A Survey on Routing in Anonymous Communication Protocols

FATEMEH SHIRAZI, KU Leuven, ESAT/COSIC and imec
MILIVOJ SIMEONOVSKI, CISPA, Saarland University, Saarland Informatics Campus
MUHAMMAD RIZWAN ASGHAR, Cyber Security Foundry, The University of Auckland
MICHAEL BACKES, CISPA Helmholtz Center i.G., Saarland Informatics Campus
CLAUDIA DIAZ, KU Leuven, ESAT/COSIC and imec

The Internet has undergone dramatic changes in the past 2 decades and now forms a global communication platform that billions of users rely on for their daily activities. While this transformation has brought tremendous benefits to society, it has also created new threats to online privacy, such as omnipotent governmental surveillance. As a result, public interest in systems for anonymous communication has drastically increased. In this work, we survey previous research on designing, developing, and deploying systems for anonymous communication. Our taxonomy and comparative assessment provide important insights about the differences between the existing classes of anonymous communication protocols.

CCS Concepts: • **Security and privacy → Pseudonymity, anonymity and untraceability**;

Additional Key Words and Phrases: Anonymous communication, routing protocols, communication networks

ACM Reference format:
Fatemeh Shirazi, Milivoj Simeonovski, Muhammad Rizwan Asghar, Michael Backes, and Claudia Diaz. 2018. A Survey on Routing in Anonymous Communication Protocols. *ACM Comput. Surv.* 51, 3, Article 51 (June 2018), 39 pages.
https://doi.org/10.1145/3182658

1 INTRODUCTION

The Internet has evolved from a mere communication network used by millions of users to a global platform for social networking, communication, education, entertainment, trade, and political activism used by billions of users. In addition to the indisputable societal benefits of this transformation, the mass reach of the Internet has created new powerful threats to online privacy.

This work was supported by the Research Council KU Leuven C16/15/058 and through KU Leuven BOF OT/13/070. In addition, it was supported by the European Commission project H2020-DS-2014-653497 PANORAMIX and by Microsoft Research through its Ph.D. Scholarship Programme. Moreover, this work was supported by the German Federal Ministry of Education and Research (BMBF) through funding for the Center for IT-Security, Privacy and Accountability (CISPA) (FKZ: 16KIS0345) and by the European Research Council Synergy Grant imPACT (n. 610150).

Authors' addresses: F. Shirazi and C. Diaz, KU Leuven ESAT/COSIC, Kasteelpark Arenberg 10 - Bus 2452, B-3001 Leuven-Heverlee, Belgium; emails: {fatemeh.shirazi, claudia.diaz}@esat.kuleuven.be; M. Simeonovski and M. Backes, CISPA, Campus E 9-1, 66123 Saarbruecken, Germany; emails: {milivoj.simeonovski, backes}@cispa.saarland; M. R. Asghar, Building 303S - Room 585, Department of Computer Science, The University of Auckland, Private Bag 92019, Auckland 1142, New Zealand; email: r.asghar@auckland.ac.nz.

The widespread dissemination of personal information that we witness today in social media platforms and applications is certainly a source of concern. The disclosure of potentially sensitive data, however, not only happens when people deliberately post content online but also becomes possible inadvertently by merely engaging in any sort of online activities. This inadvertent data disclosure is particularly worrisome because non-expert end-users cannot be expected to understand the dimensions of the collection taking place and its corresponding privacy implications.

Widely deployed communication protocols only protect, if at all, the content of conversations, but do not conceal from network observers who is communicating with whom, when, from where, and for how long. Network eavesdroppers can silently monitor users' online behavior and build up comprehensive profiles based on the aggregation of user communications' metadata. Today, users are constantly tracked, monitored, and profiled, both with the intent of monetizing their personal information through targeted advertisements and by nearly omnipotent governmental agencies that rely on the mass collection of metadata for conducting dragnet surveillance at a planetary scale.

Anonymous Communication (AC) systems have been proposed as a technical countermeasure to conceal from network observers who is communicating with whom, when, from where, and for how long, mitigating the threats of communications surveillance. The concept of AC systems was introduced by Chaum (1981) in 1981, with his proposal for implementing an anonymous email service that aimed at concealing who sent emails to whom. The further development of this concept in the past 2 decades has seen it applied to a variety of problems and scenarios, such as anonymous voting (Sako and Kilian 1995; Jakobsson et al. 2002), Private Information Retrieval (PIR) (Dingledine et al. 2000), censorship-resistance (Waldman et al. 2000; Waldman and Mazières 2001), anonymous web browsing (Goldschlag et al. 1996), hidden web services (Dingledine et al. 2004), and many others.

Public interest in AC systems has strikingly increased in the past few years. This could be explained as a response to recently revealed dragnet surveillance programs, the fact that deployed AC networks seem to become (according to leaked documents[1]) a major hurdle for communications surveillance, and to somewhat increased public awareness on the threats to privacy posed by modern information and communication technologies.

The literature offers a broad variety of proposals for anonymity network designs. Several of these designs have been implemented, and some are successfully deployed in the wild. Of the deployed systems, the most successful example to date is the Tor network, which is used daily by about two million users (The Tor Project 2017).

Existing designs take a variety of approaches to anonymous routing for implementing the AC network. Routing determines how data is sent through the network, and it as such is the core element of the AC design, determining to a large extent both security and performance of the system. State-of-the-art approaches rely on different threat models and sets of assumptions, and they provide different guarantees to their users. Even though survey articles on AC systems exist (Erdin et al. 2015; Sampigethaya and Poovendran 2006; Conrad and Shirazi 2014; AlSabah and Goldberg 2015; Ren and Wu 2010; Edman and Yener 2009; Danezis and Díaz 2008; Serjantov 2004; Raymond 2000), we still lack a systematic understanding, classification, and comparison of the routing characteristics of the plurality of existing AC approaches.

The purpose of this survey is to provide a detailed overview of the routing characteristics of current AC systems and to discuss how their routing features impact anonymity against different types of adversaries, as well as overall performance. To this end, we first *identify the routing characteristics* that are relevant for AC protocols and provide a *taxonomy* for clustering the systems

[1] https://wikileaks.org/.

with respect to their routing characteristics, deployability, and performance. Then, we apply the taxonomy to the extensive literature on AC systems, in particular including Mixnets, DCnets, Tor-related systems, and Random Walk/Distributed Hash Table (DHT)-based protocols. To select AC protocols for our examination we chose systems that have presentation value in terms of routing. Most of the reviewed protocols are systems designed as overlay networks. We excluded next-generation Internet AC solutions such as Sankey and Wright (2014) and Hsiao et al. (2012). Finally, we discuss the relationship between different routing decisions, and how they affect performance and scalability.

Outline. Section 2 provides our taxonomy for anonymous routing and describes the various routing features and dimensions that we are considering for our evaluation and discusses the relationship between these routing features. Section 3 gives a compact tabular overview describing the classification of existing systems in our taxonomy and reviews existing AC systems with respect to their routing characteristics. Section 4 compares the four main categories of AC protocols in terms of anonymity goals against different types of adversaries, scalability, and their applications. Section 5 concludes the paper.

2 ANONYMOUS ROUTING PROTOCOL CHARACTERISTICS

In this section, we introduce the routing characteristics, deployability, and performance metrics considered in our taxonomy, and we discuss the relationship between these characteristics.

2.1 Routing Characteristics

Generally, routing in a communication network refers to the selection of nodes for relaying communication through the network. Routing schemes, however, require some essential design components. For anonymous communication, we consider four building blocks that are relevant to routing in AC networks. These building blocks are node management, transfer/retrieval of node information to/by the routing decision maker, path selection, and forwarding or relaying; where path selection is the main design component of routing schemes for AC protocols.

Several taxonomies and classifications for routing protocols have been proposed in the literature (Bell and Jabbour 1986; Feeney 1999; Zou et al. 2002). However, AC networks aim to conceal the metadata of communications and thus have security requirements that make them fundamentally different from other networks.

In this section, we present a classification for anonymous routing protocols (see Table 1). Our classification (see Tables 2 and 5) is an adaptation of Feeney's taxonomy (Feeney 1999), which classifies the routing characteristics of mobile ad hoc networks into four categories:

(1) *Communication model* describes whether the communication is based on a single- or multi-channel.
(2) *Structure* describes whether or not nodes are treated equally.
(3) *State information* describes where the topology information is maintained.
(4) *Scheduling* describes whether the information about routes is maintained at the source or is instead computed on-demand.

This taxonomy does not address several relevant design features of AC networks, such as probabilistic node selection for constructing circuits and security considerations for protecting routing information from different network adversaries. In addition, not all the characteristics identified by Feeney are relevant to AC routing. For example, the distinction between single- and multi-channel features is not relevant to overlay networks, which constitutes a standard design choice for many AC networks.

Table 1. Overview of the Protocol Routing Characteristics

Feature Name			Description	Instantiation and Symbols
Network Structure	Network topology		Degree of node connectivity in the network	☒ (fully) ☐ (mostly) ⊏ (partially)
	Connection type	Direction	Data flow in connections	→ (unidirectional) ↔ (bidirectional)
		Synchronization	Timing model for connection establishment and data sending	≠ (asynchronous) ≅ (synchronous)
	Symmetry	Roles	Users operating as relays	●··●··● (peer-to-peer) ●··● (client-server) ●··⊙··● (hybrid)
		Topology	Node topology for routing	··· (flat) ✤ (hierarchical)
		Decentralization	Degree of decentralization for non-routing services	⊙ (semi decentralized) ○ (fully decentralized)
Routing Info	Network view		Network view necessary for making routing decisions	● (complete) ◐ (partial)
	Updating		Triggers for routing information updates	🕒 (periodic) ↯ (event-based)
Communication Model	Routing type		Node selection per route	●··· (source-routed) ··●·· (hop-by-hop) 📢 (broadcasted)
	Scheduling		Prioritization of traffic	≡ (fair) ⟐ (prioritized)
	Node selection	Determinism	Determinism of node selection	✓ (deterministic) ✗ (non-deterministic)
		Selection set	Permissible set of nodes per route	Ⓐ (all) 🛑 (restricted, security) 🌍 (restricted, network) ☺ (user-based)
		Selection probability	Node selection probability per route	⊛ (uniform) ◎ (weighted, static) ❋ (weighted, dynamic)
Performance, Deployability	Latency		Protocol latency	L (low-latency) H (high-latency) M (mid-latency)
	Communication mode		Longevity of connections	●–● (connection-based) 🖂 (message-based)
	Implementation		Implemented	✓ (yes) ✗ (no)
	Code availability		Open source	✓ (yes) ✗ (no)
	Context/application		The context in which the protocol is used	@ (email) ☎ (phone comm.) ℯ (web) ✉ (messaging) 📄 (file-sharing) ✎ (micro-blogging) 📶 (wireless ad hoc)

We redefine Feeney's criteria to account for design choices that are relevant to anonymous routing protocols. Nevertheless, we distinguish three groups of features inspired by Feeney's categories: *network structure*, *routing information*, and *communication model*:

(1) *Network structure* describes the characteristics of the anonymous relays, the connections between them, and the underlying network topology.
(2) *Routing information* describes the network information available to entities deciding on the route of an anonymous connection.
(3) *Communication model* defines the entities that make the routing decisions and describes how these decisions are made.

In what follows, we describe these features in more detail, including their various sub-features and corresponding notation symbols used to denote individual feature instantiations. We refer to Table 1 for a general overview of the resulting taxonomy.

2.1.1 Network Structure. We consider first the network features that are relevant to anonymous routing. These are, specifically, features related to: (a) the ***topology*** of the network, which describes how nodes are connected; (b) the ***connection type***, describing the characteristics of the connections between nodes; and (c) ***symmetry***, describing whether the entities participating in the network are all similar, or if they can take on different roles and responsibilities for routing data through the network.

(a) **Topology.** The topology describes the arrangement of various elements of the network, such as routers and communication links between those routers. We only take the logical topology of the network into account, which determines how data flows within it. We note that physical topology characteristics, such as the geographical location of computers, sometimes matters in anonymous routing decisions, for example, when considering adversaries who control an Autonomous System (AS) (Feamster and Dingledine 2004; Edman and Syverson 2009).

We consider the network as a graph in which the routers are represented by graph nodes. An edge between two nodes exists if the routing strategy allows both nodes to be directly connected as part of the same anonymous circuit.

The connectivity of nodes varies widely across AC network designs, and the advantages and disadvantages of high and low levels of connectivity have been the subject of debate for over a decade (Böhme et al. 2005).

Restricted routing proposals (Danezis 2003a) have shown that for applications that are latency-tolerant, partially connected networks with certain topological characteristics (e.g., based on expander graphs) provide optimal anonymity and latency trade-offs and mitigate certain attacks. These results further emphasize the impact of network connectivity features for anonymous routing.

We classify anonymity networks into three categories according to their connectivity: *fully connected*, *mostly connected*, and *partially connected* networks.

- We consider a network to be *fully connected* (⊠)[2] when nodes can potentially connect to most (or all) other nodes. Note that our rule of thumb is that a node on average should be able to connect to at least 95% of the other nodes.
- We call a network *mostly connected* (□) if its nodes can potentially connect to at least half of the nodes.
- Finally, in *partially connected* (⊏) networks, nodes only connect to a relatively small subset of the whole network.

Higher connectivity in the network topology leads to better resilience (availability) against node failure, such as Denial of Service (DoS) attacks; such resilience might have, in turn, a positive influence on anonymity (Böhme et al. 2005). While having a fully connected topology is better than having a very restricted network topology, such as a fixed sequence of relays, called *cascade*, Diaz et al. have shown a partially connected network structure—in particular, a stratified topology—can provide better anonymity than a fully connected network structure (Diaz et al. 2010). However, eliminating connections that might induce security problems, such as the connection between two nodes from the same IP family that may be easier to control by an adversary, can be beneficial to anonymity. The same holds for eliminating connections that would induce higher latency, which would, in turn, improve the performance of the system.

(b) **Connection Type.** Here, we consider the *direction* and *synchronization* of connections. As far as the direction is concerned, we consider the following options:

- A connection is *unidirectional* (→) if the data flow between two entities can only be in one direction.
- A connection between two entities is *bidirectional* (↔) if data can flow in both directions and the same connection is used for sending back the response to a received message.

Typically, interactive applications, such as web browsing, require bidirectional channels, while non-interactive applications, such as email, can just close the connection as soon as the message has been forwarded.

Bidirectional circuits have the advantage that they induce less overhead in terms of circuit construction. Unidirectional connections have the advantage that they are less vulnerable to timing attacks, as a malicious node can only observe data flowing in one direction, which is less informative than bidirectional connections in which patterns of requests

[2]In parenthesis, we define the symbol or the keyword that is used in the comparative Tables 2, 3, 4, and 5 to indicate the corresponding characteristic.

and response are visible to all nodes in the path. However, note that in a unidirectional connection, a larger number of nodes are going to be involved in relaying the communication between a sender and a receiver.

