Dong Lin, Micah Sherr, and Boon Thau Loo

Scalable and Anonymous Group Communication with MTor

Abstract: This paper presents *MTor*, a low-latency anonymous group communication system. We construct MTor as an extension to Tor, allowing the construction of multi-source multicast trees on top of the existing Tor infrastructure. MTor does not depend on an external service to broker the group communication, and avoids central points of failure and trust. MTor's substantial bandwidth savings and graceful scalability enable new classes of anonymous applications that are currently too bandwidth-intensive to be viable through traditional unicast Tor communication—e.g., group file transfer, collaborative editing, streaming video, and real-time audio conferencing.

We detail the design of MTor and then analyze its performance and anonymity. By simulating MTor in Shadow and TorPS using realistic models of the live Tor network's topology and recent consensus records from the live Tor network, we show that MTor achieves a 29% savings in network bandwidth and a 73% reduction in transmission time as compared to the baseline approach for anonymous group communication among 20 group members. We also demonstrate that MTor scales gracefully with the number of group participants, and allows dynamic group composition over time. Importantly, as more Tor users switch to group communication, we show that the overall performance and utilization for group communication improves. Finally, we discuss the anonymity implications of MTor and measure its resistance to traffic correlation.

Keywords: Tor, Anonymity, Multicast

DOI 10.1515/popets-2016-0003
Received August 31, 2016; revised December 2, 2015; accepted December 2, 2016.

1 Introduction

The increasing demand for practical anonymous *group* communication has spurred a large crowd of commercial ventures. Popular anonymous apps are available on iOS and Android devices that allow users to gossip anonymously with their friends [43], with nearby users [49], or with all users of the system [48]. Facebook recently unveiled its own anonymous app [42] to foster anonymous group discussion among people with similar interests. However, these solutions introduce a single point of trust, since one compromised server—or one subpoena—can break users' anonymity. This threat is not merely academic: Whisper [48] was reported to silently track their users' locations [30].

Anonymous group communication can be straightforwardly achieved using unicast anonymity networks (e.g., Tor [12]) and an external facilitator. Here, group members anonymously send their messages to the facilitator, which then "echoes" the messages to group members. While such a design may offer stronger anonymity (depending upon the underlying anonymity network), it incurs unnecessary bandwidth and latency overheads and scales poorly with group size. These bandwidth costs prohibit particularly interesting use-cases for anonymous group communication, including single-source streaming broadcasts and real-time group video conferencing.

This paper presents MTor, a practical anonymous group communication system that supports dynamic group composition with scalable performance. We construct MTor as an extension of Tor, benefitting Tor's large user base by enabling new types of anonymous applications (e.g., multiparty conferencing) while also benefitting from Tor's network infrastructure and its mature design and implementation. MTor constructs multicast trees of Tor relays across group participants. Any user can enter and leave the group communication without global coordination by joining as leaves to these trees.

Group communication for Tor. MTor provides a group communication primitive in which any member of a multicast group may originate a message; such messages are tunneled through anonymous Tor circuits to all other group members. Our security goal, which we describe in more detail below, is to prevent an adversary from (1) discerning the sender of the message and (2) enumerating group members.

We envision that any number of such groups (including zero) may exist on Tor at any given time. To participate in a group, we assume that group members obtain a succinct *group descriptor* document. The group descriptor contains sufficient information to allow a client to join the multicast group, and uses a key blinding signature scheme [32] to enforce access controls over the group while providing unlinkability between group sessions.

A significant advantage of MTor is that multicast allows for aggregation of messages, which eliminates redundancy and conserves bandwidth. To illustrate, Figure 1 shows an example multicast tree constructed with MTor. Here, five clients (1-5)

Dong Lin: University of Pennsylvania
Micah Sherr: Georgetown University
Boon Thau Loo: University of Pennsylvania

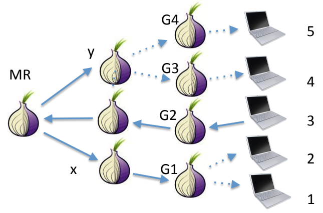


Fig. 1. Example multicast tree. Solid lines indicate points of aggregation (and bandwidth savings).

construct three-hop Tor circuits to the MR. As with unicast Tor circuits, clients begin their circuits with fixed guard relays (G1-G4). In contrast to normal Tor communication, MTor aggregates identical traffic that flows across a single connection between relays. For example, consider the case in which client 3 sends a message to the group. The message is sent via a Tor circuit to the MR, which then forwards it to its two neighbors (relays "x" and "y"). Only one copy of the message is sent for each connection. This preserves bandwidth, since x and y are each servicing two downstream clients. Similarly, bandwidth is again preserved when x forwards along the message to G1, despite the presence of clients 1 and 2 on the multicast tree.

Notably, MTor does not rely on *exit relays*—relays that serve as egress points for the Tor network. In MTor, *all* traffic is sent within Tor. Exit relays are unnecessary in MTor since traffic never leaves the Tor network. In a mixed Tor network that carries both unicast and multicast traffic, this is a desirable feature: exit relays (which are valuable since they constitute only approximately 1/3 of all relays in the live Tor network) can be reserved for unicast traffic.

Challenges. MTor presents a number of scientific and engineering challenges. First, while there is a large body of existing work that attempts to quantify sender and receiver anonymity for unicast messaging [10, 20, 36, 41, 44, 45], methods for measuring anonymity in a group setting are less well-defined.[1] We adapt previously proposed human-understandable anonymity metrics [29] to our group communication setting.

Second, MTor runs atop of the Tor network, and hence inherits many aspects of Tor's design, some of which pose interesting challenges for multicast communication. In particular, Tor currently uses TCP between all relays, which presents potential bottlenecks for multicast traffic. We introduce simple congestion techniques for achieving performant multicast communication, even when TCP is used as the underlying transport mechanism.

Finally, Tor is a highly distributed system used by hundreds of thousands [19] of geographically diverse [46] users. Tor requires only loosely synchronized clocks, which combined with the diversity of connection quality that is available to its user base, increases the complexity of decentralized algorithms for forming multicast trees. MTor mitigates clock skew and client networking issues through a robust multicast tree formation algorithm.

Summary of results. We evaluate MTor using Shadow [24], a high-fidelity discrete-event simulator that runs actual Tor code on a synthetic network topology. Shadow has recently been used to evaluate Tor's circuit scheduling algorithms [24, 27], its vulnerability to traffic correlation attacks [29], and proposed performance enhancements [16, 26]. By simulating MTor's path selection algorithm using historical records of Tor's consensus documents, we evaluate the bandwidth consumption and anonymity of MTor. Our results are encouraging: for large sized groups, MTor achieves a 63% savings in network bandwidth as compared to vanilla Tor; even for smaller-sized group of 20 clients, MTor achieves 29% savings in network bandwidth and provides a significantly improved client experience, decreasing the median transmission time of message delivery by as much as 73%. We also demonstrate that including middle relays in multicast trees helps achieve a better trade-off between bandwidth consumption and transmission time performance.

2 Threat model

We adopt Tor's threat model in which an adversary monitors or controls some fraction of the network [12]. For example, the adversary may operate Tor relays, or may monitor or control some portion of the underlying Internet. We assume the adversary cannot monitor all communication, since Tor is not designed to protect against global adversaries [12]. Finally, we conservatively assume that the adversary has access to the group descriptor document and can join any multicast group.

Tor is known to be vulnerable to an adversary who can observe and correlate traffic entering and exiting the anonymity network. This type of *traffic correlation attack* is arguably the most serious known de-anonymization attack against Tor [20, 47] and studies have demonstrated that even a moderately provisioned adversary can de-anonymize most Tor users within a few months [29]. In this paper, we focus on traffic correla-

[1] Indeed, anonymous group communication even lacks a coherent definition. For example, anonymous group messaging schemes such as DC-Nets [5], Dissent [7], Verdict [8], and Group OTR messaging [31] conceal which party sent a message, while assuming that the identities of group members are public; in contrast, MTor and M2 [38] (see Section 3) actively attempt to hide the identity of a message's sender, group members' network locations, and the members of the anonymous group.

tion attacks since, when successful, they identify a communication's endpoints and defeat Tor's anonymity goals.

We do not consider attacks that enumerate the relays involved in anonymous communication, since merely discovering which Tor relays were involved in an anonymous communication does not by itself reveal the participants of that communication. We emphasize that such "path discovery" attacks are trivially achievable in vanilla Tor by an adversary who operates a Tor relay and is chosen as the middle relay in an anonymous circuit; here, the malicious relay immediately learns its neighbors (i.e., the guard and the exit) and thus discovers the entire anonymous path. Importantly, learning the relays involved in an anonymous path does *not* by itself identify the network locations of the Tor client or the destination and thus does not break anonymity. This is in contrast to traffic correlation attacks—the focus of our security analysis—which *do* reveal the communicating parties.

Since, in MTor, messages may have multiple recipients, we consider two variants of a traffic correlation attack: We consider an adversary's ability to determine whether a given client is participating in a multicast group. If the adversary can monitor that client's (encrypted) communication with its guard, then we assume that the adversary can apply traffic analysis techniques to determine that the client is a subscriber of the group. Second, we consider attacks in which the adversary is able to identify both the receiver and sender of a multicast message; here, the adversary must monitor both clients' communications to correlate traffic.

Since MTor uses Tor as its backbone, we additionally assume that Tor's existing transport protocol is secure (e.g., that keys are randomly generated, that ciphers are strong and used correctly, that the implementation is correct, etc.). MTor does not impose any restrictions on the "last mile" connection between the client and the first relay (i.e., a guard or bridge) and is compatible with Tor pluggable transports and obfuscated bridges. We therefore consider local eavesdropping attacks such as website fingerprinting [2, 21, 37] orthogonal to this work, since solutions [2] and mitigations [40] intended for vanilla Tor are also applicable to MTor.

