Title Goes Here

Anonymous Communication

Mike Reiter

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

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

[Mazieres & Kaashoek 1998]

- **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

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

The Anonymizer

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

- Challenge: rewriting links in web pages
  - For example, a web page containing
    <A HREF=http://www.eff.org>
    is rewritten to
    <A HREF=http://www.anonymizer.com:8080/http://www.eff.org>
  - 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**

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*

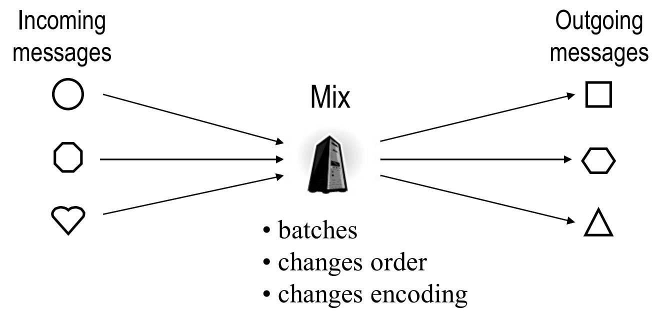
An Example Use of Janus (cont.)

- **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

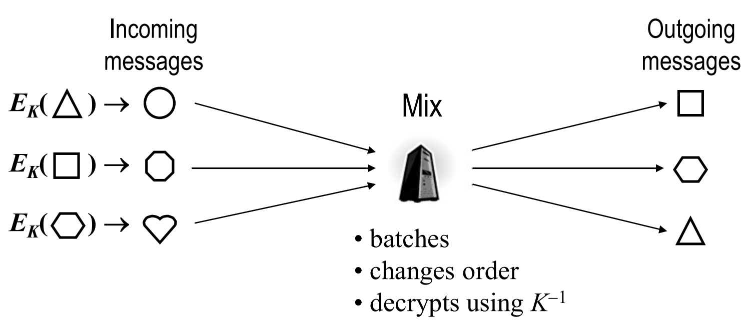


11

11

Changing the Encoding

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

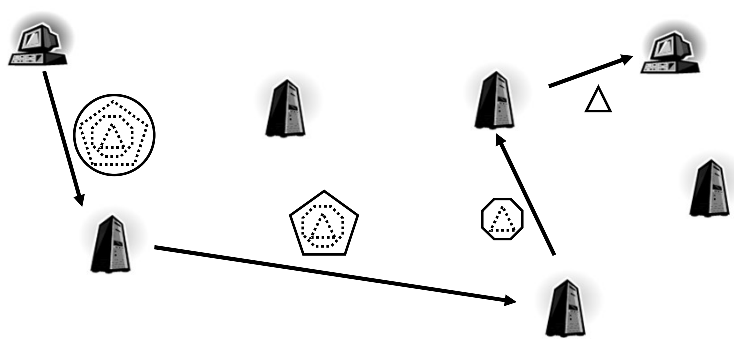


12

12

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

$$K_1 \leftarrow G(1^\lambda)$$
$$K_2 \leftarrow G(1^\lambda)$$
$$c_1 \leftarrow E_{K_M}(\langle K_1, A\rangle)$$
$$d_2 \leftarrow E_{K_B}(\langle \text{req}, K_2, c_1\rangle)$$
$$d_1 \leftarrow E_{K_M}(\langle B, d_2\rangle)$$

$$\xrightarrow{d_1}$$

$$\langle B', d_2'\rangle \leftarrow D_{K_M^{-1}}(d_1)$$

$$\xrightarrow{d_2'}$$

$$\langle \text{req}', K_2', c_1'\rangle \leftarrow D_{K_B^{-1}}(d_2')$$
$$e_1 \leftarrow E_{K_2'}(\text{rsp})$$

