Hang With Your Buddies to Resist Intersection Attacks

David Isaac Wolinsky, Ewa Syta, and Bryan Ford
Yale University
{david.wolinsky,ewa.syta,bryan.ford}@yale.edu

ABSTRACT

Some anonymity schemes might in principle protect users from pervasive network surveillance—but only if all messages are independent and unlinkable. Users in practice often need *pseudonymity*—sending messages intentionally linkable to each other but not to the sender—but pseudonymity in dynamic networks exposes users to *intersection attacks*. We present Buddies, the first systematic design for intersection attack resistance in practical anonymity systems. Buddies groups users dynamically into *buddy sets*, controlling message transmission to make buddies within a set behaviorally indistinguishable under traffic analysis. To manage the inevitable tradeoffs between anonymity guarantees and communication responsiveness, Buddies enables users to select independent attack mitigation policies for each pseudonym. Using trace-based simulations and a working prototype, we find that Buddies can guarantee non-trivial anonymity set sizes in realistic chat/microblogging scenarios, for both short-lived and long-lived pseudonyms.

Categories and Subject Descriptors

C.2.0 [**Computer-Communication Networks**]: General—*Security and protection*

Keywords

anonymity; pseudonymity; intersection; disclosure

1. INTRODUCTION

Some anonymous communication techniques promise security even against powerful adversaries capable of pervasive network traffic analysis—*provided* all messages are fully independent of each other and/or the set of participants never changes [5,9,41,52]. Practical systems, however, must tolerate *churn* in the set of online users, and must support ongoing exchanges that make messages *linkable* over time, as with Mixminion nyms [15] or Tor sessions [18]. By sending linkable messages in the presence of churn, however, users can quickly lose anonymity to statistical disclosure or intersection attacks [16,31,42,53]. Though this extensively studied attack vector could apply in almost any realistic anonymous communication scenario, no practical anonymity system we know of offers active protection against such attacks.

As an example intended merely to illustrate one possible scenario in this broad class of attacks, suppose Alice writes a blog under a pseudonym to expose corruption in her local city government. Alice always connects to the blog server via Tor [18], and never reveals personally identifying information on her blog or to the server. Carol, a corrupt city official mentioned in Alice's blog, deduces from the blog's content that its owner is local, and calls her friend Mallory, a network administrator in the monopolistic local ISP. Mallory cannot directly compromise Tor, but she reads from Alice's blog the date and time each blog entry was posted, and she learns from the ISP's access logs which customers were online and actively communicating at each of those times. While thousands of customers may be online at each posting time, every customer *except* Alice has a chance of being offline during *some* posting time, and this chance exponentially approaches certainty as Alice continues posting. Mallory simply keeps monitoring until the *intersection* of these online user sets narrows to one user, Alice. We don't know if this precise attack has occurred, but analogous intersections of hotel guest lists, IP Addresses, and e-mail accounts revealed the parties in the Petraeus/Broadwell scandal [47].

As a step toward addressing such risks we present Buddies, the first anonymous communication architecture designed to protect users systematically from long-term intersection attacks. Buddies works by continuously maintaining an anonymized database of participating users and their online status, and uses this information to *simulate* intersection attacks that a network-monitoring adversary might perform. These simulations yield two relevant anonymity metrics that Buddies reports continuously, as an indication of potential vulnerability to intersection attack: a *possibilistic* metric roughly measuring "plausible deniability," and a more conservative *indistinguishability* metric indicating vulnerability to more powerful statistical disclosure attacks [16].

Beyond just measuring vulnerability, as in prior work on metrics [17,43] and alternate forms of anonymity [30], Buddies offers *active control* over anonymity loss under intersection attack. Users specify a policy for each pseudonym that balances attack protection against communication responsiveness and availability. To enforce these policies, a *policy module* monitors and filters the set of users participating in each communication round, sometimes forcing the system to behave as if certain online users were actually offline. Through this active control mechanism, policies can enforce lower bounds on anonymity metrics, preventing Alice from revealing herself to Mallory by posting at the wrong time for example. Policies can also reduce the *rate* of anonymity loss to intersection attacks, for example by tolerating anonymity set members who are normally reliable and continuously online but who lose connectivity for brief periods. Finally, policies can adjust posting rates or periods, enabling Buddies to aggregate all users coming online within a posting period into larger anonymity sets. If Alice sets her blog's posting period to once per day, for example, then Buddies

CCS'13, November 4–8, 2013, Berlin, Germany.
ACM 978-1-4503-2477-9/13/11. DOI 10.1145/2508859.2516740.

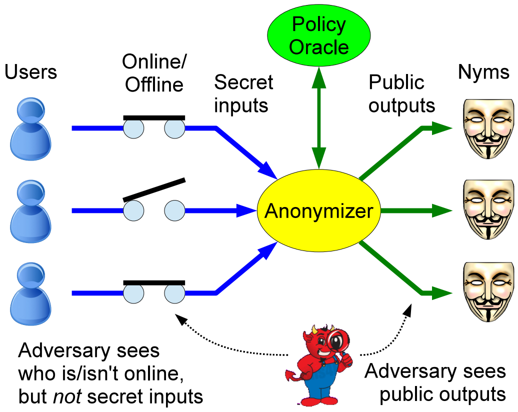


Figure 1: Conceptual model of Buddies architecture

can maintain Alice's anonymity among all users who "check in" at least once a day—*any time* during each day—even if many users check in only briefly at widely varying times.

Buddies' architecture may be treated as an extension to various existing anonymous communication schemes, but is most well-suited to schemes already offering measurable protection guarantees against traffic analyis, such as MIX cascades [5, 41], DC-nets [9, 46, 52], or verifiable shuffles [7, 25, 39]. We have built a working prototype of Buddies atop Dissent [11, 13, 52], a recent anonymous communication system that combines verifiable shuffle and DC-net techniques. The prototype's design addresses several practical challenges: to decentralize trust among independent servers, to create and manage pseudonyms while maintaining their independence, and to support user-selectable policies for each pseudonym.

To evaluate Buddies' practicality in realistic online communities, we analyze IRC trace data under a Buddies simulator, exploring questions such as how effective Buddies' anonymity metrics are, how feasible it may be to maintain nontrivial anonymity sets resistant to intersection attacks for extended periods, and how effectively Buddies can limit loss of anonymity while preserving usable levels of communication responsiveness and availability.

This paper's primary contributions are: (a) the first anonymity architecture that systematically addresses intersection attacks; (b) a modular, policy-based framework for both vulnerability monitoring and active mitigation of anonymity loss via intersection attacks; and (c) an evaluation of Buddies' practicality via a working prototype and trace-based simulations reflecting realistic online communities.

Section 2 of this paper outlines Buddies' high-level model of operation and the anonymity metrics we use. Section 3 then explores several useful attack mitigation policies in this model. Section 4 details challenges and approaches to incorporating Buddies into practical anonymity systems, and Section 5 experimentally evaluates both our working Buddies prototype and trace-based simulations. Section 6 summarizes related work, and Section 7 concludes.

2. BUDDIES ARCHITECTURE

Figure 1 shows a high-level conceptual model of the Buddies architecture. Buddies assumes there is some set of *users*, each of whom has a secret (i.e., securely encrypted) network communication path to a component we call the *Anonymizer*. For now we conceptually treat the Anonymizer as a central, trusted "black box," although later we will map this conceptually centralized component to realistic anonymization systems that decentralized trust, to avoid trusting any single physical component or administrative domain.

Buddies' model is inspired by anonymous blogging or IRC scenarios, where users post messages to a public forum, and users primarily desire sender anonymity [40]. While we expect Buddies to generalize to two-way models and metrics [44], we defer such extensions to future work. Each Buddies user "owns" some number of *Nyms*, each representing a pseudonymous identity under which the owner may post: e.g., an anonymous chat handle or blog. Users may secretly submit messages to be posted to Nyms they own, which the Anonymizer scrubs of identifying information and publicly "posts" to that Nym. To make various operational decisions, the Anonymizer consults a *Policy Oracle*. By design the Policy Oracle has no access to sensitive information, such as who owns each Nym: the Policy Oracle makes decisions based purely on public information available to anyone.

We assume the network-monitoring adversary identifies users by some network identifier or locator, such as IP address. By monitoring these locators the adversary can tell which users are online or offline at any given moment, and how much data they transmit or receive, but *cannot* see the actual content of data communicated between honest users and the Anonymizer. These assumptions model an ISP-grade adversary that can implement "wholesale" network-level monitoring of users connected via that ISP.

2.1 Overview of Operation

In Buddies' conceptual architecture, communication proceeds synchronously through a series of *rounds*. The Anonymizer drives the operation of each round i, as follows:

1. Registration: At the start of round i the Anonymizer updates the membership roster, $\mathbb{M}_i$, to include members who may have recently joined.

2. Nym creation: The Anonymizer next creates and announces one "fresh" Nym N_i each round. For each new Nym, the Anonymizer chooses one User *uniformly at random* as the Nym's owner, keeping this ownership secret. A Nym's lifetime is in principle unlimited: over time users acquire fresh Nyms at random but "statistically fair" times. (We later address creation of larger "batches" of Nyms efficiently, so new users need not wait a long time before they can post.)

3. Scheduling: The Anonymizer consults the Policy Oracle to choose one Nym, T_i, for transmission in this round, from all Nyms in existence. The Policy Oracle also specifies the number of bits B_i that the owner of Nym T_i may post. (Scheduling multiple Nyms per round is a straightforward extension.) As the Policy Oracle can access only public information, scheduling cannot depend directly on which Users currently "have messages to post." Scheduling can depend on other factors, however, such as Nyms' lifetimes, recent usage, or the interest of other users as indicated in messages previously posted anonymously via other Nyms.

4. Message submission: The Anonymizer announces the scheduled Nym T_i and transmission length B_i to the Users currently online. Each online user submits exactly B_i secret bits to the Anonymizer. These secret bits may contain either "real" data, or a *null* message of B_i zero bits, if the user has nothing useful to transmit at the moment. The bits sent from any user j other than the owner of Nym T_i represent "cover traffic" necessary to hide the Nym-owner's message submission from traffic analysis. The Anonymizer forms an *online user set*, $\mathbb{O}_i \subseteq \mathbb{M}_i$, consisting of the users who submitted a (real, null, or cover) message in round i.