In addition to achieving anonymity against passive attacks, MTor is also designed to avoid introducing work-amplification and DoS attacks. For example, a malicious client may create a multicast group and join it many times from many Sybil identities. MTor's protections mitigate such attacks by requiring that the Sybils receive traffic that is proportional to the amount of bandwidth they are adding to the MTor network. An active adversary may also attempt to inject spurious messages into group communication. Although active adversaries may inject useless messages in an open multicast group, MTor supports private group messaging that permits restricting the set of potential senders.

3 Related Work

Multicast. IP multicast [9] provides efficient group communication at the network layer by reducing message duplication on physical links. However, it requires router support and adds complexity by requiring routers to maintain per group state. Multicast is susceptible to DoS, since by design, it amplifies messages. As a result, IP multicast is not widely deployed.

Unlike IP multicast, overlay multicast provides multicast service at the application layer. It does not rely on router support and allows multicast functionality to be incrementally deployed as more nodes join the network. The key concern regarding overlay multicast is the performance penalty involved in disseminating data using overlays rather than native IP multicast. A number of systems have been proposed to provide efficient overlay multicast, including Scribe [3], SplitStream [4], Narada [6], Overcast [23], and Yoid [15]. In comparison to these existing efforts, MTor uses an existing overlay network (Tor) and provides *anonymity*.

Anonymous multicast. A number of existing anonymous multicast schemes provide provable (unconditional) anonymity guarantees, but at the cost of limited performance or requiring that the group's composition be static. Classic DC-nets [5, 8] provides provable anonymity even against traffic analysis. But communication and computation costs have in practice limited its performance and anonymity set size. Herbivore [18] supports mass participation by securely dividing large networks into smaller DC-net groups, but guarantees anonymity only within each group, showing only scalability to 40-node groups. Dissent [7] significantly improves the scalability and performance of DC-nets by switching to a client/server architecture. But Dissent's protocol halts completely if even a single server goes offline, and its group composition cannot change after the initial setup. Furthermore, Dissent's performance relies heavily on a small set of physically co-located servers with high-bandwidth and low-latency communication among them, which reduces performance for geographically distributed clients. In contrast, MTor provides wide-area anonymous group communication with dynamic group composition and achieves performance greater than that offered by Tor.

There is also related work that examines multicast for low-latency anonymity networks. M2 [38] offers receiver anonymity for one-to-many multicast communication. However, M2 assumes mutually-trusting receivers, and does not protect the identity of the sender. It is thus not useful for many-to-many anonymous communication.

4 Design

MTor constructs a multicast tree at the application-layer using Tor relays as the internal nodes of the tree. The leaves of the tree are clients, who connect through their guard relays—i.e., the clients' guards are the parents of the clients in the tree. MTor does not use IP multicast and instead uses Tor's existing TLS transport mechanism between relays to provide link-level authentication and confidentiality. The multicast tree is constructed in a dynamic and decentralized fashion, and does not require global coordination.

For a given communication group, a single multicast tree is used for the duration of a *session*. After each session, a new multicast tree is established. As described below, we use a *key blinding scheme* [32] to provide partial unlinkability between sessions.

4.1 Group Descriptors

Before a client can participate in a group communication, it needs to obtain a *group descriptor* for that group. We envision that the group descriptor could be communicated through some out-of-band mechanism (e.g., emails or a distributed key-value store) and retrieved anonymously using vanilla Tor.

The group descriptor contains the following attributes:

- **Bandwidth.** The bandwidth attribute specifies a minimum bandwidth that relays must support to be a member of the multicast tree. A conservative estimate prevents message loss, which is possible when there are bottlenecks in the multicast tree.
- **Master key(s).** MTor makes use of a *key blinding scheme* similar to Tor's next-generation hidden services [32]. A *master keypair* consists of a master public and private key (pk and sk, respectively). Mathewson [32] introduces a key generation algorithm that given a nonce n and pk (resp. sk), can generate a blinded key pk_n (resp. sk_n) such that: $\langle pk_n, sk_n \rangle$ is a public key pair (and hence suitable for signing and verifying messages); without knowing pk, one cannot derive pk_n; without knowing sk, one cannot compute sk_n; and an adversary that observes blinded public keys and signatures cannot determine which signatures and blinded keys were derived from the same master keypair. The security of this scheme using Bernstein et al.'s Ed25519 signatures [1] was later proved by Hopper [22].

  An MTor group descriptor contains pk, the master public key, which as explained in Section 4.4, provides unlinkability between different communication *sessions* of the same group. Here, unlinkability is achieved with respect to adversaries that are not members of the group and do not possess the group descriptor. We note that this is the strongest such guarantee possible, since adversaries that can join the group (i.e., obtain the group descriptor) can trivially link group sessions—by design, all group members must be able to link group sessions to ensure undisturbed communication.

  MTor allows both *open* and *closed* groups, where the former allows anyone with the group descriptor to send and receive messages; the latter permits a smaller subset of group members to be senders. For open groups, the group descriptor contains both the master public and private keys, pk and sk, respectively. In a closed group setting, the descriptor contains only pk; users must have knowledge of sk through some outside means to send to the group. We envision that the initiator of the group communication can delegate the authority to send messages (i.e., share sk) to parties that it trusts.
- **Session length.** The session length attribute specifies the length (in seconds) of a communication session. It allows group participants to periodically change their blinded keys to make it more difficult for an outside (non-group-member) adversary to identify participants of the same group. A long session length reduces tree construction overhead and communication disruption, whereas a short session length provides greater unlinkability.
- **Number of candidate middle relays.** This attribute specifies the maximum number of candidate relays that may be selected as middle relays in the path selection in a given session. The parameter provides a way to balance the tradeoff between bandwidth savings and transmission time performance. A small number of candidate middle relays increases the opportunity of message aggregation (and hence bandwidth savings), whereas a lower number may decrease transmission time for message delivery.
- **Cipher identifiers, confidentiality key, and MAC key.** In contrast to unicast Tor, MTor does *not* by default offer any end-to-end confidentiality guarantees. This is necessary to allow bandwidth savings via link aggregation and message de-duplication. However, it also implies that any relay who is part of the group's multicast tree can eavesdrop on the communication. These optional cipher attributes provide a simple mechanism for secure end-to-end communication. However, *confidentiality relies on the secure dissemination of the group descriptor file*. Users who have access to the descriptor can protect the confidentiality of their messages by encrypting them with the symmetric confidentiality key and appending a MAC. Eavesdroppers who do not have access to the descriptor cannot learn the plaintext of the group

messages. These confidentiality extensions are used only at the endpoints (to encrypt and decrypt) and therefore do not incur any additional computational costs at Tor relays.

Each group session is uniquely identified by an integer n calculated from the session length and the current time, i.e., $n = \lceil \frac{t}{l} \rceil$ where t is the UNIX/epoch time and l is the session length. At the beginning of a session, a new blinded keypair $\langle pk_n, sk_n \rangle$ is generated.

Each MTor group has a *group identifier* (GID_n) that uniquely and concisely identifies a session for that group. Specifically, the GID_n for a group during session n is calculated as:

$$\text{GID}_n = h(\text{bandwidth}|\text{session length}|pk_n)$$

where h is a cryptographic hash function and $|$ denotes concatenation. For each Tor cell being sent via multicast to a group, we include the group identifier that uniquely identifies its corresponding group, which is used by relays to determine and aggregate flows of the same group.[2] The group identifier is also used to select a multicast root (MR) for the group, which is described in more detail in Section 4.2.

GID binding proofs. The construction of the GID allows for a *GID binding proof*, where a prover provides the GID, bandwidth, session length, and blinded public key. Importantly, the GID binding proof does *not* reveal master keypairs to relays. The verifier computes the GID from the provided inputs and verifies that the computed GID matches the provided GID. (We operate in the random oracle model and assume an ideal hash function, which we approximate in our implementation using SHA hashes.) GID binding proofs are used to bind the bandwidth and the public blinded key to a GID, and enable authenticated group communication, as described in Section 4.4.

4.2 MR Selection

To enable group communication, MTor forms multicast trees over the Tor relays. Clients join a group by forming circuits to the root of the desired group's multicast tree—i.e., the *multicast root* (MR). Thus, MTor requires a mechanism for ensuring that clients who wish to join the same group select the identical MR. More concretely, the MR selection algorithm should meet the following criteria: 1) **correctness**: all clients in a given group must agree on the same MR regardless of their startup time and location, to ensure the multicast tree spans across all clients during group communication; 2) **anonymity**: clients should select the MR without disclosing their network location; and 3) **efficiency**: MR selection should introduce little overhead to the Tor network, be stable enough for persistent group communication, and choose a relay that has sufficient bandwidth to not be the bottleneck for group communication.

```
Algorithm. SelectMR($min\_bw, GID$):
1.  $cur\_hour \leftarrow$ get_current_hour()
2.  fetch the $cons$ from cache/directory server such that:
        $cons \leftarrow \operatorname{argmin}_{cons}\{\text{get\_valid\_after}(cons) \mid cons \in \text{get\_recent\_cons}()$ and
                $\text{get\_valid\_after}(cons) \geq cur\_hour\}$
3.  for $relay \in cons$:        /* construct ring */
3.1   if is_stable($relay$) and is_fast($relay$) and get_bandwidth($relay$) $\geq min\_bw$:
3.1.1       $relay_{\text{pos}} \leftarrow \text{hash}(relay_{\text{digest}} + GID + cons) \mod 2^{160}$
3.1.2       put $relay$ on the $ring$ at $relay_{\text{pos}}$
4.  $begin_{\text{pos}} \leftarrow \text{hash}(GID) \mod 2^{160}$
5.  $relay \leftarrow$ find_next($begin_{\text{pos}}, ring$)
6.  while true:                  /* search for the first active relay */
6.1   if create_circuit_to_mr($relay$) == success:
6.1.1       return $relay$
6.2   $relay \leftarrow$ find_next($relay$, $ring$)
```

Fig. 2. MR selection algorithm. find_next($X, ring$) returns the relay on the ring whose identifier is the least greatest than X, modulo 2^{160}.

