

DOI:10.1145/2714561

The Dissent system aims for a quantifiably secure, collective approach to anonymous communication online.

BY JOAN FEIGENBAUM AND BRYAN FORD

Seeking Anonymity in an Internet Panopticon

IN TODAY'S "BIG DATA" Internet, users often need to assume, by default, that their every statement or action online is monitored and tracked. Users' statements and actions are routinely linked with detailed profiles built by entities ranging from commercial vendors and advertisers to state surveillance agencies to online stalkers and criminal organizations. Indeed, recent revelations have raised the stakes enormously in Internet monitoring. Documents leaked by former National Security Agency contractor Edward Snowden revealed the U.S. government is conducting warrantless surveillance on a massive scale, and the long-term goal of the National Security Agency is to be "able to collect virtually everything available in the digital world."¹⁶

Internet users often have a legitimate need to be anonymous, or "not named or identified" by *Webster's*

definition of the term, to protect their online speech and activities from being linked to their real-world identities. Although the study of anonymous-communication technology is often motivated by high-stakes use cases (such as battlefield communication, espionage, or political protest against authoritarian regimes), anonymity actually plays many well-accepted roles in established democratic societies. For example, paying cash, voting, opinion polling, browsing printed material in a book store or library, and displaying creativity and low-risk experimentalism in forums (such as Slashdot and 4chan) are everyday examples of anonymous activity. Author J.K. Rowling used a pen name on a 2013 post-Harry Potter novel, presumably not out of fear of censorship or reprisal but merely "to publish without hype or expectation and . . . to get feedback under a different name."²²

Obtaining and maintaining anonymity on the Internet is a challenge. The state of the art in deployed tools (such as Tor²⁰) uses "onion routing" to relay encrypted connections on a detour passing through randomly chosen relays scattered around the Internet. Onion routing is scalable, supports general-purpose point-to-point communication, and appears to be effective against many of the attacks

» key insights

- **With retailers, email service providers, advertisers, surveillance agencies, and stalkers all potentially monitoring, tracking, and profiling ordinary Internet users, those users can turn to anonymous communication to prevent the linking of their online activity to their real-world identities.**
- **Currently deployed anonymity tools, with Tor the best known, are based on "onion routing," a scalable general technique that is effective in many scenarios but inherently vulnerable to several attacks that are increasingly feasible.**
- **The Dissent project takes a collective approach to online anonymity, based on different algorithmic foundations from onion routing, offering concrete advantages, as well as some disadvantages, versus Tor.**



IMAGE BY ALICIA KUBISTA/ANDRIJ BORYS ASSOCIATES

currently known to be in use.¹⁰ Unfortunately, onion routing is also known to be vulnerable to several classes of attacks for which no solution is known or believed to be forthcoming soon; for example, using traffic confirmation, an attacker who compromises a major ISP or Internet exchange might in principle be able to de-anonymize many Tor users in a matter of days.¹² With intersection attacks, an adversary can rapidly narrow the anonymity of a target via actions linkable across time, much like Paula Broadwell and the “High Country Bandits” were de-anonymized.¹⁷ Finally, through software exploits or user error, an attacker can often circumvent anonymity tools entirely.²⁴

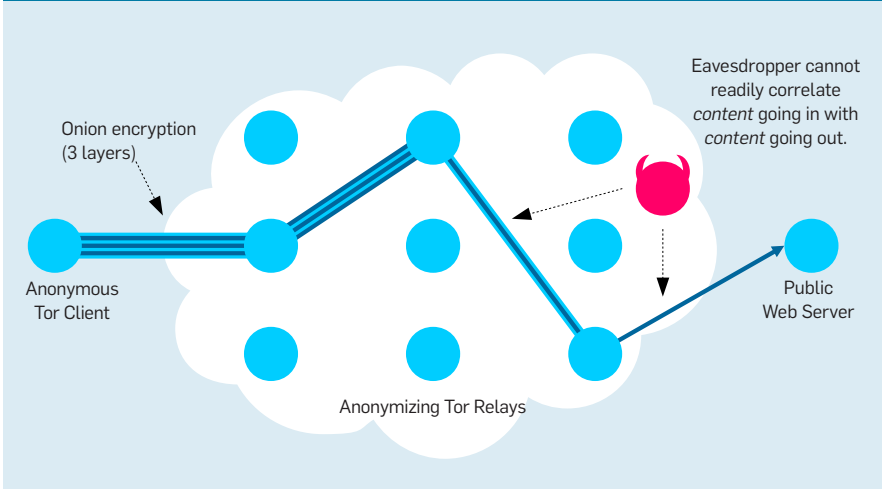
Currently deployed approaches to

anonymity also appear unable to offer accurate, principled measurement of the level or quality of anonymity a user might obtain. Considerable theoretical work has analyzed onion routing⁸ but relies on idealized formal models making assumptions that are unenforceable and may be untrue in real systems (such as users choose relays and communication partners at random) or depending on parameters unknown in practice (such as probability distributions representing user behavior).

Onion routing vulnerabilities and measurability limitations may stem from an attempt by developers of anonymity to achieve an impossible set of goals and defend an ultimately indefensible position. Currently deployed

tools offer a general-purpose, unconstrained, individualistic form of anonymous Internet access. However, many methods are available for “fingerprinting,” or tying unconstrained, individualistic network communication patterns to individual users. We suspect the only way to achieve measurable, provable levels of anonymity, and stake out a position defensible in the long term, is to develop more collective anonymity protocols and tools. It may be necessary for anonymity tools to constrain the normally individualistic behaviors of participating nodes, along with the expectations of users and possibly the set of applications and usage models to which these protocols and tools apply.

Figure 1. Onion routing.



Toward this end, we offer a high-level view of the Dissent project, a “clean-slate” effort at Yale University that began in the fall of 2009 to build practical anonymity systems embodying a collective model for anonymous communication (<http://dedis.cs.yale.edu/dissent/>). Dissent’s collective approach to anonymity is not and may never be a “drop-in” functional replacement for Tor or the individualistic, point-to-point onion routing model it implements. Rather, Dissent sets out to explore radically different territory in the anonymous-communication design domain, an approach that presents advantages, disadvantages, and many as-yet-unanswered questions. An advantage is the collective approach, making it easier to design protocols that provably guarantee certain well-defined anonymity metrics under arguably realistic environmental assumptions. A disadvantage is the collective approach is most readily applicable to multicast-oriented communication and is much less efficient or scalable than onion routing for point-to-point communication.

Dissent follows in the tradition of Herbivore,¹⁸ the first attempt (2003–2004) to build provable anonymity guarantees into a practical system and employ “dining cryptographers,” or DC-nets.³ Dissent utilizes both DC-nets and “verifiable shuffles,”¹⁵ showing for the first time how to scale the formal guarantees embodied in these techniques to offer measurable anonymity sets on the order of thousands of participants.²³ Dissent’s methods

of scaling individual anonymity sets are complementary and synergistic with techniques Herbivore pioneered for managing and subdividing large peer-to-peer anonymity networks; combining these approaches could enable further scalability improvements in the future.

Dissent incorporates the first systematic countermeasures to major classes of known attacks (such as global traffic analysis and intersection attacks).^{14,25} Because anonymity protocols alone cannot address risks (such as software exploits or accidental self-identification), the Dissent project also includes Nymix, a prototype operating system that hardens the user’s computing platform against such attacks.²⁴ Even with Nymix, however, Dissent can offer only network-level anonymity, in which the act of communicating does not reveal which user sent which message. No anonymity system can offer users personal anonymity if they disclose, say, their real-world identities in their message content.