5. User filtering: The Anonymizer now consults with the Policy Oracle, giving the Policy Oracle the set $\mathbb{O}_i$ of online users—but *not* any message content or information about which, if any, of these users owns the Nym scheduled this round. The Policy Oracle returns a new, *filtered* user set $\mathbb{P}_i \subset \mathbb{O}_i$, further constraining the set of online users whose submissions the Anonymizer will actually *accept* this round.

6. Message posting: If the owner of the scheduled Nym T_i is a member of $\mathbb{P}_i$—i.e., is online *and* was not filtered above—then the Anonymizer decrypts that user's secret message and posts it in association with Nym T_i. If the owner of T_i is not in $\mathbb{P}_i$—either because the owner was not online or was filtered above—then the Anonymizer posts B_i zero bits to Nym T_i: an output indistinguishable from a null message.

2.2 Active Mitigation of Intersection Attacks

The user filtering step above (step 5) serves as Buddies' primary "control point" through which to resist intersection attacks. The Policy Oracle uses publicly available information to *simulate* a virtual Adversary, by continuously performing an "intersection attack" against each Nym. At step 5 of each round i, the Policy Oracle first forms an attack model for the scheduled Nym T_i, based on prior history and the set $\mathbb{O}_i$ of users online in this round. The Policy Oracle computes one or more relevant anonymity metrics as detailed further below, and determines if action is required to limit or avoid anonymity loss in this round. If no action is required, the Policy Oracle returns the unfiltered user set to the Anonymizer, i.e., $\mathbb{P}_i = \mathbb{O}_i$. If action is required, however, then the Policy Oracle can filter the user set producing a $\mathbb{P}_i \subset \mathbb{O}_i$, thus preventing any user not in $\mathbb{P}_i$ from posting, as if more users were offline than are *actually* offline.

To illustrate how this filtering enables the Policy Oracle to mitigate intersection attacks, consider the following strawman policy. In step 5 of each round, the Policy Oracle simply checks whether *all* known users are presently online, i.e., whether $\mathbb{O}_i = \mathbb{M}_i$, returning $\mathbb{P}_i = \mathbb{M}_i$ if so, and otherwise returning $\mathbb{P}_i = \emptyset$. In effect, the Policy Oracle forbids the system from making progress—allowing *anyone* to post to *any* Nym—except when *all* users are online. Since messages are posted only when all users are online, the intersection of all *nonempty* rounds' user sets is $\mathbb{M}_i$, and the system preserves "perfect" anonymity assuming the Anonymizer performs as required. The tradeoff, of course, is effective system availability, which would be unusable in most practical situations.

The key technical challenge, and a primary contribution of this paper, is developing more nuanced methods of controlling the user filtering step in each round. By controlling these filtering choices, we seek to maintain both measurable anonymity levels under intersection attack *and* "usable" levels of availability, under arguably realistic conditions.

There are situations in which *no* active control mechanism will help. If all users but one go offline permanently, for example, the Policy Oracle has only two choices: eventually allow the one remaining user to post, giving up all anonymity under intersection attack; or filter that user forever, giving up availability completely. Thus, we must set realistic expectations. Section 5 experimentally investigates the feasibility of resisting intersection attacks in IRC communities, and tests possible control policies against feasibility metrics.

Buddies' architecture separates the Policy Oracle from the Anonymizer, giving the Policy Oracle access *only* to "public information" we assume is known to everyone including the adversary, eliminating the risk that policies may "accidentally" compromise anonymity by leaking Nym ownership. By architecturally disallowing the Policy Oracle from having access to private information, we avoid the need to analyze each policy carefully for such "side-channel" anonymity leaks, and instead can focus purely on the main question of how effectively a policy mitigates intersection attacks while maintaining usable levels of availability.

Another issue is whether the information the Policy Oracle needs to simulate the Adversary's intersection attacks—such as the set of users online in each round—*should* be considered "public information." Although an ideal global adversary would know this information anyway, more realistic adversaries may be unable to monitor *all* users. If Buddies' design "hands out" information that would otherwise be at least partially private—such as the IP addresses of all online users—we risk accidentally "strengthening" a weak adversary into an effectively omniscient adversary. In the important case of users' network identities such as IP addresses, our design mitigates this leak by replacing IP addresses with anonymized *tags* when reporting online user sets to the Policy Oracle, as discussed in Section 4.3. However, whether Buddies' simulation-based architecture may strengthen weak adversaries in other unexpected ways, by making "too much information" public, merits further study.

2.3 Analyzing Intersection Attacks

While we do not attempt full formal analysis, the simplicity of the conceptual model facilitates straightforward informal analysis. Our focus here is on a particular class of attacks, namely what an adversary can learn from users' online status over time (the "switches" in Figure 1). The many other known attacks against practical anonymity systems are important but out of this paper's scope. We also claim no particular novelty in our analysis techniques or metrics; our goal is merely to apply known attacks [16, 31, 42] and anonymity metrics [17, 43] to the Buddies model.

We assume the Anonymizer is trusted to "do its job," keeping secret the linkage between Users and the Nyms they own. We also assume honest Users—those users we care about protecting—do not "give away" their identities or the relationships between their Nyms via the messages they post.

Under these conditions, the Adversary obtains three potentially important pieces of information in each round i: (a) the set of online users $\mathbb{O}_i$, (b) the set $\mathbb{P}_i$ of online users who passed the Policy Oracle's filter in step 5, and (c) the B_i message bits that were posted to the scheduled Nym T_i. An observation that will be key to Buddies's design is that only (b) and (c) will *actually* prove relevant to intersection attack analysis: the adversary ultimately obtains no useful information from knowing which users *were online* during a given round, beyond what the adversary learns from knowing which users *were online and unfiltered.*

Since we assume honest users do not "give away" their identities in their message content, we ultimately care only whether the message posted to Nym T_i was null or non-null. If a non-null message appeared for Nym T_i in round i, then the adversary infers that the owner of T_i must be a member of the filtered user set $\mathbb{P}_i$ in that round. (If the owner of T_i

was online but filtered in round i, then a null message would have appeared, as if the owner was offline.)

If a *null* message appears for T_i, however, the Anonymizer's design ensures that the adversary cannot distinguish among the following three possibilities: (1) the owner of Nym T_i was offline, (2) the owner was online but filtered, or (3) the owner was online and unfiltered, but had nothing useful to send and thus intentionally posted a null message.

2.3.1 Possibilistic Anonymity Analysis

To construct a simple *possibilistic* anonymity set P_N for a given Nym N, the adversary intersects the filtered user sets $\mathbb{P}_i$ across all rounds i for which Nym N was scheduled *and* a non-null message appeared: i.e., $P_N = \bigcap_i \{O_i \mid T_i = N \wedge m_i \neq 0\}$. Thus, P_N represents the set of users who might conceivably own Nym N, consistent with the observed set of non-null messages that have appeared for Nym N up to any point in time. Since the adversary cannot distinguish whether the appearance of a null message means that N's owner was offline, filtered, or merely had nothing to send, null-message rounds do not eliminate the possibility that users offline during that round may own N, so such rounds leave the possibilistic anonymity set P_N unaffected.

We define the size of a Nym's possibilistic anonymity set, $|P_N|$, as Nym N's *possibilistic anonymity*, which for convenience we abbreviate as *possinymity.* Although possinymity is only one of the many useful anonymity metrics that have been proposed [32, 33], and perhaps a simplistic one, we feel it captures a useful measure of "plausible deniability." If for example a user is dragged into court, and the judge is shown network traces of a Buddies system in which the accused is one of $|P_N|$ users who *may in principle* have posted an offending message, then a large possibilistic anonymity may help sow uncertainty of the user's guilt. We fully acknowledge the weaknesses of plausible deniability in general, however, especially in environments where "innocent until proven guilty" is not the operative principle.

2.3.2 Probabilistic Anonymity Analysis

While a simplistic adversary might stop at the above analysis, a smarter adversary can probabilistically learn not only from rounds in which non-null messages appeared, but also from rounds in which only a null message appeared.

Suppose for example the adversary correctly surmises from past observation that, in each round i, the owner of Nym N will have no useful message to post with some independent and uniformly random probability p. In this case the user will "pass" by submitting a null message. With probability $1 - p$ the user will have a non-null message and will try to post it—but this post attempt fails, yielding a null message anyway, if the owner is offline or filtered in that round.

For simplicity assume there are two users A and B, the adversary observes exactly one round i, this round results in a null message, and $\mathbb{P}_i = \{A\}$: user A participated but user B did not. The null output from round i means one of two events occurred: (a) A owns N, but chose with probability p not to post in round i; or (b) B owns N, and no message appeared independently of p because $B \notin \mathbb{P}_i$. Because Nyms are assigned to users uniformly at random on creation, the "base" probability that either user owns N is $1/2$. The probability of the above events (a) and (b) occurring *conditioned* on the observed history, however, is different. To be precise, $P[(\text{a}) \mid (\text{a}) \cup (\text{b})] = P[(\text{a}) \cap ((\text{a}) \cup (\text{b}))]/P[(\text{a}) \cup (\text{b})] = (p/2)/P[(\text{a}) \cup (\text{b})]$, and $P[(\text{b}) \mid (\text{a}) \cup (\text{b})] = (1/2)/P[(\text{a}) \cup (\text{b})]$. Since (a) and (b) are disjoint events, $P[(\text{a}) \cup (\text{b})] = P[(\text{a})] + P[(\text{b})]$, so $P[(\text{a})] = (p/2)/(p/2 + 1/2) = p/(p + 1)$, and $P[(\text{b})] = (1/2)/(p/2 + 1/2) = 1/(p + 1)$.

From the adversary's perspective, observing one round in which no message appears for Nym N, and in which A participated but B did not, reduces the relative likelihood of A being the owner by a factor of p. Observing similar events across multiple rounds exponentially increases the adversary's "certainty" of B being the owner: after k such rounds, the likelihood of A being the owner is only $p^k/(p^k + 1)$.

2.3.3 Indistinguishability Under Probabilistic Attack

The above reasoning generalizes to many users, varying probabilities of posting, etc. Our focus is not on deepening such analysis, however, a goal admirably addressed in prior work [17, 43]. Instead, we wish to achieve measurable resistance to *unknown* probabilistic attacks: we do not know the probabilities with which users will attempt to post in particular rounds, how well the unknown attacker may be able to predict when the owner of a given Nym will post, etc.

