Seeking Anonymity in an Internet Panopticon

UNPUBLISHED DRAFT

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; moreover, statements and actions are linked with detailed user 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" [20].

For a wide variety of reasons, however, users sometimes have a legitimate need for *anonymity*, *e.g.*, for protection against the linking of their online speech and activities to their real-world identities. Although it is often high-stakes use cases, *e.g.*, battlefield communication, espionage, or political protest against authoritarian regimes, that are offered as motivation for the study of anonymous-communication technology, 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 activitiy. 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" [23].

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 some of the most concerted attacks currently known to be in use [13]. Unfortunately, OR is known to be vulnerable at least in principle 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 [15]. 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 was de-anonymized in the Petraeus scandal [22]. Finally, through software exploits or user error, an attacker can often circumvent anonymity tools entirely [19].

Current approaches to anonymity also appear unable to offer accurate, principled *measurement* of the level or quality of anonymity a user might obtain. We believe that these limitations stem from an attempt to achieve an impossible set of goals and to defend an ultimately indefensible position. In particular, current tools attempt to offer a completely general-purpose, unconstrained, and *individualistic* form of anonymous Internet access. However, there are just too many ways for unconstrained, individualistic uses of the Internet to be fingerprinted and tied to individual users. It is our thesis that the only way to achieve *measurable* and *provable* levels of anonymity and to stake out a position that can be defensible in the long term is to move toward 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, the first systematic effort to build a practical anonymity system based purely on foundations that offer measurable and formally provable anonymity properties. Dissent builds on two key pre-existing primitives – *verifiable shuffles* [18] and *dining cryptographers* [6] – but for the first time shows how to scale such techniques to offer measurable anonymity guarantees to thousands of participants. A Dissent-based anonymity network as a whole, once deployed, might readily scale to millions of participants; the current limiting factor is not the total number of participants but the level of anonymity we can guarantee each of them.

Further, Dissent represents the first anonymity system designed from the ground up to incorporate *some* systematic countermeasure for each of the major classes of known vulnerabilities in existing approaches, including global traffic analysis, active attacks, and intersection attacks. Finally, because no anonymity *protocol* alone can address risks such as software exploits or accidental self-identification, we introduce WiNon, an experimental operating system architecture to harden the *uses* of anonymity tools such as Tor and Dissent against such attacks.

While Dissent is at this writing still a research prototype not yet ready for widespread deployment, and it may never offer a direct "drop-in replacement" for OR tools such as Tor because of the possibly fundamental tradeoffs we discuss here, we hope that these results will open up a broader variety of approaches to online anonymity and lead to methods of formalizing and measuring the levels of security and anonymity they achieve in practice.

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.

This paper is concerned with *network-layer anonymity*. That is, the receiver of a message delivered by the anonymous-communication protocols that we consider does not know which network node is the source of that message. In Internet terms, this is *IP-layer anonymity*, meaning that the receiver would not know the IP address (or perhaps even the domain) of the source. Note that this does not guarantee *personal anonymity*; the content of the message may identify the human user who sent it even if the routing information hides that user's network location. The questions of whether and how to anonymize content are orthogonal to the network-layer anonymity question, and we do not address them in this paper.

In Section 2, we present 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 communication on the Internet is Tor [1]. Tor's technical foundation is *onion routing* [14], which in turn is derived from *mixnets* [5].

Onion routing (OR) uses successive layers of encryption to route messages through an overlay network in such a way that each node in a route knows the previous node and the next node but nothing else about the route. More precisely, let (V, E) be a connected, undirected network and $R \subseteq V$ be a set of nodes that serve as *relays*. The set R is known to all nodes in V, as is the public key PK_r, usable in some globally agreed-upon public-key cryptosystem, for each $r \in R$. 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, it first chooses a sequence $(r_1, r_2, \ldots, r_k)$ of relays. It then constructs an "onion" whose k layers contain both the message and the routing information needed to deliver it without revealing 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 k^{th} or innermost layer of the onion is

