Crowds:
Anonymity for Web Transactions

Paper by: Michael K. Reiter and Aviel D. Rubin,

Presented by Eric M. Busse

Portions excerpt from Crowds: Anonymity for Web Transactions
Michael K. Reiter and Aviel D. Rubin
AT&T Labs Research

How safe is web browsing?

- Web surfing is exposed to many types of monitoring and tracking, many of which may be undesirable
- SSL and existing technologies do not address these issues
- What can we do to prevent this sort of monitoring?

Crowds

Crowds seeks to obscure the actions of the individual within those of a group, by randomly forwarding requests from members between each other before sending them to their final destination.

This gives us deniability!

Conceptually, is this a good solution?

That really all depends…

- Joining a group makes you a co-conspirator
- You could be held accountable for fulfilling someone else's request
- Crowds can be undermined by some types of content (which are becoming progressively more common)

Overview

- Each user is represented by a *Jondo*.
- Jondos contact a *blender* to join a crowd.
- At the first request for a web page the users Jondo contacts another Jondo at random to begin constructing a path.
- Each path has a path key, meaning encryption of requested content is only preformed at the end points of the jondo chain.

Jondos

- Each jondo maintains a list of other active jondos
- Each jondo has a shared key which is known to all other jondos (by way of the blender) to allow for secure communication between jondos.
- Jondos perform limited page processing both to prevent certain attacks and remove dangerous content.

Blenders

- Authenticate jondos
- Maintain a list of active jondos and their shared keys
- Schedule “join-commit” events
- Blender failure will not entirely compromise the crowd, or disrupt communication between existing members.

Improves on Related Research…

- Anonymizer & LPWA (Proxies)
- Mixnets

Analysis

Anonymity (Security),
Performance & Scalability

General types of Anonymity

- Sender Anonymity
- Receiver Anonymity
- Unlinkability of Sender and Reciver

To this the authors add:

- Degree of Anonymity

Degrees of Anonymity

- Absolutely Privacy

- Beyond Suspicion
- Probable Innocence
- Possible Innocence

Crowds

- Provably Exposed

Most Web Browsers

Attackers and Crowds Safety

Attackers:

- Local Eavesdroppers
- End Servers
- Collaborating crowd members

Attacker	Sender anonymity	Receiver anonymity
local eavesdropper	exposed	$P(\text{beyond suspicion}) \underset{n \to \infty}{\longrightarrow} 1$
c collaborating members, $n \geq \frac{p_f}{p_f - 1/2}(c+1)$	probable innocence $P(\text{absolute privacy}) \underset{n \to \infty}{\longrightarrow} 1$	$P(\text{absolute privacy}) \underset{n \to \infty}{\longrightarrow} 1$
end server	beyond suspicion	N/A

Local Eavesdropper

- Request initiation is obvious, however the destination is obscured.
- This is only compromised in the event that the user is unlucky and is at the end of his particular chain
- The above event is unlikely as the probability is inversely proportional to crowd size.

End Servers

- Because of the nature of the crowd and the manner in which messages are passed between members it is equally likely that any member initiated the request.

Collaborating Jondos

- The goal of collaborating jondos is to determine the path back to the initiator of the request
- Assuming *pF* is > ½, n is the number of crowd members, c is the number of collaborators we have:

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

Which means that the path initiator has probable innocence

Timing Attacks

- These attacks arise out of the nature of web content, as an HTML page is parsed additional requests are generated from links on the page (images, jscript, etc).
- By timing the gap between a page request and the subsequent requests of its linked content a corrupt jondo on the path can attempt to deduce the position of the initiator
- This is avoided by the mechanism mentioned earlier.

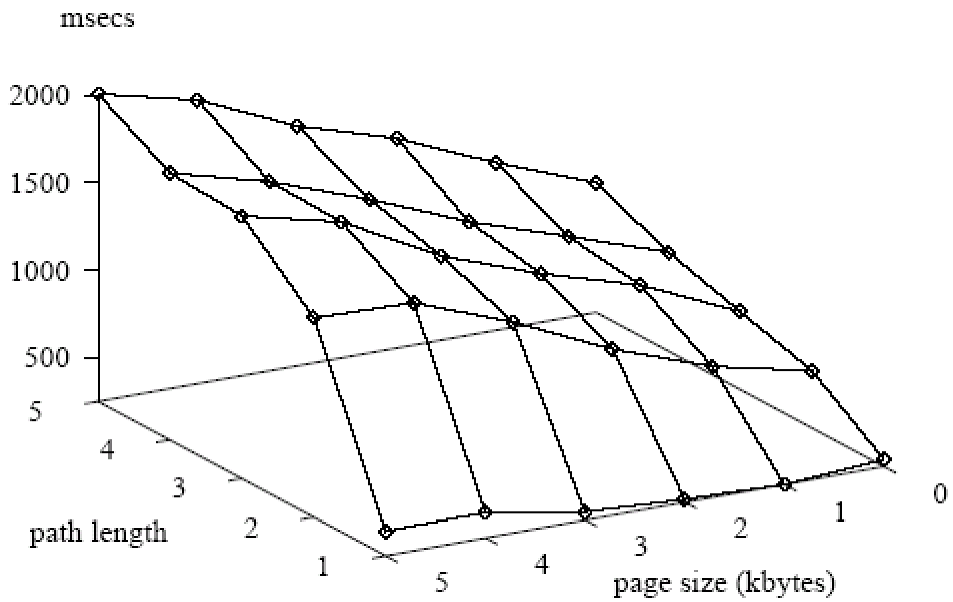
Path Reasoning

- Static vs. Dynamic
- Dynamic changes increase the odds of a collaborator being on your path
- A path will only be altered at a “join-commit” or because a node sends a “fail stop”
- A malicious jondo(s) executing a “fail stop” will not compromise the initiator