Further, we consider whether the anonymity system involves connection *synchronization*:

- A connection is *asynchronous* ($\neq$) if the establishment of connections and relaying of messages is initiated by a user without any timing coordination with other participants.
- Connections are *synchronous* ($\cong$) if they begin and end at specific timings and messages are also relayed at specific moments in time, based on some timing coordination between network entities.

Asynchronous systems are conceptually simpler as they impose fewer constraints on the activity of network participants. However, the distinct timing of actions leaks information valuable to perform traffic analysis and, for example, reveals long-term communication patterns (Danezis 2003b) or perform end-to-end correlation attacks (Levine et al. 2004; Bauer et al. 2007; Zhu et al. 2010).

Synchronous systems are often more difficult to engineer and come with a performance or usability penalty; moreover, secure and reliable time becomes an additional dependency of the system, and a possible point of failure or vulnerability to attack. However, synchronization constitutes a very powerful design feature to offer robust anonymity guarantees in the presence of powerful adversaries, because it disables trivial end-to-end correlation attacks based on start and end times of connections (Murdoch and Zielinski 2007), and other timing data that synchronization makes less granular, enabling the aggregation of participants, connections, and events in *anonymity sets*. Synchronous anonymity systems were proposed in the early 1990s by Pfitzmann et al. to anonymize ISDN telephony calls (Pfitzmann et al. 1991). These proposals were both feasible from an engineering perspective (compatible with the network requirements and introducing a low-efficiency cost) and clearly spelled-out anonymity guarantees as well as full unobservability for local calls.

(c) **Symmetry.** We consider symmetry in the roles of the network entities. An anonymity system is intuitively "more symmetric" when all the participating entities have similar roles and responsibilities and "less symmetric" if there are different roles, capabilities, and trust assumptions among the entities that participate in the routing.

We thus first examine the overlap between the *roles* of end-users who initiate communications and relaying nodes. We distinguish three types of systems.

- We classify a system as *peer-to-peer* (●· · ·●· · ·●) when end-users are expected (often even obliged) to operate as relaying nodes in order to use the AC network.
- At the other end of the spectrum, in *client-server* (●· · · ●) systems, users are not expected (often even forbidden) to operate as relaying nodes in order to use the system.
- We call a system *hybrid* (●· · · ○ · · · ●) if it combines characteristics of both *peer-to-peer* and *client-server* systems, i.e., end-users may or may not operate as relaying nodes.

These different levels of symmetry come with advantages and disadvantages (Böhme et al. 2005). Peer-to-peer systems can better scale as the number of users grows, because new users also increase the capacity of the network. Further, peer-to-peer networks are more resilient to node failures and have better availability properties. In client-server architectures, however, it is possible to run nodes more reliably and securely (as nodes are not necessarily run by laymen end-users), which in particular helps to handle liability issues with respect to complaints. Having end-users run just client software has a lower cost for end-users in terms of resources and offers opportunities for simpler, and thus often more usable, client software.

Second, we distinguish whether nodes are organized in a flat or a hierarchical structure with respect to the routing. We call the resulting feature the *topology*:

- A network has a *flat* ($\cdots$) structure if every node has the same importance and rank when making routing decisions.
- A network has a *hierarchical* (✤) structure if nodes have different capabilities and priorities toward the routing algorithm.

Hierarchical structures are often introduced to improve efficiency and performance. However, a non-flat hierarchy can make the network less resilient to attacks, as the failure of a node that is placed high in the hierarchy has a severe impact on the performance of the network.

The third and last dimension of symmetry addresses the degree of *decentralization* of network services other than (but auxiliary to) the routing itself. Note that we are not considering *centralized* models, because they are a single point of failure for surveillance and insecure by design.

- A network is *semi decentralized* ($\odot$) if it includes one or a small number of entities performing a service critical to routing (e.g., compiling and distributing network directory information). This accounts for the fact that especially high levels of trust are placed on these entities, which constitute more of a point of failure than a simple relay.
- A network is *fully decentralized* ($\bigcirc$) if the system design does not include entities that have to be especially trusted for the provision of functionalities that enable the routing. Fully decentralized systems have a better distribution of trust.

2.1.2 Routing Information. We now consider the information available to the entity (or entities) that decides on the route of a connection and how that information is made available.

(a) **Network View**. This determines the completeness of information available to establish a route.
- The routing decision-maker has a *complete view* (●) of the system if routing information about all nodes is available.
- The decision maker has a *partial view* (◐) of the system if the available routing information only cover a subset of the nodes that form the AC network.

A complete view allows the decision maker to choose among the full set of nodes. However, a partial view improves the scalability of the network, as the distribution of routing information for the full network may consume significant bandwidth and network resources. There are also some attacks that become possible when the routing decision makers only have a partial view of the network. For example, route fingerprinting attacks (Danezis and Clayton 2006; Danezis and Syverson 2008) are possible if each user knows different subsets of routers. In these attacks, the initiator of a connection can be identified by the nodes that make up the route, since typically a very small number of users will know a certain combination of network nodes.

(b) **Updating**. This determines how frequently routing information is updated.
- Routing information is updated *periodically* (🕒) if it is updated at predefined time intervals.
- Routing information is updated *event-based* (ϟ) if the updates are triggered by events in the network other than timeouts.
- No updating mechanism is in place (✗).

2.1.3 Communication Model. We finally consider features that describe the creation of anonymous routes.

(a) **Routing Type.** This refers to the selection of nodes to determine a route.
- The routing decision is *source-routed* (●· · ·) if the initiator of the communication selects the set of nodes that will form the anonymous route.
- The routing decision is *hop-by-hop* (· · ·● · ·) (also called "random routing") if the initiator only selects the first relay node, which in turn picks the second, and so on, until the message reaches its final destination.
- We refer to the routing type *broadcasting* (📢) when there is no routing decision made explicitly, but rather the data is only forwarded to multiple or all nodes. Flooding refers to broadcasting over more than one hop and when the data is broadcasted without any restriction; gossiping (Haas et al. 2006) refers to forwarding the data only to a subset of nodes, which is technically not a broadcast but multicast.

Source-routing enables the initiator to pick nodes it trusts, and prevents adversaries from biasing the node selection towards compromised nodes. A variation of the basic source-routed model is found in some systems that provide receiver anonymity. In these systems, the initiator and the receiver select, respectively, the first and second halves of the route, which are joined in the middle at a rendezvous point. An advantage of hop-by-hop routing is that even if the initiator only knows a subset of nodes, her connections might be routed throughout the whole network, mitigating route fingerprinting attacks (Danezis and Clayton 2006). In literature, other node selection strategies have been proposed, which we have not taken into consideration such as dynamic routing schemes using distance vector routing (i.e., Perkins and Royer (1997)) and link-state routing (i.e., Moy (1998)). In fact, such algorithms are often disregarded for AC networks because of the predictability they offer, which is in conflict with anonymity.

(b) **Scheduling.** This refers to the way a node serves incoming scheduling requests.
- *Fair* (≡) scheduling means that all types of connection are treated same.
- *Prioritized* (◇) scheduling means that certain connections are given priority over others.

Prioritized scheduling can improve performance and reduce congestion. However, differential treatment of traffic may undermine anonymity as the traffic of different priorities would be distinguishable and thus not conform a single (larger) anonymity set. An example of prioritized scheduling is when the scheduling follows an economic model, which might mitigate flooding attacks (Grothoff 2003).

(c) **Node Selection.** This refers to the protocol features that determine which nodes are selected to be part of an anonymous route. The number of nodes that are selected to form the anonymous connection can either be fixed (deterministically) or be computed probabilistically according to some distribution.
- Node selection can either be *deterministic* (✔) or non-deterministic (probabilistic) (✗). To characterize node selection, we consider the *selection set* that determines which nodes are eligible for being on the route and the *selection (probability) distribution* that describes the likelihood of each of the nodes in the selection set being chosen for a route.
- The selection set may contain *all nodes* (Ⓐ) of the network.
- It may contain a *security-restricted subset* (🛑) of all network nodes, i.e., a subset that is selected according to some *security-restrictions*, for example, establishing that all the nodes in a route must be in different /16 IP subnets.
- It may contain a *network-restricted subset* (🌍) of all network nodes, e.g., a subset aimed at guaranteeing the quality of the communication by, for example, avoiding congested links and nodes.

- And finally, the selection set may be user-specific, considering *user preferences and trust assumptions* (☺).

We are left to define the selection probability with which individual nodes are chosen.

- The probability distribution that describes how nodes are selected may be *uniform* (⊛).
- The probability distribution is *statically weighted*, i.e., weighted based on *general, static parameters* (◎), for example, the bandwidth of the nodes.
- The probability distribution is *dynamically weighted* based on *state-specific dependencies* (❋), for example, the nodes' response time.

Even for general parameters, the weighted selection often requires frequent updates so they reflect the current state of the network. In other words, we consider parameters that are calculated in real-time to be *dynamic* biases, and parameters based on routing information that is unchanged until the next periodic update to be *static*. The uniform selection typically offers better anonymity levels, while the weighted selection often improves performance.

2.2 Performance and Deployability

In addition to the routing characteristics identified before, we identify the following list of metrics that can be used to evaluate performance and deployability characteristics of AC protocols.

(1) **Latency.** In the literature, AC protocols are usually classified into three performance categories:
   - Protocols with *low-latency* (L) incorporate no latency to the communication and typically support applications that require real-time communication (e.g., web browsing).
   - Protocols with *high-latency* (H) do not require real-time communications and support applications that can tolerate a certain delay between requests and responses (e.g., email communication).
   - Protocols with *mid-latency* (M) introduce a random delay and may induce a restricted latency; hence, these protocols support applications that can tolerate a restricted delay between requests and responses (e.g., file-sharing).

(2) **Communication Mode.** We distinguish two kinds of communication modes, depending on the longevity of individual connections.
   - We classify protocols as *connection-based* (•–•) if routes between senders and receivers are maintained for a certain amount of time and used for exchanging multiple data transfers.
   - If routes are created just to send a message and no state is maintained for further exchanges, then we classify a protocol as *message-based* (⊠).

(3) **Implementation and Code Availability.** This indicates whether or not a prototype of the protocol has been implemented and if the code is publicly available. In both cases, the answer is either yes (✔) or no (✗).

(4) **Context/application.** We specify the context/application in which the protocols are designed to be used. We identify several basic context/applications: namely, protocols for anonymous messaging (✉), email communication (@); protocols for real-time communication such as telephony (☎); web communication, such as anonymous browsing (ⓔ) that needs to be low-latency and microblogging (✎) that can tolerate more latency; bulletin boards, auctions, voting, group messaging (👥); file-sharing (📄); and protocol that are used in the context of a wireless ad hoc networks (📶). If the protocols do not specify explicitly the context in which they are used, then we assign context/application to them mainly based on the latency and the number of intended recipients of the

communication. However, this does not rule out that AC protocols could be used for other contexts/applications as well (in some cases with minor changes). For example, AC systems that are used for e often also can also be used for @ and are categorized as e in most cases, because they provide low-latency anonymous communication that is not necessary for email applications.

2.3 Correlation Among Routing Features

Next, we address some direct and indirect correlations (i.e., dependencies and conflicts) among the routing features. We have defined the network topology based only on the connectivity of the relaying routers (see Section 2.1), where users and administrative entities are not taken into account. Therefore, according to our definition in Section 2.1, there is no correlation between topology and the roles feature, and neither between topology and the decentralization feature.

There is an evident correlation between hierarchy and network topology of AC networks. A hierarchical AC network does not have a fully connected network structure. Moreover, the network view of the routing decision maker can have an influence on the topology of the AC network. Generally speaking, a partial network view might lead to a partially connected network topology for the AC network, because the routing decision maker might have difficulties accessing routing information of certain nodes. For example, if the topology is partially connected, it might be that the routing decision maker has a partial view.

Another evident correlation exists between the network topology and the selection set. Restrictions in the selection set lead to reduced connectivity of the network topology.

Although the synchronization feature is not directly correlated to scheduling, depending on the forwarding strategy of the of relays, there can be a correlation. There is also an evident correlation between scheduling and latency, because, in a prioritized scheduling algorithm, some traffic is delayed.

AC networks with a hierarchical structure have a partially connected network structure. By definition, hierarchical organization of nodes restricts the selection set. Moreover, a partial network view limits the selection set, because the routing decision maker can only select nodes within its view.

Finally, there is an apparent correlation between latency and communication mode. High latency AC networks usually use a message-based communication mode and vice versa. Due to the temporal nature of the message-based communication, where connections are not going to be used further (e.g., replies are not going to be sent in a short time), setting up a circuit is unnecessary.

3 ROUTING CLASSIFICATION OF AC PROTOCOLS

In this section, we present a categorization of AC protocols. We have categorized AC protocols based on their routing type: being source-routed, hop-by-hop routed, or broadcast. Moreover, we classified the protocols into five main families, namely: (1) Mixnet protocols and (2) Tor-related protocols as source-routed protocols, (3) Random Walk and DHT-based protocols as hop-by-hop routed, and (4) DCnet protocols as the broadcast routing type. Finally, (5) Miscellaneous include a few protocols that do not fit into the aforementioned categories. A few protocols are presented in the most representative category, albeit they technically borrow some minor techniques from other categories as well. We summarize our classification of the routing aspects in four comparative tables (namely, Table 2, Table 3, Table 4, and Table 5).