One straightforward solution is for the group initiator to register MR information in a lookup service that all other Tor clients can access, in much the same way that Tor hidden services register and advertise their introduction point [33]. This solution is easy to deploy and does not introduce any overhead to the Tor network. However, this requires exactly one client to be designated to monitor and update the MR throughout the group communication. We desire a more flexible approach that allows for MR-migration (i.e., switching the MR from one relay to another) and does not require the client that originated the group session to stay online for the session's duration.

MTor uses an alternative design that leverages Tor's existing infrastructure. In Tor, clients periodically retrieve a *consensus document* that lists the available relays, their public keys, network addresses, exit policy, status, and other information. These documents are polled either from authoritative directories—which undergo a voting protocol to form the (digitally signed) consensus—or directory caches. In either case, clients authenticate the consensus document by verifying that it has been signed by a majority of the directory authorities. As its name implies, the consensus document should be approximately consistent among all clients. To mitigate edge cases (e.g., in which a client retrieves the consensus moments before the directory authorities generate a new consensus), MTor selects MR from the oldest available consensus whose `valid-after` attribute is larger than the current hour time.

In MTor, clients independently select the MR using a local variant of consistent hashing. Since a bad MR will slow down or disrupt the communication of the entire group, MTor first applies a filtering process to weed out undesirable relays. Only

[2] Although assigning GIDs per relay (as opposed to per session) may provide stronger anonymity properties, it eliminates the possibility of aggregating message schemes, thus reducing the benefits of MTor.

relays that have earned the STABLE and FAST flags (respectively indicators of stability and performance) and can provide at least the bandwidth specified in the group descriptor's bandwidth field are considered. The remaining Tor relays are then logically organized in a ring over $[0, 2^{160})$, with each relay's value in the ring being equal to a hash over its digest (a fingerprint of the relay's public key), GID, and consensus document used for the MR selection. The GID is included to evenly distribute the multicast roots of different groups across relays. These rings are computed locally for each client using only local knowledge and a cached copy of the consensus. The client selects the MR by finding the relay whose value in the ring is the least greater (modulo 2^{160}) than the GID. The client then attempts to create a unicast Tor circuit to the MR (the mechanism for selecting relays in this circuit is described in Section 4.3). If it is unsuccessful, then the next closest value in the ring is considered the MR, and this process repeats until a live candidate relay is discovered. The complete MR selection algorithm is presented in Figure 2.

4.3 Tree Formation

A client joins a multicast group by constructing a Tor circuit to the group's MR. The procedure for building a circuit is similar to normal circuit construction in Tor, with the exceptions that the (1) MR is used in place of an exit relay; (2) to prevent certain deadlock conditions and to improve performance, we restrict the set of candidate middle relays; (3) each relay on the circuit must provide at least the bandwidth specified in the *group descriptor*; and (4) if a chosen relay is already in the tree (e.g., selected by other group members), the client stops circuit construction and uses whatever is upstream from that relay.

Relay selection. A client who wishes to use group communication first either establishes a new group by creating a new group descriptor or obtains an existing group descriptor through some out-of-band mechanism. Next, the client selects a 3-hop circuit, consisting of one of its guard relays, followed by a middle relay, and ending with the MR. All three relays should provide at least the bandwidth specified in *group descriptor*. The guard relay is selected using Tor's default bandwidth weighting strategy. The MR is selected following the algorithm described in Section 4.2.

Middle relay selection is parameterized by the *bandwidth* and *number of candidate middle relays* values specified in group descriptors. Specifically, each client independently determines a random set of candidate middle relays of the specified number of relays from Tor's available relays, such that each candidate relay does not have the GUARD flag and provides at least the specified *bandwidth*. The no-GUARD constraint prevents our distributed tree construction algorithm from running into deadlock. As an example, consider the case where client A selects path $x \rightarrow y \rightarrow MR$ and client B selects path $y \rightarrow x \rightarrow MR$. The algorithm might deadlock if the two clients begin path construction at roughly the same time. This set of relays are randomly selected using the GID$_n$ and consensus documents as random seeds to ensure that each client will select the same set of candidate relays. The middle relay is then selected from this set using Tor's default bandwidth weighting strategy.

Note that exit relays are not necessary here since all group communication is carried over Tor's TLS connections, and never "exits" the anonymity network. Clients who use MTor for group communication rather than constructing multiple unicast circuits (each of which consumes bandwidth at exit relays) are effectively saving valuable exit relay bandwidth for non-group communication.

Tree construction. After the guard, middle, and MR relays have been selected, the client starts constructing the circuit to the MR by sending a `CREATE` cell with the GID and a GID binding proof to its chosen guard. (In Tor, `CREATE` cells signal the creation or extension of an anonymous circuit.) The circuit is similarly extended to the middle and then the MR by tunneling additional `CREATE` cells, again including the group identifier and a GID binding proof. However, if the chosen relay is already in the multicast tree (e.g., selected by other group members), the client stops circuit construction and uses whatever is upstream from that relay. Clients' MTor circuits may contain fewer than three hops if either the guard or middle relay is already forwarding messages for that multicast group.

To support forwarding of multicast messages, each relay maintains a local key-value store called the *multicast forwarding table* that is keyed on the group identifier (which is communicated through `CREATE` cells) and whose values contain routing information for a group.

Upon receiving a `CREATE` cell, a relay looks up the included group identifier in its multicast forwarding table, and responds as follows:

– If the relay has not previously received a `CREATE` cell, it replies with a `CREATED` cell, mirroring Tor's default behavior. After receiving the `CREATED` cell, the client will continue its path construction towards MR via this relay.
– If the relay has already received a `CREATE` cell from another client, it replies with a `HOLD` cell, indicating that tree construction is already under progress. After receiving the `HOLD` cell, the client will wait for a `BEGIN` cell. The `BEGIN` cell is described below; conceptually, it signals that the tree has been created.

- If the relay has already received a BEGIN cell, it replies with a BEGIN cell, indicating that the tree construction is completed. The client can now start group communication.

In all cases, the relay records in its multicast forwarding table the *previous hop* from which it received the CREATE cell and the *next hop* to which it forwards CREATE. This information represents the relay's parent and children in the multicast tree. The table is later used to forward messages during group communication.

Lastly, exactly one client will receive a CREATED cell from MR. When that happens, the client informs the MR of its role, which then multicasts a BEGIN cell across the multicast tree, informing all relays and clients on the tree to start group communication.

4.4 Sending and Receiving

A client can begin sending and receiving group messages once it has received the BEGIN cell. Outgoing messages are sent via the client's Tor circuit towards the MR. In MTor, messages should traverse each edge in a multicast tree only once. When a relay receives a message, it looks up its neighbors in the tree by searching its multicast forwarding table for the records that are keyed by the group identifier. The incoming message is then forwarded to the relay's adjacent edges, excluding the message's incoming edge. Group messages percolate down the multicast tree, and are eventually delivered by guard relays to the subscribed clients.

MTor has the potential to offer significant bandwidth savings over group communication that relies on unicast Tor circuits. Consider, for example, a Strawman Solution based on vanilla Tor in which clients use an external service such as a bulletin board, IRC server, or Google Hangouts to aid in group communication. The service supports group communication by "echoing" incoming messages to all connected clients (i.e., through their Tor connections). MTor uses significantly less bandwidth than this unicast-based approach, since the former (i) offers the possibility of message de-duplication by aggregating data on shared links in the multicast tree, (ii) uses a single multicast root rather than multiple exit relays, which both frees up exit relay resources and reduces the number of relays that are involved in the group communication, and (iii) avoids the overhead of communicating with the external service. In Section 5, we empirically measure these bandwidth savings under realistic network conditions and workloads.

Message authenticity. MTor supports both open and closed forms of group communication in which the sets of potential senders are (respectively) unrestricted or restricted. Descriptors for open groups contain both pk and sk; descriptors for closed groups contain only pk.

A sender (who must possess sk and therefore be able to derive sk_n in session n) generates a message and appends an Ed25519 signature to its message. Relays that receive MTor cells verify the cell's signatures and drop messages that fail this verification step—recall that they have knowledge of pk_n via the GID binding proofs. Similarly, clients verify the authenticity of MTor cells using pk_n. We evaluate the overhead of signing and verifying in Section 5.5.

Message confidentiality. If the group descriptor contains a cipher identifier and encryption key, then all group messages are presumed to be encrypted by the messages' senders. Receivers use the decryption and MAC keys from the group descriptor to decrypt and authenticate messages. Our design is purposefully flexible and allows the creator of the group to specify the particular symmetric key cipher and MAC algorithm. Importantly, this "end-to-end" encryption of group messages is transparent to Tor relays, since messages are encrypted/decrypted only by the group members.

4.5 Churn Handling

To effectively detect and recover from link or relay failures, MTor maintains multicast tree states (i.e., its upstream and downstream links) as soft-state in each relay. The MR periodically multicasts heartbeat cell across the tree. The relay receiving the heartbeat cell will reset its timer for the incoming link; and the relay successfully sending the heartbeat message will reset its timer corresponding to the outgoing link. If any relay fails, timers of its downstream relays and clients will expire. As a result of timer expiration, the relay will remove themselves from the multicast tree, while the client will re-construct the path to MR as described in Section 4.3.

4.6 Flow Control and Message Loss

Tor uses TCP to transmit cells reliably between neighboring relays. In MTor, when a multicast cell arrives at a relay, it is duplicated and enqueued on the internal output queues associated with each of the next hops. If an output queue reaches its capacity limit, incoming cells will be dropped on that queue; if all output queues reach their limit, then the relay blocks receiving from its previous hop. Due to potential message dropping at the application-layer, MTor offers best-effort, but potentially lossy multicast messaging (as do most multicast schemes). The use of the predetermined bandwidth attribute in the group descriptor reduces the probability of loss, as relays are selected based on their ability to handle the group's predicted data rate.