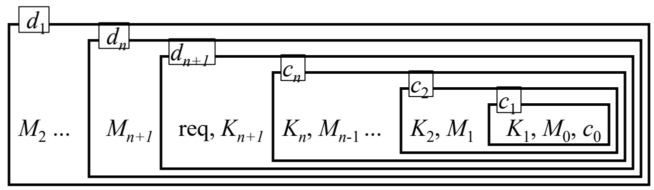
$$\xleftarrow{c_1', e_1}$$

$$\langle K_1', A'\rangle \leftarrow D_{K_M^{-1}}(c_1')$$
$$e_0 \leftarrow E_{K_1'}(e_1)$$

$$\xleftarrow{e_0}$$

$$e_1' \leftarrow D_{K_1}(e_0)$$
$$\text{rsp}' \leftarrow D_{K_2}(e_1')$$

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
  - $B = M_{n+1}$

15

Chaining Return Addresses (cont.)

$A = M_0$ | M_i | $B = M_{n+1}$

A:

$\text{for } i = 1 \ldots n+1 : K_i \leftarrow G(1^\lambda)$

$c_0 \leftarrow \perp$

$\text{for } i = 1 \ldots n : c_i \leftarrow E_{K_{M_i}}(\langle K_i, M_{i-1}, c_{i-1} \rangle)$

$d_{n+1} \leftarrow E_{K_{M_{i+1}}}(\langle \text{req}, K_{n+1}, c_n \rangle)$

$\text{for } i = n \ldots 1 : d_i \leftarrow E_{K_{M_i}}(\langle M_{i+1}, d_{i+1} \rangle)$

$\xrightarrow{d_i}$

M_i: $\langle M'_{i+1}, d'_{i+1} \rangle \leftarrow D_{K^{-1}_{M_i}}(d_i)$

$\xrightarrow{d'_{i+1}}$

B: $\langle \text{req}', K'_{n+1}, c'_n \rangle \leftarrow D_{K^{-1}_{M_{n+1}}}(d'_{n+1})$

$e_n \leftarrow E_{K'_{n+1}}(\text{rsp})$

$\xleftarrow{c'_i, e_i}$

M_i: $\langle K'_i, M'_{i-1}, c'_{i-1} \rangle \leftarrow D_{K^{-1}_{M_i}}(c'_i)$

$e_{i-1} \leftarrow E_{K'_i}(e_i)$

$\xleftarrow{c'_{i-1}, e_{i-1}}$

A: $e'_0 \leftarrow e_0$

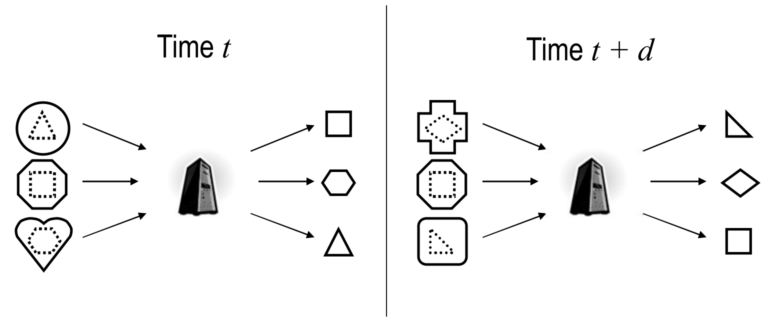
$\text{for } i = 1 \ldots n : e'_i \leftarrow D_{K_i}(e'_{i-1})$

$\text{rsp}' \leftarrow D_{K_{n+1}}(e'_n)$

16

Attacks on Mixes

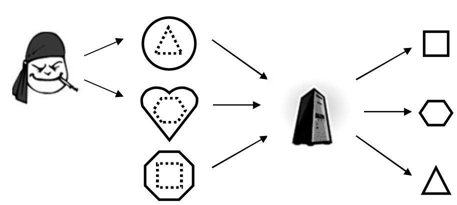
- **Replay: Send the same message through twice**



- **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.)