$$O_k = (r_k, E((d, M), PK_{r_k})),$$

i.e., the k^{th} relay node and the encryption of the core under the k^{th} relay's public key. More generally, the i^{th} layer O_i, $1 \leq i \leq k-1$, 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:

$$O_i = (r_i, E(O_{i+1}, PK_{r_i})).$$

Once it has finished constructing the outermost layer $O_1 = (r_1, E(O_2, PK_{r_1}))$, s sends $E(O_2, PK_{r_1})$ to r_1, using the routing protocol of the underlay network (V, E). When relay r_i, $1 \leq i \leq k$, receives $E(O_{i+1}, PK_{r_i})$, it decrypts it using the private key SK_{r_i} corresponding to PK_{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 underlaying routing protocol); when $i = k$, the message is just the core (d, M), because, strictly speaking, there is no O_{k+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.

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].

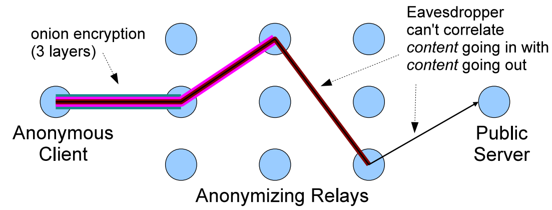

Figure 1: Onion routing.

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. It 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. For this reason, the Tor project also supports *bridges*, *i.e.*, 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, many potential users reject Tor because of its performance impact [9].

3. ATTACKS ON ONION ROUTING

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

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

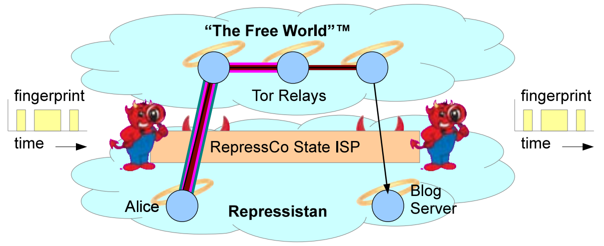

Figure 2: Traffic confirmation or fingerprinting to de-anonymize onion-routing circuits

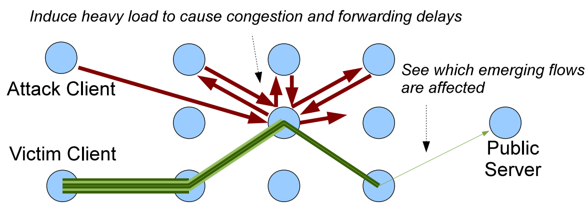

Figure 3: Example of a congestion-based active attack

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, 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. Unfortunately, it is now clear that this adversaral model is not only realistic but *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* [10]. 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.

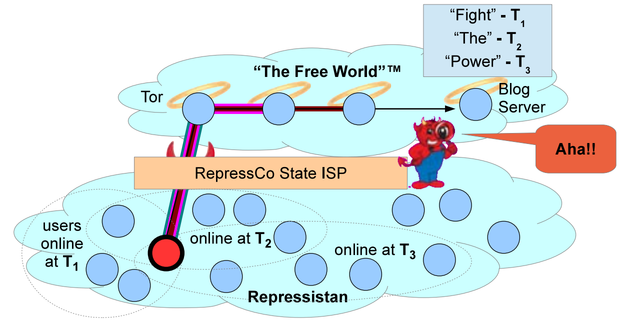

Figure 4: Example of an intersection attack

Intersection attacks.

