An innovative way to become an invisible user is simply to get lost in the crowd. After all, anonymity loves company.

Anonymous Web Tran

with Crowds

WEB SERVER LOG files are riddled with information about the users who 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 is using, the Web page that referred her to this site and, with some effort, the server that she visits next [7]. Even when the user's IP address changes between browsing sessions (for example, the IP address is assigned dynamically using DHCP [4]), 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, for example, the user's ISP 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 employee 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 been logged so universally and revealed so much about the personal tastes of its users. Thus, our thesis is Web users should have the ability to limit what information is revealed about them and to whom it is revealed. Some

Michael K. Reiter and Aviel D. Rubin

sactions

approaches, notably P3P (see Reagle and Cranor in this section), enable users to negotiate the conditions under which they 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 (for example, not the user's IP address), cannot be enforced, and are not presently widely practiced, we believe other methods are needed as well.

This article presents a system called Crowds that enables the retrieval of information over the Web without revealing so much potentially private information to several parties. The goal of Crowds is to make browsing anonymous, so that information about either the user or what information he or 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 messages from the user's machine. For example, in the case of the user's local gateway administrator, Crowds hides what sites the user is visiting from the administrator. 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 them.

How Crowds Works

Crowds works by collecting Web users into a geographically diverse group called a "crowd" that performs Web transactions on behalf of its members. 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.

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 his or her browser to employ the local jondo as a proxy for all network services, so that all communication by the browser is sent directly to the jondo. When the user then requests a URL via the 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 randomly chooses another member of the crowd, 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 the request was destined. This random choice is biased in favor of forwarding; that is, there is a systemwide parameter $p_f > 1/2$ that indicates the probability with which a jondo will choose to forward. The value of p_f impacts the anonymity properties offered by the system, as we will discuss later.

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 will show 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 backward 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 [8]). 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 (that is, 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, the paths of all jondos are forgotten and rerouted from scratch, so 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 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 jondos. This is useful for providing certain classes of anonymity properties. Other details of the Crowds protocols are described in [8].

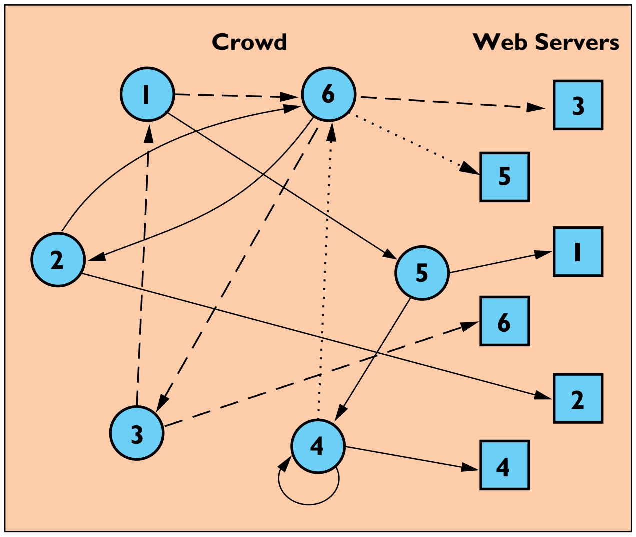

Figure 1. Paths in a crowd. The initiator and Web server of each path are labeled the same. Series of contiguous arrows of the same color, starting at a crowd member and ending at a Web server with the same label, denote a path through the crowd.

Anonymity Properties

Anonymous communication properties can be characterized on at least three different axes. The first 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 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 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 (for example, 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, from the attacker's point of view, if each sender appears no more likely to be the originator than to not be the originator.

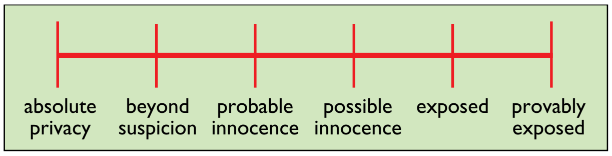

Figure 2. Degrees of anonymity range from absolute privacy, where the attacker cannot perceive the presence of communication, to provably exposed, where the attacker can prove the sender, receiver, or their relationship to others.

The anonymity properties achieved by Crowds are summarized in Table 1. 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 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.

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 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 collaborators know it is on the path. However, in [8], 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 1/2 provided the relationship between c, n, and p_f is as shown in Table 1. For example, if the probability of forwarding is $p_f = 3/4$ 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, for example, 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 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 (that is, 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.

