - Gives a way to specify how much information about identity is revealed in a system.
- Goal: Build and Measure a Pseudonymous IP Network.
  - There should be an Anonymous IP Infrastructure underneath.
- How to measure success?
  - Commercialize it and see if people use it. :-)
  - `http://jobs.zeroknowledge.com/`