Seeking Anonymity in an Internet Panopticon

Joan Feigenbaum and Bryan Ford
Yale University

1. INTRODUCTION

In today's "Big Data" Internet, users often need to assume that, by default, 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 events have raised the stakes in Internet monitoring enormously. Documents leaked by Edward Snowden have revealed that the US government is conducting warrantless surveillance on a massive scale and, in particular, that the long-term goal of the National Security Agency is to be "able to collect virtually everything available in the digital world" [18].

Internet users often have legitimate need to be *anonymous* – *i.e.*, "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 or 4chan are everyday examples of anonymous activity. Author JK Rowling used a pen name on a recent post-Harry Potter novel, presumably not out of any 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 challenging, however. The state of the art in deployed tools, such as Tor [1], uses *onion routing* (OR) to relay encrypted connections on a detour passing through randomly chosen relays scattered around the Internet. OR is scalable, supports general-purpose point-to-point communication, and appears to be effective against many of the attacks currently known to be in use [12]. Unfortunately, OR is known to be vulnerable to several classes of attacks for which no solution is known or believed to be forthcoming soon. For example, via *traffic confirmation*, an attacker who compromises a major ISP or Internet exchange might in principle de-anonymize many Tor users in a matter of days [14]. Through *intersection attacks*, an adversary can rapidly narrow the anonymity of a target via actions linkable across time, in much the same way Paula Broadwell and the "High Country Bandits" were de-anonymized [19]. Finally, through software exploits or user error, an attacker can often circumvent anonymity tools entirely [24].

Current 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 analyzes onion routing [10], but relies on idealized formal models making assumptions that are unenforceable and may be untrue in real systems – such as that users choose relays and communication partners at random – or depending on parameters that are unknown in practice, such as probability distributions representing user behavior.

We believe the vulnerabilities and measurability limitations of onion routing may stem from an attempt to achieve an impossible set of goals and to defend an ultimately indefensible position. Current tools offer a general-purpose, unconstrained, and *individualistic* form of anonymous Internet access. However, there are many ways for unconstrained, individualistic uses of the Internet to be fingerprinted and tied to individual users. We suspect that the only way to achieve measurable and provable levels of anonymity, and to stake out a position defensible in the long term, is to develop more *collective* anonymity protocols and tools. It may be necessary to constrain the normally individualistic behaviors of participating nodes, the expectations of users, and possibly the set of applications and usage models to which these protocols and tools apply.

Toward this end, we offer a high-level view of the Dissent project, a "clean-slate" effort to build practical anonymity systems embodying a collective model for anonymous communication. 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. Instead, Dissent sets out to explore radically different territory in the anonymous-communication design space, an approach that presents advantages, disadvantages, and many as-yet-unanswered questions. An advantage is that the collective approach makes it easier to design protocols that provably guarantee certain well defined anonymity metrics under arguably realistic environmental assumptions. A disadvantage is that the collective approach is most readily applicable to multicast-oriented communication, and currently much less efficient or scalable than OR for point-to-point communication.

Dissent follows in the tradition of Herbivore [20], the first attempt to build provable anonymity guarantees into a practical system, and to employ dining cryptographers or DC-nets [5]. Dissent utilizes both DC-nets and verifiable shuffles [17], 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-

This material is based upon work supported by the Defense Advanced Research Projects Agency (DARPA) and SPAWAR Systems Center Pacific, Contract No. N66001-11-C-4018.

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 [16, 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]. Dissent and Nymix OS can of course 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, for example, they disclose their real-world identities in their message content.

While at this time Dissent is still a research prototype not yet ready for widespread deployment, and may never be a direct replacement for OR tools such as Tor because of possibly fundamental tradeoffs, we hope that it will increase the diversity of practical approaches and tools available for obtaining anonymity online.

Section 2 presents the basics of OR and Tor. In Section 3, we describe four problems with OR that have gone unsolved for many years and may unfortunately be unsolvable. Section 4 provides an overview of the Dissent approach to anonymous communication, and Section 5 contains open problems and future directions.