While Dissent is still a research prototype, not yet ready for widespread deployment and may never be a direct replacement for onion routing tools like Tor due to possibly fundamental trade-offs, we hope it will increase the diversity of practical approaches and tools available for obtaining anonymity online.

Next, we present onion routing and Tor basics. We then describe four problems with onion routing that have remained unsolved for many years and may, unfortunately, be unsolvable. We then provide an overview of the Dissent

approach to anonymous communication and, finally, discuss open problems and future directions.

Onion Routing and Tor

Tor is the most widely deployed, general-purpose system for anonymous Internet communication.²⁰ Tor’s technical foundation is onion routing¹¹ derived in turn from mixnets.⁵

Onion routing uses successive layers of encryption to route messages through an overlay network, such that each node knows the previous and the next node in the route but nothing else. More precisely, let (V, E) be a connected, undirected network and $R \subseteq V$ be a set of nodes serving as relays. The set R is known to all nodes in V , as is the public key K_r , usable in some globally agreed-upon public-key cryptosystem, for each node $r \in R$. There is a routing protocol any node in V can use to send a message to any other node, but the nodes do not need to know the topology (V, E) .

If node s wishes to send message M to node d anonymously, s first chooses a sequence (r_1, r_2, \dots, r_n) of relays. It then constructs an “onion” with n layers containing both the message and the routing information needed to deliver it without revealing node s ’s identity to any node except the first relay r_1 . The core of the onion is (d, M) , or the destination node and the message itself. The n th, or innermost, layer of the onion is

$$O_n = (r_n, \text{ENC}_{K_{r_n}}(d, M))$$

or the n th relay node and the encryption of the core under the n th relay’s public key. More generally, the i th layer O_i , $1 \leq i \leq n - 1$, is formed by encrypting the $(i + 1)$ st layer under the public key of the i th relay and then prepending the i th relay’s identity r_i :

$$O_i = (r_i, \text{ENC}_{K_{r_i}}(O_{i+1}))$$

When it has finished constructing the outermost layer

$$O_1 = (r_1, \text{ENC}_{K_{r_1}}(O_2))$$

node s sends $\text{ENC}_{K_{r_1}}(O_2)$ to r_1 , using the routing protocol of the underlay network (V, E) . When relay r_i , $1 \leq i \leq n$, receives the encryption of O_i with public

key K_{r_i} , it decrypts it using the private key k_{r_i} corresponding to K_{r_i} , thus obtaining both the identity of the next node in the route and the message it needs to send to this next node it sends using the underlying routing protocol. When $i = n$, the message is just the core (d, M) , because, strictly speaking, there is no O_{n+1} . We assume d can infer from routing protocol “header fields” of M that it is the intended recipient and need not decrypt and forward (see Figure 1).

Tor is a popular free-software suite based on onion routing. As explained on the Tor project website, <https://www.torproject.org>,²⁰ “Tor protects you by bouncing your communications around a distributed network of relays run by volunteers all around the world; it prevents somebody watching your Internet connection from learning what sites you visit, and it prevents the sites you visit from learning your [network] location.” The project provides free application software that can be used for Web browsing, email, instant messaging, Internet relay chat, file transfer, and other common Internet activities. Users can also obtain free downloads that integrate the underlying Tor protocol with established browsers and email clients. Moreover, Tor users can easily (but are not required to) transform their Tor installations into Tor relays, thus contributing to the overall capacity of the Tor network. Tor has more than two million daily users worldwide, with slightly over 15% of them in the U.S., and approximately 6,000 relays. These and other statistics are regularly updated on the Tor Metrics Portal.²¹

The IP addresses of Tor relays are listed in a public directory so Tor clients can find them when building circuits. (Tor refers to routes as “circuits,” presumably because Tor is typically used for Web browsing and other TCP-based applications in which traffic flows in both directions between the endpoints.) This makes it possible for a network operator to prevent its users from accessing Tor. The operator can simply disconnect the first hop in a circuit, or the connection between the client and the first Tor relay, because the former is inside the network and the latter is outside; this forces the Tor traffic to flow through a network gateway

where the operator can block it. Several countries that operate national networks, including China and Iran, have blocked Tor in precisely this way. Website operators can also block Tor users simply by refusing connections from the last relay in a Tor circuit; Craigslist is an example of a U.S.-based website that does so. As a partial solution, the Tor project supports “bridges,” or relays whose IP addresses are not listed in the public directory, of which there are approximately 3,000 today. Tor bridges are just one of several anti-blocking, or “censorship-circumvention,” technologies.

There is inherent tension in onion routing between low latency, one aspect of which is short routes (or, equivalently, low values of k), and strong anonymity. Because its goal is to be a low-latency anonymous-communication mechanism, usable in interactive, real-time applications, Tor uses three-layer onions, or sets $k = 3$, as in Figure 1. Despite this choice of small k , many potential users reject Tor due to its performance impact.⁶

Attacks on Onion Routing

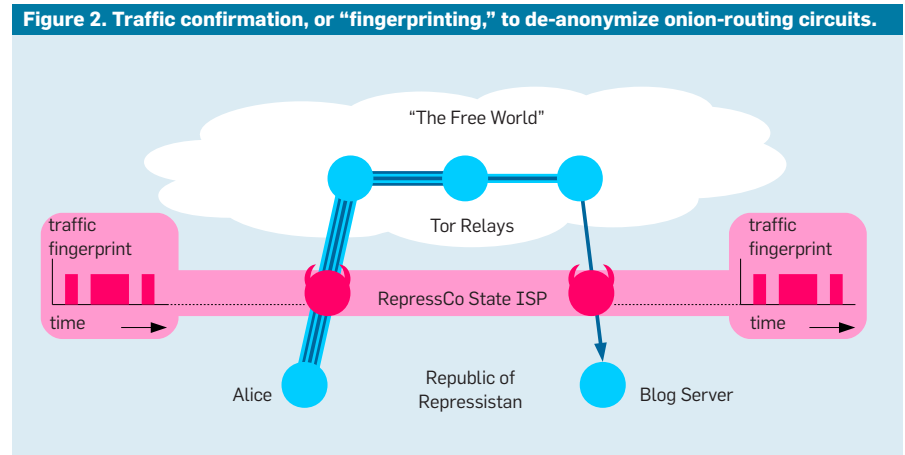
Four categories of known attacks to which onion routing is vulnerable and for which no general defenses are known are outlined in the following sections.

Global traffic analysis. Onion routing was designed to be secure against a local adversary, or one that might eavesdrop on some network links and/or compromise some relay nodes but only a small percentage of each. It was not designed for security against traffic analysis by a global adversary able to monitor large portions of the network constantly.

The most well known global-traffic-analysis attack—“traffic confirmation”—was understood by Tor’s designers but considered an unrealistically strong attack model and too costly to defend against.²⁰ In the standard scenario (see Figure 2), we assume the attacker cannot break Tor’s encryption but can monitor both the encrypted traffic flowing from the user to the first, or “entry” relay, and the traffic flowing from the final, or “exit” relay, to the user’s communication partner. This situation, while unlikely a decade ago, might be realistic today if both the user and the communication target are located in a single country, and the attacker is an ISP controlled or compromised by a state-level surveillance agency. In this case, the attacker needs to monitor, in principle, only the entry and exit traffic streams and correlate them through known fingerprinting methods.

For decades, this “global-passive-adversary” attack model was regarded as unrealistically strong and used to justify “conservative” assumptions in formal models.⁸ Unfortunately, this adversarial model is now not only realistic but in fact too weak. With the commercialization and widespread deployment of routers able to perform deep packet inspection and modification, including “man-in-the-middle” attacks against encrypted SSL streams at line rate,⁹ it has become clear to security and privacy professionals that any realistic adversary must be assumed to be active, or able to modify traffic streams at will.