4.7 Security against Active Attacks

In this section, we describe how MTor mitigates active attacks. Since MTor uses Tor as its backbone, many active attacks [14, 28, 35] against Tor and solutions intended for Tor are applicable to MTor as well. Therefore we identify and focus on attacks that are made possible due to the following unique features found in anonymous group communication: 1) a group communication system is more vulnerable to DDoS because it amplifies senders' traffic; 2) flow control is far more difficult in anonymous group communication since the effects of a single bottleneck are potentially magnified across the participating nodes; and 3) it is more difficult for clients to selectively refuse messages from anonymous adversarial clients.

Un-authenticated message flooding. An adversary may attempt to disrupt group communication by flooding the group with spurious messages. To mitigate such DoS attacks, MTor provides a weak form of authenticated multicast: only clients that have knowledge of sk_n (or sk) may send messages. (All other messages are quickly discarded by honest relays.) Senders sign their cells, storing the signature and a timestamp (to prevent replay) as an added field.

Recall that relays are given both the GID and a GID binding proof, and hence relays can extract pk_n from the proof. Relays enforce authentication by verifying received cells' signatures and dropping cells that fail verification. This mitigates potential DoS against the Tor network by preventing both malicious clients and relays from propagating unauthentic messages.

To reduce the bandwidth and computation overheads of using the above scheme, we use Ed25519 [1], a high-speed public key signature system, for authenticating cells. Ed25519 offers similar security to a 3072-bit RSA signature using only a 512-bit signature. It therefore costs 12.5% of network bandwidth to put one signature in each 512-byte cell.

For single-source streaming multicast messages, this cost may be amortized. Here, the sender may transmit special signature cells that contain a signed list of upcoming (yet-to-be-received) cell hashes. After receiving a signature cell, a relay verifies the signature over the hashes, and then verifies that the cells it subsequently receives have those hash values. Since the cost of verifying a forged packet is relatively low and an adversary has a limited opportunity of finding a collision, MTor can use truncated hash digests. For example, if hashes are truncated to 40-bits, then a single 512-byte signature cell can hold 86 40-bit hashes, a 512-bit Ed25519 signature, plus an 18-byte header, reducing the verification and storage cost by nearly two orders of magnitude.

Message flooding in Sybil attacks. A multicast network is potentially more vulnerable to flooding attacks than a unicast network, since by definition, it amplifies the traffic of the sender. To exhaust bandwidth resources in the network, an adversary can create a multicast group and join it many times from many Sybil identities.

MTor mitigates such a Sybil attack using the flow control mechanism described in Section 4.6: if all downstream clients of a relay want to jam the group communication by neglecting to receive traffic, the relay will block receiving from the upstream node to save bandwidth. If all but one client in the group stop receiving traffic, the traffic block will be pushed all the way from these clients to the guard of the client which is sending traffic. This ensures that the adversary has to receive traffic that is proportional to the load he imposes on MTor.

Message dropping at clients. A malicious client may intentionally drop packets to disrupt group communication. However, MTor offers best-effort multicast messaging by dropping messages on slow links. This ensures that a slow or malicious client cannot exhaust the memory of intermediate relays (due to message buffering) or slow down communication between other members.

Message dropping at relays. A malicious relay may also intentionally drop packets from upstream links to disrupt group communication. To mitigate such attacks, as well as to reduce the group's vulnerability to slow relays, the periodic heartbeat message described in Section 4.5 includes the count of cells transmitted during the session, signed with the signing key of the MR (signing keys are specified in Tor descriptor documents). By comparing the received number of cells with the advertised count in the heartbeat message, the downstream clients can re-connect to the MR via a different path when the packet loss rate exceeds a threshold.

5 Performance Evaluation

To provide an estimate of MTor bandwidth's consumption *on the actual Tor network*, we modified the Tor Path Simulator (TorPS) [29] to simulate MTor tree construction over a one month period of September 2014[3] , using historical records of Tor consensus documents collected by the Tor Metrics Project [46]. During the simulation the multicast tree is reconstructed every hour. The bandwidth consumption is then derived from the average size of the multicast tree. TorPS simulates the actual event of relays joining and leaving the Tor network using real relay and consensus data from Tor's historical records, and thus models the actual live Tor network as it

[3] Using the September 2014 dataset, TorPS includes 6192 relays.

existed at a specific past period in time. Using TorPS thus allows us to obtain an accurate estimate of MTor's performance had it been deployed on the live Tor network.

TorPS also provides the ability to estimate the probability of unreliability due to relay failure, as well as the resilience of Tor and MTor communication against traffic correlation attacks. In section 5.4, we define the probability of unreliability and discuss MTor's churn handling performance. In section 6, we adapt the security analysis techniques introduced by Jansen et al. [29] and measure the ability of a malicious adversary who controls some fraction of relays on the Tor network to de-anonymize group members.

To measure the network latency and transmission time as experienced by group members using MTor, we have implemented a prototype of MTor based on Tor version 0.2.3.25[4] . We also simulated our prototype using Shadow [24] following a standard Tor network modeling approach [25]. Shadow is a discrete-event network simulator that runs actual Tor code using a synthetic network stack. Shadow allows us to simulate large-scale Tor deployments and measure performance for different application scenarios. Shadow has recently been used to evaluate Tor's circuit scheduling [16, 24] and congestion management algorithms [27], as well as its anonymity properties [29].

Because Shadow bypasses many OpenSSL encryption functions in order to allow researchers to track cells, we do not use authenticated group messages or end-to-end message encryption. The Shadow experiments assume open groups that anyone can join and send/receive messages. We separately evaluate the overhead of MTor's cryptographic operations using micro-benchmarks.

Experimental setup. We use Shadow to simulate a Tor network of 455 relays, 1800 clients, and 500 client destinations (which we generically refer to below as "servers"). Relay capacities, the geographic locations of the relays, and the link latencies between the relays are configured according to the configuration supplied with Shadow, which itself is configured using data from the Tor Metrics Portal [46] following Tor modeling best practices [25, 27]. Each server is assigned 100 MBps bandwidth and clients are assigned unlimited bandwidth, which is much higher than relay capacities and thus moves the performance bottleneck to the Tor network.

To model a loaded Tor network, we include 1800 clients that fetch files from any of the 500 servers via unicast Tor circuits. To match existing studies of behavior on the live Tor network [34], 1350 clients behave as interactive web clients that fetch files of 320KB in size, and sleep for up to one minute. Additionally, 300 clients repeatedly fetch 50KB, 1MB or 5MB files, sleeping one minute in between each fetch. These types of clients continuously repeat a fetch-sleep cycle where they fetch files from a randomly selected server. Finally, another 150 clients behave as bulk clients (e.g., file sharers) that continuously fetch data from a random server and switch to a different server after every 5MB data transmission.

We include an additional 20 group messaging clients in our Shadow topology. To support our baseline comparison (see below), we also add one additional server that serves as a facilitator to support group communication via traditional unicast Tor circuits. To obtain a fair comparison with our baseline, the group descriptor used in MTor does not impose a minimum bandwidth constraint on the selected relays or limit the number of candidate middle relays.

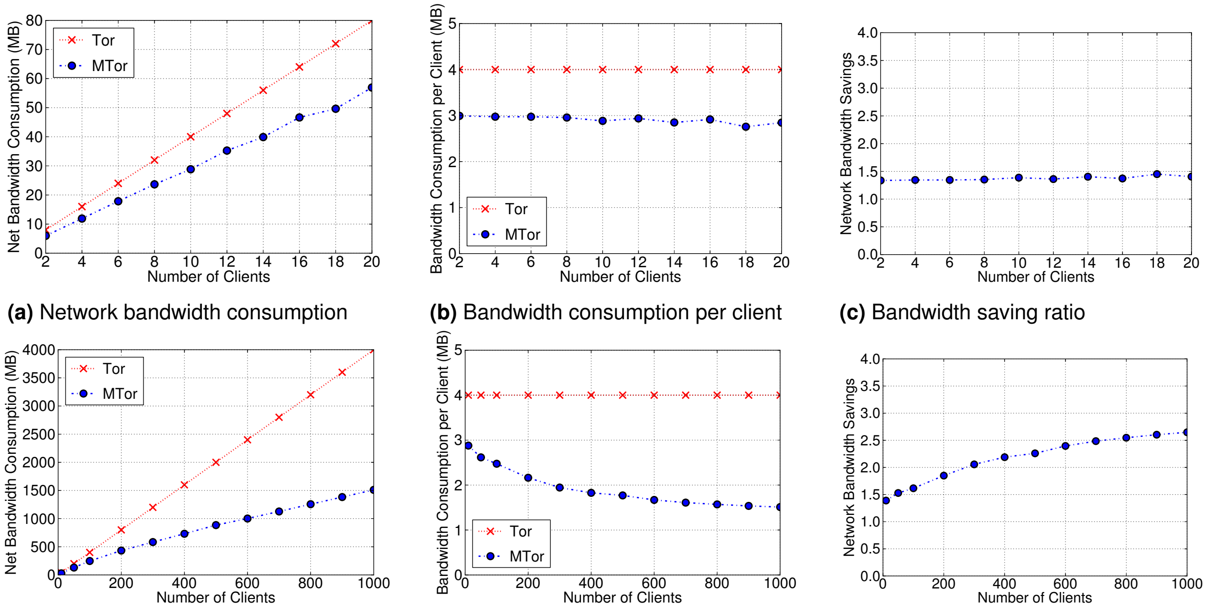
Metrics. We consider four metrics: (1) the overall *network bandwidth* that is consumed to transmit the data to all clients; (2) the *transmission time*, which measures the time it takes for a receiver to receive the sender's complete message (time-to-last-byte); (3) the packet loss rate due to a mismatch between bandwidth capacity and the bandwidth requirements of real-time communication applications; and (4) the probability of unreliability due to relay failure.