2. ONION ROUTING AND TOR

Currently the most widely deployed, general-purpose system for anonymous Internet communication is Tor [1]. Tor's technical foundation is onion routing [13], derived in turn from mixnets [7].

Onion routing (OR) 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 that any node in V can use to send a message to any other node, but the nodes need not know the topology (V, E).

If node s wishes to send message M to node d anonymously, s first chooses a sequence $(r_1, r_2, \ldots, r_n)$ of relays. It then constructs an "onion" whose n layers contain 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), *i.e.,* the destination node and the message itself. The n^{th} or innermost layer of the onion is

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

i.e., 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, $1 \le i \le k-1$, is formed by encrypting the $(i+1)^{\text{st}}$ layer under the public key of the i^{th} relay and then prepending the i^{th} relay's identity r_i:

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

Once 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, $1 \le i \le n$, receives $\text{ENC}_{K_{r_i}}(O_{i+1})$, 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 that it needs to send to this next node (which 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 that 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.

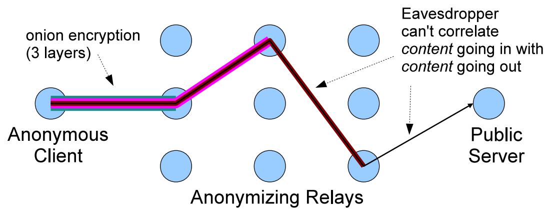

onion encryption (3 layers)
Eavesdropper can't correlate content going in with content going out
Anonymous Client
Public Server
Anonymizing Relays

Figure 1: Onion routing (OR).

Tor is a popular free-software suite based on OR. As explained on the Torproject website [1], "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, email clients, *etc.* Importantly, 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. Currently, there are approximately 40M "mean daily users" of Tor worldwide, slightly over 10% of whom are in the United States, and approximately 4700 relays. These and other statistics are regularly updated on the Tor Metrics Portal [2].

The IP addresses of Tor relays are listed in a public directory so that 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.) Clearly, 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, *i.e.*, 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, at which the operator can block it. Several countries that operate national networks, including China and Iran, have blocked Tor in precisely this way. Similarly, website operators can block Tor users simply by refusing connections from the last relay in a Tor circuit; Craigslist is an example of a US-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 currently approximately 2000. Tor bridges are just one of several anti-blocking or *censorship-circumvention* technologies.

There is inherent tension in OR 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 3-layer onions, *i.e.*, sets $k = 3$ as in Figure 1. Despite this choice of small k, many potential users reject Tor because of its performance impact [8].

3. ATTACKS ON ONION ROUTING

We now summarize four categories of known attacks to which OR is vulnerable and for which no general defenses are known.

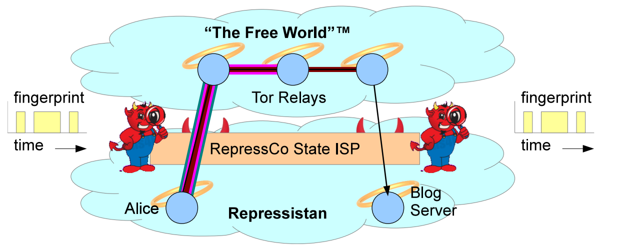


Figure 2: Traffic confirmation or fingerprinting to de-anonymize onion-routing circuits

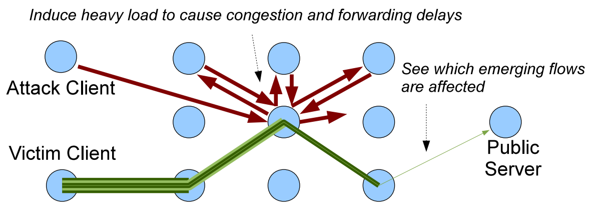

Induce heavy load to cause congestion and forwarding delays
Attack Client
See which emerging flows are affected
Victim Client
Public Server

Figure 3: Example of a congestion-based active attack

Global traffic analysis.