Active attacks. An attacker’s ability to interfere actively in an anonymity network creates an array of new attacks, as




well as ways to strengthen existing traffic-analysis attacks. Figure 3 outlines one type of congestion attack⁷ in which we assume the attacker can directly monitor only one hop of a Tor circuit (such as the traffic from the exit relay to the target Web server). The attacker in this case might be “in the network” or simply own or have compromised the Web server. The attacker wishes to determine the set of relays through which a long-lived circuit owned by a particular user has passed.

The attacker chooses one relay at a time from Tor’s public database and remotely attempts to increase its load by congesting it; for example, the attacker might simulate many ordinary Tor users to launch a denial-of-service attack on the relay. The attacker’s power can be amplified by creating artificially long “flowerpetal” circuits that visit the target relay multiple times, each visit interspersed with a visit to another relay, as in Figure 3. Regardless of how congestion is incurred, it slows all circuits passing through the relay, including the victim circuit, if and only if the circuit passes through the targeted relay. The attacker can thus test whether a particular victim circuit flows through a particular router simply by checking whether the victim circuit’s average throughput (which can be measured at any point along the circuit) slows down during the period of attacker-generated congestion. The attacker repeatedly probes different relays this way until the victim’s entry and middle relays are identified. Finally, the attacker might fully de-anonymize the user by focusing traffic analysis on, or hacking, the user’s entry relay.

Intersection attacks. In most practical uses of anonymous communication, a user typically needs to send not just a single “one-off” message anonymously but a sequence of messages explicitly related and hence inherently linkable to each other; for example, Tor clients must maintain persistent TCP connections and engage in back-and-forth “conversations” with websites in order to support interactive communication, sending new HTTP requests that depend on the Web server’s responses to the client’s previous HTTP requests. It is manifestly obvious, at least to the Web server (and probably



Dissent preserves maximum security provided only that not all of a group’s servers maliciously collude against their clients.



to any eavesdropper who can monitor the connection between the Tor exit relay and the website), which packets comprise the same Web communication session, even if it is not (yet) clear who initiated the session. Further, if the user leaves an anonymous browser window open for an extended period or regularly logs into the same Web-based online email account, an eavesdropper might be able to link many of the user’s browsing sessions together over a long period of time. Even if each message gives the attacker only a small and statistically uncertain amount of information, just slightly narrowing the identity of the anonymous user, combining this information across many observation points at different times rapidly strengthens the attacker’s knowledge and can eventually identify and de-anonymize the target.

In one example of this attack (see Figure 4), an authoritarian government compels its ISPs or cellular carriers to turn over logs of which customers were online and actively using the network during which periods of time. An anonymous dissident posts blog entries to a pseudonymous blog at different points in time. Assume the attacker controls none of the user’s onion relays. Neither does the attacker control the blog server but merely observes the times at which the blog entries appeared and the fact the posts are manifestly linkable to each other, and so can correlate this information with the ISP logs. Perhaps the subject of the blog is official corruption in a particular city, enabling the authoritarian state to guess the dissident lives in that city and narrow attention to a small set of local ISPs. The attacker merely retrieves the sets of users who were online at each time a blog post appeared and intersects those sets. Although many thousands of users may be online at each of these posting times individually, all users other than the dissident in question are likely to have gone offline during at least one of these times (due to normal churn, the partly random comings and goings of most users), allowing the attacker to eliminate them from the victim’s anonymity set. The attacker needs only to “wait and watch” until the dissident has posted enough blog entries, and

the intersection of the online-user sets will shrink to a singleton.

The strength of this attack in practice is amply demonstrated by the fact that similar reasoning is used regularly in law enforcement.¹⁷ When an anonymous bomb threat was posted at Harvard via Tor in December 2013, the FBI caught the student responsible by effectively intersecting the sets of Tor users and Harvard network users at the relevant time. Paula Broadwell, whose extramarital affair with General David Petraeus led to the end of his career as director of the CIA in 2012, was de-anonymized through the equivalent of an intersection attack. De-anonymized in similar fashion were the “High Country Bandits” in 2010, as, per Ars Technica, “... a rather grandiose name for a pair of middle-aged white men who had been knocking down rural banks in northern Arizona and Colorado, grabbing a few thousand dollars from a teller’s cash drawer and sometimes escaping on a stolen all-terrain vehicle.” Intersection attacks also are the foundation of the National Security Agency’s CO-TRAVELER cellphone-location program linking known surveillance targets with unknown potential targets as their respective cellphones move together from one cell tower to another.

Software exploits and self-identification. No anonymous communication system can succeed if other software the user is running gives away the user’s network location. In an attack against the Tor network detected in August 2013, a number of “hidden services,” or websites with locations protected by Tor and accessible only through Tor, were compromised so as to send malicious JavaScript code to all Tor clients that connected to them (see Figure 5). This JavaScript code exploited a vulnerability in a particular version of Firefox distributed as part of the Tor Browser Bundle. This code effectively “broke out” of the usual JavaScript sandbox and ran native code as part of the browser’s process. This native code then invoked the host operating system to learn the client’s true (de-anonymized) IP address, MAC address, and more, sending them to an attacker-controlled server. The attacker in this case was initially suspected and later confirmed to be the FBI, employing “black hat” hacking techniques to take