5.1 Group Communication Applications

To evaluate MTor's performance properties under different communication scenarios, we model three types of group communication applications. In all the MTor experiments, clients communicate directly via the multicast tree.

- **Single-source streaming.** In the single-source streaming application, a single non-anonymous server multicasts a file (e.g., representing a video or document) of 10MB to a group of 20 anonymous clients. In our baseline scenario, all clients connect to and receive data from the server via unicast Tor circuits. This scenario explores the transmission time improvement from using MTor in a typical initiator-responder scenario, where many initiators request the same data at around the same time.
- **Multi-source group streaming.** In the multi-source streaming application, we consider a group of 20 anonymous clients communicating with each other.
  When measuring the performance of MTor, the traffic is transmitted via the multicast tree. Since vanilla Tor does not support anonymous group communication, as a baseline for comparison, we consider a scenario in which all clients connect to an external service that "echoes" messages to all

[4] We chose version 0.2.3.25 because it is the latest version supported by Shadow at the time of implementing MTor. While there may be changes that affect performance, we expect these changes to affect MTor and Tor alike, which reduces its impact on our comparison results.



(a) Network bandwidth consumption **(b)** Bandwidth consumption per client **(c)** Bandwidth saving ratio

(d) Network bandwidth consumption **(e)** Bandwidth consumption per client **(f)** Bandwidth saving ratio

Fig. 3. Network bandwidth consumption by MTor and unicast Tor for groups of up to 20 clients (top) and groups of up to 1000 clients (bottom) for the multi-source group streaming application. Bandwidth consumption is evaluated with respect to 1MB worth of group messages that are collectively transmitted by the group's members. *(a,d)* The overall network bandwidth consumption for small and large groups. *(b,e)* The average network bandwidth consumed per client, for small and large groups. *(c,f)* The network bandwidth consumption ratio of MTor to Tor for small and large groups.

other connected clients. Tor clients connect to this external service, which we call the *facilitator*, through unicast Tor circuits. (This is effectively the Strawman Solution proposed in Section 4.4.)

- **Audio conferencing.** Our third use-case considers a group of 20 anonymous clients doing real-time voice-over-IP communication. We assume VoIP is performed using Internet Low Bit-rate codec (iLBC) [17] at 1666 Bps (iLBC is a mandatory standard for VoIP over Cable and is also used by Google Voice and Skype). Again, for our baseline configuration, all clients that rely on vanilla Tor connect to a facilitator using dedicated circuits.

  To model audio conferencing as a real-time application, each client queues a 1666-byte message per second for transmission to other clients. Old messages are dropped if they are not sent before the new message is queued. We simulate the audio conferencing for 30 minutes to measure the message loss rate and transmission time distribution.

5.2 Bandwidth Consumption

MTor offers the potential for significant bandwidth savings due to message de-duplication. To investigate how the Tor network could benefit from these savings (i.e., by having to forward less traffic), we focus in this section on the multi-source group streaming scenario. Recall that in the baseline setup, every client connects to an external service via unicast Tor circuits. We consider data transmissions from each client to every other client in the group, and evaluate the resulting load on the Tor network as a function of the group's size. Our evaluation is based on paths produced by our modified TorPS path simulator. We evaluate the bandwidth consumption when clients collectively transmit 1MB of messages to the group members (i.e., each client receives 1MB worth of message content); as we show below, the overhead of sending 1MB to the group varies considerably between vanilla Tor and MTor.

Figure 3 shows the network bandwidth consumption that results from MTor and Tor as the number of clients increases from 1 to 1000. Tor's bandwidth consumption is derived theoretically as 4×client#×1MB, since 1MB data is transmitted along 3-hop Tor circuits to the facilitator for each client in the group. To measure MTor's bandwidth costs, we simulate tree construction 720 times and compute the average number of links in the resulting multicast trees; the bandwidth is then computed as the average number of links times 1MB. These figures show the bandwidth savings MTor achieves for small (top row) and large (bottom row) group sizes.

We make the following observations: bandwidth consumption in Tor increases linearly with the number of clients by a factor of 4, whereas in MTor bandwidth consumption is sublinear. The advantage of using MTor increases with group size; MTor reduces the bandwidth cost by approximately 62% over vanilla Tor for a large group with 1000 members (Figures 3a and 3d). The bandwidth savings in MTor are due to two factors: (i) in MTor, clients' paths are shorter (consisting of two hops from the client to the MR) since they do not include links from exits to the sender; and (ii) MTor removes unnecessary cell duplication when links are shared.

Figures 3b and 3e further highlight the benefits of cell de-duplication. Here, the figures plot the bandwidth that is consumed in the Tor network, averaged across the clients, as the size of the group increases. For Tor, each group member consumes a fixed amount of 4MB bandwidth for each 1MB data transmitted, since no de-duplication occurs and each client receives the sender's communication via its own 3-hop Tor circuit. For MTor, when the size of group is 10, each client consumes on average 2.8MB bandwidth for each 1MB data transmitted. As the size of group increases to 1000, each client consumes on average only 1.5MB bandwidth, much closer to the theoretical lower bound of 1MB bandwidth necessary to serve a client. This is a direct result of de-duplication: links in the multicast tree are used by more than one client, providing the opportunity for bandwidth savings. As more clients join the group, these opportunities increase. For example, if a new client joins and its guard is already part of the group's multicast tree, then the only additional bandwidth cost due to that client is the cost of sending a copy of group messages from the guard to the client.

Figures 3c and 3f plot the savings in network bandwidth consumption when MTor is used in place of Tor for group communication, and highlight our earlier results. MTor offers increasingly efficient group communication as the size of the group increases. The savings increase from 29% for a group of 10 members to 63% for a group of 1000 members.

5.3 Impact to Client Performance

We next consider performance from the perspective of Tor users. Here, we run Tor and MTor on top of the Shadow simulator. We use *transmission time* to capture the delay experienced by end users to receive a message, since it encompasses both network congestion as well as queueing delay at the sender, receiver, and the intermediate relays. In other words, transmission time is an intuitive notion of a user's experience, which also captures the bandwidth capacity between sender and receivers.

Figure 4 compares the transmission time distribution offered by vanilla Tor (using our baseline configuration) and MTor for each of our applications. We remark that the performance improvement from using MTor is largely attributed to reduced network congestion in the Tor network, instead of performance bottlenecks at the server: although the server in the baseline setup handles one order of magnitude more traffic than clients, it is configured with 100 MBps bandwidth, much higher than relays' bandwidths.

Single-source streaming. Figure 4a shows the cumulative distribution (CDF) of transmission time to transmit a 10MB file from a single server to 20 clients. For both MTor and Tor experiments, 20 files were received and their time-to-last-byte were measured. Our Shadow experiments show that MTor provides observably improved transmission time and a much shorter tail than Tor.

MTor reduces the median transmission time by 22% from 86.7 seconds to 67.6 seconds. In the 99th percentile, the time is reduced by 43% from 317.3 seconds to 183.9 seconds. Overall, MTor reduces the latency for 55% of clients.

Multi-source group streaming. Figure 4b shows the CDF of transmission time to transmit a 10MB file during anonymous group communication. Since each client sends a copy of the file to the other 19 clients, in total 380 file copies are received by clients.

MTor significantly improves transmission time over vanilla Tor for anonymous multi-source group communication. For a small group of 20 clients, MTor reduces the median transmission time by 70% from 3482 seconds to 1032 seconds. In the 99th percentile, the time is reduced by 70% from 6921 seconds to 2046 seconds. Overall, MTor reduces the latency for 55% of clients.

In the baseline experiment, for every message that it receives, the external facilitator must transmit 19 copies (via 19 circuits) to the other group members. MTor improves performance by using message de-duplication, avoiding potentially congested exit relays, and eliminating the need to forward messages through facilitators.

Audio conferencing. Figure 4c shows the cumulative distribution of transmission times for the real-time audio conferencing application. Each client in the group attempts to send a 1666-byte message per second. To deliver real-time audio messages in a timely fashion, clients favor newer "audio samples" and drop unsent messages if a new 1666-byte message is available.

We make the following observations: MTor successfully delivers 100% of the messages, while vanilla Tor delivers only 93% of all messages. At the 50 percentile, MTor reduces the transmission time by 73% from 1.1 to 0.3 seconds. The slowest message takes 2.5 seconds to be delivered in MTor, whereas it takes 53 seconds in Tor. The result shows that MTor enables anonymous group communication with real-time delivery requirements, while Tor's message loss rate and long-tailed dis-

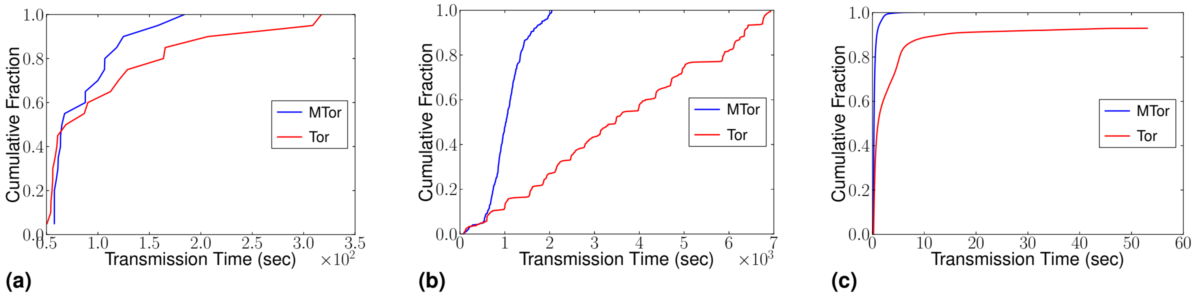


Fig. 4. Cumulative distribution of transmission time (determined using Shadow) of *(a)* 10MB files from one sender to the group during single-source streaming, *(b)* 10MB files from every client to the group during multi-source streaming, and *(c)* 1666-byte message per second from each client to the group during audio conferencing.