Instead of relying on the relevance of any *particular* probabilistic analysis—which may break each time a known attack is refined—Buddies relies on an *indistinguishability* principle that applies to all attacks of this class. If two users A and B have exhibited *identical* histories with respect to inclusion in each round's filtered user set $\mathbb{P}_i$, across *all* rounds i in which a Nym N was scheduled so far, then under any probabilistic analysis of the above form the adversary must assign identical probabilities to A and B owning Nym N. That is, if for every round i, it holds that $(A \in \mathbb{P}_i) \iff (B \in \mathbb{P}_i)$, then users A and B are *probabilistically indistinguishable* from each other, hence equally likely to own Nym N.

For any user A and Nym N, we define A's *buddy set* $B_N(A)$ as the set of users probabilistically indistinguishable from A, including A itself, with respect to potential ownership of Nym N. If n users are probabilistically indistinguishable from A, then under the attacker's analysis each such user in $B_N(A)$ has an individual probability no greater than $1/|B_N(A)|$ of being the owner of N. Intuitively, buddy-sets form equivalence classes of users who "hang together" against probabilistic intersection attacks—so that individual buddies do not "hang separately."

We next define a second anonymity metric, *indistinguishability set size*, or *indinymity* for short, as the size of the *smallest* buddy-set for a given Nym N. Since we do not know how a real attacker will actually assign probabilities to users, indinymity represents the minimum level of anonymity a member of *any* buddy set can expect to retain, even if the adversary correctly intersects the owner's anonymity set down to the members of that buddy set. Thus, the attacker cannot (correctly) assign a probability greater than $1/|B_N|$ to *any* user—including, but not limited to, the owner of N.

One might argue that we "mainly" care about the buddy set containing the true owner of N, not about other buddy sets not containing the owner. A counter-argument, however, is that a particular observation history might make some other buddy set falsely appear to the adversary as "more likely" to own N. In this case, we may well care how much protection the innocent members of that "unlucky" buddy set have against being "falsely accused" of owning N. Thus, to ensure that *all* users have the "strength in numbers" of being indistinguishable in a crowd of at least n users, regardless of the adversary's probabilistic reasoning, we must ensure that *all* buddy sets have size at least n.

3. ATTACK MITIGATION POLICIES

Based on the above architecture, we now explore possible intersection attack mitigation policies. We make no claim that these are the "right" policies, merely a starting point for ongoing refinement. Two key benefits of Buddies' architecture, however, are to modularize these policies into replaceable components independent of the rest of the anonymous communication system so they can be further evolved easily, and to ensure by system construction that policies cannot leak sensitive information *other than* by failing to protect adequately against intersection attacks.

We first explore policies for maintaining possinymity, then policies to enforce a lower bound on indinymity. An important caveat with any anonymity metric is that Buddies cannot guarantee that *measured* anonymity necessarily represents *useful* anonymity, if for example an attacker can compromise many users or create many Sybil identities [19]. Section 4.4 discusses these issues in more detail.

3.1 Maximizing Possinymity

The possinymity metric defined in Section 2.3.1 considers only rounds in which non-null messages appear for some Nym N, intersecting the filtered user sets across all such rounds to determine N's possinymity set P_N. We consider several relevant goals: maintaining a minimum possinymity level, mitigating the *rate* of possinymity loss, or both.

Maintaining a Possinymity Threshold.

Suppose a dissident, posting anonymously in a public chat room under a Nym N, wishes to maintain "plausible deniability" by ensuring that $|P_N| \geq 100$ throughout the conversation—and would rather be abruptly disconnected from the conversation (or have Nym N effectively "squelched") than risk $|P_N|$ going below this threshold. As a straightforward policy for this case, at step 5 of each round i, the Policy Oracle computes the new possinymity that N would have if $\mathbb{O}_i$ is intersected with N's "running" possinymity set from the prior round. The Policy Oracle returns $\mathbb{P}_i = \mathbb{O}_i$ if the new possinymity remains above threshold, or $\mathbb{P}_i = \emptyset$ otherwise.

In practice, the effect is that N's possinymity starts out at an initial maximum of the total set of users online when the user first posts via N, then decreases down to (but not below) the possinymity threshold as other users go offline, either temporarily or permanently. This policy has the advantage of not reducing the usability of N, or artificially delaying the time at which the user's posts appear, as long as the possinymity set remains above-threshold. Once N's possinymity set reaches the set threshold, however, all of the users remaining in this set become *critical*, in that N becomes unusable for posting once *any* remaining member goes offline. In the "dissident scenario" we envision this event might be the user's signal to move to a new network location: e.g., get a fresh IP address at a different Internet cafe.

Limiting Possinymity Loss Rate.

An alternative, or complementary, goal is to reduce the rate at which N's possinymity decreases. In realistic scenarios, as our trace data in Section 5 illustrates, clients often get delayed or disconnected temporarily but return soon thereafter. Thus, a more refined policy might temporarily halt all posting for Nym N—by returning $\mathbb{P}_i = \emptyset$—when members of N's current possinymity set go offline, in hopes that the missing members will soon return. To ensure progress and get N "unstuck" if members remain offline for a sufficient number of rounds, however, the policy eliminates these persistently offline members from N's permanent possinymity set by returning a smaller (but nonempty) $\mathbb{P}_i$. Such a policy may "filter out" possinymity set losses due to otherwise-reliable users going offline briefly, at the cost of delaying a user's posting to Nym N for a few rounds if some current possinymity set member goes offline permanently. Of course, a loss rate limiting policy may readily be combined with a threshold-maintaining policy of the form above. A further refinement of this combination might be to increase the loss limiting policy's "tolerance"—number of rounds a user may remain offline before being eliminated—as the Nym's possinymity set size falls to approach the user's specified lower bound. Such a policy in essence trades more *temporary* unavailability for greater total Nym longevity.

Users Worth Waiting For.

While the above simple variants suggest starting points, we envision many ways to refine policies further, for example by recognizing that a user's record of past reliability is often a predictor of future reliability. To maximize a Nym N's possinymity and minimize anonymity loss rate, while also limiting delays caused by *unreliable* users, we may wish to consider some members of N's current possinymity set to be "more valuable" than others: e.g., users who have remained online and participating reliably for a long period with at most a few brief offline periods. In particular, a policy might apply an offline-time threshold as discussed above to limit loss rate, but apply *different* thresholds to different members of N's current possinymity set, giving longer, more generous thresholds to more "valued" users.

The Policy Oracle can build up reliability information about users starting when those users first appear—not just when a particular *pseudonym* of interest is created. Thus, a policy that Buddies applies to a particular Nym N can benefit from user history state that the Policy Oracle may have built up since long before N was created.

3.2 Guaranteeing Minimum Indinymity

The possinymity metric considers intersection attacks only across rounds in which non-null messages appear, but in realistic chat or blogging scenarios a user's posts will be interspersed with idle periods during which *no* message appears. A smarter adversary can use the predictive techniques discussed in Section 2.3.2 to glean probabilistic information from such rounds. We therefore wish to guarantee users some level of indinymity, even under probabilistic attacks.

Forming and Enforcing Buddy-Sets.

To guarantee a Nym N will have a minimum indinymity of some value K, the Policy Oracle must ensure that all of N's buddy-sets—subsets of users whose members exhibit *identical* online behavior across rounds after user filtering—are all of size at least K. As a straw-man approach to buddy-set formation, on the creation of Nym N, the Policy Oracle could divide N's initial user roster $\mathbb{M}$ into $\lfloor |\mathbb{M}|/K \rfloor$ arbitrary buddy-sets "up-front," each containing at least K users. At step 5 of each round i, for each buddy-set B containing *any* offline user $u \notin \mathbb{O}_i$, the Policy Oracle removes *all* members of u's buddy set B from its filtered user set $\mathbb{P}_i$. The

Policy Oracle effectively *forces* members of each buddy set to come online or go offline "in unison," keeping them permanently indistinguishable under probabilistic intersection attack (even if the adversary can distinguish one buddy set from another). This straw-man policy is likely to yield poor availability, however, as it prevents N's owner from posting whenever *any* member of the owner's buddy-set is offline, making N unusable in the fairly likely event that any member of this buddy set is unreliable or disappears permanently.

Lazy Buddy-Set Formation.

As a first refinement, therefore, the Policy Oracle can delay buddy-set decisions until users actually go offline. At creation, a Nym N starts with one large buddy-set containing its entire user roster $\mathbb{M}$. In the first round i in which member(s) of $\mathbb{M}$ go offline, the Policy Oracle might first delay *all* posting for Nym N by returning $\mathbb{P}_i = \emptyset$, in hopes the missing member(s) will return online soon, as discussed above. Once the Policy Oracle "gives up" on one or more members, however, it splits N's current buddy-sets into two, isolating all persistently offline members into one of the resulting buddy-sets. By delaying the decision of how to split buddy-sets, the Policy Oracle guarantees that after the split-point, the buddy-set containing only online members will have a chance to "make progress"—at least until more users go offline for sufficiently long to force another buddy set split.

After a split, each of of the resulting buddy-sets must be at least of size K to maintain an indinymity lower bound. If fewer than K total users are actually offline ($|\mathbb{M} - \mathbb{O}_i| < K$), then the Policy Oracle must "sacrifice" a few online users, placing them in the offline users' buddy-set. Otherwise, if for example a single offline user forcing a split is actually the owner of Nym N, then placing the offline user in a buddy set of size 1 would quickly reveal to a probabilistic attacker that the now-offline user owned the pseudonym, once the attacker notices that posts have stopped appearing on Nym N. By "sacrificing" $K - 1$ additional users at the split point, even if the attacker infers from the absence of posts that N's owner is in the now-offline buddy-set, he cannot tell whether the owner is the user who *caused* the split, or is one of those sacrificed and *forced* offline to "keep him company." If the offline users in a buddy set eventually return online, then the whole buddy set rejoins in unison, making the Nym usable again if the owner was a member of this buddy set.

Choosing Whom to Sacrifice.

When the Policy Oracle must "sacrifice" online users to pad an offline buddy set to size K, an important issue is how to choose which online users to sacrifice. We investigated two classes of sacrificial policies: random, and least reliable users. Random choice clusters users into buddy sets regardless of reliability, which is simple but risks sacrificing reliable users by mixing them into buddy sets containing unreliable or permanently offline users. Alternatively, the Policy Oracle might first sacrifice users with the weakest historical record of reliability: e.g., users who arrived recently or exhibited long offline periods. By this heuristic we hope to retain the most reliable users in the buddy set that remains online immediately after the split—though this buddy set may split further if more nodes go offline in the future.