OR was designed to be secure against a *local adversary*, *i.e.*, 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* that can 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 [1]. In the standard scenario illustrated in Figure 2, we assume that 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 her 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 in principle need only monitor the entry and exit traffic streams and correlate them via 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 [10]. Unfortunately, this adversarial model is now not only realistic but in fact too weak. With the commercialization and widespread deployment of routers that can perform deep packet inspection and modification, including "Man-in-the-Middle attacks" against encrypted SSL streams at line rate [11], it has become clear that any realistic adversary must be assumed to be active, *i.e.*, able to modify traffic streams at will.

Active attacks.

The ability for an attacker to interfere actively in an anonymity network creates a wide array of new attacks as well as ways to strengthen existing traffic-analysis attacks. Figure 3 illustrates one example of a *congestion attack* [9]. In this scenario, we assume that the attacker can directly monitor only one hop of a Tor circuit, *e.g.*, the traffic from the exit relay to the target web server. The attacker in this case might be "in the network" or might 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 passes.

The attacker chooses one relay at a time from Tor's public database and remotely attempts to increase that relay's 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 can amplify his power by creating artificially long "flower-petal" circuits that visit the target relay multiple times, each visit interspersed with a visit to another relay, as shown in Figure 3. Regardless of how congestion is incurred, it slows all circuits passing through this relay, including the victim circuit, if and only if that circuit passes through the targeted relay. The attacker can therefore 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 he identifies the victim's entry and middle relays. Finally, the attacker might fully de-anonymize the user by focusing traffic analysis on, or hacking, the user's entry relay.

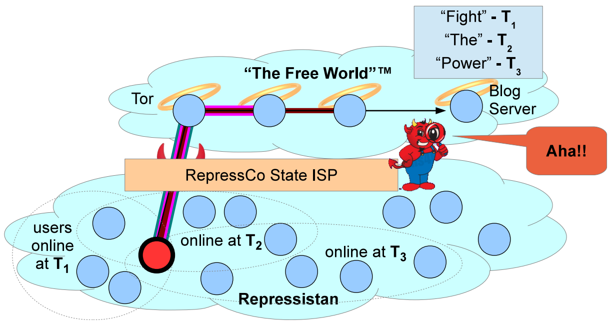


Figure 4: Example of an intersection attack

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 that are explicitly related and hence inherently linkable to each other. For example, Tor clients need to maintain persistent TCP connections and engage in back-and-forth "conversations" with web sites 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 to any eavesdropper who can monitor the connection between the Tor exit relay and the web site) which packets comprise the same Web communication session, even if it is not (yet) clear who initiated that session. Further, if the user leaves an anonymous browser window open for an extended period or regularly logs into the same anonymous Web mail account, an eavesdropper may 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.

In one example of this attack illustrated in 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