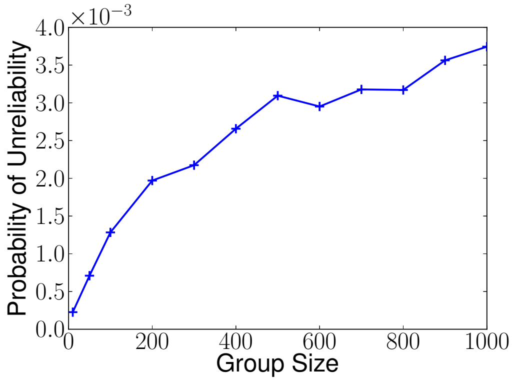


Fig. 5. Probability of unreliability

tribution of transmission times would considerably reduce the user experience for these applications.

5.4 Churn Handling Evaluation

In this section we evaluate the efficiency of the churn handling mechanism described in Section 4.5.

Suppose the heartbeat cell is sent by MR every h seconds, the timer expires after t seconds if not refreshed by heartbeat cell, and the construction of a new path from client to MR takes p seconds. If any relay fails, the downstream clients will be disconnected from MR for $t+p$ seconds, during which each client detects timer expiration and reconnects to MR via a 2-hop circuit. Since each cell has 512 bytes, the heartbeat cells will consume $512/t$ Bps of bandwidth per link.

To quantify the unreliability due to network churn, we define the probability of unreliability as the percentage of the time that *any* client in the group is disconnected from the MR. We note that this is a conservative measure of unreliability, since it assumes the disconnection of any one client will impact all other clients in the group.

As part of our experimental setup, we assume that the heartbeat cell is sent every $h = 3$ seconds, and the timer expires after $t = 9$ seconds. As evaluated using the Torflow utility [39], the construction of a 3-hop path takes roughly $p = 6$ seconds. Under this setup, the heartbeat message consumes only 170 Bps bandwidth per link. Figure 5 shows the probability of unreliability for groups of size 10 to 1000, estimated via simulation in our TorPS variant over the one month period of September 2014. Recall that TorPS uses historical data from the live Tor network and captures relay churn. We remark that for a group of size 1000, the probability of unreliability is only 0.37% — that is, less than 15 seconds of communication will be disrupted during an hourlong communication.

5.5 Authentication Microbenchmarks

Neither Shadow nor TorPS allows us to measure the computational overhead of the message authentication scheme. As microbenchmarks, we use OpenSSL version 1.0.1's speed test to measure the cost of computing SHA2 hashes. And we derive the overhead of signing and verifying 512-bit Ed25519 signatures by dividing CPU frequency by number of cycles needed for signing and verifying a signature – for Nehalem/Westmere-based processors, it takes 87548 and 273364 cycles [1] to sign a message and verify a signature, respectively.

Our "client" runs on a MacBook Pro with quad-core 2.2GHz Intel Core i7 processor and is able to generate 25K signatures and 267K hashes per second. Measurements for our "relay" are taken from a commodity server with a quad-core 2.67GHz Xeon X3450; the relay is able to verify 10K signatures per second and can perform 375K hashes per second. All measurements are pinned to a single core.

As described in Section 4.4, we can fit 86 hashes into a single signature cell when the hashes are truncated to 40-bits. Based on the above measurements, the client can send 240K authenticated cells per second (equivalently 123 MBps). The amortized verification rate for the relay is 262K cells per second; it is able to forward authenticated group message data at 134 MBps.

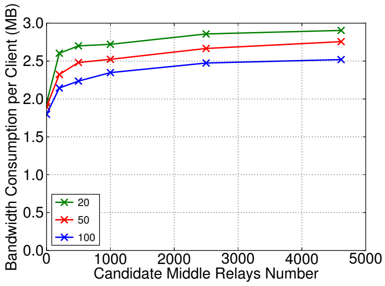


Fig. 6. Network bandwidth consumption per client.

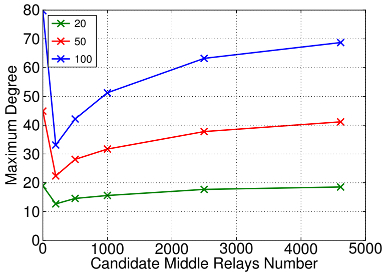


Fig. 7. Maximum degree in the multicast tree.

5.6 Middle Relay Selection

Since anonymity in MTor is independent of middle relays, we can bias the selection of middle relays to improve MTor's performance – middle relays can be sampled from a subset of all relays to improve message aggregation. We can even remove middle relays entirely from the path to the MR to save more network bandwidth. To quantify the impact of these strategies, we consider 1) network bandwidth consumption per client and 2) maximum degree in the multicast tree, as the number of candidate middle relays varies.

Figure 6 shows network bandwidth consumption per client when clients collectively transmit 1MB of messages to the group members, as the number of candidate middle relays increases from 0 to the total number of relays in Tor. Similarly, Figure 7 shows the maximum degree in the network as the number of candidate middle relays increases from 0 to the number of relays in Tor. When the number of middle relays is 0, it denotes the case where middle relays are not used in tree construction. Both metrics are evaluated for groups of 20, 50, and 100 clients, respectively.

We make the following observations: when middle relays are used in tree construction, both the bandwidth consumption in MTor and maximum degree in the multicast tree increase with the number of candidate middle relays. Therefore, it is preferable to select middle relays from a subset rather than all of the possible middle relays. On the other hand, removing middle relays entirely from multicast trees (i.e., each client uses a 2-hop circuit from its guard to the MR) reduces bandwidth consumption at the cost of significantly increasing the maximum degree in the multicast tree, which in turn increases the transmission time needed to deliver a message. This is true since the node with maximum degree in the multicast tree will send most traffic among relays on the tree. This node will thus become the bottleneck on the tree, and the time required to deliver a message will therefore be proportional to the node's degree.

For a group of 100 clients, removing the middle relay reduces bandwidth consumption per client by 14% from 2.1MB to 1.8MB, as compared to the case where 200 candidate middle relays are used in tree construction. However, it increases the (average) maximum degree by 140% from 33.1 to 79.7. Importantly, this increase in maximum degree, and thus the transmission time performance, will be more significant as the number of clients increases. Thus, enabling and selecting middle relays from a small subset of Tor relays helps achieve a reasonable trade-off between bandwidth consumption and transmission time performance.

6 Anonymity Analysis

We assume an adversary that runs relays in the Tor network and uses these relays to observe traffic, correlate flows, and de-anonymize users. We conservatively assume that an adversary is able to perfectly correlate traffic—i.e., if it observes Tor cells belonging to the same flow at two different points in the network, then the adversary can discern with perfect accuracy that those packets do indeed belong to the same Tor circuit.

We provision the adversary with an *observed bandwidth budget* of 131MBps, 327MBps or 656MBps, which it may use to operate one or more relays in the Tor network, such that the combined bandwidth of his relays does not exceed his bandwidth budget. The adversary must fix his selection of relays and is not allowed to change which relays it controls during the course of an experiment. As shown in Table 1, our bandwidth budgets conservatively model adversaries that have up to twice the observed bandwidth of the largest Tor families as of September 30th, 2014. (A Tor *family* consists of relays that report that they are administered by the same entity.) These bandwidth budgets respectively correspond to 1%, 2.5%, and 5% of the total observed bandwidth reported by all relays as of September 30th, 2014.

To carry out a traffic correlation attack in vanilla (unicast) Tor, the adversary needs to control both sides of a circuit (i.e., the guard and exit relay) to observe and later correlate the source and destination of communication. The definition of compromise in a group communication setting is less clear since there are potentially many receivers for a given message. In our anonymity evaluation, we consider two types of compromise due to traffic correlation:

Rank	Bandwidth (MBps)	Largest family member
1	327	bolobolo1
2	207	torpidsUAitlas
3	190	PrivacyRepublic0019
4	189	orion
5	155	AccessNow14

Table 1. Tor *families* with the top observed bandwidth on September 30th, 2014. The total observed bandwidth of all relays is 13 GB/s

- **Linkage.** We say two clients in the same communication group are *linked* if the adversary observes each of their guard traffic simultaneously. Traffic belonging to the same group in the same session may be identified by inspecting the messages' GID.
- **Membership identification.** A client's membership in the group is compromised if the adversary observes the client's guard traffic. By evaluating the probability of membership identification, we find an upper bound on the probability that the client is linked with *any* other client in the group.

We conservatively assume that the adversary's guard relay exhibits enough uptime to obtain the GUARD and STABLE flags. We additionally assume that the adversary's exit relay does not have the GUARD flag but does have an exit policy that allows communication to all addresses and ports; this increases its chance of being selected as an exit. All of the adversary's relays have sufficient bandwidth to obtain the FAST flag.

To measure susceptibility to traffic correlation attacks, we adopt the following security metrics from Johnson et al. [29] since we believe they are the most relevant to users of Tor:

- **Compromise rate:** the probability distribution on the fraction of paths that are compromised (w.r.t. linkage or membership identification correlation attacks) for a given user (in a given period);
- **Time to first compromise:** the probability distribution on the time until the first path compromise (w.r.t. linkage or membership identification correlation attacks).

6.1 Workloads

We envision that most users will continue to use vanilla Tor as their primary means of anonymous communication: that is, they will continue to use unicast communication to browse the web, send emails, etc. Simultaneously, a smaller percentage of Tor users will use MTor to participate in group communication.