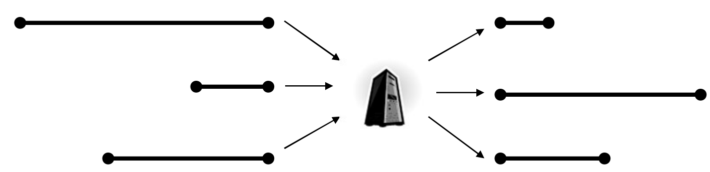
- **Bridging: Attacker submits all but one mixed message**



- **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 $k_{i,j}$**
- **actual sender s_i broadcasts**

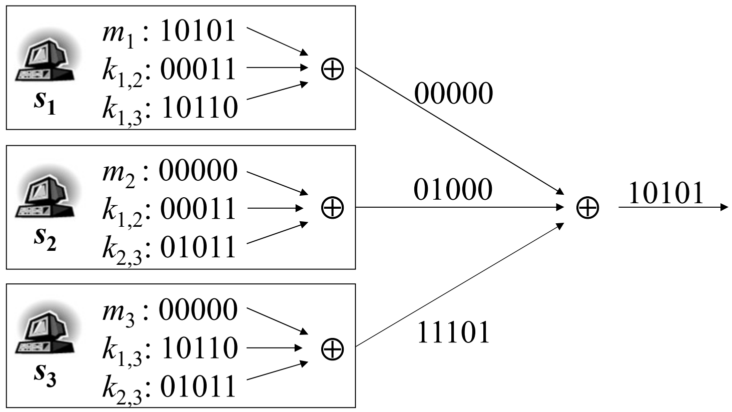
$$b \oplus k_{i,1} \oplus \ldots \oplus k_{i,i-1} \oplus k_{i,i+1} \oplus \ldots \quad k_{i,n}$$

- **each other s_i broadcasts**

$$k_{i,1} \oplus \ldots \oplus k_{i,i-1} \oplus k_{i,i+1} \oplus \ldots \oplus k_{i,n}$$

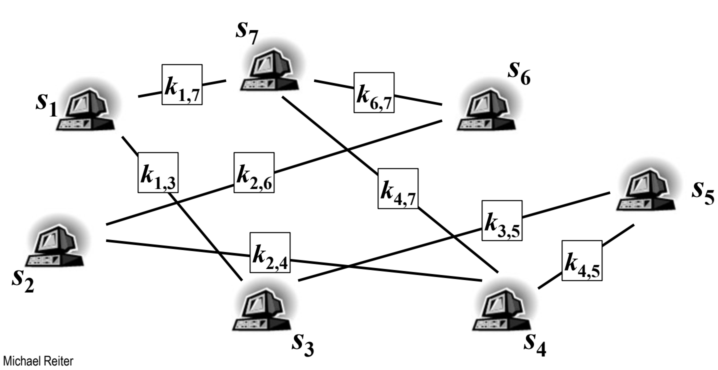
- **XOR of all broadcast messages is b**

DC-Nets (cont.)



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

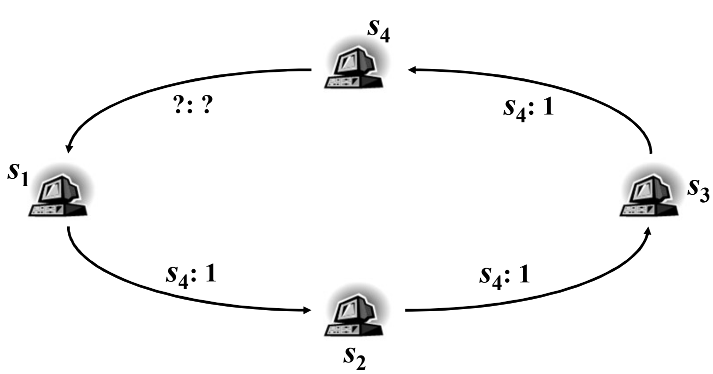


Security of DC-Nets (cont.)