Next, we discuss the AC protocols individually, starting with Mixnet protocols (from Section 3.1 to Section 3.1.3), and then we proceeding with Tor-related protocols (Section 3.2), Random Walk

Table 2. Routing Classification of Anonymous Communication Protocols: Mixnet Protocols

	Network Structure						Routing Information		Communication Model					Performance and Deployability				
		Connection Type		Symmetry							Node Selection							
	Topology	Direction	Synchronization	Roles	Hierarchy	Decentralization	Network view	Updating	Routing type	Scheduling	Determinism	Selection set	Selection probability	Latency	Communication mode	Implementation	Code availability	Context/application
Chaum's Mix Cascades (Chaum 1981)	⊠	→	≠	●··●	···	✗	●	✗	●···	≡	✓	Ⓐ	◎	H	🖂	✗	✗	@
ISDN and Real-time (Pfitzmann et al. 1991) (Jerichow et al. 1998)	⊠	↔	≅	●··●	···	⊙	◐	✗	●···	≡	✓	Ⓐ	◎	L	●–●	✗	✗	☎e
Babel (Gülcü and Tsudik 1996)	⊠	→	≠	●··●	···	⊙	●	✗	●··· ··●··	≡	✗	Ⓐ	⊛	H	🖂	✗	✗	@
Stop-and-Go-MIXes (Kesdogan et al. 1998)	⊠	→	≅	●··●	···	⊙	●	◷	●···	≡	✗	Ⓐ	⊛	H	🖂	✗	✗	@
Webmixes (Berthold et al. 2000b) (Berthold et al. 2000a)	⊏	↔	≅	●··●	···	⊙	●	◷	●···	≡	✓	☺	◎	L	●–●	✓	✓	e
A Reputation System for Mixnets (Dingledine et al. 2001)	⊠	→	≅	●··●	···	⊙	●	◷	●··· ··●··	≡	✗	🛑	❋	H	🖂	✗	✗	@
Reliable Mix cascades (Dingledine and Syverson 2002)	⊠	→	≅	●··●	···	⊙	●	◷	●···	≡	✗	Ⓐ	❋	H	🖂	✗	✗	@
Mixmaster (Möller et al. 2003)	⊠	→	≠	●··●	···	⊙	◐	✗	●···	≡	✗	Ⓐ	⊛	H	🖂	✓	✓	@
Mixminion (Danezis et al. 2003)	⊠	→	≠	●··●	···	⊙	●	◷	●···	≡	✗	Ⓐ	⊛	H	🖂	✓	✓	@
Mix-Networks with Restricted Routes (Danezis 2003a)	⊏	→	≠	●··●	···	⊙	●	◷	●···	≡	✗	🌍	◎	H	🖂	✗	✗	@

and DHT-based protocols (Sections 3.3), DCnet protocols (Section 3.4), and finally, the class of miscellaneous protocols (Section 3.5).

3.1 Mixnet-Based Protocols

The idea of anonymous communication was originally proposed by Chaum in 1981 (Chaum 1981) and initiated a new field of privacy research. The central concept proposed by Chaum is the use of *mix nodes*, or *mixes* for short. Mix nodes cryptographically transform messages so that they cannot be traced based on their content. Further, mixes shuffle ("mix") input messages and output them in a reshuffled form. Thereby, they hide the input-output relation between individual messages, such that an adversary is not able to establish a correlation between input and output messages. In Chaumian mixes, the mix node does not output the messages immediately upon arrival but instead collects a certain number of messages (up to a threshold) into a so-called *batch*, which introduces a delay in message transmission. The mix shuffles input messages within a batch and flushes them out, which are ordered lexicographically.

3.1.1 Mix Selection Strategies. To distribute trust, Chaum proposed to relay messages through a fixed sequence of mix nodes[3] called a *mix cascade*. Chaum proposes a deterministic node selection without specifying how the nodes are selected (node selection strategy) for mix cascades. He only suggests that certain factors such as the networks topology and user's trust can be used for mix node selection. In a mix cascade, messages are successively encrypted (in a layered fashion) with the public key of each mix in the cascade (see Figure 1).

As the message is transferred from one mix to the next, the current mix peels off (decrypts) the corresponding layer (i.e., remove one layer of encryption with its private key), obtains the inner layer together with the corresponding address of the next destination, and sends the

[3]In the literature, a sequence of mixes is usually referred to as *path* or *route*.

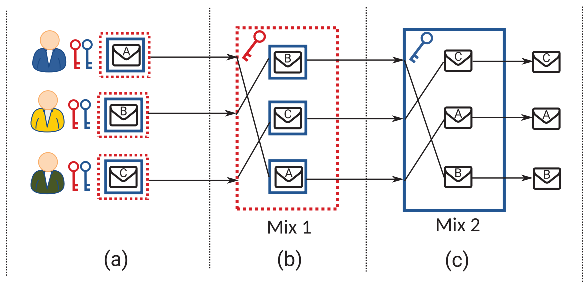

Fig. 1. A mix cascade with two mixes: (a) The users encrypt their message successively with the public key of each mix in the cascade and send it to the first mix (Mix 1). (b) The mix receives the messages, removes one layer of the encryption using its own private key, shuffles the messages, and forwards them to the next mix in the cascade. (c) Finally, once the last mix in the cascade (Mix 2) receives the messages, it decrypts the final layer of the encryption, shuffles the messages, and outputs the original messages in a reshuffled form. Note that the input-output relation between individual messages is hidden.

message to that destination. This procedure is repeated until the last mix delivers the data to its final destination. To receive replies for messages while staying untraceable (to obtain recipient anonymity (Pfitzmann and Köhntopp 2000)), return addresses are used. Chaum proposed to encrypt the address of the recipient of replies separately so that the respondent only needs to append the untraceable return address to its replies. The anonymous replies are also sent similarly in a layered fashion to the respondent. From now on, we refer to the encrypted return address block as the reply block. Note that in the case of the anonymous replies, the recipient of the reply is the routing decision maker. An alternative cryptographic model for mix cascades is using re-encryption mixes, where mixes re-encrypt messages instead of decrypting them.

To overcome a single point of failure in availability of mix cascades, *free-route* mix networks have been proposed (Gülcü and Tsudik 1996; Möller et al. 2003). In free-route mix networks, the route is not fixed and any sequence of nodes from the network can be used for relaying messages. An important aspect of mix cascades and free-route mix networks design is the node selection strategy. Selecting mixes for a mix cascade or for a path in a free-route mix network may follow different strategies. Namely, a deterministic strategy, a uniformly random selection, or a variation, such as random selection biased by network state or reputation/reliability scores. When multiple mix cascades are available for the users to choose from, node selection has two dimensions: selecting a set of mixes for building the cascades and selecting a particular mix cascade for relaying the messages. Moreover, predefined probability distributions and topological restrictions can also be taken into account for mix selection. Danezis (2003a) proposed the *restricted routes* mix networks that leverage the mix cascade and free-route mix networks by being less vulnerable to intersection attacks and being secure against global adversaries like mix cascades and being scalable like free-route mix networks. He proposes a mix network topology that is based on constant degree graphs (sparse expander graphs), where each mix only communicates with a few neighboring nodes based on a predefined probability distribution. Next, we review two variants of mix selection, one for free-route mix networks and one for mix cascades.

Mixes that fail lead to further delays in mix networks, thus selecting reliable mix nodes can lead to better performance. Dingledine et al. (2001) developed a method to identify mixes that fail and used a reputation system for mix selection leading to better reliability and efficiency for the mix network. In their proposed system, mixes issue receipts for each received message. After a mix has sent a message to the next mix, if it is not receiving a receipt within a restricted time, then it asks a set of witnesses to resend the message and receive the receipt and forward it to the original mix. The system establishes routing paths following the free-route node selection strategy, where the mixes are selected based on their past behavior (reputation score). Such a strategy suggests use of a non-deterministic node selection, biased toward mix nodes with high reputation scores. Mixes that have no positive ratings at all are avoided for mix selection. The main weakness of their scheme is that the reliability depends on the witnesses that need to be trusted, or at least a core group of trusted witnesses.

Unlike the previous system, which relies upon trusted global witnesses, Dingledine and Syverson (2002) proposed a mix cascade protocol with distributed trust. To obtain more reliable cascades, they propose to use a reputation mechanism for rearranging mix cascades. The construction of such cascade utilizes communal randomness and reputation scores provided by all of the mixes; therefore, there is no need of a trusted central authority. To mitigate the weakness of the previous work, mix nodes of a cascade act as witnesses for the reliability of their own cascade. All mixes submit random values to the configuration servers, which order mixes based on their reputation score and pick the top mix nodes to create a pool of mixes. From this pool, the mixes are selected randomly. For each cascade, routing relevant information such as available bandwidth or expected waiting time is published. Based on this information and the reputation score of the mixes, users choose mix cascade for their messages. Note that if the mix network is large, the network view might not be complete for the users.

3.1.2 Variations of Flushing Strategies. Flushing algorithm (or batching strategies) specifies the precise timing at when a batch of collected messages is flushed out of the mix to be simultaneously delivered to the respective recipients. Flushing strategies are analogous to the forwarding component of the routing and they highly influence the scheduling routing characteristic defined in Section 2.1. Recall that Chaumian mixes collect messages until a certain threshold is reached. Such mixes are called threshold mixes. Threshold mixes might induce very high-latency if the traffic load is low. Thereafter, other flushing algorithms have been proposed in the literature.

Mixes that delay messages individually, for example, based on a certain probability distribution, and lead to continuous flushing are called continuous mixes. One example of continuous mixes is the *Stop-and-Go* mixes (*SG-mix*) (Kesdogan et al. 1998) system. The initiator of a message assigns for each mix in the path a randomly selected delay (from an exponential distribution). The independent random delays that are assigned to each message make the performance and anonymity of each message independent of the other users in the system. However, a drawback of their system is that SG-mixes are vulnerable when incoming traffic is low (Díaz and Preneel 2004). Another type of flushing algorithms is pool mixes that only flush out a fraction of messages of a batch at each round and keep the remainder in the memory of the mix (pool) for next flushing rounds. In pool mixes, the number of messages that are forwarded may be determined by deterministic or non-deterministic functions, and the message selection may be uniformly random or weighted based on dynamic conditions (e.g., based on incoming traffic). When the average delay of the messages is equal, pool mixes offer better anonymity, since the anonymity set is bigger. Another advantage of pool mixes is that they are suitable for networks with fluctuating traffic load. Pool mixes, however, still need to specify *when* messages are flushed out and therefore combined with other flushing techniques such as threshold (described above) or time restrictions. Timed mixes enforce a time restriction for flushing out messages. The anonymity of timed mixes is vulnerable to low traffic, since if only one message arrives before the time restriction is met, the mix provides no anonymity measure for that message. Moreover, a combination of the aforementioned flushing strategies can also be used by mixes (Díaz and Preneel 2004; Serjantov 2004). For example, the two prominent *remailers*, namely Mixmaster (Möller et al. 2003) and Mixminion (Danezis et al. 2003), use *timed* *dynamic pool mixes* as flushing strategies (Serjantov et al. 2003), which are a combination of timed and threshold pool flushing techniques, where the parameters depend on the network traffic. The flushing algorithm of Mixmaster has been characterized by generalized mixes (Díaz and Serjantov 2003). We review these remailer protocols in Section 3.1.3.

Next, we review some mix protocols in the literature that have been suggested for applications such as ISDN telephone, web browsing, and anonymous emails. To anonymize ISDN telephone communication with its intrinsic requirements on low-latency, Pfitzmann et al. (1991) introduced the concept of *ISDN mixes*. An important feature of ISDN mixes is to maintain constant traffic in the network to avoid traffic analysis. ISDN mixes are a type of threshold mixes. To obtain sender and receiver anonymity simultaneously, ISDN mixes use two mix cascades that are selected by the sender and receiver, respectively, connected either by a connecting mix or when used in long distance communications by the long distance network operators. Initially, a broadcast takes place to exchange the connecting details and the time where the communication takes place. To achieve constant traffic, a number of ISDN channels, with an equal amount of messages, need to start and end their communication at the same time (in a so-called *time-slice*). However, this is time-consuming and would lead to blocking the connection, which is not suitable, since ISDN mixes use narrow-banded channels and were designed for low-latency communication. In Table 2, we disregard the setup broadcast message used for exchanging information for ISDN mixes. The basic design of ISDN mixes was later generalized by Jerichow et al. (1998) to a system that enables low-latency, real-time communication.

A real-world realization built on ISDN mixes are *Webmixes* (also known as JAP) (Berthold et al. 2000a, 2000b) designed for real-time Internet applications, passing the traffic to several available mix cascades. In Webmixes, the mixes transform the messages cryptographically and reshuffle their order before flushing them out. However, messages are not delayed by flushing strategies. Webmixes use an adaptation of the time-slice method introduced by ISDN mixes. Routes in Webmixes consist of *JAP proxies*, which are local software at the users, one (or several) mix cascade(s) consisting of reliable and high capacity mix nodes, and a cache-server. Web requests are sent from the users JAP proxy through the mix cascade and the cache-server and, furthermore, delivered to the destination server. The web replies are sent back through the same route and a copy of the reply is saved at the *cache-server*. Hourly mix cascade information is published by so-called *info servers*. Users can choose among the published mix cascades by the info servers. To build mix cascades, ISDN mixes, real-time mixes, and Webmixes use deterministic node selection, where nodes selection for the cascades relies on the network state.

3.1.3 Prominent Applications of Mixes: Remailers. The original concept of mixes has an immediate application to *high-latency* remailer systems for providing anonymous e-mail service.

Babel (Gülcü and Tsudik 1996) aims at mitigating traffic analysis attacks by delaying only some messages of the batches. Babel uses independent forward routes and return routes. Forward routes may include a reply block (where the return route mix addresses are encrypted in a layered fashion) that may be used by recipients for anonymous replies. Forward routes are considered to have better anonymity; one of the reasons for this is that reply blocks enable replay attacks on anonymous replies (Danezis et al. 2010). Babel introduces *intermix detours*, where mix nodes choose a random sequence of mixes and relay the message through them before forwarding the message further to the next mix of the original route. In Babel, the flushing algorithm uses time restrictions (intervals) and thresholds for flushing out messages. Another technique Babel proposes to use is *probabilistic deferment*, where a number of messages (determined by a biased coin) are delayed at each mix (this is similar to pool mixes). Babel proposes to use of free-route mix networks, where mixes are chosen uniformly random for each route by the user. However, there were no details given about how routing information could be communicated to users.

Mixmaster (Möller et al. 2003) is an anonymous remailer, where mixes transform messages cryptographically into uniform sizes by adding random data at the end of each data packet. If a message is too large, then Mixmaster splits up the message to achieve uniform sized packets and sends these packets independently of each other through a series of mixes, which do not necessarily need to be all the same. Only the last mix needs to be the same for all packets of one email message, which has been split up before. Mixmaster adopts a free-route path selection (Danezis et al. 2010), where the users choose nodes with a non-deterministic algorithm that uses statistics on the reliability of mixes bias node selection (Danezis 2003a). Although the Mixmaster protocol did not specify details about maintaining mix information, later implementations of Mixmaster adopted an ad hoc scheme for distributing routing information (Danezis et al. 2003). One of the main weaknesses of Mixmaster is that it only guarantees sender anonymity, since reply blocks are not used in Mixmaster.