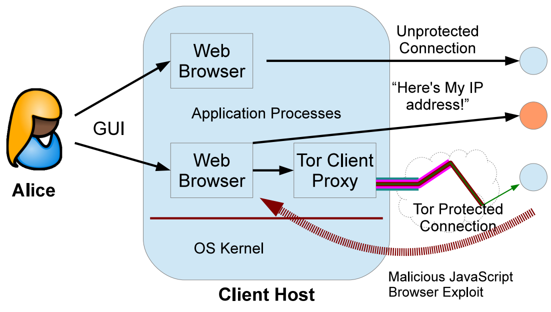


Figure 5: Example of a software-exploit attack

that the attacker controls none of the user's onion relays. Nor does he control the blog server; he merely observes the times at which the blog entries appeared and the fact that the posts are manifestly linkable to each other, and he 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 that 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 there may be many thousands of users 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 (because of 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 simply needs 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 [19]. The FBI caught a Harvard student who used Tor to post a bomb threat by effectively intersecting the sets of Tor users and Harvard network users at the relevant time. Paula Broadwell was de-anonymized via the equivalent of an intersection attack, as were the "High Country Bandits". Intersection attacks also form the foundation of the NSA's CO-TRAVELER program, which links 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 his network location. In a recent attack against the Tor network, illustrated in Figure 5, a number of *hidden services* (web sites whose locations are protected by Tor and which can be accessed only via Tor) were compromised so as to send malicious JavaScript code to all Tor clients who connected to them. This malicious JavaScript exploited a vulnerability in a particular version of Firefox distributed as part of the Tor Browser Bundle. This exploit effectively "broke out" of the usual JavaScript sandbox and ran native code as part of the browser's process. This native code simply invoked the host operating system to learn the client's true (de-anonymized) IP address, MAC address, *etc.*, and sent them to an attacker-controlled server.

4. COLLECTIVE ANONYMITY IN DISSENT

As a step toward addressing these challenges, we now introduce Dissent, a project that expands the design space and explores starkly contrasting foundations for anonymous communication.

4.1 Alternative foundations for anonymity

Quantification and formal analysis of OR security under realistic conditions has proven an elusive goal [10]. Dissent therefore builds on alternative anonymity primitives with more readily provable properties: verifiable shuffles and dining cryptographers.

Verifiable shuffles.

In a typical *cryptographic shuffle*, participating nodes play two disjoint roles: there is a set of n *clients* with messages to send and a set of m *shufflers* that randomly permute those messages. Communication proceeds in synchronous rounds. In each round, 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, 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, for example, 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, then 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 [4, 6], each client inspects the shuffle's output to ensure that 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 Neff's [17], enable each shuffler to prove to all observers the correctness of its entire shuffle, *i.e.*, 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 above, 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 OR 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* [7], which were popular be-

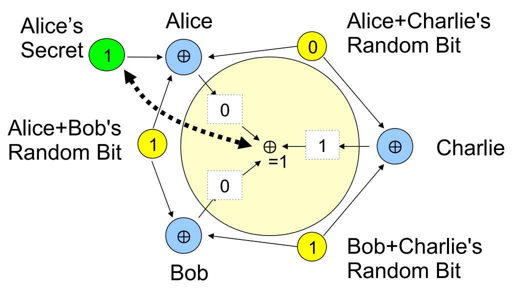


Figure 6: The Dining Cryptographers approach to anonymous communication. Alice reveals a 1-bit secret to the group, but neither Bob nor Charlie learn which of the other two members sent this message.

fore onion routing. Practical remailer systems have never to our knowledge employed state-of-the-art verifiable shuffles featuring anonymity proofs, however, and were vulnerable to active attacks analogous to the message duplication attack mentioned above.

Dining cryptographers.

The only well studied foundation for anonymity not based on sequential relaying is *Dining Cryptographers* or *DC-nets*, invented by David Chaum in the late 1980s [5] but never used in practical systems until two decades later by Herbivore [20]. Instead of relaying, DC-nets build on information-coding methods.

Consider Chaum's standard scenario, illustrated in Figure 6. Three cryptographers are dining at a restaurant when the waiter informs them that their meal has been paid for. Growing 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 that only the two can see. Each cryptographer XORs the coins to his left and right and writes the 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 of the final result, leaving a 1 if any cryptographer paid the bill (and lied about the XOR) or a 0 if no cryptographer paid. A 1 outcome provably reveals no information about which cryptographer paid the bill, however: Bob and Charlie cannot tell which of the other two cryptographers paid it (unless of course they collude against Alice).

DC-nets generalize readily to support larger groups and transmission of longer messages. Typically each pair of cryptographers uses Diffie-Hellman key exchange to agree on a shared seed for a standard pseudorandom-bit generator, which efficiently produces the many "coin flips" needed to anonymize multi-bit messages. While theoretically appealing, however, DC-nets have not been perceived as practical, for at least three reasons illustrated in Figure 7. First, in groups of size N, optimal security normally requires all pairs of cryptographers to 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. 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.