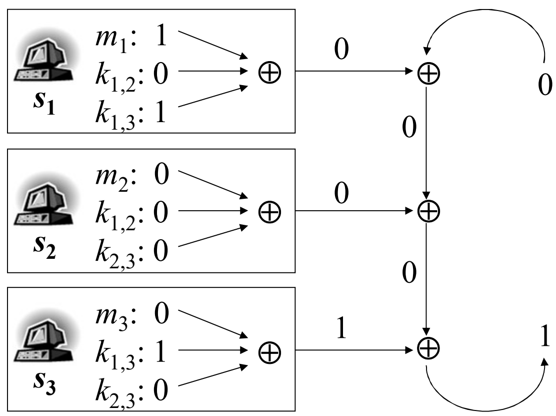
- **Definition: An <u>anonymity set seen by a set K of keys</u> is a set of connected vertices in the graph formed by removing the edges corresponding to K.**
- **Examples:**
  - anonymity set seen by $K = \emptyset$? ***All vertices V***
  - anonymity sets seen by K = {all edges}? **$\{s\}$ for each $s \in V$**
  - in a complete graph, the anonymity set seen by all keys incident on a set S of vertices? $\boldsymbol{V \setminus S}$
  - in a biconnected graph, the anonymity set seen by all keys incident on one vertex s ? $\boldsymbol{V \setminus \{s\}}$
- **<u>Theorem</u>: Any attacker knowing keys K can learn only the parity of the messages of an anonymity set seen by K.**

Ring Networks

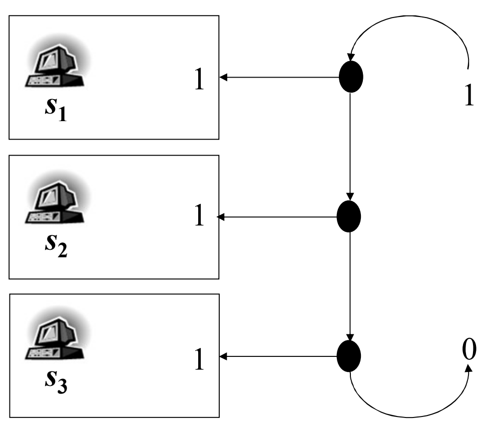
- **Communication systems based on cycles, called <u>rings</u>, are a common structure for LANs (e.g., token ring)**
  - a bit travels around ring from sender to destination



DC-Nets Over a Ring (Round 1)



DC-Nets Over a Ring (Round 2)



Properties of Ring Implementation

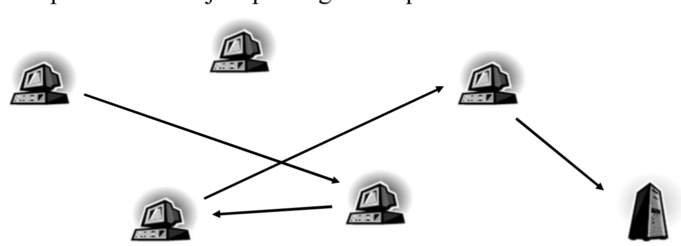
- **A threefold (at least) decrease in bandwidth compared to one in which messages travel half-way around ring on average**
- **May incur collisions due to concurrent senders**
  - avoid token reservation; may reveal sender
  - collisions detected after full trip around the ring
  - after detection, sender can wait a random time to retransmit

27

27

Sender Anonymity: Crowds

- **A proxy-based approach developed for web browsing**
  - each user joins a “crowd” and runs a local proxy
  - each user request is routed along a random path to destination server
  - each proxy on the path cannot tell if its predecessor is the source or if its predecessor is just passing the request on behalf of another



- **Main adversaries addressed**
  - web server
  - other crowd members

28

28

Crowds Proxy Algorithm

```
⟨client, request⟩ ← receive_request();

if (client = browser)
  sanitize(request);              /* strip cookies &
                                     identifying headers */
  if (my_path_id = ⊥)             /* if my_path_id is not initialized … */
    next[my_path_id] ←R Crowd;    /* select next proxy at random */
  forward_request(my_path_id);    /* send request to next proxy */
else                              /* client is a proxy */
  path_id ← remove_path_id(request);  /* remove incoming path id */

  if (translate[path_id] = ⊥)      /* incoming path_id is new */
    coin ← coin_flip(pf);         /* tails with probability pf */
    if (coin = 'heads')
      translate[path_id] ← 'submit';
    else
      translate[path_id] ← new_path_id(); /* outgoing path id */
      next[translate[path_id]] ←R Crowd;  /* select next proxy */

  if (translate[path_id] = 'submit')
    submit_request();             /* send request to destination server */
  else
    forward_request(translate[path_id]); /* send request to next proxy */
```