Mixminion (or Type III remailer) (Danezis et al. 2003) are widely considered as the state-of-the-art remailer. To guarantee equal routing information for all senders, Mixminion deploys a group of redundant and a synchronized system of *directory servers*, which was not considered in the Mixmaster design. Note that we disregard the directory servers synchronization for our classification in Table 2. Like Mixmaster, Mixminion also uses "timed dynamic pool." Mixminion uses reply blocks. Generally, reply blocks enable replay attacks; hence, Mixminion introduces *Single Use* Reply Blocks (SURB), where for each reply message, the content of the reply is appended to the SURB and sent through the mix network. In the Mixminion communication model, the routing path is divided into two so-called *legs*, each consisting of half of the mixes in the route. For reply messages, where both sender and receiver anonymity is desirable, in the first leg of the route, the sender of the reply chooses the mixes and appends the SURB for the second leg. When the message is traversing the route, at a crossover point (the last mix in the first leg), the SURB replaces the first leg, and the message is routed further to the intended recipient. In such cases, the route consists of mixes, which are half chosen by the sender and half chosen by the recipient. Thus, Mixminion aims at providing sender anonymity and recipient anonymity for email messages. Moreover, since forward and reply messages are not distinguishable from each other by outsiders and intermediate mix nodes themselves, they share the same anonymity set. The exceptions are the crossover points that have partial knowledge and the exit mix nodes, because they can observe whether the content has been encrypted or is in plain text. Mixminion also suggests choosing nodes from preferably a large pool; however, further details on the node selection strategy have not been specified in Mixminion.

3.1.4 *Discussion.*

Mixnets, as classified in Table 2, show very heterogeneous routing designs due to their routing diversity on multiple routing building blocks, which in turn lead to topological differences.

As mentioned earlier, existing routing strategies can be classified into *free-routes* mix networks and *mix cascades*. However, we distinguish whether a connection is potentially allowed between two nodes or not based on routing of the messages. Hence, we marked most of the mix cascade networks as fully connected. The connectivity in restricted route mix networks and Webmixes is restricted due to restrictions in the selection set, which leads to a partially connected network topology.

Generally speaking, mix cascade networks employ rather synchronized connection, because messages are sent in batches and mostly depend on their flushing algorithms on a timely schedule. For example, timed mixes lead to synchronized message transmission. Recall that the flushing algorithm in Mixmaster and Mixminion partially uses time restrictions. However, we consider these two protocols with asynchronous message transmissions due to the possibility that low traffic might lead to a threshold restriction instead of a time restriction. As for free-route systems, in SG-mixes, message transmission is also synchronized due to assigned time ranges by the routing initiator. Nevertheless, these timing ranges are not coordinated with other users or mix nodes.

In Mixnets, node management is not always specified in the protocol description. For example, in Chaumian mixes, the view of the routing decision maker is not discussed; however, it can be implicitly deduced that it is complete. The anonymous remailer Mixmaster does not discuss node management either; however, the later implementation uses ad hoc systems, which suggests a partial view (Danezis et al. 2003). The remailer Mixminion defines a node management strategy to insure a complete view for the routing decision maker.

Source-routing is one of the inherent routing features of mix cascade protocols, because the routing paths are fixed beforehand. Choosing the mixes for the mix cascade might be either deterministic, such as in the case of Webmixes, or non-deterministic, such as in the case of Reliable mix cascades.

Flushing algorithms do apparently impact scheduling. Note that some of the protocols in Table 2 use randomness in the scheduling process (e.g., pool mixes). Consequently, some messages are forwarded later than others. Since individual messages do not have priorities by themselves, we categorized them also as fair. How the set of nodes is derived for node selection has also not been specified precisely for mix networks. The same holds for selection probability, such as for Chaumian mixes. For mix networks, we categorized the selection probability as deterministic, because all mixes are chosen for a single mix cascade. For both mix cascade protocols and free-route mix networks, the selection set varies depending on the application of the AC network and on the potential anonymity properties.

As mentioned in Section 3.1, in mix cascades, the selection probability has two dimensions when more than one cascade exists. For instance, Webmixes can provide multiple mix cascades, where mixes are chosen by the network administrator for each mix cascade. Thereafter, the user manually selects one of these mix cascades for routing her messages. Another mix cascade protocol, where mixes are selected deterministic, is ISDN mixes.

All mix cascade protocols are high-latency AC networks and have a message-based communication mode; exceptions are ISDNs, Real-time mixes, and Webmixes, which are designed for low-latency applications, such as web browsing. Note that the latencies might be restricted, for instance in case of SG-mixes, where the delays are randomly selected from a restricted time range.

3.2 Tor-Related Protocols

Tor is based on onion-routing (Goldschlag et al. 1996; Reed et al. 1998) designed for anonymous communication for applications with low-latency constraints, such as web browsing. An onion-routing network consists of a set of nodes so-called *Onion Routers (ORs)*. Users choose an ordered sequence of ORs to establish a bidirectional channel, so-called *circuit*, for relaying their data through the onion-routing network. The communication is encrypted in a layered fashion and the ORs in the circuit each can decrypt their corresponding layer. When the communication is relayed by an OR in the circuit, the OR removes the corresponding layer of encryption and forwards the data to the next OR in the circuit (see Figure 2). The last OR forwards the data to the destination. Each OR only knows their predecessor and successor in the circuit, and the complete sequence is only known to the circuit initiator (the user). Therefore, only the first OR in a circuit is aware of the IP address of the user who has initiated the circuit and only the last OR of a circuit is aware of the destination of the communication, which is relayed through the circuit. The response of the receiver is relayed back to the initiator through the same circuit. Similar to

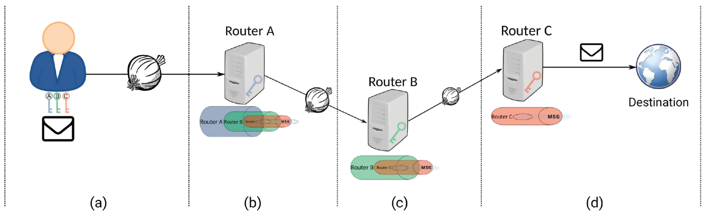

Fig. 2. The concept of onion-routing: (a) The sender chooses an ordered sequence of onion routers for relaying his message. For instance, the sender chooses Router A, B, and C. She then encrypts the message in a layered fashion starting with the key of the last onion router (Router C) along with an instruction that indicates the next hop in the sequence. The newly created data structure is called onion. Finally, she forwards the onion to the first onion router in the sequence (Router A), so called entry node. (b) Once the message is delivered to the entry node, the router decrypts one layer of encryption with its corresponding key and forwards it to the next hop following the instruction within the layer. At this point, the onion router is aware of the identity of the sender and the next hop. (c) Router B receives the onion, removes one layer of encryption, and forwards it to the next hop (Router C). (d) Finally, Router C receives the onion, removes the last layer of the encryption and forwards the message to its final destination. Note that, the exit node (Router C) does not who is the sender, it is only aware of the final destination and its predecessor node.

Table 3. Routing Classification of Anonymous Communication Protocols: Tor-Related Protocols

	Network Structure						Routing Inf.		Communication Model					Performance and Deployability				
		Connection Type		Symmetry							Node Selection							
	Topology	Direction	Synchronization	Roles	Hierarchy	Decentralization	Network view	Updating	Routing type	Scheduling	Determinism	Selection set	Selection probability	Latency	Communication mode	Implementation	Code availability	Context/application
Tor (Dingledine et al. 2004)	□	↔	≠	●··⊙··●	···	⊙	●	◷	●···	≡	✗	🌍🛑	◎	L	●–●	✓	✓	e
AS-Aware Node Selection (Edman and Syverson 2009)	□	↔	≠	●··⊙··●	···	⊙	●	◷	●···	≡	✗	🌍🛑	◎	L	●–●	✓	✓	e
LASTor (Akhoondi et al. 2014)	□	↔	≠	●··⊙··●	···	⊙	●	◷	●···	≡	✗	🌍🛑	◎	L	●–●	✓	✗	e
Coordinate Node Selection (Sherr et al. 2009)	□	↔	≠	●··⊙··●	···	⊙	●	◷	●···	≡	✗	🌍🛑	◎	L	●–●	✓	✓	e
Tuneup for Node Selection (Snader and Borisov 2011)	□	↔	≠	●··⊙··●	···	⊙	●	◷	●···	≡	✗	🌍🛑	❋	L	●–●	✓	✗	e
Congestion-aware Node Selection (Wang et al. 2012)	□	↔	≠	●··⊙··●	···	⊙	●	◷	●···	≡	✗	🌍🛑	❋	L	●–●	✓	✗	e
Panchenko Node Selection **MaTor** **DeNasa** (Panchenko et al. 2012) (Backes et al. 2014) (Barton and Wright 2016)	□	↔	≠	●··⊙··●	···	⊙	●	◷	●···	≡	✗	🌍🛑	◎	L	●–●	✓	✗	e
Torchestra, PCTCP, IMUX (Gopal and Heninger 2012) (AlSabah and Goldberg 2013) (Geddes et al. 2014)	□	↔	≠	●··⊙··●	···	⊙	●	◷	●···	≡	✗	🌍🛑	◎	L	●–●	✓	✗	e
Conflux (AlSabah et al. 2013)	□	↔	≠	●··⊙··●	···	⊙	●	◷	●···	≡	✗	🌍🛑	❋	L	●–●	✓	✗	e
Prioritized Scheduling **DiffTor** (Tang and Goldberg 2010) (AlSabah et al. 2012)	□	↔	≠	●··⊙··●	···	⊙	●	◷	●···	◇	✗	🌍🛑	◎	L	●–●	✓	✗	e
PIR-Tor (Mittal et al. 2011b)	□	↔	≠	●··⊙··●	···	⊙	◐	◷	●···	≡	✗	🌍🛑	◎	L	●–●	✓	✗	e
MTor (Lin et al. 2016)	□	↔	≠	●··⊙··●	✤	⊙	◐	◷	●···	≡	✗	🌍🛑	◎	L	●–●	✓	✗	👥

Webmixes, in onion-routing, the ORs implement First-In First-Out (FIFO)-like forwarding strategy to provide low-latency services. Having no delays at the ORs and due to missing cover traffic onion-routing are susceptible to a number of attacks, such as traffic analysis and timing attacks, where an adversary may identify and correlate traffic patterns at the initiator and receiver (Danezis and Díaz 2008; Ren and Wu 2010), thus de-anonymizing the connection. Nonetheless, onion-routing is a promising design to provide a low-latency AC network, and many currently used systems can build upon this design.

Onion-routing used in Tor (Dingledine et al. 2004) constitutes an extension of the original onion-routing design, with some modifications to achieve better security, efficiency, and deployability. The Tor network, an open-source and a free-to-use framework, consists of a large set of volunteering routers (at the time of writing, there exist more than 7,000 routers (The Tor Project 2017)). The network is mostly connected, because routers can connect to any router from the Tor network, except for connections between routers located in the same IP /16 subnet space, which is not possible. Tor's services are used daily by approximately 2,000,000 users (The Tor Project 2017). Each user runs a piece of software called Onion Proxy (OP) that manages all Tor-related processes, e.g., establishing circuits or handling connections from user applications. Tor deploys a group of well-known and trusted authoritative servers that publish on a regular basis (typically, every hour) a list of all active Tor nodes with their characteristics, e.g., estimated bandwidth, IP addresses, and cryptographic keys. This list is called a *consensus*. After the user has obtained the consensus, the OP of the user chooses an ordered set of usually three ORs to build a circuit. The first node in a circuit is called the *entry* node, the second node is the *middle* node, and the last node in the circuit is the *exit* node. The first node that is selected is the exit node, then the entry node of the circuit is selected, and last the middle node of the circuit is selected. After selecting a set of ORs, the OP contacts the entry node and builds a circuit with it. This newly created circuit is used to contact the middle OR to extend the circuit and similarly through the middle node the exit node is contacted to extend the circuit. The established circuit can now be used to anonymously relay data.

In 2002, Wright et al. introduced the predecessor attack (Wright et al. 2002) on onion-routing. To defend against this and related attacks, selecting a small set of nodes was introduced for Tor (Wright et al. 2003). Previously, each user maintained a list of three randomly pre-selected (so-called *guard*) nodes with high bandwidth and uptime. This list was updated every 30–60 days and the user could choose uniformly random an entry node from this list for each path construction. This has changed recently, because Tor is starting to let each user select only one fixed entry guard node for 9 months (Dingledine et al. 2014).

In the early onion-routing design, uniformly random node selection was suggested (Syverson et al. 2001). Due to performance considerations, Tor's routing policy does not select nodes with the same probability, but rather preference is given to high-bandwidth nodes. The likelihood that nodes are chosen for certain positions in a given route depends on the ratios of overall node bandwidths and node characteristics such as the IP addresses and whether they can be selected as entry node or as exit node. Moreover, some additional bandwidth weights are used to balance off the node selection. As mentioned before, a further development in the routing policy is to disallow a communication to pass through two nodes within the same /16 subnet IP address. The implications of these changes with respect to structural node corruption have been recently explored by Backes et al. (2013) and Backes et al. (2014).

Next, we review two prominent attacks on Tor's routing. Murdoch et al. have proposed a traffic-analysis attack using timing information to identify Tor nodes and to infer traffic load to a specific initiator. Their investigation shows a degradation of Tor's anonymity against such attacks. They furthermore propose some strategies to prevent the risk of such attacks, mainly by increasing communication latency (Murdoch and Danezis 2005). Bauer et al. have proposed a traffic analysis attack aimed at decreasing the anonymity of Tor (Bauer et al. 2007). Their attack investigates the load balancing that is performed by Tor, where high-bandwidth nodes are preferred in the node selection strategy. They show that performance optimization impairs the anonymity of Tor against end-to-end traffic analysis attacks.

Since Tor has been proposed, there has been a great deal of research on extending Tor's routing strategy. The proposed extensions to the Tor routing protocol aim mostly at improving either the achieved anonymity of Tor or the performance that Tor users experience.

Improvements to Tor's anonymity have been often realized by aiming at an improved node selection. For example, improving anonymity by using better weighting at the node selection phase has been proposed in Panchenko et al. (2012) and Backes et al. (2014). Involving AS-level information in the node selection has been proposed by Edman and Syverson (2009) and Akhoondi et al. (2014). Moreover, offering the user a tune-up option between uniformly random node selection (for high anonymity) and weighted random node selection with a bias toward high-bandwidth nodes (for better performance) has been suggested by Snader and Borisov Snader and Borisov (2011).