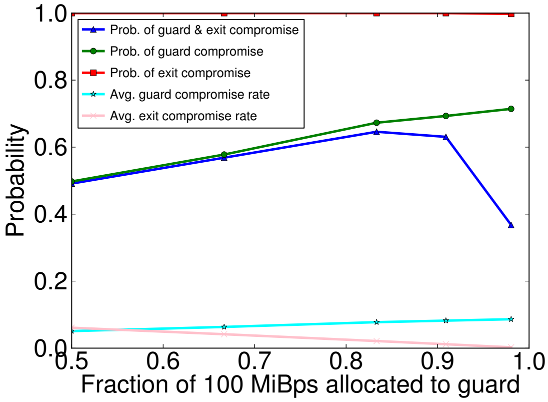


Fig. 8. Probability of observing traffic (y-axis) for various bandwidth allocation strategies between the guard and exit (x-axis), using Tor consensus data from April 2014 through September 2014.

For the unicast Tor users, we adopt the user models introduced by Johnson et al. [29] that are intended to reflect the behavior of actual users of the live Tor network. These user models consist of a sequence of Tor streams and the times at which they occur. Here, streams include DNS resolution requests in addition to TCP connections to specific destinations. Johnson et al. construct these models by using client applications on the live Tor network and tracing the behavior of the local Tor client. We use models consisting of Tor users who use *(i)* Gmail/Google Chat, *(ii)* Google Calendar/Docs, *(iii)* Facebook, and *(iv)* perform web searches.

For MTor clients, we consider a large "webcasting" scenario in which 5000 MTor clients participate in the same group and receive multicasted messages from a single sender. These webcasting sessions last for an hour, after which time the clients all leave the group and join a new group webcasting session with (w.h.p.) a new MR. This process repeats for the duration of the simulation.

6.2 TorPS Results

Attacker configurations. We first determine the bandwidth allocation between guard and exit relays that maximizes the adversary's ability to de-anonymize ordinary unicast Tor users. We tested guard-to-exit bandwidth ratios of 1:1, 2:1, 5:1, 10:1 and 50:1 using the TorPS path simulator. Figure 8 shows the compromise rate with varying bandwidth allocation ratios between guard and exit relays. A 5:1 ratio maximizes the probability of compromising both sides of at least one stream during the simulation period (blue line), which we adopt in the rest of this section. This confirms an earlier result by [29].

Since exit relays are not used by MTor, adversaries who attempt to de-anonymize group communication will spend their entire bandwidth budget in controlling guard relays. Recall that an adversary succeeds in linkage and membership identification correlation attacks by controlling the guard relay(s) used by a group's clients.

Simulation results. For both unicast and MTor clients, we use TorPS to conduct 5000 Monte Carlo simulations of six months of client activity spanning the period from April 2014 to September 2014. We use the output of these simulations to evaluate the compromise rate and time to first compromise for Tor and MTor, for the linkage and membership identification attacks described above.

The adversary's ability to perform membership identification in MTor is depicted in Figure 9. The figure shows the cumulative distribution over the fraction of streams that an adversary is able to compromise (i.e., determine that the client is a member of the group). Our results indicate that an adversary who continually contributes 131MBps of guard bandwidth to the network fails to identify more than 58% of the MTor clients during the simulation's six-month window. For 90% of the clients, the adversary is able to successfully determine group membership for only approximately 12% of the clients' multicast groups. MTor fares worse against even more well-provisioned adversaries, although we note that even against an adversary who would constitute the largest contributor to Tor (the 327MBps case), 70% of clients have fewer than 12% of their streams compromised.

Figure 9 plots the cumulative distribution of the time to first compromise for MTor and Tor. A direct comparison between MTor and Tor is not possible, since the latter uses unicast workloads (web browsing, etc.) while the former is based on group communication. Generally, however, we expect MTor to provide *greater* resistance to linkage attacks than vanilla Tor for most clients: In vanilla Tor, exit relays are chosen independently for each new circuit, while the choice of guard relays persists across circuits.[5] An adversary who controls an exit relay can therefore wait until his relay is chosen. In contrast, MTor avoids the use of exit relays, requiring the adversary to control the guard relays of the two clients it is attempting to directly link. Adversaries who are not sufficiently lucky to operate the guards must wait potentially months before clients select other guards. This trend is observable in Figure 9 for all tested attacker strengths: albeit with different underlying workloads, the adversary is more quickly able to perform linkage correlation attacks against Tor than it is against MTor.

Against our 131MBps adversary, approximately 68% of clients were not identified as being a group member within 100 days. Against an adversary who would constitute the largest contributor to Tor (the 327MBps case), roughly 42% of clients were not identified. These results should be considered conservative measures of MTor's anonymity since, in practice, most users would presumably not continuously participate in a group for such a long duration.

In Figure 9, we observe that the time to first compromise increases roughly linearly with the adversary's provisioned bandwidth budget, for both membership and linkage attacks. This is due to Tor's bandwidth-weighted relay selection policy: clients choose relays proportional to how much bandwidth they contribute to the network, thus increasing the adversary's bandwidth budget by a constant factor also increases the probability that clients will select its relays by roughly the same factor.

7 Discussion and Limitations

Incremental deployment. MTor requires changes to both Tor clients and relays. Importantly, however, since MTor works alongside standard unicast Tor, it does not require that all clients and relays support anonymous multicast communication. A straightforward approach to incrementally deploying MTor involves the introduction of a new Tor flag, MTOR, that is assigned to relays by the Tor directories if those relays support group communication. Once a sufficiently large number of relays advertise the MTOR flag in their descriptors (hence offering diverse options for relay selection), MTor-capable clients can then choose amongst those relays when selecting and building a path to the MR.

Growth of the Tor network. MTor offers bandwidth savings due in part to its de-duplication of messages. If the Tor network expands to include more relays with the STABLE and FAST flags, then the probability that clients using MTor will select the same relays in their paths to the MR will decrease, thus providing fewer opportunities for de-duplication. One possible approach to counter this effect is to adopt the MTOR flag described above, and assign it only to a fixed number of relays such that the opportunities for de-duplication also remain fixed.

Fortunately, our TorPS simulation using recent consensus data from the live Tor network indicates that opportunities for de-duplication *do* exist in current Tor (see Section 5.2). And, independent of de-duplication, MTor offers other bandwidth

[5] The Tor Project is currently investigating how often Tor guard relays should be rotated [11, 13]. In the current version of Tor, a client rotates guards between 30 and 60 days (uniformly chosen).

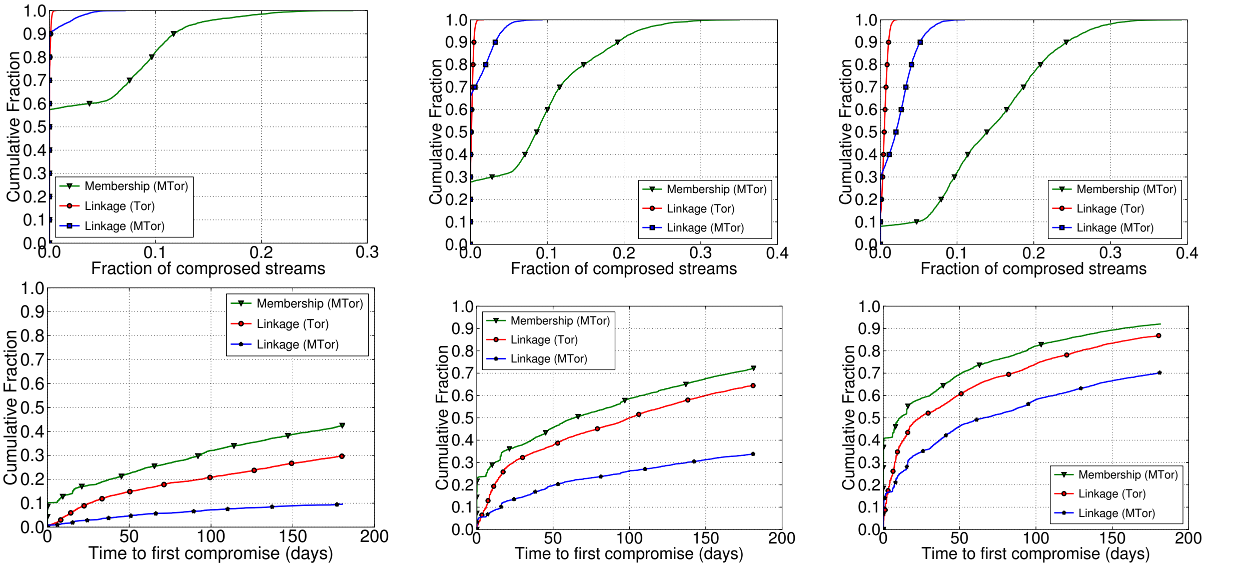


Fig. 9. Fraction of compromised streams *(top)* and time to first compromise *(bottom)* with an adversary budget of *(left)* 131MBps, *(middle)* 327MBps, and *(right)* 656MBps.

savings. Since it handles message distribution *within* the Tor network, MTor (i) eliminates the need to burden exit relays and, more importantly, (ii) reduces network consumption by removing at least two hops between clients in the same group.

Susceptibility to traffic correlation attacks. In MTor, all traffic is sent within the Tor network. MTor does not use exit policies since exit relays are not used, and thus selecting the relay path to the MR is not affected by the group members' choice of application. This is in contrast to standard unicast Tor where the choice of application (or really, the destination port) influences relay selection since a compatible exit relay must be chosen. This has an interesting effect on anonymity: unlike vanilla Tor, MTor's susceptibility to traffic correlation attacks is independent of its users' choice of application.

8 Conclusion

This paper presents the design and implementation of MTor, which to the best of our knowledge is the first system that provides low-latency anonymous group communication with a decentralized trust infrastructure. MTor gracefully scales with the size of the communication group by constructing multicast trees on top of the Tor overlay network, and allows dynamic group composition without relying on global coordination.