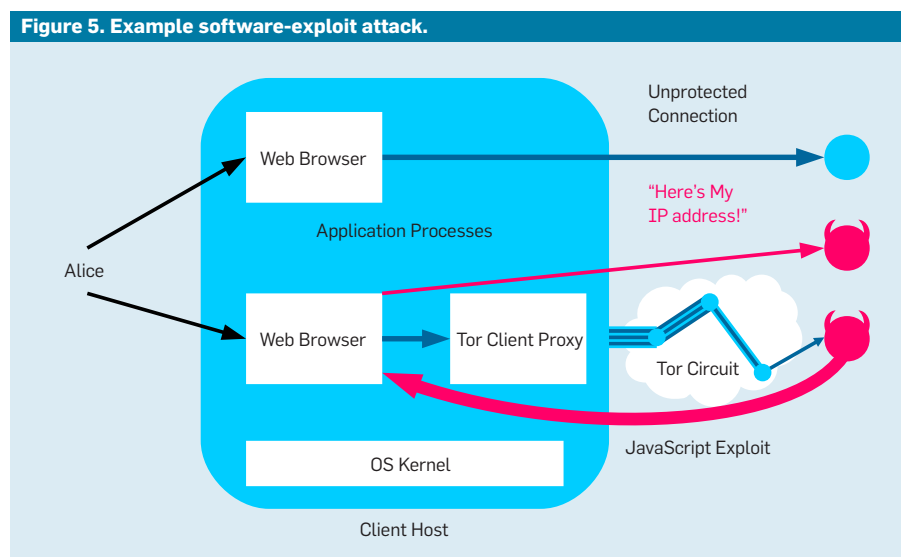
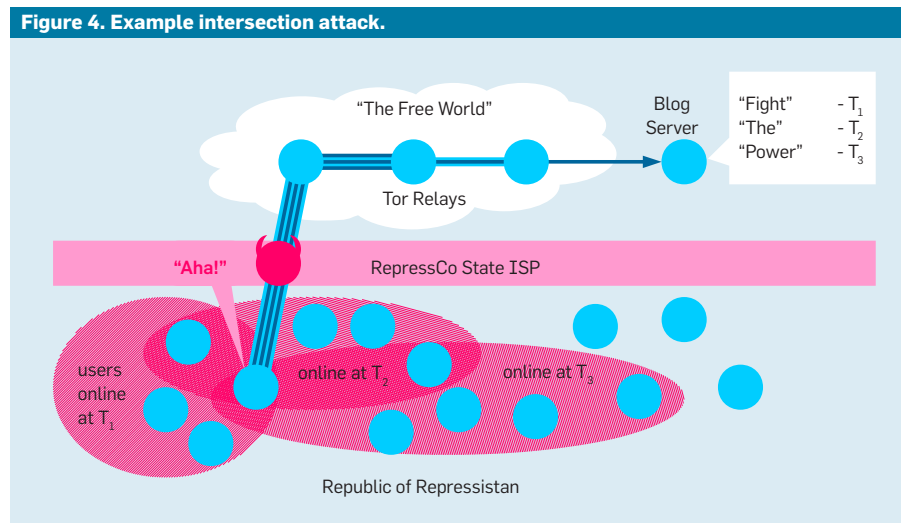
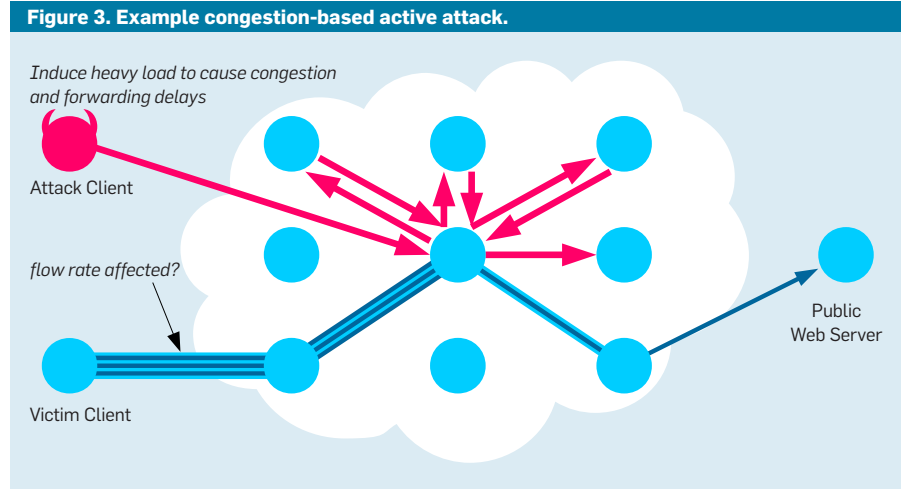
down hidden services carrying child pornography and trace their users.

Collective Anonymity in Dissent

As a step toward addressing these challenges, we introduce Dissent, a project

at Yale University that expands the design space and explores starkly contrasting foundations for anonymous communication.

Alternative foundations for anonymity. Quantification and formal analysis



of onion routing security under realistic conditions has proved an elusive goal.⁸ Dissent thus builds on alternative anonymity primitives (such as verifiable shuffles and dining cryptographers) with more readily provable properties.

Verifiable shuffles. In a typical cryptographic shuffle, participating nodes play two disjoint roles: a set of n clients with messages to send and a set of m shufflers that randomly permute these messages. Communication proceeds in synchronous rounds. In each, each of the n clients encrypts a single message under m concentric layers of public-key encryption, using each of the m shufflers' public keys, in a standardized order. All n clients send their ciphertexts to the first shuffler, which holds the private key to the outermost layer of encryption in all the clients' ciphertexts. The first shuffler waits until it receives all n clients' ciphertexts, then unwraps this outermost encryption layer, randomly permutes the entire set of ciphertexts, and forwards the permuted batch of n ciphertexts to the next shuffler. Each shuffler in turn unwraps another layer of encryption, permutes the batch of ciphertexts, and then forwards them to the next shuffler. The final shuffler then broadcasts all the fully decrypted cleartexts to all potentially interested recipients.

In an "honest-but-curious" security model in which we assume each shuffler correctly follows the protocol

(without, say, inserting, removing, or modifying any ciphertexts), the output from the last shuffler offers provable anonymity among all non-colluding clients, provided at least one of the shufflers keeps its random permutation secret. Unfortunately, if any of the shufflers is actively dishonest, this anonymity is easily broken. For example, if the first shuffler duplicates the ciphertext of some attacker-chosen client, the attacker may be able to distinguish the victim's cleartext in the shuffle's final output simply by looking for the cleartext that appears twice in the otherwise-anonymized output batch.

A substantial body of work addresses these vulnerabilities to such active attacks. In a "sender-verifiable" shuffle,^{2,4} each client inspects the shuffle's output to ensure its own message was not dropped, modified, or duplicated before allowing the shuffled messages to be fully decrypted and used. More sophisticated and complex provable shuffles (such as one by Neff¹⁵) enable each shuffler to prove to all observers the correctness of its entire shuffle, or that the shuffler's output is a correct permutation of its input, without revealing any information about which permutation it chose.

Both types of verifiable shuffles offer cryptographic guarantees that the process of shuffling reveals no information about which of the n clients submitted a given message appearing

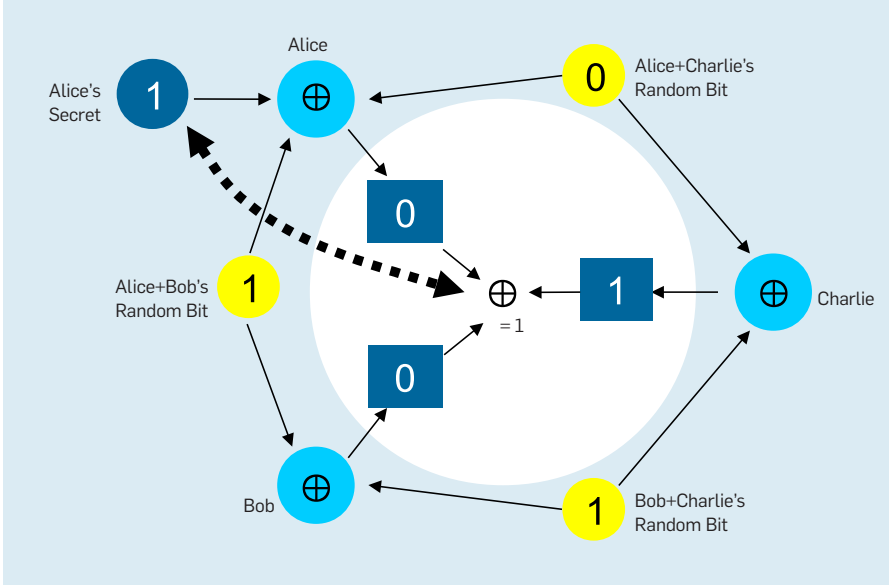
in the shuffled output. Shuffling has the practical disadvantage that the level of security achievable against potentially compromised shufflers depends on the number of shufflers in the path, and multiple shufflers must inherently be placed in sequence to improve security; in essence, latency is inversely proportional to security. The typical cascade arrangement, where all clients send their messages through the same sequence of shufflers at the same time, is most amenable to formal anonymity proofs but exacerbates the performance problem by creating the "worst possible congestion" at each shuffler in succession instead of randomly distributing load across many shufflers as an ad hoc, individualistic onion router network would.

For these reasons, verifiable shuffles may be practical only when high latencies are tolerable and shufflers are well provisioned. One relevant application is electronic voting, for which some shuffle schemes were specifically intended and which might readily tolerate minutes or hours of latency. A second application that arguably fits this model is "anonymous remailers,"¹⁵ which was popular before onion routing. Practical remailer systems have never, to our knowledge, employed state-of-the-art verifiable shuffles featuring anonymity proofs and were and remain vulnerable to active attacks analogous to the message-duplication attack described earlier.