Tor's performance problems have several causes, and hence suggested improvements aim at different aspects of the Tor routing protocol. One cause of Tor performance is high congestion (AlSabah and Goldberg 2015; Dingledine and Murdoch 2009), often caused by bulk traffic, which induces high-latency for interactive/web traffic. Several solutions to solve the problem of high waiting times for interactive traffic have been proposed. One possible solution is to increase the number of connections between two nodes (Geddes et al. 2014; AlSabah and Goldberg 2013; Gopal and Heninger 2012; AlSabah et al. 2013), which can be used to separate interactive and bulk traffic into different connections. Another solution is to prioritize interactive traffic in the scheduling phase (Tang and Goldberg 2010; AlSabah et al. 2012). An alternative solution is to improve how Tor's resources are used by improving node selection with a more realistic estimation of the available bandwidth of nodes (Panchenko et al. 2012). Furthermore, another solution to Tor's congestion problem is to enforce avoiding congested nodes at the node selection phase (Wang et al. 2012). Another reason for Tor's high-latency is circuitous paths (Akhoondi et al. 2014). To solve this problem, node selection strategies have been proposed that take the destination between chosen nodes into account (Akhoondi et al. 2014; Sherr et al. 2009; Panchenko et al. 2012).

The scalability of Tor has also been subject to new proposals for the Tor routing protocol in the literature. One proposal to tackle scalability issues is to give the user only the information about the necessary nodes for path construction and to hide the complete view of the system from the user by either managing Tor nodes as a DHT table and using Kademlia for node retrieval (McLachlan et al. 2009) or by using private node retrieval (Mittal et al. 2011b).

3.2.1 Discussion. On a conceptual level, all *Tor-related protocols* are equally characterized by their routing features. However, there are three exceptions that affect this: the completeness of the network view, the fairness of scheduling, and the node selection probability (leaving apart the non-technical question if the code has been made publicly available). Their differences, however, often lie in implementation details, which are not necessarily relevant to routing, such as reducing buffer size (AlSabah et al. 2011). In addition, differences in the routing policy, which do not change the routing feature on a conceptual level, such as changing node selection probabilities (Backes et al. 2014; Panchenko et al. 2012), are equally classified in the table, even though node selection probabilities could be different.

One inherent routing feature of Tor-related protocols, due to preventing additional latency, is to have no synchronization, which makes such protocols sensitive to timing attacks and global adversaries. Another inherent feature is that all Tor-related protocols have a client-server model, which improves their usability and leads to a higher number of users, thus contributing to better

Table 4. Routing Classification of Anonymous Communication Protocols: Random Walks and DHT-Based Protocols

	Network Structure						Routing Info.		Communication Model					Performance and Deployability				
		Connection Type		Symmetry							Node Selection							
	Topology	Direction	Synchronization	Roles	Hierarchy	Decentralization	Network view	Updating	Routing type	Scheduling	Determinism	Selection set	Selection probability	Latency	Communication mode	Implementation	Code availability	Context/application
Crowds (Reiter and Rubin 1998)	⊠	↔	≠	●··●··●	···	⊙	●	↯	··●··	≡	✗	Ⓐ	⊛	L	✉	✓	✗	e
MorphMix (Rennhard and Plattner 2002) (Rennhard and Plattner 2004)	⊏	↔	≠	●··●··●	···	⊙	◐	↯	··●··	≡	✗	🌍	✱	L	●–●	✓	✓	e
Tarzan (Freedman et al. 2002) (Freedman and Morris 2002)	☐	↔	≠	●··●··●	···	○	●	↯	●···	≡	✗	🛑	⊛	L	●–●	✓	✓	e
Torsk (McLachlan et al. 2009)	⊏	↔	≠	●··⊙··●	···	⊙	◐	↯	●···	≡	✗	🌍	⊛	L	●–●	✓	✗	e
NISAN (Panchenko et al. 2009)	⊏	↔	≠	●··●··●	···	○	◐	↯	●···	≡	✗	Ⓐ	⊛	L	✉	✓	✗	e
AP3 (Mislove et al. 2004)	⊏	↔	≠	●··●··●	···	○	◐	↯	··●··	≡	✗	Ⓐ	⊛	L	●–●	✗	✗	📢 @ e
Salsa (Nambiar and Wright 2006)	⊏	↔	≠	●··●··● ●··⊙··●	···	○	◐	↯	●···	≡	✗	Ⓐ	⊛	L	●–●	✓	✗	e
Octopus (Wang and Borisov 2012)	⊏	↔	≠	●··●··●	···	⊙	◐	↯	●···	≡	✗	Ⓐ	⊛	M	✉	✓	✗	✎ @
Freenet Opennet (Clarke et al. 2001)	⊏	↔	≠	●··●··●	···	⊙	◐	🕒	··●··	≡	✓	Ⓐ	✱	L	✉	✓	✓	📄
Freenet Darknet (Clarke et al. 2010)	⊏	↔	≠	●··●··●	···	○	◐	🕒	··●··	≡	✓	☺	✱	L	✉	✓	✓	📄
GAP from GNUnet (Bennett et al. 2002) (Bennett and Grothoff 2003)	⊏	↔	≠	●··●··●	···	○	◐	↯	··●··	⟐	✓	Ⓐ	✱	M	✉	✓	✓	📄

anonymity for Tor-related protocols (Dingledine and Mathewson 2006). They are characterized as complete network view due to a central authority, which distributes the list of Tor routers. One exception is Mittal et al. (2011b), which realizes private node retrieval and thereby constrains the decision maker's view of the network. A complete view helps against adversary biasing node selection and is preferred in source-routing to prevent the decision maker from choosing from a smaller set of nodes.

Further inherent routing features concerning the communication model include routing type, scheduling, determinism in the node selection, and the selection set. The exceptions here are Tang and Goldberg (2010) and AlSabah et al. (2012), who suggest a prioritization at the scheduling phase in favor of interactive traffic to reduce delays that interactive users might experience.

Node selection in all Tor-related protocols is non-deterministic. This is important, since the Tor network consists of volunteers and is very likely to have a fraction of malicious nodes among them. A non-deterministic node selection reduces the chances of consistently selecting malicious nodes. Given a large number of Tor relays that are spread around the world, and since the adversary is assumed to be local, a non-deterministic node selection makes targeted surveillance harder. Barton and Wright proposed an AS aware node selection for Tor (Barton and Wright 2016) to improve Tor's resilience against adversaries controlling an AS.

Furthermore, the node selection probability is generally weighted using static parameters, except for a few approaches that dynamically adjust weights, e.g., for balancing security versus performance (Snader and Borisov 2011) and for avoiding congestion (Wang et al. 2012; AlSabah et al. 2013). Tor-related protocols are low-latency and have circuit-based communication mode, which are both inherent routing features of these protocols. Although we classify Tor as a protocol where the routing decision maker has a complete view, it is worth mentioning that the unlisted relays,

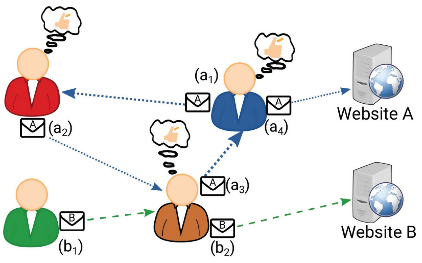

Fig. 3. The basic concept of Crowds: In this simple scenario, two users want to send their messages (message A and message B) to Website A and Website B, respectively. First, they send their message to a randomly selected node (step a_1) and (step b_1). The receiving node flips a coin to decide whether to send the message to another node or to its final destination. The receiving node of the message A randomly decides to send the message to another node (step a_2), while the receiving node of the message B decides to send the message to its final destination (step b_2). The receiving node of message A, again by flipping a coin, decides to send the message to another node (step a_3). The message goes further until a node decides to send it to the final destination (step a_4). Finally, the third receiving node decides to forward the message to its destination (step a_4).

known as *bridges*, are not part of this view. MTor is an extension built on top of Tor to facilitate group communication within Tor (Lin et al. 2016), which is designed to be scalable.

3.3 Random Walks, Structured and Unstructured DHT-Based Protocols

In this section, we review *Random Walk protocols*, where the communication is relayed randomly through the network. We consider a protocol a Random Walk protocol if node selection is hop-by-hop routed and a random selection. Random Walk protocols are often combined with peer-to-peer network structures.

Crowds (Reiter and Rubin 1998) is one of the early AC systems designed for anonymous web browsing. The key design feature of Crowds is a random peer selection and a peer-to-peer structure, where all users of the system are nodes themselves. In Crowds, all nodes are grouped into so-called *crowds* and a so-called *blender* is responsible for managing and administrating nodes; all nodes within a crowd might connect to each other for relaying a communication. To use Crowds, a user randomly selects a node and sends her message (i.e., website request). Upon receiving the request, this node flips a biased coin to decide whether to send the request directly to the receiver or to forward it to another node selected uniform at random. This continues until the message arrives at the destination (see Figure 3). The server replies are relayed through the same nodes in reverse order. Wright et al. showed that Crowds is vulnerable to so-called predecessor attacks (Wright et al. 2002, 2004). In this attack, the attacker tracks an identifiable stream of communications over a number path reformations and logs the nodes that send the message. Eventually, the attacker sees the initiation more often than the other nodes. To prevent such type of attacks, Crowds suggests to employ static route (a user keeps the route for a while) such that an attacker does not have multiple routes to link to the same node (Reiter and Rubin 1998). However, even keeping routes static for a day is not enough to prevent predecessor attacks (Danezis et al. 2010).

MorphMix (Rennhard and Plattner 2002, 2004) is a dynamic peer-to-peer AC network. Technically, MorphMix establishes circuit-based connections using layered encryption, where the anonymous route is established iteratively by the nodes on the route. Each node is typically only aware of a set of network nodes, which is not necessarily covering all nodes. To avoid repeated connections with the same set of nodes, a node has to forget about nodes it has not been connected and

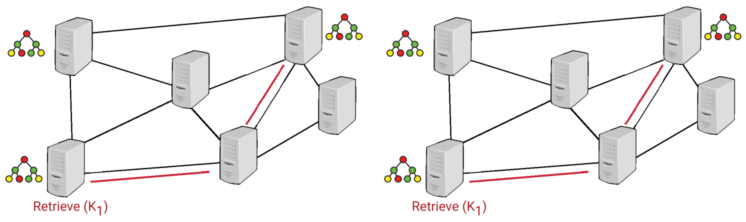

Fig. 4. An overview of a general DHT network (left image) and a DHT network where the search algorithm for routing indexes uses binary trees (image on the right), such as Salsa or DHT networks using the Kademlia look-up method.

constantly require new node information. After an initiator selects the first node, it selects randomly a witness for each hop thereafter, randomly chosen from the nodes in its local database and asks the next hop to extend the route with the assistance of the witness. Each node then proposes a set of candidate nodes for the next hop and the corresponding witness chooses one of these candidates as the next hop. To prevent path compromise, nodes can only propose nodes with different IP prefix to its own IP address to the witness. The witness should not be selected from the nodes to which the initiator is connected currently to avoid initiators being identified by witness nodes. To mitigate guessing whether a node was the initiator of the path by the next hop, the initiator adds random delays to its communication before forwarding them in the tunnel establishment phase.

Tarzan (Freedman et al. 2002; Freedman and Morris 2002) is a peer-to-peer anonymous fully decentralized IP-level network overlay. All participants are peers; they are all potential originators of traffic and also potential relays. Tarzan nodes do not implement any mixing strategies and simply forward incoming traffic. After the initiator node selects a set of nodes (preferably from existing connections from previous communication rounds) to form a route through the overlay network, a tunnel via these nodes is established for relaying communication. Unlike the early design of the protocol (Freedman et al. 2002), where the peers only needed to know about a random subset of nodes, the later version (Freedman and Morris 2002) introduces a *gossip-based* protocol based on the Name-Dropper protocol (Harchol-Balter et al. 1999), where more node information is requested from randomly chosen nodes. The aim is to gain information about all available servers in the network to avoid attacks that are facilitated due to churn, such as fingerprinting attacks (Danezis and Clayton 2006). Node information is stored in a ring model and lookups are carried out using the Chord algorithm (Stoica et al. 2001). The initiator only selects nodes randomly from distinct IP subnets, a three-layer hierarchy selection is used to make sure nodes are from distinct subnets.

Efficiency is one of the main problems in Random Walk protocols. In the next section, we review Random Walk protocols that employ DHT lookups to gain better efficiency (e.g., AP3 protocol (Mislove et al. 2004)).

3.3.1 DHT-Based Protocols. In distributed systems, where there are network administrators, a challenge is to locate a node. One solution is to use DHTs to manage the distributed nature of the data (relaying nodes or distributed storage). Generally, DHT refers to a trust-distributing, structured data management model for storing (value, key) pairs and is accompanied with key-based lookups for locating the corresponding stored value (see Figure 4). The value might be, for example, either the router information of relaying nodes in a distributed network or a stored content (file). The keys are hashed from the identifier of the value (for nodes, their IP addresses are hashed into keys). In the literature, several lookup strategies for the DHT-based structures have been proposed, aiming at efficient searching. Some popular lookup strategies are Kademlia

(Maymounkov and Mazières 2002) (locating the nodes based on their estimated distance using an XOR metric), Chord (Stoica et al. 2001) (using a clockwise circle metric, where at each hop of the lookup, the distance to the node is decreased, at least half), and Pastry (Rowstron and Druschel 2001) (carrying out lookups based on numerical identifiers).

DHT structures enable efficient routing even when the peers of a DHT structure keep only information (key-value pairs) of a partial subset of all the other peers of the DHT structure; this, in turn, leads also to improved scalability of such systems. Another important feature of DHT-based structures is having better load balancing. For systems where nodes have only a partial view of the structure, hop-by-hop routing is preferable. Some AC protocols use randomness in the routing strategy in addition to the typical lookup method, for example, by selecting a randomly constructed key and a classical lookup method, such as an adaptation of Chord, Kademlia, or Pastry, to find that key. Next, we review AC protocols that use an adaptation from Kademlia, Chord, Pastry for their node lookup (considered as structured DHT-based protocols). We proceed by reviewing independent DHT-based routing proposals for AC that are considered unstructured DHT-based protocols. We start with *AP3* (Mislove et al. 2004), a Random Walk protocol aiming at offering anonymity when a large part of the nodes is compromised. AP3 uses the same routing strategy as Crowds, with the difference that the node information is retrieved using Pastry and that the node does not have a complete view of the system.

Next, we review two protocols that aim at replacing node selection of source-routed protocols such as Tor-related protocols with structured DHT systems making them suitable to be combined with onion-routing. *Salsa* (Nambiar and Wright 2006), proposed by Nambiar et al., aims to provide scalability and prevent malicious colluding nodes from biasing routing. Salsa virtually divides nodes into groups, which are organized in a binary tree form. For routing, simultaneous redundant lookups and bound checking are used to avoid malicious nodes returning wrong addresses. The lookup queries are carried out similar to the Chord lookup in a recursive fashion. In Salsa, the routing information that is available to each node is partial; however, the tree structure allows nodes to carry out source-routing.