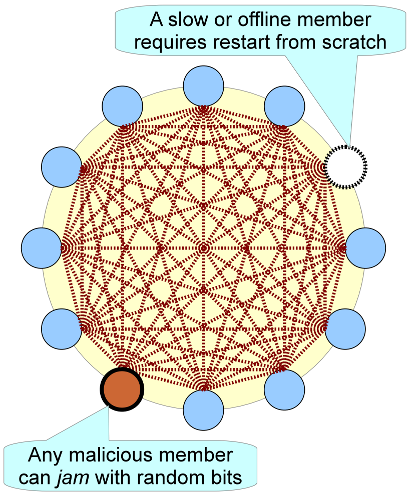

A slow or offline member requires restart from scratch
Any malicious member can *jam* with random bits

Figure 7: Why DC-nets are hard to scale in practice: (1) worst-case $N \times N$ coin-sharing matrix; (2) network churn requires rounds to start over; (3) malicious members can anonymously jam the group.

4.2 Practical dining cryptographers

Utilizing the DC-nets foundation in practical systems requires solving two main challenges: jamming and scalability. Herbivore [20] pioneered the exploration of practical solutions to both of these problems, and the Dissent project continues this work.

The jamming problem.

Both Chaum's original paper [5] and many follow-up 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 a peer who finds himself in an unreliable or jammed group to switch groups until he finds a functioning one. This design has the advantage of scaling to support arbitrary-sized networks, with the downside that each peer obtains provable anonymity only within his own group – typically tens of nodes at most – and not guaranteeing anonymity within the larger network. A second downside of switching groups to avoid jamming is that an attacker who runs many Sybil nodes and selectively jams only groups he cannot compromise completely, while offering good service in groups in which he has isolated a single "victim" node, can make it more likely that a victim "settles" in a compromised group than an uncompromised one [3].

Dissent, the only system since Herbivore to put DC-nets into practice, explores different solutions to these challenges. First, Dissent addresses the jamming problem by implementing *accountability* mechanisms, allowing the group to revoke the anonymity of any peer found to be attempting to jam communication maliciously while preserving strong anonymity protection for peers who "play by the rules." Dissent's first version introduced a conceptually simple and clean accountability mechanism that leveraged the verifiable-shuffle primitive discussed above, at the cost of requiring a high-latency shuffle between each round of (otherwise more efficient) DC-nets communication. The next version [23] 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.

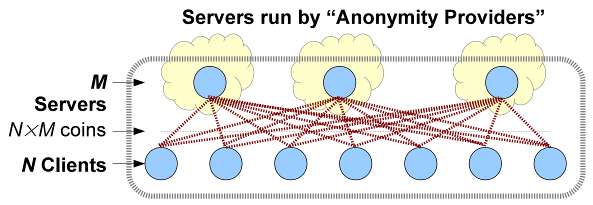


Figure 8: Improving scalability and churn resistance through an asymmetric, client/server DC-nets architecture.

An adversary who manages to infiltrate a group with many malicious nodes, however, could still "sacrifice" them one-by-one to create extended denial-of-service attacks. Addressing this risk, Dissent's most recent incarnation [6] 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 only briefly to enable the jamming victim to broadcast an *accusation*, yielding a more efficient retroactive-blame mechanism.

Scaling and network churn.

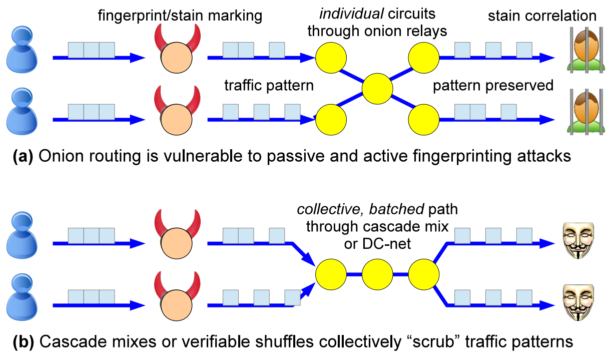
Even with multiple realistic solutions to the jamming problem now available, DC-nets cannot offer useful anonymity if they can guarantee anonymity-set sizes of at most tens of members. Herbivore addressed the $N \times N$ communication complexity problem via a star topology, in which a designated member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

To address churn and scale DC-nets further, Dissent now adopts a client/multi-server model with trust split across several servers, preferably administered independently. No single server is trusted; in fact, Dissent preserves maximum 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 merely trust that at least one trustworthy server exists.