We expect reliability-sensitive policies to maximize a Nym's effective lifetime, *provided* the Nym's true owner is one of the more reliable users and does not get sacrificed into an unreliable or permanently offline buddy set. A short-lived or unreliable user cannot expect his Nyms to be long-lived in any case: a long-lived Nym must have a "base" of reliable, long-lived users to maintain anonymity under intersection attack, and the Nym's owner must obviously *be* a member of the long-lived anonymity set he wishes to "hide in."

3.3 Varying Policies and Nym Independence

So far we have assumed the Policy Oracle enforces a "global policy" on all Nyms, but this would limit flexibility and perhaps unrealistically require all users to "agree on" one policy. Instead, Buddies allows each Nym to have a separate policy—e.g., different lower bounds and anonymity loss mitigation tradeoffs—chosen by the Nym's owner.

Since intersection attacks are by definition not an issue until a Nym N has been in use for more than one round, N's owner specifies the policy parameters for a Nym N in its first post to N. The set of users online in this first message round, in which N's policy is set, forms N's initial user roster $\mathbb{M}$, which is also by definition the maximum anonymity set N can ever achieve under intersection attack. In subsequent rounds in which Nym N is scheduled, the announced policy for N determines the Policy Oracle's behavior in filtering N's user sets to mitigate intersection attacks.

Each Nym's policy is thus independent of other Nyms—including other Nyms owned by the same user. This policy independence, and the correctness of Section 2's analysis, depend on the assumption made in Section 2.1 that the Anonymizer assigns Nyms to users *uniformly at random and independent of all other Nyms*. Otherwise, the choices the Policy Oracle makes on behalf of one Nym might well leak information about other Nyms. This leads to some specific design challenges addressed below in Section 4.2.

To illustrate the importance of independent Nym assignment, suppose there are two users, to whom the Anonymizer *non-independently* issues two Nyms N_1 and N_2, via a random 1-to-1 permutation—always giving each user exactly one Nym. In the first communication round, N_1's owner announces a weak policy with a minimum buddy set size of 1, but N_2 demands a buddy set size of 2. The adversary later sees a non-null post to N_1 while user B is offline, and as N_1's weak policy permits, infers that A must own N_1. If each user owns exactly one Nym, then the adversary can *also* infer that B must own N_2, violating N_2's stronger policy. With independent Nym assignment, in contrast, the knowledge that A owns N_1 gives the adversary no information about which user owns N_2, because it is just as likely that A owns *both* N_1 and N_2, as it is that each user owns exactly one Nym.

4. BUDDIES IN PRACTICAL SYSTEMS

Since Buddies' conceptual model in Section 2 is unrealistically simple in several ways, we now address key challenges of implementing Buddies in practical anonymity systems. For concreteness we will focus on the design of our Buddies prototype built as an extension to Dissent [11, 13, 52], but we also discuss the Buddies architecture's potential applicability to other existing anonymous communication systems.

4.1 Decentralizing the Anonymizer

So far we have treated the Anonymizer as a trusted "black box" component, but in a practical anonymity system we do not wish to require users to trust any single component. In a practical design, therefore, we replace Buddies' Anonymizer with one of the standard decentralized schemes for anonymous message transmission, such as mix-nets [6,8,15], DC-nets [9,50], or verifiable shuffles [7,25,39].

While agnostic in principle to specific anonymous communication mechanisms, Buddies makes two important assumptions about the Anonymizer's design. First, Buddies assumes the Anonymizer is already resistant to basic, *short-term* network-level traffic analysis and timing attacks [34,38,45]. Without basic traffic analysis protection necessary for *unlinkable* message traffic, we cannot expect to achieve reliable long-term protection for *linkable* posts via pseudonyms. Second, we assume the Anonymizer distributes trust across some group of servers, and that *there exists* at least one trustworthy server in this group—although the user need not know *which* server is trustworthy. This *anytrust* model [51] is already embodied in the relays used in mix-nets or Tor circuits [18], the "authorities" assumed in verifiable shuffles [39], or Dissent's multi-provider cloud model [52].

While "ad hoc" mix-nets and onion routing schemes are vulnerable to many traffic analysis and active attacks, variants such as MIX cascades [5,7,41] and verifiable shuffles [25,39] offer formally provable traffic analysis resistance. These systems typically work in synchronous rounds, where users submit onion-encrypted ciphertexts to a common set of mixes, who serially decrypt and permute the ciphertexts to reveal the anonymous plaintexts. To resist traffic analysis, users with no useful message to send in a round must submit encrypted "empty" messages as cover traffic.

DC-nets [9,50] similarly operate in synchronous rounds, but derive anonymity from *parallel* information coding techniques rather than *serial* mixing, achieving traffic analysis protection in fewer communication hops. Dissent [11,13,52] adapts DC-nets to a practical and scalable client-server model. By leveraging the parallel communication structure of DC-nets, Dissent achieves per-round latencies orders of magnitudes lower than a verifiable shuffle or cascade mix guaranteeing equivalent security. Dissent thus forms a natural foundation for our Buddies prototype to build on.

4.2 Creating and Extending Nyms

Buddies can conveniently represent Nyms as public/private key pairs, so that anyone may learn the public keys of all Nyms in existence, but only a Nym's owner holds the corresponding private key. Dissent runs Neff's verifiable key-shuffle [39], once per communication *epoch*, to generate a list of public keys forming a DC-nets transmission schedule for that epoch. Each client submits one public key to the shuffle, which the servers re-encrypt and randomly permute, to produce a well-known list of *slot* keys. The client holding a slot's matching key can identify its own slot, but neither the servers nor *other* clients learn who owns any honest client's slot. Adapting this mechanism to create fresh Nyms in Buddies introduces two further technical challenges: assigning Nyms to users *independently* at random, and enabling Nyms to have unlimited lifetimes.

Creating Nyms via Lotteries.

A simple way to create independent Nyms meeting the requirements in Section 3.3 is by "lottery." Each user submits a fresh public key to a verifiable shuffle, the servers jointly pick one re-encrypted key at random from the shuffled output to be a new Nym, and discard all other keys. For efficiency, we would prefer to generate fresh Nyms in batches, amortizing the cost of the shuffle across lotteries for many Nyms, while ensuring that each Nym is assigned independently. One conceivable approach is for the servers to mint a batch of e-cash "coins" [10], encrypt each coin to a random key chosen from the shuffle's output, and finally run a verifiable DC-nets round [13, 27], with one slot per coin, allowing each coin's "winner" to spend the coin and publish a fresh public Nym key. We leave detailed exploration of this challenge to future work.

Extending Nyms Across Epochs.

To enable Nyms to persist beyond one epoch, clients can use each winning ticket from a Nym lottery either to publish a fresh Nym key, or to re-publish an old Nym key, effectively "reviving" the old Nym and giving it a transmission slot in the new epoch. A lottery winner might even publish *another* user's public Nym key, effectively delegating the winning ticket's share of bandwidth in the new epoch an arbitrary Nym whose content the lottery winner finds interesting.

When a client revives its own Nym in a new epoch, the client must ensure that the set of users participating in the Nym lottery is consistent with the existing Nym's anonymity policy: e.g., that the user's buddies are also online. If some participants go offline *during* a Nym lottery, to avoid policy-checking races the servers must restart the lottery, enabling clients to re-check the new participant set before exposing an old Nym in the new epoch.

4.3 Implementing the Policy Oracle

Implementing the Policy Oracle as a single independent server would require that all clients trust the Policy Oracle server to implement their requested attack mitigation policies correctly. Although a bad Policy Oracle server cannot *directly* de-anonymize users since it does not know which users own each Nym, it could—by intent or negligence—simply make intersection attacks easy. For this reason, Buddies leverages the *anytrust* server model that the underlying Anonymizer already uses, running a virtual replica of the Policy Oracle in "lock-step" on each of the anonymization servers. The servers use standard distributed accountability techniques [28] to cross-check each others' computation of Policy Oracle decisions, halting progress and raising an alarm if any server deviates from an agreed-upon deterministic algorithm implementing the Policy Oracle. These techniques apply readily to the Policy Oracle precisely because it architecturally has access to no sensitive state, and hence all of its state may be safely replicated.

Identifying Users to the Policy Oracle.

Buddies' Anonymizer shares the set of users currently online in each round with the Policy Oracle, which implies that we must treat these online sets as "public information" that the adversary may also obtain. As discussed in Section 2.2, however, revealing *actual* user identities or locators for this purpose, such as users' public keys or IP addresses, risks *strengthening* a weak adversary into an "omniscient" adversary for intersection attack purposes.

Buddies addresses this problem by permitting clients to authenticate with the servers via *linkable ring signatures* [24, 35]. To connect, each client generates a cryptographic proof that it holds the private key corresponding to one of a *ring* of public keys, without revealing *which* key the client holds. In addition, the client generates and proves the correctness of a *linkage tag*, which has a 1-to-1 relationship with the client's private key, but is cryptographically unlinkable to any of the public keys without knowledge of the corresponding private keys. The servers track which clients are online via their linkage tags, and provide only the list of online tags to the Policy Oracle in each round, so the Policy Oracle can simulate an adversary's intersection attacks without knowing which *actual* users are online each round.

Of course, the server that a client connects to directly can associate the client's network-level IP address with its linkage tag, and a compromised server may share this information with an adversary. This linkage information does not help a global passive adversary, who by definition obtains all the same information the Policy Oracle obtains merely by monitoring the network, but may help weaker adversaries perform intersection attacks against those users who connect via compromised servers. We see this risk as equivalent to the risk clients run of connecting to a compromised "entry relay" in existing anonymity systems [15, 18]. Compromised servers are just one of the many avenues through which we must assume an adversary might monitor the network.

4.4 Malicious Users and Sybil Attacks

While Buddies can measure, and optionally enforce a lower bound on, the number of users comprising a Nym's possinymity or indinymity set, Buddies cannot guarantee that all those users are providing *useful anonymity.* In particular, if the owner of a Nym N has specified a policy mandating a minimum buddy-set size of K, but up to F other clients may be colluding with the adversary, then N's owner may have to assume that its *actual* minimum anonymity set size may be as little as $K - F$, if all F bad clients happen to—or somehow arrange to—land in the same buddy set as N's owner. Since in practical systems we don't expect users to have a reliable way of "knowing" how many other clients are conspiring against them, we treat F as an unknown variable that users may simply have to "guess" and factor into their choices of possinymity or indinymity lower-bounds. In this respect Buddies is no different from any other anonymity system some of whose users may be compromised.