Attacker	**Sender anonymity**	**Receiver anonymity**
local eavesdropper	exposed	P (beyond suspicion) $\xrightarrow[n\to\infty]{} 1$
c collaborating members, $n > \frac{p_f}{p_f - 1/2}(c+1)$	probable innocence P (absolute privacy) $\xrightarrow[n\to\infty]{} 1$	P (absolute privacy) $\xrightarrow[n\to\infty]{} 1$
end serves	beyond suspicion	N/A

Table 1. Anonymity properties provided by Crowds

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 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 he or she was running Crowds and did not originate the request. 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 (that is, 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, for example, 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 the user disable the proxy settings in his or 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, like via SSL (see [6]). However, our present implementation does not support this alternative.

Third, like any proxy-based anonymizing service, Crowds can be circumvented if the user's browser downloads and executes mobile code (such as 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 generally increases network traffic and load on the machines that execute jondos. (We refer the reader to [8] for details of response time experiments 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 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, indicating 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 Perl 5 language in which it is presently implemented. Moreover, organizing crowds so crowd members are in relatively close network proximity to each other should minimize inter-jondo communication delays.

Crowds prevents a Web server from learning any potentially identifying information about the user, INCLUDING EVEN THE USER'S IP ADDRESS OR DOMAIN NAME.

Comparison to Other Systems

The Anonymizer[1] 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; that is, the end server has no information about who initiated the request. A similar mechanism is the Lucent Personalized Web Assistant (LPWA), discussed in this section. This

[1] www.anonymizer.com

proxy provides explicit support for personalized Web browsing (such as browsing Web sites that require an account name and password) without revealing the user to the end server [5].

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 [1] have been developed to support anonymous Web communication, including Internet communication in general (for example, [9]). 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, but briefly, mix networks are designed to provide different anonymity properties than Crowds and do so using a more heavyweight protocol. We believe the anonymity properties provided by Crowds and the protocol by which they are achieved are better suited to synchronous communication (including Web transactions) than are mix networks.

Deployment Issues

At press time, we have distributed approximately 1,400 copies of the Crowds code in response to user requests, and presently maintain an active Crowds 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 presently on the Internet. 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 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, for example, 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 affects 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 user's connectivity adds latency not only to his or her own Web requests, but also to the requests sent on any path of which their 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 [8].

A final deployment issue we face is U.S. export controls on software containing strong cryptography (see [3] for a discussion of this issue), which Crowds does. As such, Crowds is presently available only in the U.S. and Canada.

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 (for example, see [2, 3]). Thus, it is 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 currently have the right to institute such policies, just as they have the right to read employee email and monitor their phone conversations.

In conclusion, Crowds provides users with the ability to retrieve Web content anonymously. It offers 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 article 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. C

References

1. Chaum, D. Untraceable electronic mail, return addresses, and digital pseudonyms. *Commun. ACM 24,* 2 (Feb. 1981), 84–88.
2. Denning, D. et.al. To tap or not to tap. *Commun. ACM 36,* 3 (Mar. 1993), 24–44.
3. Diffie, W., and Landau, S. *Privacy on the Line: The Politics of Wiretapping and Encryption.* MIT Press, Cambridge, Mass., 1998.
4. Droms, R. *Dynamic Host Configuration Protocol.* RFC-1541, Oct. 27, 1993.
5. Gabber, E., Gibbons, P., Matias, Y., and Mayer, A. How to make personalized Web browsing simple, secure, and anonymous. In *Proceedings of Financial Cryptography '97* (1997).
6. Garfinkel, S. and Spafford, G. *Web Security and Commerce.* O'Reilly, 1997.
7. Reiter, M.K., Anupam, V., and Mayer, A. Detecting hit-shaving in click-through payment schemes. In *Proceedings of the 3rd USENIX Workshop on Electronic Commerce.* (Aug. 1998), 155–166.
8. Reiter, M.K., and Rubin, A.D. Crowds: Anonymity for Web transactions. *ACM Trans. Info. Syst. Security 1,* 1 (Apr. 1998).
9. Syverson, D., Goldschlag, M., and Reed, M.G. Anonymous connections and onion routing. In *Proceedings of the 1997 IEEE Symposium on Security and Privacy.* (May 1997).

The code is supported and freely available to citizens of the U.S. and Canada from www.research.att.com/projects/crowds.

Michael K. Reiter (reiter@research.bell-labs.com) is Head of the Secure Systems Research Department at Bell Laboratories, Murray Hill, N.J.
Aviel D. Rubin (rubin@research.att.com) is a senior technical staff member at AT&T Labs–Research, Florham Park, N.J.

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, to republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.

© 1999 ACM 0002-0782/99/0200 $5.00