McLachlan et al. have proposed *Torsk* (McLachlan et al. 2009), a peer-to-peer AC protocol, replacing Tor's node selection and directory service with a DHT design to provide better scalability for Tor. Their design uses DHT tables for node selection by using a randomly chosen key that is looked up in the table using Kademlia. To secure lookups, Torsk uses the "root certification" approach proposed by Myrmic (Wang et al. 2006) and randomly selected secret "secret buddies."

Panchenko et al. proposed *NISAN* (Panchenko et al. 2009), an AC protocol that aims at achieving high scalability and preventing adversaries from bias routing. NISAN uses iterative search to select nodes randomly for constructing anonymous paths and uses an adaptation of Chord, where the node lookups are aggregated to hide which node is exactly looked up for the next hop. Moreover, NISAN hides the node that it is looking up, by requiring the complete routing table and enforcing bound checking to further prevent selecting nodes from routing tables, which were manipulated by malicious nodes.

Octopus (Wang and Borisov 2012) aims at preventing malicious nodes from biasing the routing procedure and provides anonymity by hiding which nodes have been looked up for anonymous paths. For routing, Octopus uses iterative lookups by sending the query to the closest node to the searched key in the local routing table and then retrieving the routing table from that node until the node containing the corresponding value to the key is found. Node selection is carried out in two phases, where in the first phase nodes are selected by the path initiator (user) and in the second phase the last node selected in the first phase chooses the remaining nodes. Therefore, Octopus is not purely a Random Walk protocol. After establishing anonymous paths, Octopus splits queries to different paths and adds dummy traffic to hide the real queries among them. Furthermore, as security measures, Octopus enforces bound-checking on the received routing tables to prevent using manipulated routing tables, and it proactively tries to identify and remove malicious nodes.

Next, we review two file-sharing protocols that use DHT for routing file requests and their responses. They, however, use unstructured routing. Clarke et al. proposed *Freenet* (Clarke et al. 2001), a peer-to-peer censorship-resistant system for sharing storage space. Freenet offers strong decentralization to provide privacy and robustness against attacks, where the key design feature is based on storage replication and plausible deniability. The relaying nodes only know their predecessor and the successor in order provide privacy and when a request is sent or receiver the node does not know whether the predecessor was the initiator or whether the successor is the recipient. Files are stored multiple times at the nodes, are indexed by binary file keys, and can be looked up by their corresponding key. Replication of files provides resilience against node failure and node overloads. Each node has a dynamic routing table including the node information with the stored keys, where routing tables are updated periodically to achieve better performance. Freenet uses an adaptive routing using DHTs with keys that are location-independent. The original Freenet design uses a heuristic-based deterministic routing that is a distance-directed depth-first search (with backtracking) (Clarke et al. 2010; Roos et al. 2014) and uses potentially all participating nodes choosing mostly neighborhood nodes (currently called Opennet mode).

When a file request arrives at a node, including a key and a value for *hops-to-live*, if the file is not stored locally, the node looks up the node with the nearest key in the routing table and forwards the file request to the corresponding node. The node receiving the request repeats the process until either the file is found or the hops-to-live is reached. If the requested file is found, then the node forwards the file to the node from which it has received the request, stores a copy of the file locally and updates its routing table into optimize routing for future requests. In case the node that is contacted is not responding, the node sends the request to the node with the second-nearest key and if that node is also unresponsive, it contacts the third-nearest one, and so on. If the file is not retrieved within the hops-to-live number of hops, then the search is aborted and the file requester is informed. The nodes that are forwarding the requested file back to the file requester change randomly the sender address, providing reasonable deniability for the node that has stored the file (Clarke et al. 2001).

The Opennet mode was vulnerable to various attacks. In particular, nodes participating in Freenet were not protected, and an attacker could easily find out whether a router is a participating Freenet node. In 2010, Freenet has been extended by a membership-concealing Darknet mode that uses trusted connections for routing (Clarke et al. 2010), where such shortcomings are addressed. In the Darknet mode, the routing table is consisting of nodes derived from a fixed graph, which is the social graph of the node and the user chooses the nodes from her trusted nodes (Clarke et al. 2010). Since the Darknet mode is based on the trusted network of a user, the structure of the network is following Kleinberg's small-world model (Kleinberg 2000).

GNUnet (Bennett et al. 2002) was originally designed as a peer-to-peer censorship-resistant content sharing system, but has been expanded into other applications such as anonymous file-sharing using the *GAP* protocol (Bennett and Grothoff 2003). GAP aims at providing requester and responder anonymity for file search and file-sharing. In GAP, a node that is relaying a message in the forward route has the option to "drop out" from the reply route (for example, due to network state and its own heavy load) and when the reply is sent back, the node is over-jumped. Moreover, when queries arrive at the nodes, they can be dropped if the node has already too much load. Routing in GAP uses a credit rating scheme, where relaying requests and replies increase the credit and sending uses the credit. The credit score is local at each node. In GAP, the file request can either be sent to newly selected nodes or to a node where there is already a connection established. This is decided based on the node's current CPU and load, the credit rating and a random factor. The

Table 5. Routing Classification of Anonymous Communication Protocols: DCnets and Miscellaneous Protocols

		Network Structure						**Routing Info.**		**Communication Model**					**Performance and Deployability**				
			Connection Type		Symmetry							Node Selection							
		Topology	Direction	Synchronization	Roles	Hierarchy	Decentralization	Network view	Updating	Routing type	Scheduling	Determinism	Selection set	Selection probability	Latency	Communication mode	Implementation	Code availability	Context/application
DCnets	**Chaum's DCnet** **Revisited DCnet** (Chaum 1988) (Waidner and Pfitzmann 1990) (Golle and Juels 2004)	⊠	→	≠	●··●··●	···	✗	●	ϟ	📢	≡	✓	Ⓐ	◎	H	🖂	✗	✗	✉
	Herbivore (Goel et al. 2003)	⊏	→	≠	●··●··●	✤	⊙	◐	ϟ	📢	≡	✓	🌍	◎	M	🖂	✓	✗	✉
	Dissent (Corrigan-Gibbs and Ford 2010)	⊠	→	≠	●··●··●	···	⊙	●	ϟ	📢	≡	✓	Ⓐ	◎	H	🖂	✓	✓	✉
	Dissent in Numbers (Wolinsky et al. 2012a) (Wolinsky et al. 2012b)	⊏	→	≠	●··●	✤	⊙	◐	ϟ	📢	≡	✓	🌍	◎	H	🖂	✓	✓	✉
Misc.	**I2P** (Schimmer 2009)	□	→	≠	●··●··●	···	○	●	🕒	●···	⟐	✗	🌍🛑	❋	L	●–●	✓	✓	e @ 📄
	P5 (Sherwood et al. 2002)	⊏	→	≠	●··●··●	✤	⊙	◐	ϟ	📢	≡	✓	☺	◎	H	🖂	✓	✗	✉
	CAR (Shokri et al. 2007)	⊠	↔	≠	●··●··●	···	✗	◐	ϟ	📢	≡	✓	🌍	◎	M	●–●	✓	✗	📶

node selection is random with a bias towards nodes, which have a closer identifier to the hash value of the file that is queried. Moreover, the network activity is also taken into account in node selection (giving preference to "hot paths"). GAP uses a *time-to-live* restriction to avoid routing loops and when time-to-live is reached, the query is forwarded directly to the destination with a certain probability. For flushing in GAP, nodes use a combination of time and threshold mixes for flushing batches of messages, where the time restriction is selected randomly.

3.3.2 Discussion. Crowds and Morphmix are two of the early Random Walk protocols that were proposed for anonymous communication. However, they present conceptual differences in terms of routing features. Both *Crowds* and *Morphmix* have fully connected topologies, since every node may build a connection with every other node, resulting in better availability of the system, which leads to a bigger attack surface for timing attacks. *Tarzan* originally had a partially connected topology that was due to its partial network view of the route initiator. However, in the later version of Tarzan, a gossip-based strategy has been proposed to have a complete view for the route initiator, which leads to a fully connected topology as marked in Table 5.

The path length in Crowds may vary and is determined in a non-deterministic manner to make simple timing attacks harder for external, local, and passive adversaries. Still, this does not necessarily hold for the case that at least one of the nodes in the path is malicious. In Morphmix, the initiator does not select the nodes of the route itself, rather decides on the number of nodes and establishes the connection.

Crowds is semi-decentralized, because routing information of nodes is distributed by a central entity (the blender), which introduces a single point of failure with respect to node administration. Morphmix, however, has a fully decentralized structure. The network view is complete in Crowds, which, on the one hand, protects Crowds from eclipse attacks and, on the other hand, is important, since Crowds has a hop-by-hop routing type that makes the node selection sensitive to be biased by adversaries. In Morphmix, the network view is partial, and therefore, witnesses were introduced to prevent the biased node selection. Moreover, an inherent feature of Random Walk protocols is that the node selection is non-deterministic. In Crowds, each node is chosen from the set of all nodes based on a geometric distribution (Danezis et al. 2009); whereas, in Morphmix, the initiator knows a subset of nodes.

An inherent routing feature of DHT-based protocols is partially connected topology and a partial network view. The routing information is distributed among nodes and no single node has the complete list. Such a design increases the scalability of the protocols. A partially connected network topology makes DHT-based protocols less resilient against DoS attacks, which aim at disconnecting the network as much as possible compared to Tor-related protocols. The connection direction is bidirectional for the majority of protocols with two exceptions. The exceptions are the file-sharing applications Gnunet and Freenet Opennet mode.

Generally, DHT-based protocols are fully peer-to-peer protocols. There are two exceptions in this category, namely, Torsk and Salsa, where the first one has a hybrid role structure while the latter one allows both hybrid and fully peer-to-peer role structures. For being partially connected, DHT-based protocols provide a partial view of the network to the routing decision maker. Note that this may introduce a series of attacks. Examples of attacks against protocols that provide only a partial view of the network to the routing decision maker are route fingerprinting attacks (Danezis and Clayton 2006), and route bridging attacks (Danezis and Syverson 2008). Another series of attacks, which might be possible due to a partial network view, are attacks that aim at disconnecting target nodes from the rest of the network, such as eclipse attacks (Castro et al. 2002).

Most of the DHT-based protocols are characterized by a hop-by-hop routing type. Exceptions are NISAN, Salsa, and Octopus, with source-routing. In Octopus, there are two decision makers for node selection; the path initiator who decides only about a segment of the path and the last node of that segment, which initiates the rest of the path. In our study, we could not find much information about the scheduling of DHT-based protocols, in particular for protocols that have not been deployed. Most of the DHT-based protocols have non-deterministic node selection, where again, exceptions are the file-sharing applications, where the routing path does not need to be anonymous.

The set selection for DHT-based protocols is, in most cases, all nodes within the routing table (i.e., all nodes available to the decision maker). However, there are two exceptions: Torsk, where the set selection is restricted by security and network restrictions, and Freenet in the Darknet mode, where the set selection is based on trust assumptions of the user. For most of DHT-based protocols, the selection probability is uniform, exceptions are Freenet and Gnunet. Both protocols do not aim at achieving unlinkability (Pfitzmann and Köhntopp 2000) nor they hide that a user is participating in the network. Nevertheless, they hide the role of the peer in the network. Most of the DHT-based protocols are message-based except Torsk, AP3, and Salsa.

3.4 DCnets

The idea of DCnets was first proposed by Chaum (1988) and later revisited (Golle and Juels 2004; Waidner and Pfitzmann 1990). As reflected by its name, DCnet is inspired by a scenario in which three cryptographers sit together for dinner in a restaurant and are interested in identifying whether the dinner is paid by the National Security Agent (NSA) or one of them. Each cryptographer flips a coin and shares the outcome with the cryptographer on her right. Next, each cryptographer tells to all whether the two coins she can see are the same or different. The cryptographer who pays the bill states the opposite. One can count the number of differences: an odd number of differences indicates that a cryptographer has paid, while an even number shows that the NSA has paid. If it is a cryptographer, then the other two do not learn who has paid, unless they collude together (see Figure 5).

The key concept is that every participant outputs a message that is disguised by XORing them with the keys the participants are sharing pairwise with other participants. The participants

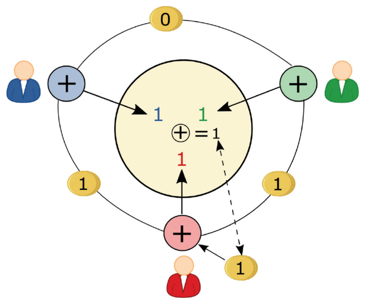

Fig. 5. The concept of Dining Cryptographers Networks: Each cryptographer flips a coin and shares the outcome with the cryptographer on her right. Next, each cryptographer tells whether the two coins she sees are the same or different. The cryptographer who pays the bill states the opposite. An odd number of differences indicates that a cryptographer has paid, while an even number shows that the NSA has paid.

combine their outputs and share the output with each other (i.e., they broadcast their output). When the encrypted messages are combined, the keys cancel each other out, and the message is revealed; however, the sender remains unknown.

DCnets are an important alternative to mix-based schemes and their extensions due to their resistance against traffic analysis attacks. DCnets offer non-interactive anonymous communication using secure multi-party computation with information-theoretically secure anonymity, guaranteeing sender anonymity while enabling all participants to verify the final outcome.

The DCnet concept can be generalized, to transmit large messages simply by repeating the protocol as desired (Golle and Juels 2004). DCnet expects all participants to be involved in every run of the protocol and requires pairwise shared keys between the participants. Moreover, every participant needs to disclose the same number of bits in each round. The participants can share the keys for every round, or they can repeatedly use the same key; this makes DCnet unconditionally or computationally secure, under the assumption that the protocol is executed correctly. Moreover, DCnets also have practical challenges, such as the message transmission or avoiding collisions (unintentional) and disruptions (intentional collisions). Since a collision invalidates the message (bit), when only one-bit messages are sent, just one of the participants may transmit at a time (although all participants are involved in each round). If multiple participants want to send messages within a block of communication, then they need to occupy different positions within the block. One proposed solution is to randomly pick a position (slot) in the block that is going to transmit and reserve the position in earlier rounds (pre-transmission round). However, this might only shift problem and again in the reservation round collisions might occur. The basic DCnet does not prevent any disruption, such as actively blocking participants from sending the message; hence, it is susceptible to anonymous DoS attacks. To partially address this problem, some solutions to detect disrupters in DCnets have been proposed in the literature (Bos and den Boer 1990; Waidner 1990). Furthermore, recovering from a fault is only possible by re-broadcasting the messages.