Reducing vulnerability to malicious clients may be an argument in favor of random buddy-set formation (Section 3.2). Randomized policies may offer some guarantee that the malicious users present in a Nym's initial user set become "evenly distributed" among buddy-sets. In any preferential, "reputation-based" formation scheme, if the attacker can learn or correctly guess the general "level of reliability" of a Nym N's true owner—which may well be inferable from N's posting record—then the attacker's compromised nodes might deliberately exhibit a similar level of reliability in hopes of getting clustered together in the owner's buddy set. For such attacks to succeed, however, the malicious users must be present at N's creation in order to fall in N's possinymity set in the first place, and the attacker must adjust their reliability profile *after* learning "enough" about N's owner, but *before* too many buddy set splits have already occurred for N. Thus, if the Policy Oracle builds up user reputation information in a relatively conservative, long-term fashion across the users' entire histories (e.g., from before N appeared), this may make it difficult for an attacker to "steer" malicious users' reliability profiles "late in the game" to implement a cluster attack N. Clustering attacks nevertheless present a risk that more randomized buddy set formation policies may reduce.

As in any distributed system, an attacker may be able to amplify the effective numbers of malicious clients via Sybil attacks [19], creating many fake user identities. Buddies addresses this risk by requiring users to be authenticated—via linkable ring signatures as detailed above—as owners of "real" identities in some Sybil attack resistant identity space. Buddies is agnostic as to the exact nature of this public identity space or how it is made resistant to Sybil attacks. The current prototype simply is defined for "closed" groups, defined by a static *roster* of public keys listing all members, so the group is exactly as Sybil attack resistant as whatever method the group's creater uses to form the roster. To support open-ended groups, Buddies could build on one of the many Sybil attack resistance schemes, such as those based on social networks [48, 54]—or could simply rate-limit Sybil attacks via some "barrier to entry," e.g., requiring users to solve a CAPTCHA or receive a phone callback to "register" an unknown public key for participation.

5. EVALUATION

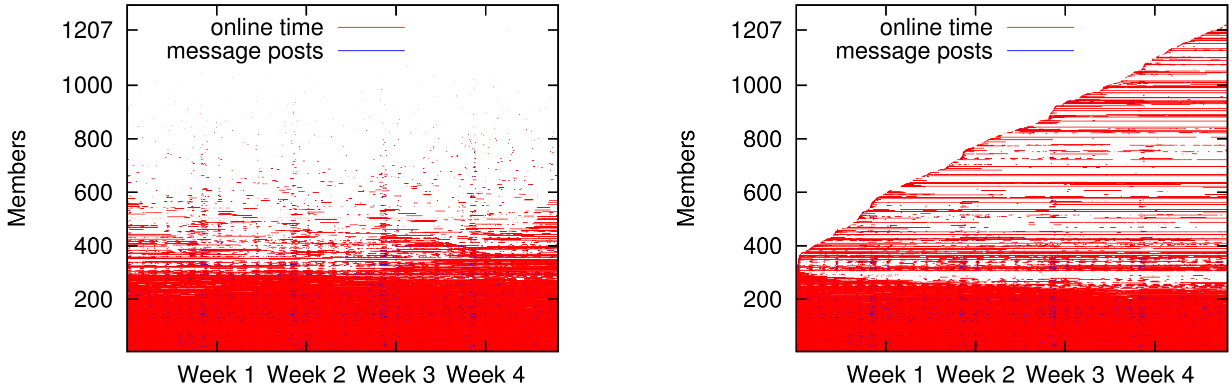
We now evaluate Buddies' utility, using data we collected from popular public IRC (Internet Relay Chat) chat rooms on EFnet servers. After introducing our data collection and simulation approach, we first explore "ideal" metrics quantifying levels of anonymity achievable in principle under given conditions—metrics that depend *only* on the user behavior dataset and not on any particular Buddies policy or loss mitigation algorithm. Next, we apply these traces to an event-based Buddies simulator to evaluate more realistic policies against these ideals. We consider naïve anonymous posting without Buddies, then posting under policies that enforce minimum buddy-set sizes, and policies that attempt to maximize possinymity. Finally, we analyze the overheads Buddies induces in the context of Dissent.

5.1 Datasets and Simulation Methodology

To evaluate Buddies' utility, we use traces taken from popular public IRC (Internet Relay Chat) chat rooms on EFnet servers. Unlike web traffic [53], IRC logs record participants online status in addition to activity or transmission of messages. The online status becomes critical for systems in which inactive but online users submit cover traffic, such as Buddies assumes. While BitTorrent traffic [2] supports similar conventions as IRC, online status and activity, user behavior focuses transferring data between peers, which does not make a strong correlation to the need for long term intersection resistant Nyms. Furthermore, the buddy system focuses on anonymous group communication systems that reveal all anonymous cleartexts to all users with at least all servers privy to the input; such behavior maps better to IRC and BitTorrent traffic than to Web traffic.

We monitored 100 (a limitation imposed by EFnet) of the most active EFnet-based IRC rooms, for over a month dating from November 26th through December 30th, 2012. For each room, we obtained the following for each member: joins, leaves, nickname changes, and messages. Anecdotally, we found that users often temporarily disconnect from IRC without IRC recognizing the disconnection. This creates a period of time in which the user must use a secondary nickname, then switch back to the original nickname once IRC recognizes the disconnect. Unfortunately we have no statistics on average disconnection time, but we were able to identify these scenarios and "fix" the data such that the affected users appear to be continuously online.

Figure 2 visually illustrates the trace collected from one sample IRC room, **football**, plotting the time period a given



(a) Members sorted by total time online

(b) Members sorted by first appearance online

Figure 2: Visualization of user online periods over one month in EFnet's **football** IRC chatroom.

user was online as a horizontal line. Figure 2(a) sorts the 1207 unique users observed during the trace vertically by total time the user was online during the observation period. This graph shows that the online forum has a "core" set of around 300 users who stay online most of the time, while the remaining users come online for shorter periods at varying times, with denser vertical stripes showing periodic patterns (e.g., football games). Figure 2(b) shows the same data with users sorted by *first* appearance online: again about 300 users were online already at the start of the trace, while new users appear subsequently at a fairly constant rate—with a fraction of these "late arrivals" remaining online for the remaining trace period.

To analyze this data we implemented an event-driven Buddies simulator in Python, cleverly called the Anonymity Simulator (AS). AS plays the role of users, the Anonymizer, and the Policy Oracle. As input the AS takes an IRC trace, time between rounds, system-wide buddy and possinymity set sizes, and buddy set formation policies. We primarily focus on random buddy set formation policies and user online times. For the latter, the AS can either use an initial period of the trace, a bootstrap period, or use a deity mode and review the entire data set. To better compare apples-to-apples, random, bootstrapped, and deity mode evaluations all began at the same time in the trace.

5.2 Ideal Anonymity Analysis

We first use our IRC traces to explore upper bounds on the anonymity we expect to be achievable in *any* system resistant to intersection attacks, under a trace-driven scenario, but independent of particular anonymity mechanisms or policies. These experiments depend *only* on analysis of the IRC data, and do not depend on Buddies' design or the Buddies simulator. This analysis serves to deepen our understanding of user behavior in realistic online forums, and to establish realistic expectations of what a system such as Buddies might achieve in principle. We first consider anonymity potentially achievable for low-latency communication using pseudonyms of varying lifetimes, then focus on *long-lived* pseudonyms in communication scenarios that can tolerate varying offline times in members of anonymity sets.

Low-latency pseudonyms of varying lifetime.

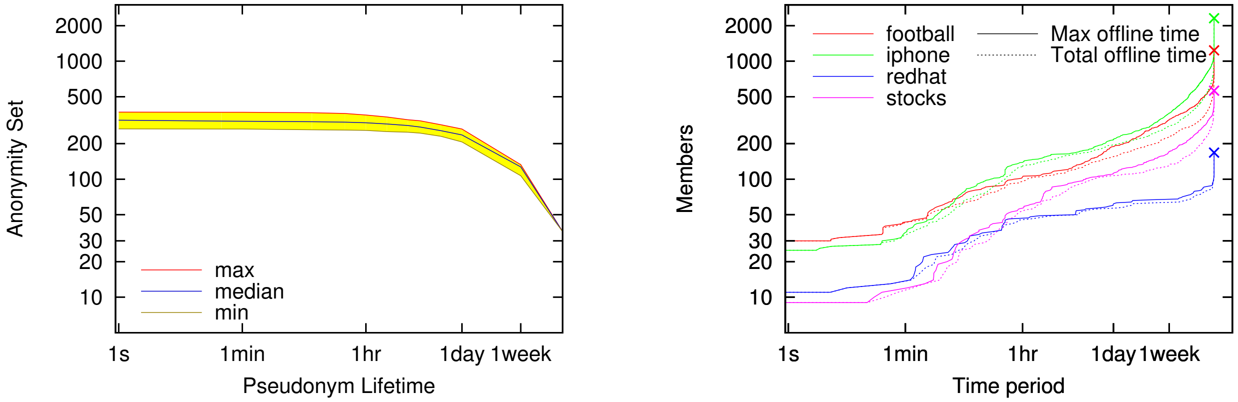
We focus on the **football** dataset, considering all online periods of all 1207 users appearing in the trace as visualized in Figure 2. We treat each contiguous online period lasting at least time x as representing a pseudonym with lifetime x, and compute an ideal anonymity set for that pseudonym as the total number of users *also contiguously online* during that pseudonym's lifetime. This analysis pessimistically eliminates from the anonymity set users with *any* offline period, however brief, during the pseudonym's lifetime.

Figure 3(a) summarizes the distribution of these ideal anonymity set sizes, for pseudonym lifetimes varying on the log-scale x-axis. Pseudonyms used for up to about one hour reliably achieve anonymity sets of at least 250 members, and sometimes up to 375 members—between 20% and 30% of the total user population observed—suggesting that substantial resistance to intersection attack may be achievable in large forums for short-lived pseudonyms. Achievable anonymity under these assumptions falls off rapidly as pseudonym lifetime increases further, however.

Long-lived pseudonyms tolerant of offline periods.