Dining cryptographers. The only well-studied foundation for anonymity not based on sequential relaying is "dining cryptographers," or DC-nets, invented by Chaum³ in the late 1980s but never used in practical systems until two decades later by Herbivore.¹⁸ Instead of multi-hop message or packet relaying, DC-nets build on information-coding methods.

To illustrate how DC-nets operates, consider Chaum's classic scenario (see Figure 6), in which three cryptographers are dining at a restaurant when the waiter says their meal has been paid for. Suspicious, they wish to learn whether one of their group paid the bill anonymously or NSA agents at the next table paid it. So each adjacent pair of cryptographers flips a coin only the two can see. Each cryptographer XORs the coins to his left and right and writes the

Figure 6. The dining-cryptographers approach to anonymous communication; Alice reveals a one-bit secret to the group, but neither Bob nor Charlie learn which of the other two members sent the message.



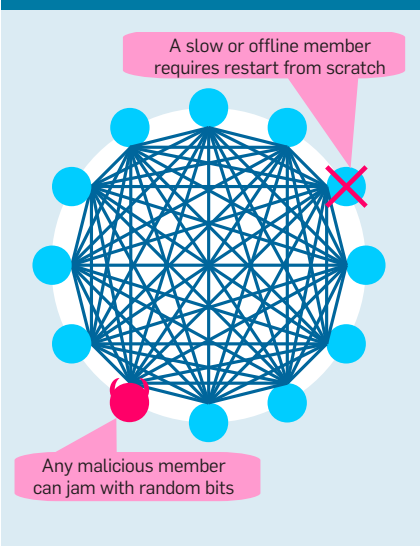
result on a napkin everyone can see—except any cryptographer who paid the bill (Alice in this case), who flips the result of the XOR. The cryptographers then XOR together the values written on all the napkins. Because each coin toss affects the values of exactly two napkins, the effects of the coins cancel out and have no effect on the final result, leaving a 1 if any cryptographer paid the bill (and lied about the XOR) or a 0 if no cryptographer paid. However, a 1 outcome provably reveals no information about which cryptographer paid the bill; Bob and Charlie cannot tell which of the other two cryptographers paid it, unless of course they collude against Alice.

DC-nets generalize to support larger groups and transmission of longer messages. Each pair of cryptographers typically uses Diffie-Hellman key exchange to agree on a shared seed for a standard pseudorandom-bit generator that efficiently produces the many “coin flips” needed to anonymize multi-bit messages. However, while theoretically appealing, DC-nets have not been perceived by anonymous communication tool developers as practical, for at least three reasons (see Figure 7). First, in groups of size N , optimal security normally requires all pairs of cryptographers share coins, yielding complexity $\Omega(N^2)$, both computational and communication. Second, large networks of “peer-to-peer” clients invariably exhibit high churn, with clients going offline at inopportune times; if a DC-nets group member disappears during a round, the results of the round become unusable and must be restarted from scratch. And third, large groups are more likely to be infiltrated by misbehaving members who might wish to block communication, and any member of a basic DC-nets group can trivially—and anonymously—jam all communication simply by transmitting a constant stream of random bits.

Practical dining cryptographers. Utilizing the DC-nets foundation in practical systems requires solving two main challenges: jamming and scalability. Herbivore¹⁸ pioneered exploration of practical solutions to both problems, and the Dissent project continues this work.

The jamming problem. Both Chaum’s original paper³ and many follow-up

Figure 7. Why scaling DC-nets is difficult in practice: worst case $N \times N$ coin-sharing matrix; network churn requires communications rounds to start over; and malicious members can anonymously jam the group.



works studied theoretical solutions to the jamming problem but were complex and to our knowledge never put into practice. Herbivore sidestepped the jamming problem by securely dividing a large peer-to-peer network into many smaller DC-nets groups, enabling participants who find themselves in an unreliable or jammed group to switch groups until they find a functioning one. This design has the advantage of scaling to support arbitrary-size networks, with the downside that participants obtain provable anonymity only within their own group—typically tens of nodes at most—and not guaranteeing anonymity within the larger network. Switching groups to avoid jamming can also introduce weaknesses to more intelligent attackers, who might run many Sybil nodes and selectively jam only groups they cannot compromise completely, all while offering good service in groups in which they have isolated a single “victim” node. The active attacker can thereby “prod” potential victims to switch groups until they land in a completely compromised group.¹

Dissent, the only system since Herbivore to put DC-nets into practice, explores different solutions to these challenges. First, it addresses the jamming problem by implementing accountability mechanisms, allowing the group to revoke the anonymity of any peer found to be attempting to jam commu-

nication maliciously while preserving strong anonymity protection for peers who “play by the rules.” Dissent’s first publicly available version introduced a conceptually simple and clean accountability mechanism that leveraged the verifiable-shuffle primitive discussed earlier, at the cost of requiring a high-latency shuffle between each round of (otherwise more efficient) DC-nets communication. The next version²³ in 2012 introduced a more efficient but complex retroactive-blame mechanism, allowing lower-latency DC-nets rounds to be performed “back-to-back” in the absence of jamming and requiring an expensive shuffle only once per detected jamming attempt.


However, an adversary who manages to infiltrate a group with many malicious nodes could still “sacrifice” them one-by-one to create extended denial-of-service attacks. Addressing this risk, a more recent incarnation of Dissent⁴ replaces the “coins” of classic DC-nets with pseudorandom elliptic-curve group elements, replaces the XOR combining operator with group multiplication, and requires clients to prove their DC-nets ciphertexts correct on submission, using zero-knowledge proofs. To avoid the costs of using elliptic-curve cryptography all the time, Dissent implements a hybrid mode that uses XOR-based DC-nets unless jamming is detected, at which point the system switches to elliptic-curve DC-nets briefly to enable the jamming victim to broadcast an accusation, yielding a more efficient retroactive-blame mechanism.

Scaling and network churn. Even with multiple realistic solutions to the jamming problem now available, DC-nets cannot offer useful anonymity if tools built using DC-nets can guarantee only anonymity-set size of at most tens of members. Herbivore addressed the $N \times N$ communication-complexity problem through a star topology in which a designated member of each group collects other members’ ciphertexts, XORs them together, and broadcasts the results to all members. However, without a general solution to the network churn and jamming problems, both Herbivore and the first version of Dissent were limited in practice to small anonymity sets comprising at most tens of nodes.


Addressing churn and scaling DC-nets further, Dissent now adopts a client/multi-server model with trust split across multiple servers, preferably administered independently. No single server is trusted; in fact, Dissent preserves full security provided only that not all of a group's servers maliciously collude against their clients. The clients need not know or guess which server is trustworthy but must trust only that at least one trustworthy server exists.

When a Dissent group is formed, the group's creator defines both the set of servers to support the group and the client-admission policy; in the simplest case, the policy is simply a list of public keys representing group members. Dissent servers thus play a role analogous to relays in Tor, serving to support the anonymity needs of many different clients and groups. Like Tor relays, the Dissent servers supporting a new group might be chosen automatically from a public directory of available servers to balance load. Choosing the servers for each group from a larger "cloud" of available servers in this way enables, in principle, Dissent's design to support an arbitrary number of groups, though the degree to which an individual group scales may be more limited. If a particular logical group becomes extremely popular, Herbivore's technique of splitting a large group into multiple smaller groups may be applicable. Our current Dissent prototype does not yet implement either a directory service or Herbivore-style subdivision of large networks.

