Anonymity Loves Company: Anonymous Web Transactions with Crowds

Michael K. Reiter
Bell Laboratories
Murray Hill, NJ, USA
reiter@research.bell-labs.com

Aviel D. Rubin
AT&T Labs—Research
Florham Park, NJ, USA
rubin@research.att.com

1 Motivation

Web servers' log files are riddled with information about the users that visit them. Obviously a server can record the content that each visitor accesses. In addition, however, the server can record the user's IP address—and thus often the user's Internet domain name, workplace, and/or approximate location—the type of computing platform she's using, the web page that referred her to this site and, with some effort, the server that she visits next [Rei98]. Even when the user's IP address changes between browsing sessions (e.g., her IP address is assigned dynamically using DHCP [Dro93]), a web server can link multiple sessions by the same user by planting a unique *cookie* in the user's browser during the first browsing session, and retrieving that cookie in subsequent sessions. Moreover, virtually the same monitoring capabilities are available to other parties, e.g., the user's Internet service provider or local gateway administrator, who can observe all communication in which the user participates.

The user profiling made possible by such monitoring capabilities is viewed as a useful tool by many businesses and consumers: it makes it possible for a web server to personalize its content for its users, and for businesses to monitor its employees' activities. However, the negative ramifications for user privacy are considerable. While a lack of privacy has, in principle, always characterized Internet communication, never before has a type of Internet communication both been logged so universally and revealed so much about the personal tastes of its users. Our thesis is thus that a web user should have the ability to limit what information is revealed about her and to whom it is revealed. Some approaches, notably P3P [RC98], enable a user to negotiate the conditions under which she will share potentially private information with a web server. However, since these negotiations apply only to the server (not others who can capture private information), only for certain kinds of information (e.g., not the user's IP address), cannot be enforced, and are not presently widely practiced, we believe that other methods are needed as well.

This paper presents a system called Crowds, which enables the retrieval of information over the web without revealing so much potentially private information to so many parties. The goal of Crowds is to make browsing *anonymous*, so that information about either the user or what information she retrieves is hidden from web servers and other parties. Crowds prevents a web server from learning any potentially identifying information about the user, including even the user's IP address or domain name. Crowds also prevents web servers from learning a variety of other information, such as the page that referred the user to its site or the user's computing platform. Crowds provides complementary protections when the party of concern can directly monitor the

To appear in *Communications of the ACM*.

messages from the user's machine, e.g., the user's local gateway administrator: in this case, Crowds hides what sites the user is visiting from these parties. And, since Crowds achieves these properties by employing other users' machines for the purpose of issuing web requests, Crowds takes measures to conceal user-specific information from these parties, too.

2 How Crowds works

Crowds works by collecting web users into a geographically diverse group called a *crowd* that performs web transactions on its members' behalves. A user is represented in the crowd by a process on her local machine called a *jondo* (pronounced "John Doe" and meant to connote a faceless participant in the crowd). That is, a user joins the crowd by starting her jondo on her local machine. Once started, the jondo engages in a protocol to join the crowd, during which it is informed of the other current crowd members and in which the other crowd members are informed of the new jondo's membership in the crowd.

Once a user's jondo has been admitted to the crowd, it can employ the crowd to issue requests to web servers in a way that prevents web servers and other crowd members from determining who initiated those requests. To do this, the user configures her browser to employ her local jondo as a proxy for all network services, so that all communication by her browser is sent directly to the jondo. When the user then requests a URL via her browser, the HTTP request for that URL is sent to the jondo, rather than the end server that serves that URL. Upon receiving this request, this jondo chooses another member of the crowd uniformly at random, and sends the request to that member. Whenever a crowd member receives a request from another jondo in the crowd, it makes a random choice to either *forward* the request to another crowd member (which it chooses uniformly at random from all crowd members) or to *submit* the request to the end server to which it was destined. This random choice is biased in favor of forwarding; i.e., there is a system-wide parameter $p_f > 1/2$ that indicates the probability with which a jondo will choose to forward. As we will see in Section 3.1, the value of p_f impacts the anonymity properties offered by the system.

To summarize, a request is issued by the browser, forwarded through some number of jondos, and eventually submitted to the end server. The sequence of jondos that a request traverses is called a *path*. An important feature of the Crowds protocol is that the request is sent in the same form along each "hop" of the path, so that each jondo cannot tell whether its predecessor initiated the request or is just forwarding it from another jondo; we show in Section 3.1 that this an important ingredient for the anonymity properties of Crowds. The server's reply to the request (typically an HTML page or image) is sent backwards along the path, with each jondo sending it to its predecessor on the path. When the originating jondo receives the reply, it delivers the reply to the browser for rendering to the user. Figure 1 shows paths in a crowd.