When a Dissent group is formed, the group 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 in principle enables Dissent's design to support an arbitrary number of groups, but 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, however.

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]. As illustrated in Figure 8, clients no longer share secret "coins" directly with other clients but only with each of the group's servers. Since the number of servers in each group is typically small (*e.g.*, 3–5, comparable to the number of Tor relays supporting a circuit), the number of pseudorandom strings each client must compute is substantially reduced. This change does not reduce anonymity, however, subject to Dissent's assumption that at least one server is honest. Chaum's DC-nets security proof [5] ensures ideal anonymity provided all honest nodes are connected via the coin-sharing graph; Dissent satisfies this requirement, because the one honest server assumed to exist shares coins directly with all honest clients.

More importantly 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 at most corrupt a round and cannot de-anonymize clients except by colluding with all other servers.


fingerprint/stain marking
individual circuits through onion relays
stain correlation
traffic pattern
pattern preserved
collective, batched path through cascade mix or DC-net

(a) Onion routing is vulnerable to passive and active fingerprinting attacks

(b) Cascade mixes or verifiable shuffles collectively "scrub" traffic patterns

Figure 9: Fingerprinting or staining attacks

4.3 How Dissent handles attacks

We now summarize how Dissent handles the attacks in Section 3.

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 [21], and 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 OR system. For example, they send identical amounts of network traffic in each round, although amounts and allocations may vary from round to round.

Active attacks.

One countermeasure to traffic analysis in OR is to "pad" connections to a common bit rate. While padding may limit passive traffic analysis, it often fails against active attacks, for reasons illustrated in Figure 9. Suppose a set of OR 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 OR network, which 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 via timing. A sustained attack deployed against Tor last year exploited another subtle protocol side-channel to mark and correlate circuits, going undetected for five months before being discovered and thwarted last July.

The collective-anonymity primitives underlying Herbivore and Dissent, in contrast, structurally keep the clients comprising an anonymity set in "lock-step," under the direction of a common, collective control plane. As in the popular 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 such as those above, because its 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 via their decisions, even accidentally.

Contrary to the intuition that defense against global traffic analysis and active attacks require 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 who 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 recent project to strengthen OR security, employs a similar collective-control philosophy to normalize flow rates dynamically across an anonymity set [15]. In this way, a collective control plane can in principle not only protect against both passive and active attacks but, ironically, can also improve efficiency over padding traffic to a constant bit rate.

Intersection attacks.

While the power and generality of intersection attacks has been extensively studied in the past decade, there has been scant work on actually building mechanisms to protect users of practical systems against intersection attacks. The nearest precedents we are aware of are suggestions that traffic padding may make intersection attacks more difficult [16]. To the best of our knowledge, such proposals have never been implemented, 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 both to measure potential vulnerability to intersection attacks, using formally grounded but plausibly realistic metrics, and to offer 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 [16].

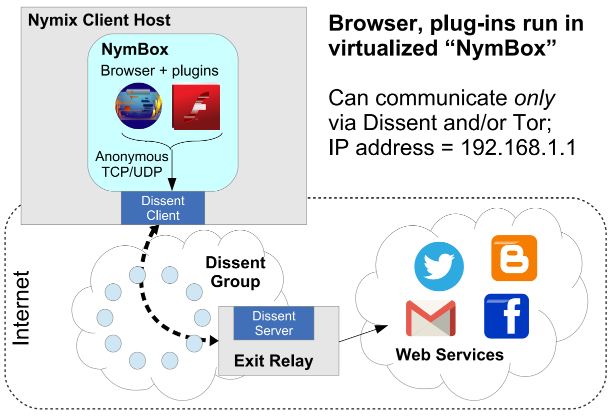


Figure 10: Nymix: using per-pseudonym virtual machines or NymBoxes to harden the client operating system against software exploits, staining, and self-identification

Users may set policies for long-lived pseudonyms limiting the rate at which measured possinymity or indinymity may be lost, or setting a threshold below which these metrics must not 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. We believe this cost reflects a fundamental tradeoff between anonymity and availability.