In most practical uses of anonymous communication (voting being a rare exception), 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 the attacker can obtain only a very small and perhaps 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 in which periods of time. An anonymous dissident posts blog entries to a pseudonymous blog at different points in time. Assume 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

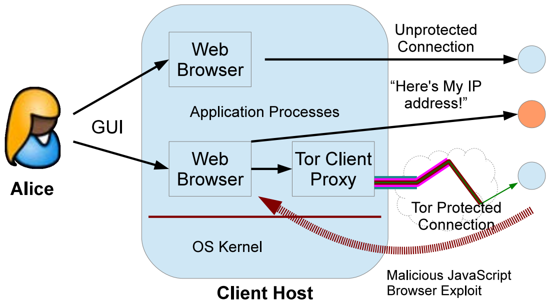

Figure 5: Example of a software-exploit attack

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 posts (because of normal *churn* – 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 be reduced 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; for example, Paula Broadwell was de-anonymized via the equivalent of an intersection attack [22], as were the High Country Bandits [3]. Intersection attacks also appear to form the foundation of the NSA's recently revealed CO-TRAVELER program, which links known surveillance targets with unknown potential targets as their respective cellphones move together from one cell tower to another [12].

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 recently observed 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, a version of Firefox specifically configured to work with Tor. This JavaScript-based 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 over a normal network connection (not via Tor) to an attacker-controlled server collecting this information.

4. DISSENT: TOWARDS QUANTIFIABLE ANONYMITY

We now offer a high-level overview of Dissent, an ongoing project aimed at rethinking the foundations of anonymous communication and providing quantifiable, provable anonymity in some, if not all, realistic scenarios.

4.1 Alternative approaches to anonymity

While unconstrained OR does not appear amenable to provable security properties under realistic conditions, there are anonymity primitives that have formally provable properties. Dissent achieves traffic-analysis resistance by building on two such constructions: *verifiable shuffles* and *dining cryptographers*.

Verifiable shuffles.

In this construction, 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 the n 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], 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 complex and sophisticated *fully verifiable* shuffles, such as Neff's [18], enable each shuffler to *prove* in zero knowledge the correctness of its shuffle, *i.e.*, to prove that its output is a correct permutation of its input without revealing any information about *which* permutation it chose.

Verifiable shuffles offer a cryptographically provable guarantee that the *process of shuffling* reveals no information about which of the n clients submitted a particular message appearing in the shuffled output. This guarantee, unfortunately, does not automatically extend to the larger system in which such a shuffle might be used. For example, a client might still de-anonymize itself with some information in the *content* of its message or even by the client's presence or absence in a particular messaging round. Further, this basic approach to anonymity has the practical disadvantage that the level of security achievable against potentially compromised shufflers tends to depend 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 popular *cascade* arrangement, in which all clients send their messages through *the same* sequence of shufflers at the same time, effectively exacerbates this performance problem by deliberately creating the "worst possible congestion" at each shuffler in succession, rather than randomly distributing load across a large number of shufflers as an ad hoc, individualistic OR network would. For these reasons, verifiable shuffles appear to be practical only in applications in which rather high latencies are tolerable, and shufflers

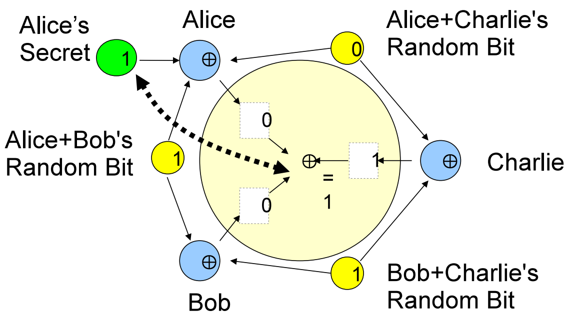

Figure 6: The Dining Cryptographers approach to anonymous communication. Alice reveals a 1-bit secret to the group, but neither Bob nor Charlie learn which of the other two members sent this message.