Crowd Control

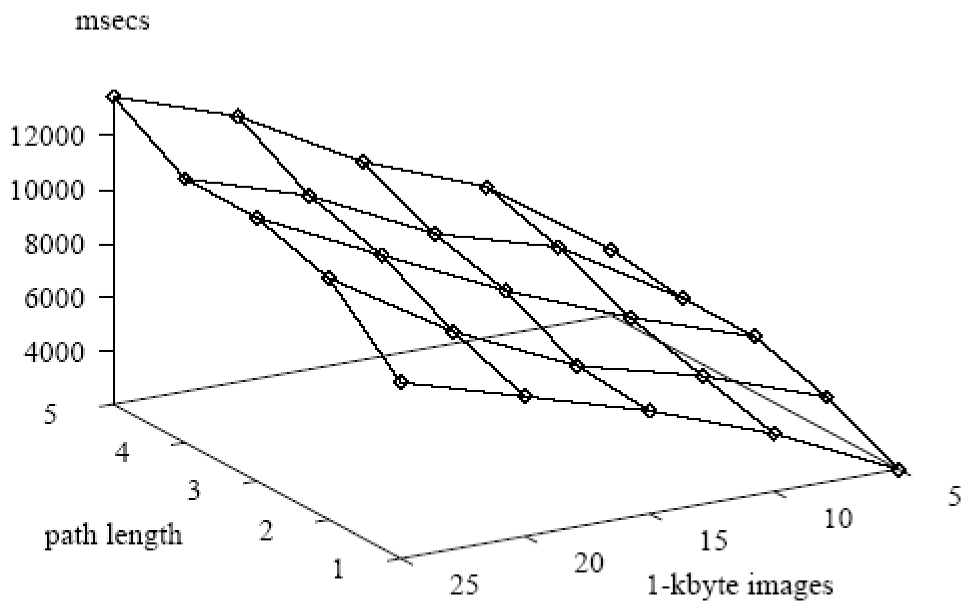
- The blender should have limits on the number of jondos allowed to associated with a single username/IP
- Two types of crowds should exist, large public crowds, and smaller personal crowds

Performance



Path length	Page size (kbytes) 0	1	2	3	4	5
1	288	247	264	294	393	386
2	573	700	900	1157	1369	1384
3	692	945	1113	1316	1612	1748
4	814	1004	1191	1421	1623	1774
5	992	1205	1446	1620	1870	2007

Performance, cont'd



Path length	Number of 1-kbyte images 5	10	15	20	25
1	2069	4200	5866	7219	8557
2	3313	4915	6101	8195	10994
3	4127	5654	7464	9611	11809
4	4122	6840	8156	10380	11823
5	4508	7644	9388	11889	13438

Performance Implications

- Paths are relatively fixed, hence slow links on a path can dramatically impact performance.
- Path length, and therefore *pF* also factor heavily into the performance.

$$
\begin{aligned}
(1-p_f)\sum_{k=0}^{\infty}(k+2)(p_f)^k &= (1-p_f)\left[\sum_{k=0}^{\infty}k(p_f)^k + 2\sum_{k=0}^{\infty}(p_f)^k\right] \\
&= (1-p_f)\left[\frac{p_f}{(1-p_f)^2} + \frac{2}{1-p_f}\right] \\
&= \frac{p_f}{1-p_f} + 2
\end{aligned}
$$

Scale

- The upper bound on the number of times a jondo appears on a given path is
  $O \{ 1/(1-pF)^2 [1 + (1 + (1/n))] \}$
- As a consequence of this result the load on any given jondo will remain constant as the number of crowd members increases
- Throughput on the network increases as the number of crowd members increases

Other Concerns

Firewalls pose a special concern for Crowds users as they prevent jondos outside the wall from forming paths involving jondos within the wall. While a jondo inside a wall can create a path involving those outside his security is seriously compromised.

Questions?

To clarify the “Wide Mouth Frog” protocol is also known as the “Otway-Rees Protocol”

When Alice wants to talk to Bob she asks Troy, the trusted third party, to assist in the key exchange.

The process is as follows:

A - Identity or location of Alice

B - Identity or location of Bob

Ka - Key shared between Troy and Alice

Kb - Key shared between Troy and Bob

Sab - Secret shared between Alice and Bob for session communication

Exchange:

Alice -> Troy {B,Sab}Ka

Troy -> Bob {A,Sab}Kb

In this manner Alice uses Troy to securely share a secret with Bob.