While individual groups do not scale indefinitely, Dissent exploits its client/multi-server architecture to make groups scale two orders of magnitude beyond prior DC-nets designs.²³ Clients no longer share secret "coins" directly with other clients but only with each of the group's servers, as in Figure 8. Since the number of servers in each group is typically small (such as three to five, comparable to the number of Tor relays supporting a circuit), the number of pseudorandom strings each client must compute is substantially reduced. However, this change does not reduce anonymity, subject to Dissent's assumption that at least one server is honest. Chaum's DC-nets security proof³ ensures ideal anonymity, provided all



Public demand for anonymity online may intensify as a result of the ongoing surveillance scandal, thereby providing an opportunity to deploy new anonymity tools.



honest nodes are connected through the coin-sharing graph; Dissent satisfies this requirement, as the one honest server assumed to exist shares coins directly with all honest clients.

More important in practice, Dissent's client/multi-server coin-sharing design addresses network churn by making the composition of client ciphertexts independent of the set of other clients online in a given round. The servers set a deadline, and all clients currently online must submit their ciphertexts by that deadline or risk being "left out" of the round. Unlike prior DC-nets designs, if some Dissent clients miss the deadline, the other clients' ciphertexts remain usable. The servers merely adjust the set of client/server-shared secrets they use to compute their server-side DC-net ciphertexts.

Because each client's ciphertext depends on secrets it shares with all servers, no client's ciphertext can be used or decrypted unless all servers agree on the same set of online clients in the round and produce correct server-side ciphertexts based on that agreement. Malicious servers can do no worse than corrupt a round, and cannot de-anonymize clients except by colluding with all other servers.

How Dissent addresses attacks. Here, we outline how Dissent addresses the types of attacks discussed earlier.

Global traffic analysis. Dissent builds on anonymity primitives that have formal security proofs in a model where the attacker is assumed to monitor all network traffic sent among all participating nodes but cannot break the encryption. We have extended these formal security proofs to cover the first version of the full Dissent protocol;¹⁹ formal analysis of subsequent versions is in progress. Although verifiable shuffles differ from DC-nets in their details, both approaches share one key property that enables formal anonymity proofs: All participants act collectively under a common "control plane" rather than individually as in an ad hoc onion routing system; for example, they send identical amounts of network traffic in each round, though amounts and allocations may vary from round to round.

Active attacks. One countermeasure to traffic analysis in an onion router is

to “pad” connections to a common bit rate. While padding may limit passive traffic analysis, it often fails against active attacks, for reasons outlined in Figure 9. Suppose a set of onion router users pad the traffic they send to a common rate, but a compromised upstream ISP wishes to “mark” or “stain” each client’s traffic by delaying packets with a distinctive timing pattern. An onion router network that handles each client’s circuit individually preserves this recognizable timing pattern (with some noise) as it passes through the relays, at which point the attacker might recognize the timing pattern at the egress more readily than would be feasible with a traffic-confirmation attack alone. Active attacks also need not mark circuits solely through timing. A sustained attack deployed against Tor starting in January 2014 exploited another subtle protocol side-channel to mark and correlate circuits, going undetected for five months before being discovered by Tor project members on July 4, 2014 and subsequently thwarted (<https://blog.torproject.org/blog/tor-security-advisory-relay-early-traffic-confirmation-attack>).

In contrast, the collective-anonymity primitives underlying Herbivore and Dissent structurally keep the clients comprising an anonymity set in “lock-step” under the direction of a common, collective control plane. As in the popu-

lar children’s game “Simon Says,” participants transmit when and how much the collective control plane tells them to transmit. A client’s network-visible communication behavior does not leave a trackable fingerprint or stain, even under active attacks, because the client’s network-visible behavior depends only on this anonymized, collective control state; that is, a client’s visible behavior never depends directly on individual client state. Further, the Dissent servers implementing this collective control plane do not know which user owns which pseudonym or DC-nets transmission slot and thus cannot leak that information through their decisions, even accidentally.

Contrary to the intuition that defense against global traffic analysis and active attacks requires padding

traffic to a constant rate, Dissent’s control plane can adapt flow rates to client demand by scheduling future rounds based on (public) results from prior rounds. For example, the control-plane scheduler dynamically allocates DC-nets transmission bandwidth to pseudonyms that in prior rounds anonymously indicated a desire to transmit and hence avoids wasting network bandwidth or computation effort when no one has anything useful to say. Aqua, a project launched in 2013 at the Max Planck Institute for Software Systems in Germany to strengthen onion router security, employs a similar collective-control philosophy to normalize flow rates dynamically across an anonymity set.¹³ In this way, a collective control plane can in principle not only protect against

Figure 8. Improving scalability and churn resistance through asymmetric, client/server DC-nets architecture.

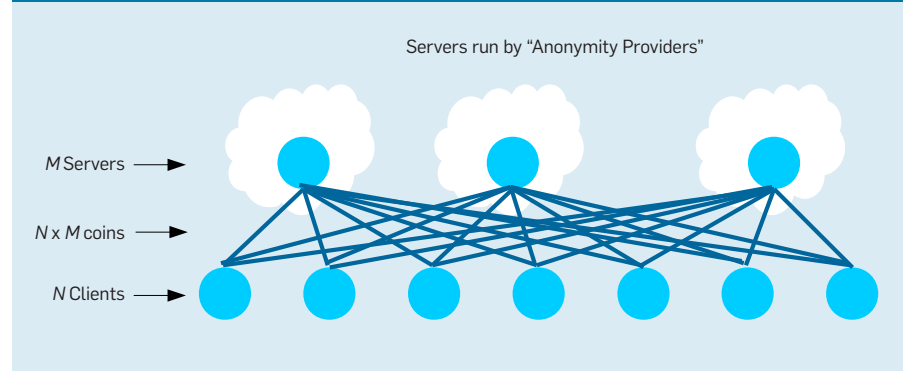
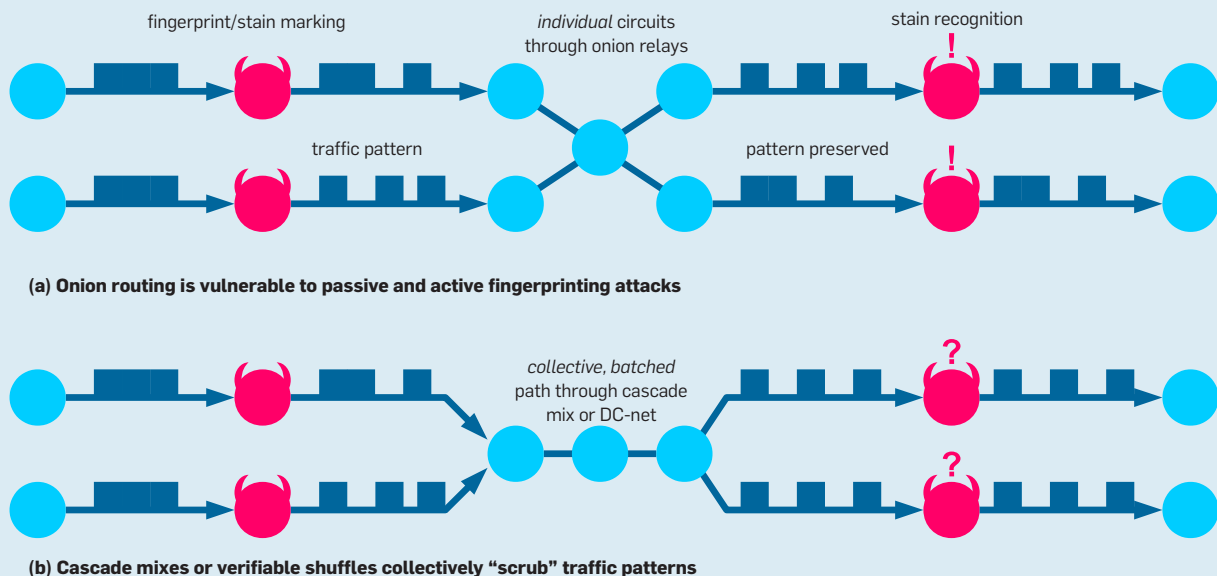


Figure 9. Fingerprinting or staining attacks.



both passive and active attacks but ironically also improve efficiency over padding traffic to a constant bit rate.