Applications that demand truly long-lived pseudonyms, such as blogs, can often tolerate longer communication latencies. Many anonymous communication schemes can *aggregate* user behavior into selectable time periods, such as batches in mix-nets [8,15], or dropoff windows in DC-net systems such as BlogDrop [12] or Verdict [13]. Users who come online and participate at *any time* during each window are indistinguishable to the adversary, enabling the system to tolerate users who go offline briefly for periods shorter than that window. Buddies policies can similarly be configured to tolerate users who go offline for brief periods without anonymity loss, as discussed in Section 3.1. In all these cases, the main cost is to increase communication latency up to that offline tolerance period. Without focusing on any particular mechanism, we now parameterize our ideal anonymity analysis on the assumption that a long-lived pseudonym has *some* way to tolerate users who go offline for periods up to a varying *maximum offline time*.

Figure 3(b) shows the ideal anonymity achievable for a hypothetical long-lived pseudonym whose lifetime is the *entire* 1-month observation period of the IRC trace, assuming that pseudonym can tolerate a varying maximum offline time shown on the log-scale x-axis. Since this scenario gives us only one "data point" per IRC trace for a given maximum offline period, the figure shows the same analysis for several different IRC datasets. Consistent with the



(a) Ideal anonymity set size potentially achievable by a *low-latency* pseudonym of varying lifetime. The pseudonym can tolerate *no* offline time in members of its anonymity set.

(b) Ideal anonymity potentially achievable for *long-lived* (1-month) pseudonyms, tolerating anonymity set members who go offline for a varying maximum offline time period.

Figure 3: Analysis of *ideal anonymity* potentially achievable based on user behavior in the IRC **football** room.

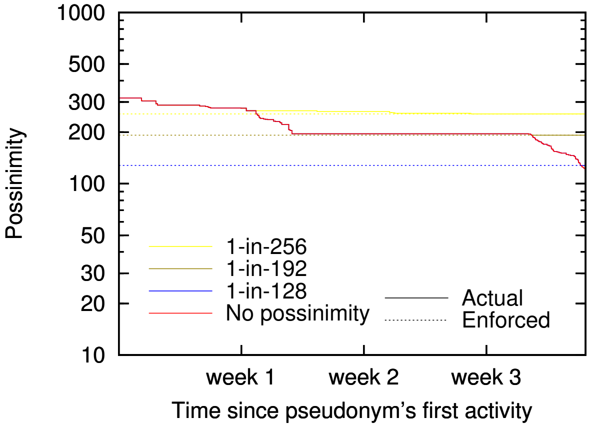


Figure 4: Evolution of worst-case possinymity over trace

right edge of Figure 3(a), long-lived pseudonyms that cannot tolerate even short offline periods—as required for low-latency communication—achieve small anonymity sets consisting of only the 10–30 users in each trace who were continuously throughout the trace. (The leftmost portion of this graph may be optimistic, in fact, as the IRC server may not have logged temporary network disconnections shorter than the TCP timeout of around 30 seconds.) A long-lived pseudonym that can tolerate disconnections up to one hour, however, can achieve 100-user anonymity sets across the 1-month trace in the **football** and **iphone** rooms. A pseudonym that can tolerate 1-day disconnections—realistic for a blog whose author posts at most once per day anyway—can achieve still larger anonymity sets of around 200 users, or 6% of the **football** group's total observed membership.

In general, the IRC datasets exhibit a common pattern where a small set of dedicated users is almost always online, a larger set–roughly 15% to 20%–who show up about once an hour to once a day, and a large set of ephemeral users comprise around 80% of the total population. We do not expect ephemeral users to contribute usefully to the anonymity sets of long-lived Nyms, but it is important for Buddies to operate in their presence, and ephemeral users can increase anonymity for short-lived Nyms as Figure 3(a) shows.

5.3 Possinymity Analysis and Enforcement

We now explore the behavior of Buddies' possibilistic anonymity metric, as defined in Section 2.3.1, under our IRC trace workloads. While the above ideal anonymity analysis generically explored the levels of anonymity a "hypothetical" pseudonym might achieve, we now wish to use our traces to model *specific* pseudonyms and explore their behavior under specific Buddies policies. To do so, we consider the messages posted by each IRC user to represent one pseudonym in our model, we take the times these IRC messages appeared to represent a "schedule" of the times the modeled pseudonym owner had a message to post, and we define the nominal *lifetime* of each modeled pseudonym as the time from the first message to the last message. We then replay this activity under AS, the Anonymity Simulator, to evaluate the behavior of Buddies' metrics under these activity traces, and the effect of Buddies policies on *actual* communication behavior, taking into account the (sometimes indefinite) delays that Buddies sometimes imposes to preserve anonymity.

Figure 4 shows how possinymity evolves across the lifetimes of pseudonyms with 1-second round times (no offline time tolerance). The x-axis represents time since since each pseudonym's inception—i.e., from its first scheduled message. The y-axis value of the solid lines show minimum or *worst-case* possinymity across all pseudonyms whose lifetimes extend at least to that point. Each line color in the graph reflects behavior under Buddies policy that enforces a different possinymity lower bound, whose enforced level is represented by the corresponding horizontal dotted line.

Consistent with our expectations from Figures 2 and 3(a), possinymity starts at the number of users online at time of first post, and gradually decreases as users "churn" offline and get intersected out of the pseudonym's possinymity set at a relatively constant rate. In practice, enforcing a lower bound leaves pseudonyms' behavior unaffected until just before a message post would decrease the pseudonym's possinymity below the lower bound. At this point the pseudonym becomes unable to post—i.e., Buddies deliberately compromises availability to preserve anonymity—leaving possinymity constant after that point in the trace.

5.4 Indinymity Analysis and Buddy Sets

We now explore probabilistic attacks of the form discussed in Section 2.3.2, and Buddies' ability to protect users against such attacks via buddy sets. To validate the protection Buddies provides, we model one *particular* probabilistic intersection attack, in which the attacker observes the messages posted by a pseudonym, assumes the owner decided whether to post in each round with a fixed, independent probability p, and uses the analysis outlined in Section 2.3.2 to divide users into classes based on likeliness of owning the pseudonym. The attacker then makes a "best guess" at which user owns that pseudonym, and we compute the attacker's chance of guessing correctly: $1/k$ if the owner is within the k-user equivalence class the attacker judges most likely, and 0 if not. We make no claim that this particular attack is representative of the attacks most likely to be mounted by real-world adversaries, but merely use it as an example to test and visualize the effectiveness of buddy sets. We reiterate that Buddies' security is based ultimately not on knowledge or expectations of particular attacks, but on an indistinguishability principle that applies to all probabilistic attacks using observed online/offline behavior as inputs.

Figure 5 shows pseudonyms' measured worst-case anonymity under this probabilistic attack, again as a function of the length of time each pseudonym has been active. Each line color represents a different buddy set size enforcement policy, with dotted lines representing effective anonymity against the probabilistic guessing attack, solid lines representing measured possinymity for reference, and dashed lines representing the buddy set size—and hence indinymity lower bound—enforced by the respective policy. Note that these graphs have logarithmic x-axes, unlike Figure 4. While possinymity decreases at a relatively constant rate with user churn, the probabilistic attack is much faster—effectively eliminating the vast majority of an unprotected user's initial anonymity set within the first hour since a pseudonym starts posting, for example. Nevertheless, for a given enforced buddy set size, the probabilistic attack only approaches, but never violates, the enforced indinymity bound.

5.5 Effect on Usable Pseudonym Lifetime

While the above experiments validate Buddies' effectiveness at mitigating anonymity loss through both possibilistic and probabilistic intersection attacks, this security necessarily comes with usability tradeoffs. Buddies policies may make a pseudonym unusable for posting messages before the owner naturally finishes using the pseudonym, and in still usable pseudonyms, messages may be delayed.

To explore Buddies' effect on pseudonym usability, Figure 6 contains CDFs showing the distributions of *useful lifetimes* of pseudonyms under various enforced buddy set sizes. As the relevant comparison baseline, the black line representing the *nominal* case shows the distribution of pseudonym lifetimes in the IRC trace without any Buddies policy. We take this line to represent the distribution of time periods during which traced users *intended* to use a pseudonym to post messages. Each colored line in contrast shows the distribution of *actual*, useful pseudonym lifetimes upon enforcing a given enforced buddy set size.

For many pseudonyms, unfortunately, resistance to probabilistic intersection attack comes at a high cost. As Figure 6(a) shows, for example, 50% of the pseudonyms modeled in the trace have nominal lifetimes of at least one day—meaning that their owners "would like to" use them for one day—but under enforced buddy set sizes of 32 or more, 50% of pseudonyms remain usable only for about an hour under Buddies. *Some* pseudonyms remain usable for much longer even under Buddies, however: in particular, those pseudonyms whose owners are long-lived and fall into the same buddy set with other long-lived, reliable users.

For each buddy set size, the graph additionally contrasts two schemes for dividing users into buddy sets. The *Oracle* scheme "sorts" users into buddy sets based on their maximum offline time, as measured for Figure 3(b), thereby clustering users with similar reliability levels together from the perspective of an oracle who can "see" into the future. The dotted lines, for comparison, reflect a more realistic, *dynamic* scheme designed along the lines discussed in Section 3.2. This scheme clusters users into buddy sets dynamically based only on *recent past* reliability history, up to to the point in time when a large buddy set must be subdivided in order to keep a pseudonym usable. Since past reliability is not a perfect predictor of future reliability, the realistic, dynamic scheme incurs some further loss of effective pseudonym lifetime. The current dynamic scheme is simplistic and can likely be improved, so we expect the utility actually achievable in practical systems to lie somewhere between the respective solid and dotted lines.