Subsequent requests initiated by the same jondo follow the same path through the crowd, even if these requests are targeted for different destination web servers. That is, once established, a path remains static as long as possible, in order to prevent certain attacks on anonymity (see [RR98]). To achieve this, each jondo on a path records its predecessor and successor on the path, so that when it receives another request on that path (i.e., another request from the same predecessor and labeled with the same path identifier), it can route it along the path in the same way. A path is changed only when jondos on the path fail or when new jondos join the crowd. In the latter case, all jondos' paths are "forgotten" and re-routed from scratch, so that the path initiated by the new jondo does not stand out as the only newly-formed path (which would enable jondos on the path to know that the new jondo initiated it, thereby exposing the initiator of requests on that path).

All communication between jondos is encrypted by a cryptographic key shared between those

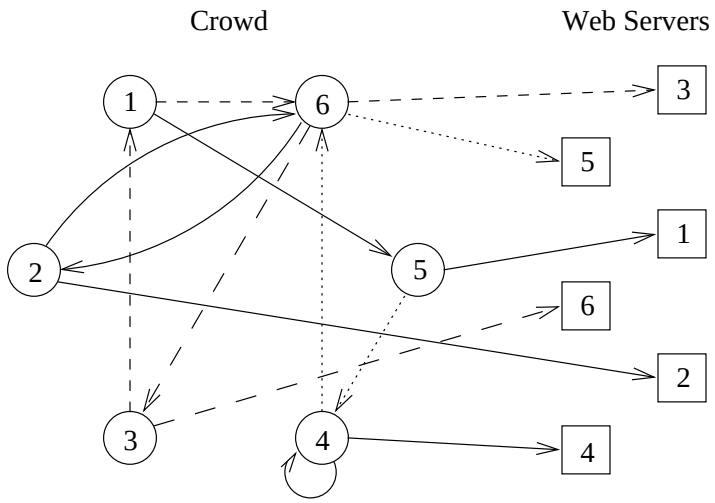

Figure 1: Paths in a crowd. The initiator and web server of each path are labeled the same. Series of contiguous arrows of the same form, starting at a crowd member and ending at a web server with the same label, denote a path through the crowd.

jondos. As shown in Section 3.1, this is useful for providing certain classes of anonymity properties. Other details of the Crowds protocols are described in [RR98].

3 Evaluating Crowds

3.1 Anonymity properties

Anonymous communication properties can be characterized on at least three different axes. The first axis is the type of anonymity; here we are concerned with *sender anonymity* and *receiver anonymity*, where the "sender" is a web user and the "receiver" is the web server. *Sender anonymity* means that the identity of the party who sent a message (web request) is hidden, while its receiver (and the message itself) might not be. *Receiver anonymity* similarly means that the identity of the receiver (web server) is hidden. The second axis is against what *adversaries* each type of anonymity is provided. In our case, an adversary can be a web server, other crowd members, or a "local eavesdropper" (e.g., the user's local gateway administrator) who can observe all (and only) communication in which the user's machine participates. The third axis is the *degree* of anonymity, which captures the strength of each type of anonymity on a spectrum (see Figure 2) ranging from *absolute privacy*, where the adversary cannot even perceive the presence of communication, to *provably exposed*, where the adversary can prove the sender or receiver to others. Of particular interest here are the *beyond suspicion* and *probable innocence* points on this spectrum. A sender's anonymity is beyond suspicion if though the attacker can see evidence of a sent message, the sender appears no more likely to be the originator of that message than any other potential sender in the system. A weaker guarantee is *probable innocence*: a sender is probably innocent if, from the attacker's point of view, each sender appears no more likely to be the originator than to not be the originator.

The anonymity properties achieved by Crowds are summarized in Table 1 and briefly justified in the rest of this section. First, since the jondo originating the request always forwards the request to a randomly chosen member of the crowd, the end server receives each request from any member

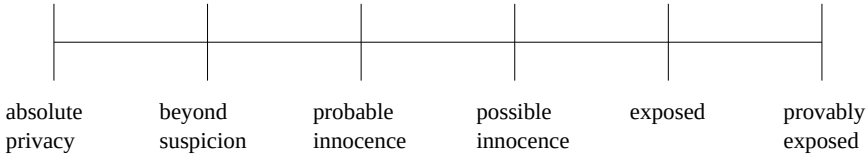

Figure 2: **Degrees of anonymity:** Degrees range from *absolute privacy*, where the attacker cannot perceive the presence of communication, to *provably exposed*, where the attacker can prove the sender, receiver, or their relationship to others.

of the crowd with equal likelihood. That is, the end server obtains no information regarding who initiated any given request, except possibly for the fact that it was initiated by a crowd member. In the language of the spectrum of Figure 2, this offers sender anonymity that is *beyond suspicion* when the adversary is the end server.