Intersection attacks. While the power and general applicability of intersection attacks have been studied extensively over the past decade, there is scant work on actually building mechanisms to protect users of practical systems against intersection attacks. The nearest precedents we are aware of suggest only that traffic padding may make intersection attacks more difficult, falling short of quantifying or controlling the effectiveness of such attacks.¹⁴ To the best of our knowledge, traffic padding proposals have never been implemented in deployed tools, in part because there is no obvious way to measure how much protection against intersection attacks a given padding scheme will provide in a real environment.

Dissent is the first anonymity system designed with mechanisms to measure potential vulnerability to intersection attacks, using formally grounded but plausibly realistic metrics, and offers users active control over anonymity loss under intersection attacks.²⁵ Dissent implements two different anonymity metrics: “possinymity,” a possibilistic measurement of anonymity-set size motivated by “plausible-deniability” arguments, and “indinymity,” an indistinguishability metric effective against stronger adversaries that may make probabilistic “guesses” via statistical disclosure.¹⁴

Users may set policies for long-lived pseudonyms, limiting the rate measured possinymity or indinymity may be lost or setting a threshold below which these metrics are not allowed to fall. Dissent’s collective control plane enforces these policies in essence by detecting when allowing a communication round to proceed might reduce a pseudonym’s possinymity or indinymity “too much” and in response suppressing or delaying communication temporarily.

The control plane can compute these metrics and enforce these policies even though its logic does not “know” which user actually owns each pseudonym. The downside is that employing these controls to resist intersection attacks can reduce the responsiveness, availability, and/or lifetime of a pseudonym. This cost reflects a fundamental trade-off between anonymity and availability.

Software exploits and self-identification. No anonymity protocol can by itself prevent de-anonymization through software exploits or user self-identification. Nevertheless, the Dis-

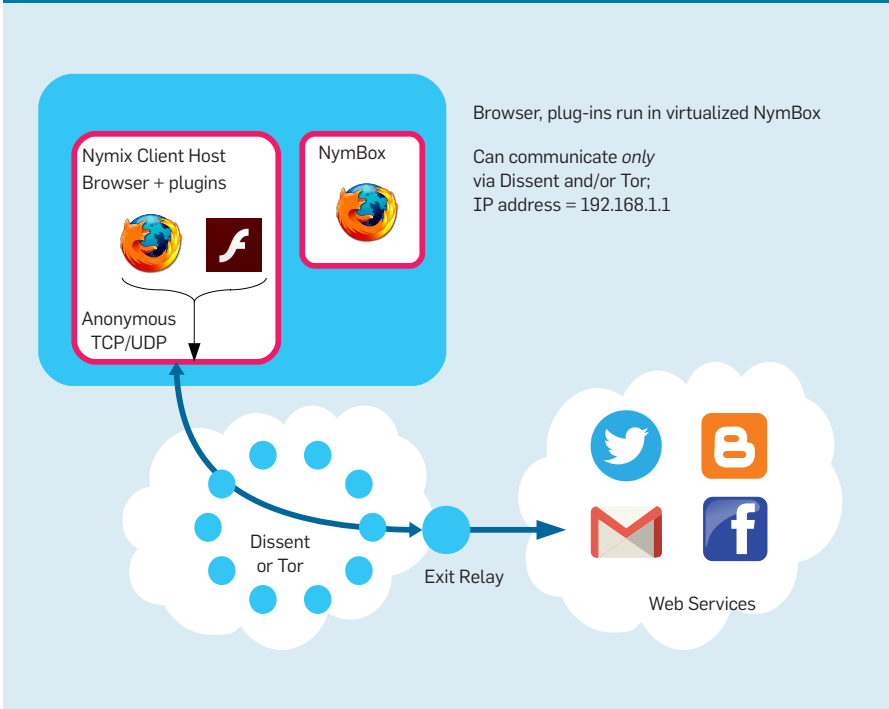
sent project is exploring system-level solutions through Nymix, a prototype USB-bootable Linux distribution that employs virtual machines (VMs) to improve resistance to exploits.²⁴

Nymix runs anonymity-client software (either Tor or Dissent) in the platform’s host operating system but isolates the browser and any plug-ins and other extensions it may depend on in a separate “guest VM,” as in Figure 10. No software in this guest VM is given access to information about the physical host OS or its network configuration; for example, the guest VM sees only a standard private (NATted) IP address (such as 192.168.1.1) and the fake MAC address of a virtual device. Even native code injected by a browser exploit (such as the one detected in August 2013 affecting the Windows version of the Tor Browser Bundle) would thus not be able to “leak” the client’s IP address without also breaking out of the VM. Escaping the VM as well may be possible, but the additional barrier increases attack difficulty.

Nymix binds guest-VM state instances to pseudonyms managed by the anonymity layer, enabling users to launch multiple simultaneous pseudonyms in different VMs, or “NymBoxes,” as in Figure 10. Nymix securely discards all pseudonym state embodied in a NymBox when appropriate to minimize the user’s long-term exposure to intersection attacks. This binding of pseudonyms to VMs makes it easy for the user to maintain state related to the context of one logical pseudonym (such as Web cookies and open logins) while offering stronger protection against the user’s accidentally linking different pseudonym VMs, because they appear as entirely separate OS environments, not just as different browser windows or tabs.

To reduce the risk of self-identification, Nymix allows the user to “move” data between non-anonymous contexts (such as personal .jpg photos stored on the host OS) and pseudonym-VM contexts only through a quarantine file system “drop box.” All files the user moves across browsing contexts in this way undergo a suite of tests to identify possibly compromising information (such as “exchangeable image file format,” or Exif, meta-

Figure 10. Using per-pseudonym virtual machines, or NymBoxes, to harden the client operating system against software exploits, stalling, and self-identification.



data within .jpg files). The quarantine system alerts users of any detected compromise risks, giving them the opportunity to scrub the file or decide not to transfer it at all. While all these defenses are inherently “soft” because there is only so much privacy-tool developers can do to prevent users from shooting themselves in the foot, Nymix combines these VM-based isolation and structuring principles to make it easier for users to make appropriate and well-informed uses of today’s, as well as tomorrow’s, anonymity tools.

Challenges and Future Work

Dissent takes a few important steps toward developing a collective approach to anonymous communication, but many practical challenges remain.

First, while DC-nets now scale to thousands of users, to support a global user population DC-nets must scale to hundreds of thousands of users or more. One approach is to combine Dissent’s scaling techniques with those of HerbiVore¹⁸ by dividing large anonymity networks into manageable anonymity sets (such as hundreds or thousands of nodes), balancing performance against anonymity guarantees. A second approach is to use small, localized Dissent clusters that already offer performance adequate for interactive Web browsing^{23,24} as a decentralized implementation for the crucial entry-relay role in a Tor circuit.²⁰ Much of a Tor user’s security depends on the user’s entry relay’s being uncompromised;¹² replacing this single point of failure with a Dissent group could distribute the user’s trust among the members of the group and further protect traffic between the user and the Tor relays from traffic analysis by “last mile” ISP adversaries.