We performed comprehensive analyses of MTor's bandwidth, latency, unreliability, and anonymity using recently proposed simulation techniques with realistic models of the Tor topology and historical datasets of Tor relay information. Our results are encouraging: the bandwidth consumption and latency performance scale gracefully as additional clients join the group communication. We show that MTor achieves significant performance improvements that enable new forms of anonymous group communication (e.g., anonymous VoIP) while providing anonymity that is comparable to that provided by vanilla Tor.

Acknowledgments

We are grateful for the helpful comments and suggestions from the anonymous reviewers, and especially from our shepherd, Matthew Wright. We thank Chris Wacek, Henry Tan, Matt Blaze, Jonathan Smith, and Clay Shields for insightful discussions about applying multicast in anonymity networks. This work is partially funded from National Science Foundation grants CNS-0845552, CNS-1117052, CNS-1149832, CNS-1218066, CNS-1527401, and DARPA contract N66001-11-C-4020. Any opinions, findings, and conclusions or recommendations expressed herein are those of the authors and do not necessarily reflect the views of the funding agencies.

References

[1] D. J. Bernstein, N. Duif, T. Lange, P. Schwabe, and B.-Y. Yang. High-speed High-security Signatures. *Journal of Cryptographic Engineering*, 2(2):77–89, 2012.

[2] X. Cai, X. C. Zhang, B. Joshi, and R. Johnson. Touching from a Distance: Website Fingerprinting Attacks and Defenses. In *ACM Conference on Computer and Communications Security (CCS)*, 2012.

[3] M. Castro, P. Druschel, A.-M. Kermarrec, and A. I. T. Rowstron. Scribe: A Large-Scale and Decentralized Application-Level Multicast Infrastructure. 20(8), October 2002.

[4] M. Castro, P. Druschel, A.-M. Kermarrec, A. Nandi, A. I. T. Rowstron, and A. Singh. SplitStream: High-Bandwidth Multicast in Cooperative Environments. In *ACM Symposium on Operating Systems Principles (SOSP)*, pages 298–313, 2003.

[5] D. Chaum. The Dining Cryptographers Problem: Unconditional Sender and Recipient Untraceability. *Journal of Cryptology*, 1(1):65–75, 1988.

[6] Y.-h. Chu, S. G. Rao, and H. Zhang. A Case for End System Multicast. In *ACM SIGMETRICS Performance Evaluation Review*, 2000.

[7] H. Corrigan-Gibbs and B. Ford. Dissent: Accountable Anonymous Group Messaging. In *ACM Conference on Computer and Communications Security (CCS)*, 2010.

[8] H. Corrigan-Gibbs, D. I. Wolinsky, and B. Ford. Proactively Accountable Anonymous Messaging in Verdict. In *USENIX Security Symposium (USENIX)*, 2013.

[9] S. E. Deering and D. R. Cheriton. Multicast Routing in Datagram Internetworks and Extended LANs. *ACM Transactions on Computer Systems (TOCS)*, 8(2):85–110, 1990.

[10] C. Díaz, S. Seys, J. Claessens, and B. Preneel. Towards Measuring Anonymity. In *Privacy Enhancing Technologies (PET)*, 2003.

[11] R. Dingledine. Research Problem: Better Guard Rotation Parameters, August 2011. Available at https://blog.torproject.org/blog/research-problem-better-guard-rotation-parameters.

[12] R. Dingledine, N. Mathewson, and P. Syverson. Tor: The Second-Generation Onion Router. In *USENIX Security Symposium (USENIX)*, August 2004.

[13] R. Dingledine, N. Hopper, G. Kadianakis, and N. Mathewson. One Fast Guard for Life (or 9 months). In *Privacy Enhancing Technologies Symposium (PETS)*, 2014.

[14] N. S. Evans, R. Dingledine, and C. Grothoff. A Practical Congestion Attack on Tor using Long Paths. In *USENIX Security Symposium (USENIX)*, 2009.

[15] P. Francis. Yoid: Extending the Internet Multicast Architecture, 2000. Unpublished manuscript, available at https://mpi-sws.org/~francis/yoidArch.pdf.

[16] J. Geddes, R. Jansen, and N. Hopper. IMUX: Managing Tor Connections from Two to Infinity, and Beyond. In *Workshop on Privacy in the Electronic Society (WPES)*, 2014.

[17] Global IP Solutions. The Internet Low Bitrate Codec (ILBC). http://tools.ietf.org/html/rfc3951.

[18] S. Goel, M. Robson, M. Polte, and E. Sirer. Herbivore: A Scalable and Efficient Protocol for Anonymous Communication. Technical report, Cornell University, 2003.

[19] S. Hahn and K. Loesing. Privacy-preserving Ways to Estimate the Number of Tor Users. Technical Report 2010-11-001, Tor Project, November 2010.

[20] A. Hamel, J.-C. Grégoire, and I. Goldberg. The Misentropists: New Approaches to Measures in Tor. Technical Report 2011-18, Cheriton School of Computer Science, University of Waterloo, 2011.

[21] D. Herrmann, R. Wendolsky, and H. Federrath. Website Fingerprinting: Attacking Popular Privacy Enhancing Technologies with the Multinomial Naive-bayes Classifier. In *ACM Workshop on Cloud Computing Security (CCSW)*, 2009.

[22] N. Hopper. Proving Security of Tor's Hidden Service Identity Blinding Protocol. Technical Report 2013-12-001, Tor Project, 2013.

[23] J. Jannotti, D. K. Gifford, K. L. Johnson, M. F. Kaashoek, and J. W. O'Toole, Jr. Overcast: Reliable Multicasting with an Overlay Network. In *Symposium on Operating System Design & Implementation (OSDI)*, 2000.

[24] R. Jansen and N. Hopper. Shadow: Running Tor in a Box for Accurate and Efficient Experimentation. In *Network and Distributed System Security Symposium (NDSS)*, 2012.

[25] R. Jansen, K. S. Bauer, N. Hopper, and R. Dingledine. Methodically Modeling the Tor Network. In *CSET*, 2012.

[26] R. Jansen, A. Johnson, and P. F. Syverson. LIRA: Lightweight Incentivized Routing for Anonymity. In *Network and Distributed System Security Symposium (NDSS)*, 2013.

[27] R. Jansen, J. Geddes, C. Wacek, M. Sherr, and P. Syverson. Never Been KIST: Tor's Congestion Management Blossoms with Kernel-Informed Socket Transport. In *USENIX Security Symposium (USENIX)*, August 2014.

[28] R. Jansen, F. Tschorsch, A. Johnson, and B. Scheuermann. The Sniper Attack: Anonymously Deanonymizing and Disabling the Tor Network. In *Network and Distributed System Security Symposium (NDSS)*, 2014.

[29] A. Johnson, C. Wacek, R. Jansen, M. Sherr, and P. Syverson. Users Get Routed: Traffic Correlation on Tor By Realistic Adversaries. In *ACM Conference on Computer and Communications Security (CCS)*, November 2013.

[30] P. Lewis and D. Rushe. http://www.theguardian.com/world/2014/oct/16/-sp-revealed-whisper-app-tracking-users.

[31] H. Liu, E. Y. Vasserman, and N. Hopper. Improved Group Off-the-record Messaging. In *ACM Workshop on Privacy in the Electronic Society (WPES)*, 2013.

[32] N. Mathewson. Next-Generation Hidden Service in Tor. Draft 224, Tor Project, 2013. https://gitweb.torproject.org/

torspec.git/plain/proposals/224-rend-spec-ng.txt.

[33] N. Mathewson and R. Dingledine. Tor Rendezvous Specification, 2014. Available at https://gitweb.torproject.org/torspec.git/log/rend-spec.txt?ofs=50.

[34] D. McCoy, K. Bauer, D. Grunwald, T. Kohno, and D. Sicker. Shining Light in Dark Places: Understanding the Tor Network. In *Privacy Enhancing Technologies Symposium (PETS)*, 2008.

[35] S. J. Murdoch and G. Danezis. Low-Cost Traffic Analysis of Tor. In *IEEE Symposium on Security and Privacy (Oakland)*, 2005.

[36] S. J. Murdoch and R. N. M. Watson. Metrics for Security and Performance in Low-Latency Anonymity Systems. In *Privacy Enhancing Technologies Symposium (PETS)*, 2008.

[37] A. Panchenko, L. Niessen, A. Zinnen, and T. Engel. Website Fingerprinting in Onion Routing Based Anonymization Networks. In *ACM Workshop on Privacy in the Electronic Society (WPES)*, 2011.

[38] G. Perng, M. K. Reiter, and C. Wang. M2: Multicasting Mixes for Efficient and Anonymous Communication. In *International Conference on Distributed Computing Systems (ICDCS)*, 2006.

[39] M. Perry. Torflow: Tor network analysis. *Proc. 2nd HotPETs*, pages 1–14, 2009.

[40] M. Perry. A Critique of Website Traffic Fingerprinting Attacks, 2014. Available at https://blog.torproject.org/blog/critique-website-traffic-fingerprinting-attacks.

[41] M. K. Reiter and A. D. Rubin. Crowds: Anonymity for Web Transactions. *ACM Transactions on Information and System Security*, 1(1):66–92, 1998.

[42] Rooms. http://www.rooms.me.

[43] Secret. https://www.secret.ly.

[44] A. Serjantov and G. Danezis. Towards an Information Theoretic Metric for Anonymity. In *Privacy Enhancing Technologies (PET)*, 2003.

[45] P. Syverson. Why I'm not an Entropist. In *Security Protocols Workshop*, 2009.

[46] Tor Project, Inc. Tor Metrics Portal. https://metrics.torproject.org/.

[47] Tor Project, Inc. Tor FAQ: What Attacks Remain Against Onion Routing, 2014. Available at https://www.torproject.org/docs/faq.html.en#AttacksOnOnionRouting.

[48] Whisper. http://www.whisper.sh.

[49] Yik Yak. http://www.yikyakapp.com.