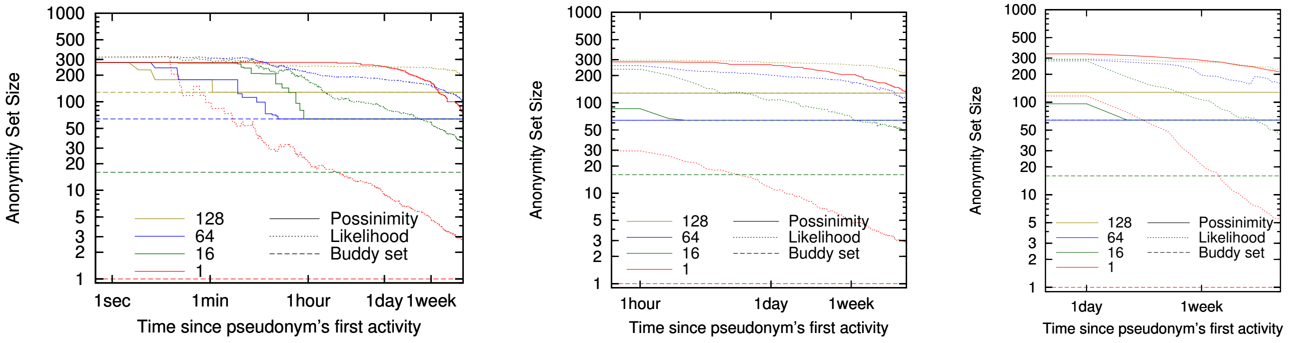
5.6 Effect on Pseudonym Messaging Delay

Finally, for messages that users successfully transmit via pseudonyms under various enforced buddy set sizes, Figure 7 shows the distribution of artificial *delays* that Buddies imposes on those message transmissions to preserve indinymity. Unsurprisingly, the percentage of messages experiencing delays increases with buddy set size, since any pseudonym's owner must effectively wait to post until all her buddies are also online, and large buddy sets increase the likelihood of some buddy being offline at the time of a desired message transmission. When a message *is* delayed, unfortunately, it is commonly delayed by at least one hour, and under large buddy set sizes often by a day or more. This result further confirms our intuition that *long-term* resistance to probabilistic intersection attacks in dynamic networks may really be feasible only for delay-tolerant transmission.

We contrast the messaging delays experienced under the two buddy set formation schemes above: one based on an oracle's perspective of users' long-term reliability, the other based on a more realistic dynamic algorithm. In this case, we find somewhat surprisingly that the dynamic scheme actually delays substantially fewer messages than the oracle scheme. This is because the oracle scheme sorts buddies into sets based only on *global* reliability measured over the entire trace, and may inappropriately group together users who were online for similar durations *but at different times.* As the dynamic scheme builds buddy sets using recent information localized in time, it better groups ephemeral users who are online at similar times. Neither of these buddy set formation schemes are likely to be close to optimal, and we consider improving them to be a useful area for future work.

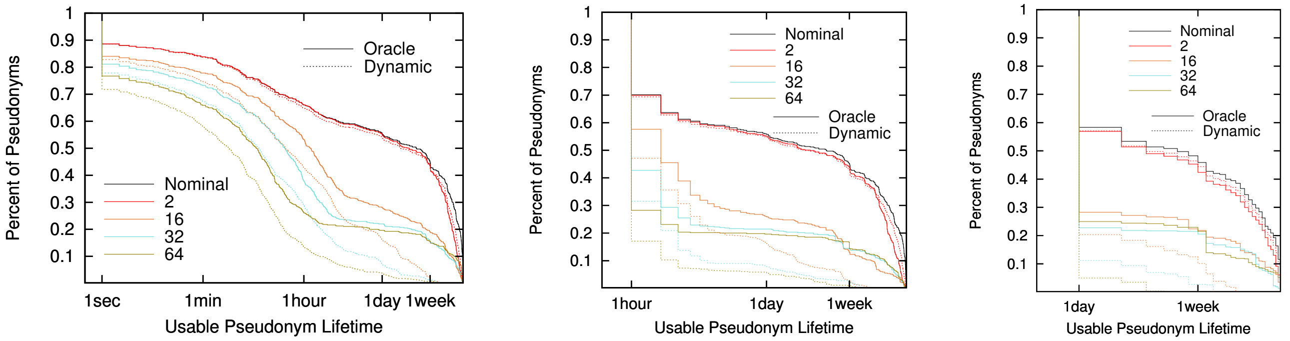
5.7 Practical Considerations

The integration of Buddies into Dissent sheds light into both the overheads of Buddies in a real system and the implementation complexities that Buddies induces. We built both Buddies and a web service for querying the various



(a) 1-second rounds **(b)** 1-hour rounds **(c)** 1-day rounds

Figure 5: Measured worst-case anonymity loss over time under one example probabilistic intersection attack.



(a) 1-second rounds **(b)** 1-hour rounds **(c)** 1-day rounds

Figure 6: CDFs showing distributions of pseudonym lifetimes, with and without indinymity enforcement via buddy sets.

Buddies meters and we modified Dissent [52] to 1) support the reactive and proactive analysis in the anonymity protocol and 2) transmit the set of online members (those with ciphertexts used in Dissent) along with the cleartext messages at the end of each round. Buddies totaled 528 lines of C++ code, while Dissent incurred only 41 lines of additional C++ code. Included within the 528 lines of C++ code, Buddies comes with both a static and dynamic policy each weighing in at 89 and 172 lines of code, respectively.

In Dissent, the buddy set concept applied cleanly; however, in the current Dissent implementation, the maintenance of possinymity is not ideal. While DC-nets theoretically allow serial processing of anonymous cleartexts allowing finer grained control over possinymity, currently, Dissent servers perform the cleartext revealing process for all scheduled Nyms in parallel, which limits possinymity evaluations to occur only before processing the first anonymous cleartext. Performing the operation iteratively would require adding additional interaction among the servers, potentially adding significant overhead. Overhead would be negligible only for rounds with interval time t in which there were m scheduled Nyms with inter-server network latency of l, where $m \times l \ll t$. Fortunately, the Dissent implementation does support scheduling Nyms during different intervals, and therefore Buddies can still make both possinymity and indinymity sets largely independent for different Nyms.

Using our Dissent implementation of Buddies, we focused evaluations on the additional delay added by Buddies. We constructed a network of 8 server machines, 64 client machines, and from 8 to 512 clients (running up to 8 clients on each client machine). Clients have a 50 millisecond delay and aggregate 100 M/bit connection to servers, while servers have a 10 millisecond delay to other servers, but each has a dedicated 100 M/bit connection to every other server. To focus on the overheads Buddies incurs, we avoided sending anonymous plaintexts across Dissent, The results, shown in Figure 8, indicate that Buddies imposes negligible overhead over Dissent—within measurement error—because performance costs are dominated by the expensive asymmetric cryptography used in signing and verifying messages.

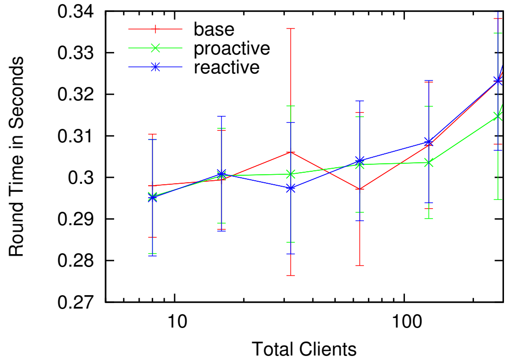
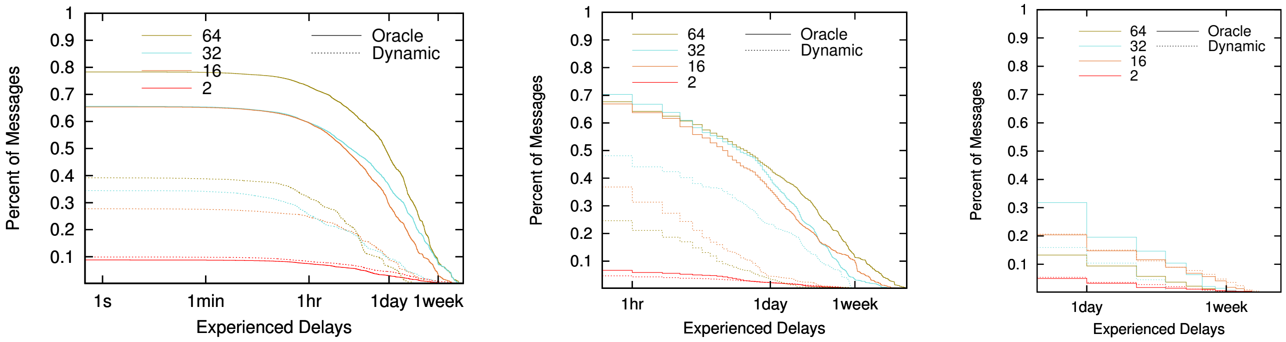


Figure 8: The overhead of using Buddies in Dissent.

6. RELATED WORK

The utility of pseudonyms has been well-recognized since Chaum's seminal paper on mix networks [8]. Pseudonymity has motivated much work in anonymous authentication [36] and signature schemes [24,35]. One way to protect distributed



(a) 1-second rounds **(b)** 1-hour rounds **(c)** 1-day rounds

Figure 7: CDFs showing distributions of delays imposed on successfully transmitted messages under Buddies.

systems from Sybil attacks [19] is to build online pseudonyms atop "real-world" identities [22]. None of these approaches protect a pseudonym's owner from being traced via network monitoring, however. We concur with recent proposals to integrate pseudonymity into network architecture [29], although this gateway-based proposal unfortunately has the same "single point of failure" weakness as a conventional single-hop proxy or commercial VPN service [1].

Multi-hop mix networks [15] and onion routing systems [18] address this single point of failure, but remain vulnerable to many traffic analysis attacks. Pseudonymous communication is by now well-known to be highly vulnerable to long-term intersection attacks [31,42], later strengthened into statistical disclosure attacks [16, 37, 53]. Padding communications to a constant rate with dummy traffic [4, 6, 14, 23] can slow—but not stop—passive intersection attacks [37].

Padding may not even slow *active* attacks, however, where an attacker deliberately perturbs performance to trace a circuit [20, 38, 45]. Active attacks are eminently realistic, being already widely used for Internet censorship [26, 49], for example. Buddies is the only architecture we are aware of that addresses both passive *and* active intersection attacks, by "collectivizing" the anonymizer's control plane logic into a Policy Oracle component that cannot see—and thus cannot leak—sensitive information (Section 2), even when replicated for accountability (Section 4.3). This architecture ensures that Buddies' attack mitigation policies apply regardless of whether clients churn "normally," or are *forced* to slow or go offline due to deliberate denial-of-service.

The Java Anonymous Proxy [3] incorporates an "Anonym-O-Meter" [21], to give users an indication of their current anonymity level. This meter does not address intersection attacks, but serves as a precedent for Buddies' computation and reporting of possinymity and indinymity metrics.

Prior systems also protect anonymity within well-defined groups. Tarzan [23] organizes an overlay of *mimics*, where each user maintains constant *cover* traffic with k other mimics to mitigate traffic analysis attacks. Systems based on DC-nets [9], such as Herbivore [46] and earlier versions of Dissent [11, 13, 52], achieve provable traffic analysis resistance for *unlinkable* messages in a *single* communication round. Since every group member typically knows the online status of every other, however, linkable transmissions using pseudonyms can make such systems *more* vulnerable to intersection attack than "amorphous" systems such as Tor. Buddies addresses this risk by using linkable ring signatures to authenticate and "tag" users (Section 4.3).