Second, while Dissent can measure vulnerability to intersection attack and control anonymity loss,²⁵ it cannot also ensure availability if users exhibit high churn and individualistic “every user for themselves” behavior. Securing long-lived pseudonyms may be feasible only in applications that incentivize users to keep communication devices online constantly, even if at low rates of activity, to reduce anonymity decay caused by churn. Further, robust intersection-attack resistance may be practical only in applications designed to encourage users to act collectively

rather than individually and optimized for these collective uses.

Applications in which users cooperatively produce collective information “feeds” consumed by many other users may be well suited to Dissent’s collective anonymity model, including the interaction models of Internet relay chat, forums like Slashdot and Twitter, and applications supporting voting, deliberating, or “town hall” meetings. Given the close relationship between collective deliberation and the foundations of democracy and freedom of speech, such applications may also represent some of the most socially important use cases for online anonymity. But how best to support and incentivize cooperative behavior remains an important open problem.

Finally, large anonymity sets clearly require widespread public demand for anonymity. Tor’s two-million daily users are dwarfed in number by the number of users of Google, Facebook, Yahoo!, and other services that do not provide anonymity—and cannot provide it, because their business models depend crucially on exploiting personal information. Public demand for anonymity online may intensify as a result of the ongoing surveillance scandal, thereby providing an opportunity to deploy new anonymity tools.

Acknowledgments

This material is based on work supported by the Defense Advanced Research Projects Agency and SPAWAR Systems Center Pacific, contract no. N66001-11-C-4018. ■

References

1. Borisov, N., Danezis, G., Mittal, P., and Tabriz, P. Denial of service or denial of security? How attacks on reliability can compromise anonymity. In *Proceedings of the 14th ACM Conference on Computer and Communications Security* (Alexandria, VA, Oct. 29–Nov. 2), ACM Press, New York, 2007.
2. Brickell, J. and Shmatikov, V. Efficient anonymity-preserving data collection. In *Proceedings of the 12th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining* (Philadelphia, PA, Aug. 20–23), ACM Press, New York, 2006.
3. Chaum, D. The dining cryptographers problem: Unconditional sender and recipient untraceability. *Journal of Cryptology* 1, 1 (1988), 65–75.
4. Corrigan-Gibbs, H., Wolinsky, D.I., and Ford, B. Proactively accountable anonymous messaging in Verdict. In *Proceedings of the 22nd USENIX Security Symposium* (Washington, D.C., Aug. 14–16), USENIX Association, Berkeley, CA, 2013.
5. Danezis, G., Dingledine, R., and Mathewson, N. Mixminion: Design of a type III anonymous remailer protocol. In *Proceedings of the 2003 IEEE Symposium on Security and Privacy* (Oakland, CA, May 11–14). IEEE Computer Society Press, Los Alamitos, CA, 2003.
6. Dingledine, R. and Murdoch, S.J. Performance improvements on Tor, or why Tor is slow and what

we’re going to do about it. Presented at DEFCON 17 (Las Vegas, NV, July 30–Aug. 2, 2009); <https://svn.torproject.org/svn/projects/roadmaps/2009-03-11-performance.pdf>

7. Evans, N.S., Dingledine, R., and Grothoff, C. A practical congestion attack on Tor using long paths. In *Proceedings of the 18th USENIX Security Symposium* (Montreal, Canada, Aug. 10–14), USENIX Association, Berkeley, CA, 2009.
8. Feigenbaum, J., Johnson, A., and Syverson, P. Probabilistic analysis of onion routing in a black-box model. *ACM Transactions on Information and System Security* 15, 3 (2012), article 14.
9. Gallagher, R. New Snowden documents show NSA deemed Google networks a ‘target’. *Slate* (Sept. 9, 2013).
10. Gellman, B., Timberg, C., and Rich, S. Secret NSA documents show campaign against Tor encrypted network. *The Washington Post* (Oct. 4, 2013).
11. Goldschlag, D.M., Reed, M.G., and Syverson, P.F. Hiding routing information. In *Proceedings of the First International Workshop on Information Hiding* (Cambridge, U.K., May 30–June 1), Springer, Berlin, 1996.
12. Johnson, A., Wacek, C., Jansen, R., Sherr, M., and Syverson, P. Users get routed: Traffic correlation on Tor by realistic adversaries. In *Proceedings of the 20th ACM Conference on Computer and Communications Security* (Berlin, Germany, Nov. 4–8), ACM Press, New York, 2013.
13. Le Blond, S., Choffnes, D., Zhou, W., Druschel, P., Ballani, H., and Francis, P. Towards efficient traffic-analysis resistant anonymity networks. In *Proceedings of ACM SIGCOMM 2013* (Hong Kong, China, Aug. 12–16), ACM Press, New York, 2013.
14. Mathewson, N. and Dingledine, R. Practical traffic analysis: Extending and resisting statistical disclosure. In *Proceedings of the Fourth Workshop on Privacy Enhancing Technologies* (Toronto, Canada, May 24–26), Springer, Berlin, 2004.
15. Neff, C.A. A verifiable secret shuffle and its application to e-voting. In *Proceedings of the Eighth ACM Conference on Computer and Communications Security* (Philadelphia, PA, Nov. 6–8), ACM Press, New York, 2001.
16. Risen, J. and Poitras, L. NSA report outlined goals for more power. *The New York Times* (Nov. 22, 2013).
17. Segal, A., Ford, B., and Feigenbaum, J. Catching bandits and only bandits: Privacy-preserving intersection warrants for lawful surveillance. In *Proceedings of the Fourth USENIX Workshop on Free and Open Communications on the Internet* (San Diego, CA, Aug. 18), USENIX Association, Berkeley, CA, 2014.
18. Sizer, E.G., Goel, S., Robson, M., and Engin, D. Eluding carnivores: File sharing with strong anonymity. In *Proceedings of the 11th ACM SIGOPS European Workshop* (Leuven, Belgium, Sept. 19–22), ACM Press, New York, 2004.
19. Syta, E., Johnson, A., Corrigan-Gibbs, H., Weng, S.-H., Wolinsky, D.I., and Ford, B. Security analysis of accountable anonymity in Dissent. *ACM Transactions on Information and System Security* 17, 1 (2014), article 4.
20. Tor. *Anonymity Online*; <https://www.torproject.org>
21. Tor. Metrics portal; <http://metrics.torproject.org>
22. Watts, R. JK Rowling unmasked as author of acclaimed detective novel. *The Telegraph* (July 13, 2013).
23. Wolinsky, D.I., Corrigan-Gibbs, H., Johnson, A., and Ford, B. Dissent in numbers: Making strong anonymity scale. In *Proceedings of the 10th USENIX Symposium on Operating Systems Design and Implementation* (Hollywood, CA, Oct. 8–10), USENIX Association, Berkeley, CA, 2012.
24. Wolinsky, D.I., Jackowitz, D., and Ford, B. Managing NymBoxes for identity and tracking protection. In *Proceedings of the 2014 Conference on Timely Results in Operating Systems* (Broomfield, CO, Oct. 5), USENIX Association, Berkeley, CA, 2014.
25. Wolinsky, D.I., Syta, E., and Ford, B. Hang with your buddies to resist intersection attacks. In *Proceedings of the 20th ACM Conference on Computer and Communications Security* (Berlin, Germany, Nov. 4–8), ACM Press, New York, 2013.

Joan Feigenbaum (joan.feigenbaum@yale.edu) is the department chair and Grace Murray Hopper Professor of Computer Science at Yale University, New Haven, CT.

Bryan Ford (bryan.ford@epfl.ch) is an associate professor of computer and communication sciences at the Swiss Federal Institute of Technology (EPFL), Lausanne, Switzerland.

Copyright held by authors.