Table 1: Anonymity properties provided by Crowds

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

Second, since a jondo on a path cannot distinguish whether its predecessor on the path initiated the request or is merely forwarding it, no jondo on the path can learn the initiator of a request sent along that path. More precisely, suppose that there are c members of the crowd collaborating to determine who originated a request on a path that none of these c initiated. The predecessor of the first collaborator on the path obviously appears to be the most likely originator, since the collaborators know that it is on the path. However, in [RR98], we show that if the crowd has n members when the path is formed, then the probability that the collaborators are immediately preceded on the path by the request originator is at most $\frac{1}{2}$ provided that the relationship between c, n, and p_f is as shown in Table 1. For example, if the probability of forwarding is $p_f = \frac{3}{4}$ and the number of crowd members is at least $3(c+1)$, then the seemingly most likely originator, from the collaborators' viewpoint, is in fact not the originator at least half the time. In the parlance of Figure 2, this says that each crowd member is "probably innocent" (in terms of sender anonymity). Moreover, since the probability that none of the c collaborators will be on any given path (and thus will be unable to observe communication on that path) grows as the crowd size grows, it follows that the probability of absolute privacy approaches 1 as n grows, for both sender and receiver anonymity against c collaborating crowd members.

Third, use of the crowd offers receiver anonymity against an eavesdropper that can observe all (and only) communication involving the user's machine, e.g., as a local gateway administrator might be able to do. Crowds encrypts all communication between jondos, and so this "local eavesdropper" is unable to determine the eventual destination of a request in the event that the originator of the request does not end up submitting the request itself. Since the originator submits its own request with probability that decreases as the crowd size increases, the receiver (i.e., the web server) is discovered by the local eavesdropper with probability that decreases as the crowd size grows. That

is, in the face of a local eavesdropper, the probability of "beyond suspicion" receiver anonymity approaches 1 as n grows.

3.2 Risks and limitations

There are several risks and limitations of which potential Crowds users should be aware. First, because any crowd member can end up submitting any request originating within the crowd, the web server's log may incorrectly record the submitting jondo's IP address as the request originator's address. Of course, if challenged about this apparent request, the owner of the submitting jondo can simply explain that she was running Crowds and did not originate the request, and indeed Crowds has been designed so that this explanation is completely plausible.

Second, while Crowds offers sender anonymity to the originator of each request, it does not protect the confidentiality of the request contents against jondos on the path that the request traverses. As such, the contents of each request will generally be more accessible to other parties (i.e., the other jondos in the crowd) than they would be if Crowds were not used. This is primarily a concern for request contents that are private to the originator, e.g., the user's account name and password for some web server, or the user's credit card number. However, because such communications typically expose the user's identity to the end server anyway, it is counterproductive to apply an anonymizing tool such as Crowds to such communication. For such communication, we recommend that the user disable the proxy settings in her browser, so that communication takes place directly between the browser and web server. Another alternative would be to adapt Crowds to support end-to-end encrypted communication between a browser and web server, e.g., via SSL (see, e.g., [GS97, Ch. 12]). However, our present implementation does not support this.

Third, like any proxy-based anonymizing service, Crowds can be circumvented if the user's browser downloads and executes mobile code (e.g., a Java applet or ActiveX control) that opens a network connection back to the server that served it. This connection will generally be made to the web server directly, bypassing the user's jondo and thus exposing the user to the web server. For this reason, we recommend that users disable Java applets and ActiveX controls (or at least their networking capabilities) in their browsers when using Crowds.

Fourth, Crowds increases web page retrieval times as observed by the user, and more generally increases network traffic and load on the machines that execute jondos. We refer the reader to [RR98] for details of response time experiments that we have performed. The main conclusion of these experiments is that the overhead intrinsic to Crowds is quite modest and is most heavily influenced by the number of images embedded in a retrieved page. Moreover, we show analytically in [RR98] that Crowds scales well, in the sense that the load on each jondo's machine stays roughly constant as crowd size grows. These results were recently validated by an informal survey of Crowds users, which indicated that Crowds performance was usually acceptable for most users. Further improvements in response times could be achieved by implementing Crowds in a compiled language such as C, rather than the partially-interpreted and much slower language (Perl 5) in which it is presently implemented. Moreover, organizing crowds so that crowd members are in relatively close network proximity to each other should minimize inter-jondo communication delays.

