Anonymous Communication

- Techniques to prevent traffic analysis, specifically discovery of source-destination patterns

- Historically important
  - Traffic patterns were very useful in WWII, for both learning about the enemy and throwing them off (with decoy traffic)

- Important today because private content is increasingly being carried by public and private networks
  - Internet telephony
  - Browsing, shopping, content delivery via Internet
  - Pay-per-view movies over cable
Basic Concepts

What do you want to hide?
- Sender anonymity: attacker cannot determine sender
- Receiver anonymity: attacker cannot determine receiver
- Unlinkability: attacker can determine senders and receivers, but cannot determine <sender, receiver> associations

From whom do you want to hide it?
- local eavesdropper (e.g., your employer)
- global eavesdropper (e.g., a government or coalition of governments)
- your communication partner (e.g., a web server)

Uses of an Email Pseudonym Server

nym.alias.net is an email pseudonym server (one of many)
- allows anyone to create an email alias without revealing her identity

Survey of users revealed numerous uses
- To make political statements, to hide their correspondents, and to encrypt email in countries with oppressive governments
- Where alternatives might lead to embarrassment, harassment, prosecution, or loss of job
  - alcoholism, depression, being a sexual minority, whistle-blowing
  - virus development, software piracy, and other illegal uses
- For protection from the unforeseen ramifications of a USENET news posting
- So that statements are judged on their own merit
Simple Proxies

- The most common technology for achieving sender anonymity from communication partner
- Much like network address translation
  - proxy replaces client’s address with its own

```
From: reiter@unc.edu       From: foo@bar.com
To: reiter@unc.edu         To: foo@bar.com
bar.com                   bar.com
```

- Has been implemented for several protocols
  - HTTP, email, ...

The Anonymizer

- Tailored to HTTP (web) traffic
- Hides user's address from web server (sender anonymity)

```
web client 10.42.6.9:1024   anonymizer 192.123.2.5:2028   web server
```

- Challenge: rewriting links in web pages
  - For example, a web page containing
    `<A HREF=http://www.eff.org>`
    is rewritten to
  - Must be done reliably, or anonymity is lost
Weaknesses of The Anonymizer

- Administrator of The Anonymizer knows all
  - Common to all single-proxy solutions
  - Even for TLS-protected connections

- Translating URLs in web page scripts is difficult, if not impossible
  - Failure to translate can expose identity
  - Safest to disable JavaScript

- Cookies must be handled with care

- If browser’s Java security policy permits applet to connect only back to web server, then applet will probably fail

Janus

- Also known as Lucent Personalized Web Assistant (LPWA)
  - No longer operational

- Similar to Anonymizer, but generates pseudonyms, email addresses, and passwords for sites that require accounts

- How it works:
  - Initially the user provides her email address and a password to Janus
  - When at a web site, user types control codes for her account, password, and email address, respectively
  - Janus replaces these codes with pseudonymous ones
An Example Use of Janus

- I give to Janus:
  - email: “reiter@unc.edu”
  - password: “tomato”

- When I visit “www.nytimes.com/subscribe”, I enter
  - subscriber ID: “\U” ← control code for account name
  - password: “\P” ← control code for password
  - email address: “\@” ← control code for email address

An Example Use of Janus (cont.)

- Janus finds “\U”, “\P”, and “\@” and replaces them:
  - “\U” → f(“reiter@unc.edu”, “tomato”, “nytimes.com”)
  - “\P” → g(“reiter@unc.edu”, “tomato”, “nytimes.com”)
  - “\@” → h(“reiter@unc.edu”, “tomato”, “nytimes.com”)

  where f, g, and h are one-way on first two inputs, and the output of h is of the form “xxxx@janus.com”

- Because f, g, and h are deterministic, future accesses will yield same account information

  “xxxx@janus.com” is forwarded to “reiter@unc.edu”
Unlinkability Via Mixes

- A “mix” is a special “router”
- Attempts to hide correspondence between incoming and outgoing messages

Mix
- batches
- changes order
- changes encoding

Incoming messages
- ○
- △

Outgoing messages
- □
- ○

Changing the Encoding

- Change of encoding should be something that only mix server can perform
  - Usually implemented with a public key cryptosystem

Incoming messages
- $E_K(\triangle) \rightarrow ○$
- $E_K(\square) \rightarrow ○$
- $E_K(\circ) \rightarrow \triangle$

Outgoing messages
- □
- ○
- △

• batches
• changes order
• decrypts using $K^{-1}$
Chaining Mixes

- Mixes can be “chained” together
  - If any mix server is trustworthy, then unlinkability can be achieved
  - Defends against compromise of mix servers

Return Addresses

\[
\begin{align*}
K_1 & \leftarrow G(\ell^1) \\
K_2 & \leftarrow G(\ell^2) \\
c_1 & \leftarrow E_{K_0}(K_1, \ell^1) \\
d_2 & \leftarrow E_{K_1}(\text{req}, K_2, \ell^2) \\
d_1 & \leftarrow E_{K_0}(B, d_2) \\
\langle B', d'_1 \rangle & \leftarrow D_{K_2}(d_1) \\
\langle \text{req}', K_2', c'_1 \rangle & \leftarrow D_{K_2}(d'_1) \\
e_1 & \leftarrow E_{K_2}(\text{rsp}) \\
\langle K_1', A' \rangle & \leftarrow D_{K_2}(c'_1) \\
e_0 & \leftarrow E_{K_2}(e_1) \\
e'_1 & \leftarrow D_{K_2}(e_0) \\
\text{rsp}' & \leftarrow D_{K_2}(e'_1)
\end{align*}
\]
Chaining Return Addresses

- \( c_i \) and \( d_i \) are encrypted under \( K_{M_i} \)
  - \( d_i \) is decrypted on outbound direction (request)
  - \( c_i \) is decrypted on return direction (response)
  - \( M_i \) uses \( K_i \) to encrypt on return direction
  - \( B = M_{n+1} \)

Chaining Return Addresses (cont.)