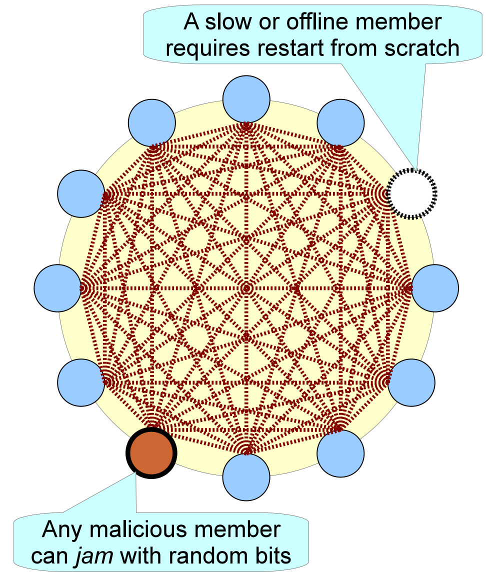

Figure 7: Why DC-nets is hard to scale in practice: (1) worst-case $N \times N$ coin-sharing matrix; (2) network churn requires rounds to start over; (3) malicious members can anonymously jam the group.

are well provisioned.

Dining cryptographers.

The only well studied foundation for online anonymity not based on sequential relaying is the *Dining Cryptographers* or *DC-nets* approach, invented by Chaum in the late 1980s [6]. Instead of relaying, DC-nets builds on information coding.

Consider Chaum's standard story, 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 the 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 generalizes easily 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 produce 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 random bits all the time.

4.2 Practical dining cryptographers

Although DC-nets has for some time been considered attractive as an alternate foundation for anonymity, it has generally been seen as impractical for two reasons: *jamming* and *scalability*.

The jamming problem.

Both Chaum's original paper [6] and many follow-up works studied theoretical solutions to the jamming problem, but Herbivore [21] was the first serious attempt to use DC-nets in a practical system. Herbivore's design securely divided a large peer-to-peer network into many smaller DC-nets groups, enabling a peer who found himself in an unreliable or jammed group to switch groups until he finds a functioning one. While scalable in principle to arbitrary-sized networks, the primary downside of this approach is that each peer obtains provable *anonymity* only with respect to his own group and not with respect to the network as a whle; the protocol was demonstrated to be practical only with unsatisfyingly small anonymity sets consisting of at most tens of nodes. 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.

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 leveraged the verifiable-shuffle primitive discussed above to introduce a conceptually simple and clean accountability mechanism, at the performance cost of requiring a high-latency verifiable shuffle between each round of (otherwise more efficient) DC-nets communication. The next version [24] 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

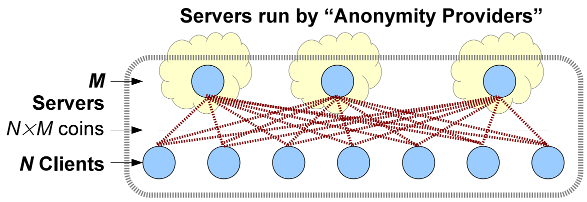

Figure 8: Improving scalability and churn resistance through an asymmetric, client/server DC-nets architecture.

shuffle only once per detected jamming attempt. However, an adversary who manages to infiltrate a group with many malicious nodes could still in principle "sacrifice" them one-by-one to create extended denial-of-service attacks. Addressing this risk, a more recent variation [7] 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 proactively on submission, using zero-knowledge proofs. To avoid the costs of using elliptic-curve cryptography all the time, a *hybrid* mode is also available 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* in 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 it can guarantee anonymity-set sizes of at most tens of members because of the problems of network churn and $N \times N$ coin-sharing. To address these challenges, Dissent adopts a client/multi-server model with trust split across several servers, preferably chosen from independent administrative domains. 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 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 (but our Dissent prototype does not yet implement such a directory service). 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 also be applicable (but also is not yet implemented in the Dissent prototype).