3.3 Comparison to other systems

The Anonymizer (see `www.anonymizer.com`) is a popular tool for anonymizing web communications. This is a web site that serves as a simple proxy for web requests. That is, a user's request for a URL is first sent to the Anonymizer, which submits the URL to the end server. When the server replies to the Anonymizer, the Anonymizer sends the response back to the user's browser.

The Anonymizer protects the anonymity for the user from the end server; i.e., the end server has no information about who initiated the request. A similar mechanism is the Lucent Personalized Web Assistant [GGMM97] (LPWA), but in addition this proxy provides explicit support for personalized web browsing (i.e., browsing web sites that require an account name and password) without revealing the user to the end server.

The most cogent advantage of Crowds over the Anonymizer and LPWA is that Crowds presents no single point at which a passive eavesdropper can compromise all users' anonymity. That is, the system administrators of the Anonymizer and LPWA have access to all users' browsing behavior, and thus these services have to be trusted to not divulge this information or make use of it in other ways. Crowds presents no such single point at which all users' browsing behavior can be monitored, and indeed to reliably monitor even a single user seems to require the adversary to either monitor all communication between all jondos (a difficult task if a crowd spans multiple administrative domains) or directly monitor the actions of the user on her own local machine.

Other systems based on *mixes* [Cha81] have been developed to support anonymous web communication, including Internet communication in general (e.g., [SGR97]). A mix network consists of a collection of dedicated routers, call mixes, and each sender routes communication through some number of these mixes on the way to its destination. This routing protocol employs a layered encryption technique as well as buffering and reordering of messages at each mix in order to hide the correlation between the messages input to a mix and the messages it outputs as seen by an external eavesdropper. Space limitations preclude a detailed comparison of Crowds and mixes (see [RR98]), but briefly, mix networks are designed to provide different anonymity properties than Crowds and do so using a more heavyweight protocol. We believe that the anonymity properties provided by Crowds and the protocol by which they are achieved are better suited to synchronous communication (including web transactions) than mix networks are.

4 Deployment issues

At the time of this writing, we have distributed approximately 1200 copies of the Crowds code in response to user requests and presently maintain an active Crowd running on the Internet. The experience of developing and deploying Crowds has been a useful one, in that it has given us insight into the practical challenges of deploying a highly distributed software system on the Internet. Some of these challenges we had anticipated; others we had not.

Several deployment challenges resulted from realities regarding network connectivity on the Internet today. First, firewalls limit the extent to which a single crowd can span administrative domains. The reason is because like all network servers, jondos are identified by their IP address and port number, and most corporate firewalls do not allow incoming connections on ports other than a few standard ones. Thus, a firewall may prevent a jondo outside the firewall from connecting to another behind the firewall. Since most firewalls are configured to allow outgoing connections on any port, it is still possible for a jondo to initiate a path that goes outside the firewall and eventually to web servers, and in fact a Crowds user behind a firewall can configure her jondo to use it in this way. However, the firewall gives the first jondo on the path outside that domain a way to verify that the initiating computer resides within the domain: it simply tries to open a connection back to its predecessor on the path, and if that fails, then the path must have originated in the predecessor's domain. Thus, a crowd member behind a firewall is not offered the same anonymity as those that are not. The barriers imposed by firewalls could be partially rectified by, e.g., having jondos communicate using SSL and the standard SSL port, as some firewalls will allow this communication to pass. However, the use of SSL would likely result in higher protocol costs, and the use of the SSL

port might conflict with other uses of it on the same machine. Moreover, allowing jondos outside a firewall connect to one behind a firewall comes at the risk of exposing firewall-protected resources (in particular, web servers behind the firewall) to the outside.

A second reality of Internet connectivity that impacts Crowds deployment is the fact that a sizeable segment of potential Crowds users have only relatively low-bandwidth and high-latency modem connectivity to the Internet. When in a crowd, each such user's connectivity adds latency not only to her own web requests, but also to the requests sent on any path of which her computer is a member. Because of this, presently we limit the crowd that we maintain to contain only members with a direct connection to the Internet, and recommend the use of a separate "slow" crowd for those users with only modem connectivity. Alternatively, such a user can use a jondo on a machine other than her own and that has a fast connection to the Internet, but doing so offers no anonymity properties against an adversary who can observe the communication between the user's browser and that jondo.