Chaum proposed to either use a ring topology for sharing the messages or use broadcast to transmit messages to all participants at once. The ring topology solution has a problem of detecting the disruptions, because malicious participants can adapt their answers to avoid being detected. Basically, if two users submit reverse bits, they cancel each other out and the disruptions remain undetected. Other topologies that have been proposed for DCnets are tree (Dolev and Ostrobsky 2000) or star topologies (Pfitzmann and Waidner 1986). The broadcast solution has the problem of being expensive and introduces the problem of collision. The major limitations of DCnet are the strong assumptions that they require: first, participants follow the protocol honestly and are expected not to collude; second, unconditional sender anonymity is guaranteed only if there is an unconditional secure channel between every pair of participants. Furthermore, DCnets are vulnerable to Sybil attacks (Douceur 2002).

Herbivore (Goel et al. 2003) is built on top of DCnets aiming at better efficiency and scalability and managing churn. To improve scalability, Herbivore breaks down the participants into smaller groups called *cliques*, a message can only be traced to a clique but not to the corresponding sender/receiver within their clique. Within a clique, participants are organized in a star topology, where the central node relays all messages between members of a clique. The central node is changed for each new round of communication. For inter-clique communication, the cliques are connected to each other in a ring topology. For locating cliques, Herbivore employs the Chord protocol (Stoica et al. 2001). To mitigate intersection attacks, nodes departure from a clique can be vetoed by the node that is in the middle of a long-run transmission. Although authors claim Herbivore can is a low-latency AC system, we decided to classify the protocols as being *mid-latency*, since it contains a central node that has to wait for messages from all other nodes in the clique. One of the main weaknesses of Herbivore is that smaller anonymity sets are achieved and the applications have a time restriction based on the cliques lifetime. Moreover, the star topology makes the design vulnerable to DoS attacks.

Dissent (Dinning-cryptographers Shuffled-Send Network) (Corrigan-Gibbs and Ford 2010) is a latency-tolerant protocol for anonymous communication. It is the first protocol that provides accountability for a small-size group, and also maintains integrity. Dissent is built on top of DCnets, but relaxes the aforementioned assumption that all participants follow the protocol correctly. Although Dissent, in fact, combines DCnets and Mixnets, we categorized it as DCnet, because its DCnet features are more essential to its design. In Dissent, anonymous communication is guaranteed for members of a group. Apart from the multi-party computation and layered encryption to hide the sender of the messages, to solve the collision problem, each group member influences the position of the messages of other group members in the final transmission block. Dissent consists of two sub-protocols: a *shuffle protocol* and a *bulk protocol.* In the bulk protocol, each member creates an assignment table for each of the other member, so-call *message descriptors.* The shuffle protocol is used to shuffle these messages descriptors. Based on these message descriptors, each participant inserts its messages to a cipher stream, which is a slice of the message block that needs to be transmitted. The shuffle protocol functions similar to mix cascades; each participant receives the set of message descriptors encrypted in a layered fashion, which she shuffles and passes over to the next participant. Thereafter, each member transmits one cipher stream. When these cipher streams are combined, a vector of concatenated messages is obtained. Dissent uses broadcasting for intermediate runs of its protocols such as sharing keys. However, the final cipher streams are not necessarily broadcasted, and can be sent to a single group member or non-member node. Hence, Dissent primarily guarantees only sender anonymity and further protocol setup details determine whether recipient anonymity is also achieved. To mitigate untraceable DoS attacks (disruptions), go/no-go messages and blame phases are used in Dissent, which identify collisions and malicious participants and enables accountability.

Wolinsky et al. have extended Dissent to improve scalability and efficiency (Wolinsky et al. 2012a). They propose to group participants and use designated servers, where the group members share keys with these servers instead of each other (the network consists of server nodes and participant nodes). In the basic version of Dissent, the group size was restricted; however, in the extended version, the participants may form larger groups, though the servers consist of a significantly smaller group, while still being not completely centralized to avoid the single point of failure. Hence, the extended Dissent builds an asymmetric topology for key sharing. At least one of the servers needs to be honest to prevent compromises. While latency introduced at the shuffle protocol made the basic version of Dissent unsuitable for interactive and low-latency applications, the extended Dissent, if used in a local-area setting, can be suitable for low-latency communication.

3.4.1 Discussion. DCnet protocols, as classified in Table 5, have some general inherent routing features that are due to the broadcast nature of their communication. These inherent routing features include unidirectional connection, asynchronous connection, and network structure involving centralized entities. Moreover, the routing type is source-routing with deterministic node selection and statically weighted selection probability.

Furthermore, DCnet protocols incur high-latency and have message-based communication models. In DCnets, we regard avoiding collusion and shuffling as routing relevant aspects of these protocols. The first designs of DCnets (Chaum 1988; Waidner and Pfitzmann 1990; Golle and Juels 2004) and Dissent (Corrigan-Gibbs and Ford 2010) are the direct realization of the original DCnet; therefore, they are similarly characterized. Inherent characteristics of such protocols are fully connected network structures, having a fully peer-to-peer role model. They support flat topologies, selecting all nodes for the selection set and offer a uniform selection probability for node selection.

To improve efficiency and performance, some DCnet protocols (Wolinsky et al. 2012a, 2012b; Goel et al. 2003) have been proposed, which vary in their routing features. Unlike the first group, in these protocols, the network structure is partially connected. For example, in Herbivore, participants are organized in star topologies, which are then connected in a ring topology. The organization of the nodes yields a hierarchical structure for the second group of DCnet protocols. Moreover, in the extended version of Dissent, users do not share keys with each other but rather with designated servers. Furthermore, the new versions of DCnet protocols enforce network restrictions to the selection set to increase efficiency and performance.

3.5 Miscellaneous Protocols

In this section, we review *I2P* that is alternative source-routed solutions to Tor and two broadcasting protocols that are alternative broadcasting solutions.

I2P (Timpanaro et al. 2012) is a distributed overlay network, originally aimed at enabling anonymous communication between two nodes within the I2P network. Note that currently there is a service built on top of I2P to allow getting connected to web servers (Timpanaro et al. 2011). Currently, the number of I2P routers is estimated to be between 40,000 and 50,000 (Project 2016b).

The network metadata (containing router contact information and destination contact information) is distributed among a subset of all nodes, so-called *floodfill* nodes, and is managed using DHT structure by employing Kademlia for node lookups. At bootstrapping, users obtain a list of I2P peers from websites and then contact two floodfill routers from the list and requests router information that is available to that floodfill node. In order to mitigate that malicious floodfill nodes are not biasing node selection by providing manipulated router information, router information is stored at eight floodfill nodes (Egger et al. 2013).

Nodes are categorized into tiers (called *peer profiling*) based on the previous performance (response times) and reliability (uptime) of nodes. Three main types of tiers are defined in I2P: high capacity, fast, and standard. The routing protocol of I2P, so-called *garlic routing*, is source-routed with a randomized node selection biased towards faster nodes (Schimmer 2009).

In I2P, communication channels are unidirectional and called tunnels; tunnels for outgoing traffic are called *outbound* and tunnels for incoming traffic are called *inbound*. Each user maintains a number of inbound and outbound tunnels; outbound/inbound tunnels of other users can be retrieved from the floodfill nodes. When users want to relay communication to each other, the nodes in the chosen inbound and outbound tunnels shape the relaying route. Moreover, there are two types of tunnels in I2P—client tunnels and exploratory tunnels—for which different peer selection strategies are used. Client tunnels are used for application traffic, and exploratory tunnels are used to send administrative information. For client tunnels, peers are selected randomly from the nodes that are categorized as fast-tier nodes, which is done locally by the client using previous measurements. For exploratory tunnels, peers are selected randomly from the set of nodes that are categorized as a standard tier. The communication through I2P is protected using *garlic encryption*, which is very similar to onion encryption with the difference that multiple data messages may be contained in a single *garlic message*. P5 (Sherwood et al. 2002) is an anonymous communication system where users are divided into subgroups correlated to P5 nodes. The P5 routing scheme uses a logical tree structure to broadcast messages to other nodes. If a user has joined one of the nodes higher in the hierarchy, then it receives communication more efficiently; however, this comes at cost of reduced anonymity. Another example of AC networks that use the broadcasting routing type is Chain-based Anonymous Routing (CAR) protocol designed by Shokri et al. (2007), which is an anonymous communication solution for on-demand ad hoc settings. CAR uses flooding for route discovery; however, once the route is constructed the data is transferred only through this route and is not broadcasted. Therefore, in CAR only the route discovery is using the broadcasting routing type, data forwarding is using source-routing. Another example of AC protocols for ad hoc settings that uses broadcasting is TEAP (Gunasekaran and Premalatha 2013).

3.5.1 Discussion. Connectivity of *I2P* is similar to Tor-related protocols due to the similarities for the node selection. I2P is characterized with unidirectional connection, which reduces the timing data that a single relay can have. However, multiple relays participate in the communication between a sender and receiver. The routing information of I2P is managed in a DHT-like fashion. Each database node (floodfill peer) has a slice of the information (Schimmer 2009), which could enable adversaries to carry out eclipse attacks targeting floodfill nodes (Egger et al. 2013). Since a user obtains node information from more than one floodfill node (up to eight), the union of this information might cover most of the I2P network and give the decision maker an almost complete view. I2P uses a source-routing approach, allowing the users to choose nodes that are faster. The selection probability in I2P is non-deterministic with a bias towards nodes that are profiled as fast responding nodes. Response times of these nodes differ among users; hence, timing attacks are more difficult to mount compared to Tor, where the node selection is biased using publicly known information (Project 2016a). Since response times are continuously measured, we have marked the selection probability with a bias based on dynamic restrictions. At the node level, I2P nodes use a prioritized scheduling mechanism, where each task has "bid," and the task with the lowest (best) bid is served first (Project 2016c).

P5 protocol uses broadcasting using logical hierarchies to improve efficiency and scalability. Compared to DCnets, they also have more flexibility for adding new nodes and changing the efficiency/anonymity trade-off after the initial setup has taken place. The *CAR* protocol uses broadcasting over several hops, while DCnets use a 1-hop broadcasting. This allows CAR nodes and users to have a partial view of the system and the participants of the protocol, which makes CAR more scalable than DCnets. However, DCnets have the advantage compared to CAR that they guarantee information-theoretically secure anonymity, while CAR's anonymity relies on the security of cryptographic primitives.

4 DISCUSSION

In this section, we discuss challenges and trade-offs between the investigated classes of AC protocols with respect to their routing characteristics and the relevance of their routing characteristics for real-world applications.

Mixnets are designed to be secure against traffic analysis and global adversaries by aggregating messages into batches; however, this comes with a latency overhead and makes them more appropriate for latency-tolerant applications, such as micro-blogging, email, questionnaires, collecting statistics, and electronic voting.

Since Mixnets are secure against a global adversary the system can consist of fewer nodes than systems that are vulnerable against a local adversary. However, like any other system, mix cascades are still vulnerable to collusion of mixes; therefore, the number of mixes needs to be appropriately large, which adds to the latency. Mixnets are also vulnerable to flooding attacks (Serjantov et al. 2003), and there needs to be a large amount of traffic entering the system to make this attack infeasible for the adversary. Mix cascades are not resilient to DoS attacks, and changing their initial setup is difficult. In mix cascades that are using layered encryption, if a mix fails to operate, all encrypted messages on the line are lost. The encryption layers of the encrypted messages need to be changed according to the new setup of the cascade and re-sent to the new mix cascade. Also, in decryption mix cascades messages need to add layers of encryption depending on the length of the cascade, which can lead to a large increase in message size. On the other hand, if a mix cascade is using re-encryption model instead, it is easier to recover from mix failure and messages have fixed sized regardless of the length of the cascade. In both types of mix cascades, since the users are not selecting the mixes, they do not need to maintain a complete view of the system, but although low overhead for users is a key factor in terms of scalability, mix cascades are not scalable, because their bandwidth can only be expanded vertically. This is because every mix needs to relay all the messages and therefore adding a mix to the mix cascade would not expand the bandwidth but improve anonymity by distributing trust one mix further. If a mix cascade is using varying sizes of mixes, then the bandwidth of the cascade is limited to the bandwidth of its smallest mix. Hence, increasing bandwidth can only be achieved by increasing the size of all nodes in the mix cascade. Expanding the bandwidth by adding a new mix cascade leads to splitting up the anonymity set size among the mix cascades and the increase in traffic would have no impact on achieving stronger anonymity. Free-route mix networks scale better than mix cascades in terms of bandwidth expansion, which helps with increasing the anonymity set size. They are more resilient against DoS attacks and increase the attack surface, which makes attacks more expensive for the adversary. However, the increased traffic needs to be load-balanced, which is a difficult task.

Tor comprises a large number of volunteer nodes including a large number of high-bandwidth nodes among them. This leads to a network with many high-bandwidth nodes that are highly connected with each other (Brinkmeier et al. 2009), which makes Tor resilient against DoS attacks (Shirazi et al. 2015), fast, and suitable for low-latency applications, such as web browsing or instant messaging. Both Mixnets and Tor are source-routed, which makes users responsible for path selection. Although this makes sense to provide sender anonymity, it makes achieving receiver anonymity difficult. Tor achieves receiver anonymity, by so-called *hidden services* (Dingledine et al. 2004), where two source-routed paths are connected together at a pre-defined meeting point. However, when the meeting point and an additional node from the hidden services part of the route are controlled by the adversary, receiver anonymity can be broken (Biryukov et al. 2013). Moreover, if the meeting point is controlled by the adversary, anonymity is weakened (Lazar and Zeldovich 2016). The fact the hidden services use six nodes, compared to three nodes for usual Tor traffic, which already weakens their anonymity by separating them from the rest of Tor traffic (Kwon et al. 2015).

Moreover, since Tor uses source-routing almost a list of all Tor nodes need to be known to the routing decision maker, except for a list of nodes that are not publicly available for censorship purposes. The complete list of relays needs to be periodically updated, which makes it more expensive for users and can hurt scalability. Roos and Strufe have proved that efficient routing in dynamic overlay networks is not scalable, in particularly for a client-server model (Roos and Strufe 2015). Furthermore, Tor is designed to be secure only against local adversaries and it is vulnerable to traffic analysis attacks (Murdoch and Danezis 2005; Chakravarty et al. 2010; Nathan S. Evans 2009; Johnson et al. 2013; Mittal et al. 2011a; O'Gorman and Blott 2009), in particular, if the adversary can see both ends of the communication.

Peer-to-peer networks rely on Random Walk protocols and DHT protocols, which use hop-by-hop routing. Using hop-by-hop routing increases scalability, because the network view for users is incomplete and does not need to be maintained, which can be problematic if the network grows. In particular, peer-to-peer networks, where peers are based on friendship relationship, are the most promising type in terms of scalability (Roos 2016). Another advantage of hop-by-hop routing is that it facilitates load balancing, which is one of the challenges of source-routing protocols. One solution to this challenge would be providing real-time network data for the routing initiator, however, this solution would worsen scalability of source-routing protocols even further. Such systems are suitable, for instance, for anonymous file-sharing, where the nodes have to dedicate a considerable amount of resources. However, being fully peer-to-peer may affect the usability of the protocol, because users are required to participate as a relay, which increases the cost of using anonymous communication. Unfortunately, this might lead to a decrease in the number of users of such systems and in turn reduce anonymity (Dingledine and Mathewson 2006). The main drawback of using hop-by-hop routing is the malicious influence that compromises nodes can have on the path selection, leading to route-capture attacks (Danezis and Clayton 2006).