While individual groups do not scale indefinitely, Dissent exploits its client/multi-server architecture to make groups scale at least two orders of magnitude better than in prior designs based on DC-nets. 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 supporting each group is typically small (*e.g.*, 3–5, comparable to the number of Tor relays supporting a circuit), the number of pseudorandom-number sequences

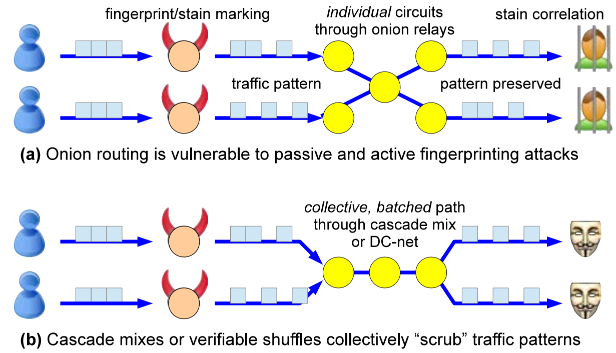

Figure 9: Fingerprinting or staining attacks

each client must compute to implement DC-nets is substantially reduced.

Handling network churn.

Even more importantly in practice, Dissent's client/multi-server coin-sharing design makes the composition of client ciphertexts independent of the precise set of *other* clients online in a given DC-nets 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. If some clients miss the deadline, however, the other clients' ciphertexts do not become unusable; instead, the servers merely need to 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.

4.3 How Dissent handles attacks

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

Global traffic analysis.

As discussed above, Dissent builds entirely on anonymity primitives, namely verifiable shuffles and DC-nets, that have formal proofs of security, even in a model in which the attacker is assumed to monitor *all* network traffic sent among *all* participating nodes (but to be unable to break the encryption). 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.

A commonly proposed approach to traffic analysis in OR is to "pad" connections to a common bit rate, as in the recent Aqua system [16]. While padding may indeed limit *passive* traffic analysis, however, it often fails against *active* attacks, for reasons illustrated in Figure 9. Suppose that 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*, will preserve 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.

By contrast, the collective-anonymity primitives Dissent builds on ensure that clients forming a Dissent group operate in "lock-step," under the direction of a common, collective control plane – much like the popular children's game "Simon Says." Participating nodes transmit when and how much the collective control plane says to transmit; the control-plane elements (the Dissent servers) do not know which user owns which pseudonym or DC-nets transmission slot and thus cannot leak that information via their decisions. Control-plane decisions in one communication round *can* depend on the results of prior rounds; for example, the Dissent 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. In this way, a collective control plane can in principle not only offer better protection against both passive and active attacks but, ironically, can also offer improved *efficiency* over merely padding all 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 anonymity systems against intersection attacks. The nearest precedents we are aware of are suggestions that traffic padding may make intersection attacks more difficult [17]. To the best of our knowledge, such proposals have never actually been implemented, in part because there is no obvious way to measure how much, *if any*, intersection-attack resistance a given padding scheme will provide in a real environment.

Dissent is the first anonymity system that incorporates in its design both *measurement* of vulnerability to intersection attacks, using formally grounded but plausibly useful metrics, and active *control* over anonymity loss to intersection attacks. In particular, Dissent implements two different anonymity metrics [25]: *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* [8].

Software exploits and self-identification.

No anonymity *protocol*, by itself, can prevent de-anonymization via software exploits or user self-identification. Nevertheless, as part of the Dissent project, we are exploring *system-level* solutions to this problem as part of *WiNon*, a prototype USB-bootable Linux distribution that employs virtual machines (VMs) to offer improved resistance to such vulnerabilities.

As shown in Figure 10, WiNon 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; so even native code injected by the recent Tor Browser Bundle exploit would not be able to "leak" the client's IP address without also breaking out of the VM (which of course may be possible, but one hopes that it is

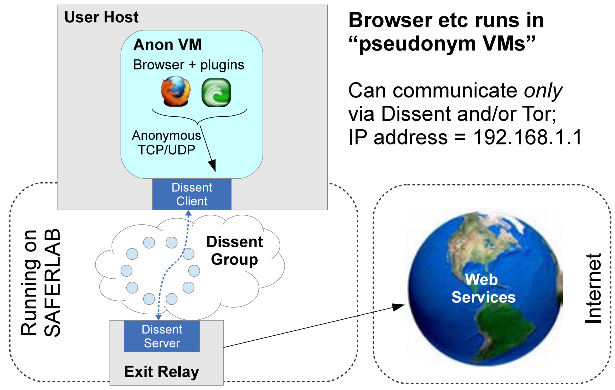

Figure 10: WiNon: using virtual machines to harden anonymity systems against software exploits, staining, and self-identification

harder than breaking out of the standard Javascript sandbox).