29

29

Crowds Proxy Algorithm (cont.)

```
subroutine forward_request(out_path_id)
  send out_path_id||request to next[out_path_id];
  reply ← await_reply(∞);        /* wait for reply or
                                     recognizable proxy failure */
  if (reply = 'proxy failed')     /* proxy failed */
    Crowd ← Crowd \ {next[out_path_id]};
    next[out_path_id] ←R Crowd;
    forward_request(out_path_id);
  else                            /* received reply from jondo */
    send reply to client;


subroutine submit_request()
  send request to destination(request);
  reply ← await_reply(timeout);  /* wait for reply, timeout,
                                     or server failure */
  send reply to client;
```


30

30

Anonymity Properties

- **Anonymity versus web server**
  - Proxy at source of request always forwards request to some proxy in the crowd
  - Web server thus receives the request from a Crowd member chosen uniformly at random
- **Anonymity versus colluding crowd members**
  - Colluding members will suspect who they receive request from
  - Define
    - H_k, $k \geq 1$, to be event that first collaborator on path occupies the k-th position on the path (where the initiator is in 0-th position)
    - $H_{k+} = H_k \vee H_{k+1} \vee H_{k+2} \vee \ldots$
    - I to be the event that first collaborator is immediately preceded by path initiator
  - Then, we want to compute $P(I \mid H_{1+})$

31

31

Computing $\mathbf{P(\mathit{I} \mid \mathit{H}_{1+})}$

$$P(I \mid H_{1+}) = \frac{P(I \wedge H_{1+})}{P(H_{1+})} = \frac{P(I)}{P(H_{1+})} \quad \text{since } I \supset H_{1+}$$

- **We need to compute $P(I)$ and $P(H_{1+})$**
- **Let n = # crowd members, c = # collaborators**

32

32

Computing $P(H_{1+})$

- To compute $P(H_{1+})$, let's first compute $P(H_i)$

$$P(H_i) = \left(\frac{p_f(n-c)}{n}\right)^{i-1}\left(\frac{c}{n}\right)$$

- $P(H_{1+})$ follows from $P(H_i)$

$$P(H_{1+}) = \sum_{i=1}^{\infty} P(H_i) = \sum_{j=0}^{\infty}\left(\frac{p_f(n-c)}{n}\right)^{j}\left(\frac{c}{n}\right) = \frac{c}{n-p_f(n-c)}$$

Computing $P(I)$

$$P(I) = P(H_1)P(I \mid H_1) + P(H_{2+})P(I \mid H_{2+})$$

$$P(H_1) = \frac{c}{n}$$

$$P(I \mid H_1) = 1$$

$$P(H_{2+}) = \sum_{i=2}^{\infty} P(H_i) = \sum_{j=1}^{\infty}\left(\frac{p_f(n-c)}{n}\right)^{j}\left(\frac{c}{n}\right) = \frac{cp_f(n-c)}{n^2-np_f(n-c)}$$