- \( A = M_0 \)
- \( M_i \)
- \( B = M_{n+1} \)

for \( i = 1 \ldots n+1; K_e \leftarrow G^{i+1} \)
\( c_0 \leftarrow 1 \)
for \( i = 1 \ldots n; e_i \leftarrow E_{K_{e_i}}((K_i, M_{i-1}, c_{i-1})) \)
\( d_{i+1} \leftarrow E_{K_{e_i}}(\text{req}, K_{e_i}, c_{i-1}) \)
for \( i = n+1; d_i \leftarrow E_{K_{e_i}}((M_{i-1}, d_{i-1})) \)
\( d_i \)
\( (M_{i-1}, d_{i-1}) \leftarrow D_{K_{e_i}}(d_i) \)
\( d_{i+1} \)
\( (\text{req}, K_{e_i}, c_{i-1}) \leftarrow D_{K_{e_i}}(d_{i+1}) \)
\( e_{n+1} \leftarrow E_{K_{e_i}}(\exp) \)
\( e_i \leftarrow E_{K_{e_i}}(\epsilon_i) \)
\( e_{i+1} \leftarrow E_{K_{e_i}}(e_i) \)
\( e_{i-1} \leftarrow E_{K_{e_i}}(e_i) \)
\( r_{i+1} \leftarrow D_{K_{e_i}}(e_i) \)

Copyright © 2018 by Michael Reiter
All rights reserved.
Attacks on Mixes

- **Replay**: Send the same message through twice

  ![Diagram](Time_t \rightarrow \text{Mix} \rightarrow \text{Timestamp} \rightarrow \text{Packet}) \hspace{1cm} \text{Time } t + d \rightarrow \text{Mix} \rightarrow \text{Timestamp} \rightarrow \text{Packet}

  - **Possible defense**
    - Sender includes timestamp within each “layer”
    - Mix drops each message with expired timestamp
    - Mix keeps copy of each message until its timestamp expires, and refuses to process the same message again

Attacks on Mixes (cont.)

- **Bridging**: Attacker submits all but one mixed message

  ![Diagram](Contact \rightarrow \text{Mix} \rightarrow \text{Packet}) \hspace{1cm} \text{Contact} \rightarrow \text{Mix} \rightarrow \text{Packet}

  - **Possible defense**
    - Authenticate senders
    - Limit number of messages from each sender per batch
    - Hope the number of colluding attackers is small
    - Output dummy messages
Attacks on Mixes (cont.)

- Length can disclose correspondence between inputs and outputs

- Possible defense
  - Break/pad messages into fix-length blocks, and make sure that mix transformation is length preserving
  - Maintain constant amount of communication between each mix server

Sender Anonymity: DC-Nets

Basic idea: for one sender to anonymously send one bit $b$

- each pair of potential senders $s_i, s_j$ share a secret key bit $k_{ij}$
- actual sender $s_i$ broadcasts
  $$b \oplus k_{i,1} \oplus \ldots \oplus k_{i,i-1} \oplus k_{i,i+1} \oplus \ldots \oplus k_{i,n}$$
- each other $s_i$ broadcasts
  $$k_{i,1} \oplus \ldots \oplus k_{i,i-1} \oplus k_{i,i+1} \oplus \ldots \oplus k_{i,n}$$
- XOR of all broadcast messages is $b$
DC-Nets (cont.)

\[ m_1 : 10101 \]
\[ k_{1,2} : 00011 \]
\[ k_{1,3} : 10110 \]

\[ m_2 : 00000 \]
\[ k_{1,2} : 00011 \]
\[ k_{2,3} : 01011 \]

\[ m_3 : 00000 \]
\[ k_{1,3} : 10110 \]
\[ k_{2,3} : 01011 \]

Security of DC-Nets

- Consider the following graph
  - each node represents a participant
  - each edge represents a key shared by its endpoints
  - each participant has a message to send (0 or 1)
  - assume that the graph is connected
Security of DC-Nets (cont.)

- **Definition:** An anonymity set seen by a set $K$ of keys is a set of connected vertices in the graph formed by removing the edges corresponding to $K$.

- **Examples:**
  - anonymity set seen by $K = \emptyset$? *All vertices* $V$
  - anonymity sets seen by $K = \{\text{all edges}\}$? *for each* $s \in V$
  - in a complete graph, the anonymity set seen by all keys incident on a set $S$ of vertices? $V \setminus S$
  - in a biconnected graph, the anonymity set seen by all keys incident on one vertex $s$? $V \setminus \{s\}$

- **Theorem:** Any attacker knowing keys $K$ can learn only the parity of the messages of an anonymity set seen by $K$.