WiNon also explicitly binds guest-VM state instances to "pseudonyms" managed by the anonymity layer and enables users to launch multiple simultaneous pseudonyms (in separate VMs) and securely discard their state when desired to minimize 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.

Finally, to reduce the risk of self-identification, WiNon 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; if a potential compromise is detected, the quarantine system warns the user 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, WiNon 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. OPEN PROBLEMS AND FUTURE DIRECTIONS

Dissent has taken a few steps toward achieving measurable, formally grounded anonymity "at scale," but significant practical challenges remain.

First, while DC-nets now scales to thousands of users, it needs to scale to hundreds of thousands or more. One promising approach is to combine Dissent's scaling techniques with those of Herbivore [21] by dividing an arbitrarily large-scale anonymity network into smaller anonymity sets (*e.g.*, hundreds or thousands of nodes) tuned to balance performance against anonymity guarantees. A second is to use small, localized Dissent clusters, which already offer performance adequate for interactive Web browsing [24], as a decentralized substitute 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 [15]; replacing this single point of failure with a Dissent group would distribute the user's trust among the members of this group and would further protect traffic between the user and the Tor relays from traffic analysis by "last mile" ISP adversaries.

Second, while Dissent has introduced techniques to measure vulnerability to intersection attack (and limited mechanisms to control this vulnerability), robust resistance to intersection attack ultimately depends on user and application behaviors and cannot be solved in a purely individualistic, "every user for himself" setting. In particular, we expect that robust intersection-attack resistance may be feasible only in applications that *incentivize* users to keep their communication devices online consistently for long periods, even if at low rates of activity, to reduce the "decay" of anonymity sets caused by user churn. Further, robust intersection-attack resistance may ultimately be practical in applications designed to encourage users to act *collectively*, rather than *individually*, and optimized for such user behavior. Tor's goal is to support thousands of separate, individualistic Web interactions by different users; it offers attractive generality but yields a usage pattern that is inherently trackable and expensive to obscure, especially against a strong adversary capable of intersection attacks.

By contrast, applications in which users cooperate and deliberate to produce a collective information "feed" consumed by all users (such as the interaction models supported by IRC, popular websites such as Twitter or Slashdot, or applications supporting voting, deliberating, or "Town Hall" meetings) may be inherently more suitable contexts in which to give users access to strong anonymity. Given the close relationship between such collective deliberation and the foundations of democracy and freedom of speech, such applications may also represent some of the most *important* use cases for strong anonymity. How to incentivize this type of cooperative behavior online, 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] Nate Anderson. How "cell tower dumps" caught the High Country Bandits—and why it matters. *arstechnica*, August 29, 2013.

[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. Untraceable electronic mail, return addresses, and digital pseudonyms. *Communications of the ACM*, 24:84–88, February 1981.

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

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

[8] George Danezis and Andrei Serjantov. Statistical disclosure or intersection attacks on anonymity systems. In *6th International Workshop on Information Hiding*, May 2004.

[9] 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.

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

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

[12] Barton Gellman and Ashkan Soltani. NSA tracking cellphone locations worldwide, Snowden documents show. *The Washington Post*, December 4, 2013.

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

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

[15] 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.

[16] 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.

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

[18] 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.

[19] Kevin Poulsen. Feds are suspects in new malware that attacks Tor anonymity. *Wired*, August 5, 2013.

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

[21] 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.

[22] Chris Soghoian. Surveillance and security lessons from the Petraeus scandal. *ACLU Blog*, November 13, 2012.

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

[24] 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.

[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.