$$P(I \mid H_{2+}) = \frac{1}{n-c}$$

$$\therefore P(I) = \frac{c(n-np_f+cp_f+p_f)}{n^2-p_f n(n-c)}$$

Computing $P(I \mid H_{1+})$

- Putting it all together …

$$P(I \mid H_{1+}) = \frac{P(I)}{P(H_{1+})} = \frac{n - p_f(n-c-1)}{n}$$

- If we want $P(I \mid H_{1+}) \le 1/2$, then it suffices for

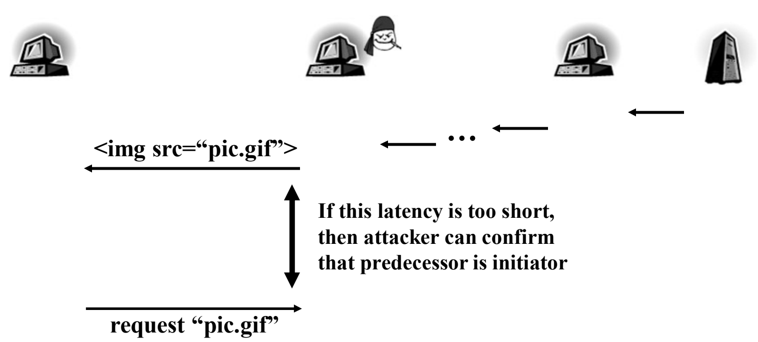
$$n \ge \frac{p_f}{p_f - 1/2}(c+1)$$

- For example, $p_f = 3/4$ and $n \ge 3(c+1)$ implies $P(I \mid H_{1+}) \le 1/2$

35

Timing Attacks

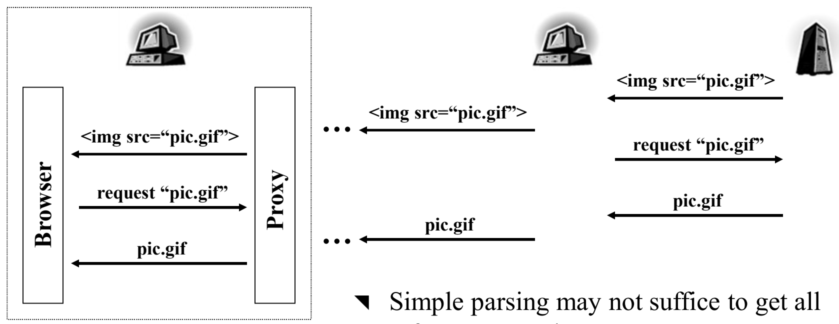
- Timing attacks arise from the structure of HTML
  - Some HTML commands elicit an immediate request from browser



36

Timing Attacks (cont.)

- **One approach to prevent timing attacks**
  - First and last proxy parse HTML to identify automatic requests
  - First (local) proxy blocks automatic requests from browser
  - Last proxy retrieves pages and sends them along



- Simple parsing may not suffice to get all references to other pages
  - Disable active content
- Makes caching less effective

37

37

Anonymity Decays Due to Path Linking

- **Anonymity decays (versus collaborating proxies) if multiple paths can be linked to the same initiator**
  - Linking can occur based on timing, content, etc.
- **Initiator precedes first collaborator with higher probability than any other proxy**
- **So, the true initiator will precede collaborators more often than any other on linked paths**
- **Over time, this exposes initiator (if paths can be linked)**
- **Can be delayed by reconfiguring paths very rarely**
  - But path reconfigurations are required for a new member to join
- **This threat applies to any sender-anonymous system**

38

38

Receiver Anonymity Via Broadcast

- **To hide receiver**
  - deliver each message to all nodes (broadcast)
  - label each message so that the addressee and nobody else can recognize it is addressed to her (an implicit address)
- **Implicit address**
  - vs. explicit address: latter names a place in the network
  - is visible if it can be publicly tested for equality, invisible otherwise
  - is public if known to every user, private if distinct and secret to a particular user

Visible Implicit Addresses

- **Visible implicit addresses: pseudonyms**
  - users choose arbitrary pseudonyms for themselves
  - pseudonyms are used to label messages
  - can be used as private address, but ideally only once
    - multiple uses enables linking of messages to same user
- **Invisible (and public) implicit addresses can be realized with a public key cryptosystem**
  - message is addressed by adding redundancy and then encrypting it with addressee's public key: $E_k(m,h(m))$
  - each receiver decrypts all messages, uses redundancy to decide which messages are addressed to it
  - can similarly be realized if sender shares a distinct secret key with each receiver