A separate set of challenges regarding Crowds deployment has to do with the transience of jondos. It has been common among Crowds users for their jondos to remain in the crowd for only brief periods of time (presumably while the corresponding user is browsing) and then to leave. In alpha releases of the software, one reason for this was undoubtedly the tendency of jondos to crash. However, this trend has continued despite the fact that jondos have become much more reliable in subsequent versions of the software. This transience has adverse effects both on the anonymity of the transient jondo's owner and on the overall anonymizing ability of the crowd. Increasing the longevity of jondos in the crowd seems now to be largely a matter of educating our users of the benefits. To further this goal, we have begun an incentive program to encourage users to allow their jondos to remain in the crowd more permanently. This goal can also be furthered by limiting the frequency with which jondos are allowed to join the crowd, which is useful for other reasons as well [RR98].

A final deployment issue that we face is the United State's export controls on software containing strong cryptography (e.g., see [DL98] for a discussion of this issue), which Crowds does. As such, Crowds is presently available only in the United States and Canada.

5 Politics

Anonymity on the Internet is among the topics at the center of a larger social and political debate over electronic privacy, alongside topics like the escrow of cryptographic keys (e.g., see [Den93, DL98]). It is thus not surprising that Crowds has not received a uniformly warm welcome.

For example, some Internet vendors have expressed concerns about the proliferation of Crowds. When the only available anonymizing service on the Internet was the Anonymizer proxy, some Internet vendors noticed that a large percentage of purchases submitted via the Anonymizer used stolen credit card numbers. Thus, these vendors configured their web servers to refuse purchases submitted via the Anonymizer. When we announced our intention to release Crowds, some of these same vendors objected on the grounds that it would be difficult to ascertain when a purchase is submitted via a crowd, and in particular because it would not be possible to use client IP address to either filter requests or to track those that submitted false credit cards. We were requested to add a special header field to each request that identifies it as coming from a crowd. However, because users could simply modify the code to eliminate any such header (the source code for Crowds is freely available), such a modification would not have been effective.

As a second example, it has come to our attention that at least one large company has instituted a policy prohibiting the use of Crowds at work. Apparently the management is concerned about

its employees viewing "inappropriate" material on the job and being unable to monitor who is requesting it. Private companies presently have the right to institute such policies, just as they have the right to read employee e-mail and monitor their phone conversations.

6 Conclusion

Crowds provides users with the ability to retrieve web content anonymously. Crowds provides strong anonymity properties that are well-suited for web transactions, and does so in the face of a broader range of adversaries than proxy-based anonymizers do. In this paper we have sketched how Crowds works and described the positive aspects and the limitations of this technology, as well as what we have learned from its deployment. As more people adopt this technology, it will become more useful because a larger crowd provides stronger anonymity with little effect on performance. The code is supported and freely available to citizens of the United States and Canada from `http://www.research.att.com/projects/crowds`.

References

[Cha81] D. Chaum. Untraceable electronic mail, return adresses, and digital pseudonyms. *Communications of the ACM*, 24(2):84–88, February 1981.

[Den93] D. Denning, et.al. To tap or not to tap. *Communications of the ACM* 36(3):24–44, March 1993.

[DL98] W. Diffie and S. Landau. *Privacy on the Line: The Politics of Wiretapping and Encryption.* MIT Press, 1998.

[Dro93] R. Droms. Dynamic host configuration protocol. RFC-1541, October 27, 1993.

[GGMM97] E. Gabber, P. Gibbons, Y. Matias, and A. Mayer. How to make personalized web browsing simple, secure, and anonymous. In *Proceedings of Financial Cryptography '97*, 1997.

[GS97] S. Garfinkel and G. Spafford. *Web Security and Commerce.* O'Reilly & Associates, 1997.

[RC98] J. Reagle and L. Cranor. The platform for privacy preferences. *Communications of the ACM*, this issue.

[Rei98] M. K. Reiter, V. Anupam, and A. Mayer. Detecting hit-shaving in click-through payment schemes. In *Proceedings of the 3rd USENIX Workshop on Electronic Commerce*, pages 155–166, August 1998.

[RR98] M. K. Reiter and A. D. Rubin. Crowds: Anonymity for web transactions. *ACM Transactions on Information and Systems Security* 1(1), April 1998.

[SGR97] P. F. Syverson, D. M. Goldschlag, and M. G. Reed. Anonymous connections and onion routing. *In Proceedings of the 1997 IEEE Symposium on Security and Privacy*, May 1997.