Software exploits and self-identification.

No anonymity protocol, by itself, can prevent de-anonymization via software exploits or user self-identification. Nevertheless, the Dissent project is exploring system-level solutions to this problem via Nymix, a prototype USB-bootable Linux distribution that employs virtual machines (VMs) to improve resistance to exploits [24].

As shown in Figure 10, Nymix runs anonymity-client software (currently 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. 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 the recent Tor Browser Bundle exploit would thus not be able to "leak" the client's IP address without also breaking out of the VM (which of course may be possible, but raises the 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*. Nymix securely discards all pseudonym state embodied in a NymBox when desired 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, open logins, *etc.*), while offering stronger protection against the user's accidentally linking different pseudonym

VMs, because they appear as entirely separate OS environments and not just 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 JPEG photos stored on the host OS, and pseudonym-VM contexts only via a *quarantine* file system "drop box." Any files the user moves across browsing contexts in this way undergoes a suite of tests for possibly compromising information, such as EXIF metadata within JPEGs. The quarantine system warns the user of any detected compromise risks and gives him the opportunity to scrub the file or decide not to transfer it at all. While all of these defenses are inherently "soft," because there is only so much we can do to prevent users from shooting themselves in the foot, Nymix combines these VM-based isolation and structuring principles in an effort to make it easier for users to make appropriate and well informed uses of today's and tomorrow's anonymity tools.

5. CHALLENGES AND FUTURE WORK

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

First, while DC-nets now scale to thousands of users, they need to scale to hundreds of thousands or more. One approach is to combine Dissent's scaling techniques with those of Herbivore [20] by dividing large anonymity networks into manageable anonymity sets (*e.g.*, hundreds or thousands of nodes), balancing performance against anonymity guarantees. A second approach is to use small, localized Dissent clusters, which already offer performance adequate for interactive Web browsing [23, 24], as a decentralized implementation for the crucial entry-relay role in a Tor circuit [1]. Much of a Tor user's security depends on his entry relay's being uncompromised [14]; replacing this single point of failure with a Dissent group could distribute the user's trust among the members of this 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 himself" behavior. Securing long-lived pseudonyms may be feasible only in applications that incentivize users to keep communication devices online consistently, 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 others users may be well suited to Dissent's collective anonymity model: *e.g.*, the interaction models of IRC, forums like Twitter or Slashdot, or 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. How best to support and incentivize cooperative behavior, however, remains an important open problem.

Finally, it is clear that large anonymity sets require widespread public demand for anonymity. Tor's 40M "mean daily users" are dwarfed in number by the users of Google, Facebook, Yahoo!, and other services that do not provide anonymity – and cannot provide it, because their business models depend crucially on exploitation of personal information. Public demand for anonymity online may rise as a result of the ongoing surveillance scandal, thereby providing an opportunity to deploy new anonymity tools.

6. REFERENCES

[1] Tor: Anonymity online. https://www.torproject.org.

[2] Tor metrics portal. http://metrics.torproject.org/.

[3] Nikita Borisov, George Danezis, Prateek Mittal, and Parisa Tabriz. 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*.

[4] Justin Brickell and Vitaly Shmatikov. Efficient anonymity-preserving data collection. In *12th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD)*, August 2006.

[5] David Chaum. The Dining Cryptographers problem: Unconditional sender and recipient untraceability. *Journal of Cryptology*, pages 65–75, January 1988.

[6] Henry Corrigan-Gibbs, David Isaac Wolinsky, and Bryan Ford. Proactively accountable anonymous messaging in Verdict. In *22nd USENIX Security Symposium*, August 2013.