Hopper and Vasserman [30] establish anonymity among sets of k members in a mix, similar to buddy sets, and explore the resistance of these k-anonymity sets to statistical disclosure attacks. Buddies builds on this approach to offer users dynamic anonymity monitoring and active controls over tradeoffs between anonymity and performance. Aqua [6] uses padded, multipath onion routing to achieve efficiency and reduce vulnerability to traffic analysis. We expect Buddies be synergistic with designs like Aqua's, by providing a stronger and more controllable notion of intersection attack resistance than currently provided by Aqua's k-sets.

7. CONCLUSION

Buddies offers the first systematic architecture addressing long-term intersection attacks in anonymity systems, by offering passive metrics of vulnerability and active control policies. While only a first step leaving many open questions, our trace-based simulations and working prototype suggest that Buddies may point to practical ways of further protecting anonymity-sensitive users of online forums.

Acknowledgments

We would like to thank Joan Feigenbaum, Aaron Johnson, Peter Druschel, our shepherd Clay Shields, and the anonymous reviewers for their helpful feedback on this paper. This material is based upon work supported by the Defense Advanced Research Agency (DARPA) and SPAWAR Systems Center Pacific, Contract No. N66001-11-C-4018.

8. REFERENCES

[1] Anonymizer, September 2012. `http://anonymizer.com/`.
[2] K. Bauer et al. Bitblender: light-weight anonymity for BitTorrent. In *AIPACa*, 2008.
[3] O. Berthold, H. Federrath, and M. Köhntopp. Project "anonymity and unobservability in the internet". In *CFP*, April 2000.
[4] O. Berthold and H. Langos. Dummy traffic against long term intersection attacks. In *2nd PET*, 2002.
[5] O. Berthold, A. Pfitzmann, and R. Standtke. The disadvantages of free MIX routes and how to overcome them. In *Workshop on Design Issues in Anonymity and Unobservability*, pages 30–45, July 2000.
[6] S. L. Blond, D. Choffnes, W. Zhou, P. Druschel, H. Ballani, and P. Francis. Towards efficient traffic-analysis resistant anonymity networks. In *SIGCOMM*, August 2013.
[7] J. Brickell and V. Shmatikov. Efficient anonymity-preserving data collection. In *12th KDD*, pages 76–85, Aug. 2006.

[8] D. Chaum. Untraceable electronic mail, return addresses, and digital pseudonyms. *Communications of the ACM*, Feb. 1981.

[9] D. Chaum. The dining cryptographers problem: Unconditional sender and recipient untraceability. *Journal of Cryptology*, pages 65–75, Jan. 1988.

[10] D. Chaum, A. Fiat, and M. Naor. Untraceable electronic cash. In *CRYPTO*, Aug. 1988.

[11] H. Corrigan-Gibbs and B. Ford. Dissent: accountable anonymous group messaging. In *17th CCS*, Oct. 2010.

[12] H. Corrigan-Gibbs and B. Ford. Scavenging for anonymity with BlogDrop (abstract). In *Provable Privacy Workshop*, July 2012.

[13] H. Corrigan-Gibbs, D. I. Wolinsky, and B. Ford. Proactively accountable anonymous messaging in Verdict. In *22nd USENIX Security*, Aug. 2013.

[14] W. Dai. PipeNet 1.1, Aug. 1996. UseNet post. `http://www.weidai.com/pipenet.txt`.

[15] G. Danezis, R. Dingledine, and N. Mathewson. Mixminion: Design of a Type III anonymous remailer protocol. In *IEEE SP*, pages 2–15, May 2003.

[16] G. Danezis and A. Serjantov. Statistical disclosure or intersection attacks on anonymity systems. In *Information Hiding Workshop*, May 2004.

[17] C. Díaz, S. Seys, J. Claessens, and B. Preneel. Towards measuring anonymity. In *Proceedings of the 2nd international conference on Privacy enhancing technologies*, PET'02, 2003.

[18] R. Dingledine, N. Mathewson, and P. Syverson. Tor: the second-generation onion router. In *12th USENIX Security*, Aug. 2004.

[19] J. R. Douceur. The Sybil attack. In *1st International Workshop on Peer-to-Peer Systems*, pages 251–260, Mar. 2002.

[20] N. S. Evans, R. Dingledine, and C. Grothoff. A practical congestion attack on Tor using long paths. In *18th USENIX Security*, Aug. 2009.

[21] H. Federrath. The Anonym-O-Meter. `http://anon.inf.tu-dresden.de/help/jap_help/en/help/jap.JAPNewView_anonymometer.html`, July 2013.

[22] B. Ford and J. Strauss. An offline foundation for online accountable pseudonyms. In *1st International Workshop on Social Network Systems (SocialNets)*, 2008.

[23] M. J. Freedman and R. Morris. Tarzan: A peer-to-peer anonymizing network layer. In *9th CCS*, pages 193–206, 2002.

[24] E. Fujisaki and K. Suzuki. Traceable ring signature. In *10th PKC*, pages 181–200, Apr. 2007.

[25] J. Furukawa and K. Sako. An efficient scheme for proving a shuffle. In *CRYPTO*, pages 368–387, Aug. 2001.

[26] P. Gill et al. Characterizing censorship of Web content worldwide, 2013. `http://www.cs.stonybrook.edu/~phillipa/papers/GCDG_May8.pdf`.

[27] P. Golle and A. Juels. Dining cryptographers revisited. *Eurocrypt*, pages 456–473, May 2004.

[28] A. Haeberlen, P. Kouznetsov, and P. Druschel. PeerReview: Practical accountability for distributed systems. In *21st SOSP*, Oct. 2007.

[29] S. Han et al. Expressive privacy control with pseudonyms. In *SIGCOMM*, Aug. 2013.

[30] N. Hopper and E. Y. Vasserman. On the effectiveness of k-anonymity against traffic analysis and surveillance. In *WPES*, Oct. 2006.

[31] D. Kedogan, D. Agrawal, and S. Penz. Limits of anonymity in open environments. In *5th International Workshop on Information Hiding*, pages 53–69, Oct. 2002.

[32] D. Kelly, R. Raines, R. Baldwin, B. Mullins, and M. Grimaila. Towards mathematically modeling the anonymity reasoning ability of an adversary. In *IPCCC*, pages 524–531. IEEE, 2008.

[33] D. J. Kelly. *A taxonomy for and analysis of anonymous communications networks*. PhD thesis, Wright Patterson AFB, OH, USA, 2009. AAI3351544.

[34] B. Levine, M. Reiter, C. Wang, and M. Wright. Timing attacks in low-latency mix systems. In *Financial Cryptography*, pages 251–265. 2004.

[35] J. K. Liu, V. K. Wei, and D. S. Wong. Linkable spontaneous anonymous group signature for ad hoc groups. In *Australian Conference on Information Security and Privacy*, pages 614–623, July 2004.

[36] A. Lysyanskaya, R. L. Rivest, A. Sahai, and S. Wolf. Pseudonym systems. In *Selected Areas in Cryptography*, pages 184–199. Springer, 2000.

[37] N. Mathewson and R. Dingledine. Practical traffic analysis: extending and resisting statistical disclosure. In *PET*, May 2004.

[38] S. J. Murdoch and G. Danezis. Low-cost traffic analysis of Tor. In *IEEE Security and Privacy*, pages 183–195, May 2005.

[39] C. A. Neff. A verifiable secret shuffle and its application to e-voting. In *CCS*, pages 116–125, Nov. 2001.

[40] A. Pfitzmann and M. Hansen. A terminology for talking about privacy by data minimization: Anonymity, unlinkability, undetectability, unobservability, pseudonymity, and identity management, Aug. 2010.

[41] A. Pfitzmann, B. Pfitzmann, and M. Waidner. ISDN-mixes: Untraceable communication with very small bandwidth overhead. In *GI/ITG Conference on Communication in Distributed Systems*, February 1991.

[42] J.-F. Raymond. Traffic analysis: Protocols, attacks, design issues and open problems. In *Workshop on Design Issues in Anonymity and Unobservability*, pages 10–29, 2000.

[43] A. Serjantov and G. Danezis. Towards an information theoretic metric for anonymity. In *Proceedings of the 2nd international conference on Privacy enhancing technologies*, PET'02, 2003.

[44] V. Shmatikov and M.-H. Wang. Measuring relationship anonymity in mix networks. In *WPES*, Oct. 2006.

[45] V. Shmatikov and M.-H. Wang. Timing analysis in low-latency mix networks: Attacks and defenses. In *Computer Security–ESORICS 2006*. Springer, 2006.

[46] E. G. Sirer, S. Goel, M. Robson, and D. Engin. Eluding carnivores: File sharing with strong anonymity. In *SIGOPS EW*, Sept. 2004.

[47] C. Soghoian. Surveillance and security lessons from the Petraeus scandal. 12 2012.

[48] N. Tran, B. Min, J. Li, and L. Submaranian. Sybil-resilient online content voting. In *6th NSDI*, pages 15–28, Apr. 2009.

[49] J.-P. Verkamp and M. Gupta. Inferring mechanics of Web censorship around the world. In *2nd FOCI*, Aug. 2012.

[50] M. Waidner and B. Pfitzmann. The dining cryptographers in the disco: Unconditional sender and recipient untraceability with computationally secure serviceability. In *Eurocrypt*, pages 302–319, Apr. 1989.

[51] D. I. Wolinsky, H. Corrigan-Gibbs, B. Ford, and A. Johnson. Scalable anonymous group communication in the anytrust model. In *EuroSec*, Apr. 2012.

[52] D. I. Wolinsky, H. Corrigan-Gibbs, A. Johnson, and B. Ford. Dissent in numbers: Making strong anonymity scale. In *10th OSDI*, pages 179–192, Oct. 2012.

[53] M. K. Wright, M. Adler, B. N. Levine, and C. Shields. Passive-logging attacks against anonymous communications systems. *TISSEC*, May 2008.

[54] H. Yu, P. B. Gibbons, M. Kaminsky, and F. Xiao. SybilLimit: A near-optimal social network defense against sybil attacks. In *IEEE Symposium on Security and Privacy*, pages 3–17, May 2008.