[7] George Danezis, Roger Dingledine, and Nick Mathewson. Mixminion: Design of a Type III anonymous remailer protocol. In *IEEE Security and Privacy (SP)*, pages 2–15, May 2003.

[8] Roger Dingledine and Steven J. Murdoch. Performance improvements on Tor or, why Tor is slow and what we're going to do about it. In *DEFCON 17*, July 2009.

[9] Nathan S. Evans, Roger Dingledine, and Christian Grothoff. A practical congestion attack on Tor using long paths. In *18th USENIX Security Symposium*, August 2009.

[10] Joan Feigenbaum, Aaron Johnson, and Paul Syverson. Probabilistic analysis of onion routing in a black-box model. *ACM Transactions on Information and System Security*, 15(3):14:1–14:28, November 2012.

[11] Ryan Gallagher. New Snowden documents show NSA deemed Google networks a "target". *Slate*, September 9, 2013.

[12] Barton Gellman, Craig Timberg, and Steven Rich. Secret NSA documents show campaign against Tor encrypted network. *The Washington Post*, October 4, 2013.

[13] David M. Goldschlag, Michael G. Reed, and Paul F. Syverson. Hiding Routing Information. In *1st International Workshop on Information Hiding*, May 1996.

[14] Aaron Johnson, Chris Wacek, Rob Jansen, Micah Sherr, and Paul Syverson. Users get routed: Traffic correlation on Tor by realistic adversaries. In *20th ACM Conference on Computer and Communications Security (CCS)*, November 2013.

[15] Stevens Le Blond, David Choffnes, Wenxuan Zhou, Peter Druschel, Hitesh Ballani, and Paul Francis. Towards efficient traffic-analysis resistant anonymity networks. In *ACM SIGCOMM*, August 2013.

[16] Nick Mathewson and Roger Dingledine. Practical traffic analysis: extending and resisting statistical disclosure. In *4th International Workshop on Privacy Enhancing Technologies (PETS)*, May 2004.

[17] C. Andrew Neff. A verifiable secret shuffle and its application to e-voting. In *8th ACM Conference on Computer and Communications Security (CCS)*, November 2001.

[18] James Risen and Laura Poitras. NSA report outlined goals for more power. *The New York Times*, November 22, 2013.

[19] Aaron Segal, Bryan Ford, and Joan Feigenbaum. Catching bandits and only bandits: Privacy-preserving intersection warrants for lawful surveillance. In *4th USENIX Workshop on Free and Open Communications on the Internet (FOCI'14)*, August 2014.

[20] Emin Gün Sirer, Sharad Goel, Mark Robson, and Dǒgan Engin. Eluding carnivores: File sharing with strong anonymity. In *11th ACM SIGOPS European Workshop*, September 2004.

[21] Ewa Syta, Aaron Johnson, Henry Corrigan-Gibbs, Shu-Chun Weng, David Isaac Wolinsky, and Bryan Ford. Security analysis of accountable anonymity in Dissent. *ACM Transactions on Information and System Security (TISSEC)*, 17(1), August 2014.

[22] Robert Watts. JK Rowling unmasked as author of acclaimed detective novel. *The Telegraph*, July 13, 2013.

[23] David Isaac Wolinsky, Henry Corrigan-Gibbs, Aaron Johnson, and Bryan Ford. Dissent in numbers: Making strong anonymity scale. In *10th USENIX Symposium on Operating Systems Design and Implementation (OSDI)*, October 2012.

[24] David Isaac Wolinsky, Daniel Jackowitz, and Bryan Ford. Managing NymBoxes for identity and tracking protection. In *USENIX Conference on Timely Results in Operating Systems*, October 2014.

[25] David Isaac Wolinsky, Ewa Syta, and Bryan Ford. Hang with your buddies to resist intersection attacks. In *20th ACM Conference on Computer and Communications Security (CCS)*, November 2013.