Classic DCnets provide information-theoretic anonymity; whereas, Mixnets, for example, only provide computational security. However, DCnets are often vulnerable against active adversaries, in particular, if the adversary is internal. Early designs of DCnets required a restricted setting, where all users or nodes need to be honest and were also not resilient against DoS attacks. Follow-up AC systems such as Dissent have incorporated detection techniques for malicious nodes into their protocol. While for DCnets, like for mix cascades, anonymity guarantees are given with the assumption that the adversary can track the users' communication route; for Tor-related protocols and Random Walk and DHT protocols, this breaks their anonymity guarantees that are based on a local adversary assumption. This is mainly because in DCnets and mix cascades the communication route of several users is common, hence, a user's communication is hidden among them. While in Tor-related protocols and Random Walk/DHT protocols the anonymity set size per route is one. Since the route does need to be kept secret for both mix cascades and DCnets, a significantly smaller number of nodes in the AC networks can be sufficient, because local adversaries are only reasonable if the system has many relays and a large user-base. Moreover, the broadcasting routing type, in comparison to source-routing and hop-by-hop routing, is more resilient against DoS attacks, and data can be sent over the shortest route (Shokri et al. 2007) without the need to transfer information about the network topology to the communication initiator. Although this comes at a cost of using bandwidth excessively. However, not all broadcasting AC systems are resilient, for example, if only one relay goes offline in Dissent the protocol halts completely and the composition of the relays cannot be changed after the initial setup (Lin et al. 2016). DCnets tend to have a large communication overhead and do not scale well, because each relay needs to process all messages like mix cascades their bandwidth can be only expanded vertically. Even the newer version of Dissent, which employs a client-server approach for better scalability, can only scale up to a few thousand clients (Wolinsky et al. 2012a). Therefore, DCnet protocols are more suitable for applications such as group communication/messaging or micro-blogging at a small scale.

Table 6 summarizes challenges that our four routing classes face in terms of routing and the adversary model (see Raymond (2001) for the definition of AC adversaries) that is assumed for anonymity.

Table 6. Overview of the Adversary Assumptions, Focus of Routing Feature, and Challenges That Our Four Routing Classes Face

Routing Class	**Challenges**	**Adversary Type**
Mixnet Protocols	Traffic analysis attacks, such as flooding attacks	Global & active
Tor-related Protocols	Traffic analysis attacks, such as timing attacks	Local & active
Random Walk/DHT Protocols	Partitioning attacks & biasing node selection	Local & active
DCnet Protocols	Collision and disruption	Global & passive

5 CONCLUDING REMARKS

In this work, we identified key anonymous routing characteristics and classified AC systems in groups according to their routing features. Moreover, we introduced and evaluated the main existing AC protocols under our classification. Furthermore, we discussed the relevance of routing characteristics in such networks and their influence on anonymity and security. We have drawn several lessons from conducting our survey. On the one hand, there are trade-offs between security, anonymity, scalability, and performance goals, because the routing decisions that support each of these goals often conflict with each other. This is especially true for achieving simultaneously strong anonymity and good performance, which is still an open problem. On the other hand, routing aspects are related to each other; for example, a partial view of the system (in the routing information) often supports the hop-by-hop routing. We observe that making certain routing decisions leads often to a trade-off between security, anonymity, scalability, and performance goals. Finally, our classification uncovers which routing decisions have to be tailored to the security, anonymity, scalability, and performance goals that are necessary for a specific use case of a given AC protocol.

REFERENCES

Masoud Akhoondi, Curtis Yu, and Harsha V. Madhyastha. 2014. LASTor: A low-latency AS-aware Tor client. *IEEE/ACM Trans. Netw.* 22, 6 (dec 2014), 1742–1755. DOI: http://dx.doi.org/10.1109/TNET.2013.2291242

Mashael AlSabah, Kevin Bauer, Ian Goldberg, Dirk Grunwald, Damon McCoy, Stefan Savage, and Geoffrey M. Voelker. 2011. DefenestraTor: Throwing out windows in Tor. In *Privacy Enhancing Technologies*, Simone Fischer-Hübner and Nicholas Hopper (Eds.). Lecture Notes in Computer Science, Vol. 6794. Springer, Berlin, 134–154.

Mashael AlSabah, Kevin S. Bauer, Tariq Elahi, and Ian Goldberg. 2013. The path less travelled: Overcoming orTor's bottlenecks with traffic splitting. In *Proceedings of the 13th International Symposium on Privacy Enhamcing Technologies (PETS'13)*. 143–163.

Mashael AlSabah, Kevin S. Bauer, and Ian Goldberg. 2012. Enhancing orTor's performance using real-time traffic classification. In *Proceedings of the ACM Conference on Computer and Communications Security (CCS'12)*. 73–84.

Mashael AlSabah and Ian Goldberg. 2013. PCTCP: Per-circuit TCP-over-IPsec transport for anonymous communication overlay networks. In *Proceedings of the 2013 ACM SIGSAC Conference on Computer and Communications Security (CCS'13)*. 349–360.

Mashael AlSabah and Ian Goldberg. 2015. Performance and Security Improvements for Tor: A Survey. Cryptology ePrint Archive, Report No. 2015/235.

M. Backes, A. Kate, P. Manoharan, S. Meiser, and E. Mohammadi. 2013. AnoA: A framework for analyzing anonymous communication protocols. In *Proceedings of the 26th IEEE Computer Security Foundations Symposium (CSF'13)*. 163–178.

Michael Backes, Aniket Kate, Sebastian Meiser, and Esfandiar Mohammadi. 2014. (Nothing else) MATor(s): Monitoring the anonymity of orTor's path selection. In *Proceedings of the 2014 ACM SIGSAC Conference on Computer and Communications Security (CCS'14)*. ACM, New York, NY, 513–524.

Armon Barton and Matthew Wright. 2016. DeNASA: Destination-naive as-awareness in anonymous communications. In *Proceedings of the 16th Privacy Enhancing Technologies Symposium (PETS'16)*.

Kevin Bauer, Damon McCoy, Dirk Grunwald, Tadayoshi Kohno, and Douglas Sicker. 2007. Low-resource routing attacks against Tor. In *Proceedings of the 2007 ACM Workshop on Privacy in Electronic Society (WPES'07)*. ACM, New York, NY, 11–20.

P. Bell and K. Jabbour. 1986. Review of point-to-point network routing algorithms. *Comm. Mag.* 24, 1 (Jan. 1986), 34–38. DOI:http://dx.doi.org/10.1109/MCOM.1986.1092937

Krista Bennett and Christian Grothoff. 2003. Gap—Practical anonymous networking. In *Privacy Enhancing Technologies*, Roger Dingledine (Ed.). Springer, Berlin, 141–160.

Krista Bennett, Tiberius Stef, Christian Grothoff, Tzvetan Horozov, and Ioana Patrascu. 2002. The GNet whitepaper. Technical report, Purdue University, 21 pages.

Oliver Berthold, Hannes Federrath, and Marit Köhntopp. 2000a. Project anonymity and unobservability in the internet. In *Proceedings of the 10th Conference on Computers, Freedom and Privacy: Challenging the Assumptions (CFP'00)*. ACM, New York, NY, 57–65.

Oliver Berthold, Hannes Federrath, and Stefan Köpsell. 2000b. Web MIXes: A system for anonymous and unobservable internet access. In *Proceedings of the International Workshop on Design Issues in Anonymity and Unobservability: Designing Privacy Enhancing Technologies*. 115–129.

A. Biryukov, I. Pustogarov, and R. Weinmann. 2013. Trawling for Tor hidden services: Detection, measurement, deanonymization. In *Proceedings of the IEEE Symposium on Security and Privacy (SP'13)*. 80–94.

Rainer Böhme, George Danezis, Claudia Díaz, Stefan Köpsell, and Andreas Pfitzmann. 2005. On the PET workshop panel mix cascades versus peer-to-peer: Is one concept superior? In *Privacy Enhancing Technologies*, David Martin and Andrei Serjantov (Eds.). Lecture Notes in Computer Science, Vol. 3424. Springer, Berlin, 243–255.

Jurjen Bos and Bert den Boer. 1990. Detection of disrupters in the DC protocol. In *Advances in Cryptology—EUROCRYPT '89*, Jean-Jacques Quisquater and Joos Vandewalle (Eds.). Lecture Notes in Computer Science, Vol. 434. Springer, Berlin, 320–327.

Michael Brinkmeier, Mathias Fischer, Sascha Grau, Günter Schäfer, and Thorsten Strufe. 2009. Methods for improving resilience in communication networks and P2P overlays. *Praxis der Informationsverarbeitung und Kommunikation* 32, 1 (2009), 64–78. DOI:http://dx.doi.org/10.1515/piko.2009.0013

Miguel Castro, Peter Druschel, Ayalvadi Ganesh, Antony Rowstron, and Dan S. Wallach. 2002. Secure routing for structured peer-to-peer overlay networks. *SIGOPS Oper. Syst. Rev.* 36, SI (Dec 2002), 299–314.

Sambuddho Chakravarty, Angelos Stavrou, and Angelos D. Keromytis. 2010. Traffic analysis against low-latency anonymity networks using available bandwidth estimation. In *Computer Security—ESORICS 2010*, Dimitris Gritzalis, Bart Preneel, and Marianthi Theoharidou (Eds.). Lecture Notes in Computer Science, Vol. 6345. Springer, Berlin, 249–267.

David Chaum. 1981. Untraceable electronic mail, return addresses, and digital pseudonyms. *Commun. ACM* 24, 2 (1981), 84–88.

David Chaum. 1988. The dining cryptographers problem: Unconditional sender and recipient untraceability. *J. Cryptol.* 1, 1 (1988), 65–75.

Ian Clarke, Oskar Sandberg, Matthew Toseland, and Vilhelm Verendel. 2010. Private communication through a network of trusted connections: The dark freenet. *Network* (2010).

Ian Clarke, Oskar Sandberg, Brandon Wiley, and Theodore W. Hong. 2001. Freenet: A distributed anonymous information storage and retrieval system. In *Proceedings of the International Workshop on Designing Privacy Enhancing Technologies: Design Issues in Anonymity and Unobservability*. Springer-Verlag, New York, 46–66.

Bernd Conrad and Fatemeh Shirazi. 2014. A survey on Tor and I2P. In *Proceedings of the 9th International Conference on Internet Monitoring and Protection (ICIMP'14)*. 22–28.

Henry Corrigan-Gibbs and Bryan Ford. 2010. Dissent: Accountable anonymous group messaging. In *Proceedings of the 17th ACM Conference on Computer and Communications Security (CCS'10)*. 340–350.

George Danezis. 2003a. Mix-networks with restricted routes. In *Proceedings of the 3rd International Workshop on Privacy Enhancing Technologies (PET'03)*. 1–17.

George Danezis. 2003b. Statistical disclosure attacks. In *Proceedings of the 18th International Conference on Information Security: Security and Privacy in the Age of Uncertainty (SEC'03)*. 421–426.

George Danezis and Richard Clayton. 2006. Route fingerprinting in anonymous communications. In *Proceedings of the 6th IEEE International Conference on Peer-to-Peer Computing (P2P'06)*. IEEE, 69–72.

George Danezis and Claudia Díaz. 2008. *A Survey of Anonymous Communication Channels*. Technical Report. Microsoft Research.

George Danezis, Claudia Diaz, Emilia Ksper, and Carmela Troncoso. 2009. The wisdom of crowds: Attacks and optimal constructions. In *Computer Security—ESORICS 2009*, Michael Backes and Peng Ning (Eds.). Lecture Notes in Computer Science, Vol. 5789. Springer, Berlin, 406–423.

George Danezis, Claudia Diaz, and Paul F. Syverson. 2010. Systems for anonymous communication. In *CRC Handbook of Financial Cryptography and Security*, B. Rosenberg and D. Stinson (Eds.). Chapman & Hall, 341–390.

George Danezis, Roger Dingledine, and Nick Mathewson. 2003. Mixminion: Design of a type III anonymous remailer protocol. In *Proceedings of the 2003 IEEE Symposium on Security and Privacy (SP'03)*. 2–15.

George Danezis and Paul Syverson. 2008. Bridging and fingerprinting: Epistemic attacks on route selection. In *Proceedings of the 8th International Symposium on Privacy Enhancing Technologies (PETS'08)*. Springer-Verlag, Berlin, 151–166.

Claudia Diaz, Steven J. Murdoch, and Carmela Troncoso. 2010. Impact of network topology on anonymity and overhead in low-latency anonymity networks. In *Proceedings of the 10th International Conference on Privacy Enhancing Technologies (PETS'10)*. Springer-Verlag, Berlin, 184–201.

Claudia Díaz and Bart Preneel. 2004. Taxonomy of mixes and dummy traffic. In *Proceedings of the 18th World Computer Congress on Information Security Management, Education and Privacy (IFIP'04), and the TC11 19th International Information Security Workshops*. 215–230.

Claudia Díaz and Andrei Serjantov. 2003. Generalising mixes. In *Privacy Enhancing Technologies*, Roger Dingledine (Ed.). Lecture Notes in Computer Science, Vol. 2760. Springer, Berlin, 18–31.

Roger Dingledine, Michael J. Freedman, David Hopwood, and David Molnar. 2001. A reputation system to increase MIX-Net reliability. In *Information Hiding*, IraS. Moskowitz (Ed.). Lecture Notes in Computer Science, Vol. 2137. Springer, Berlin, 126–141.

Roger Dingledine, Michael J. Freedman, and David Molnar. 2000. The free haven project: Distributed anonymous storage service. In *Designing Privacy Enhancing Technologies, International Workshop on Design Issues in Anonymity and Unobservability, Berkeley, CA, July 25-26, 2000*. 67–95.

Roger Dingledine, Nicholas Hopper, George Kadianakis, and Nick Mathewson. 2014. One fast guard for life (or 9 months). *Proceedings of the 7th Workshop on Hot Topics in Privacy Enhancing Technologies (HotPETs'14).*

Roger Dingledine and Nick Mathewson. 2006. Anonymity loves company: Usability and the network effect. In *Proceedings of the 5th Workshop on the Economics of Information Security (WEIS'06)*, Ross Anderson (Ed.). Cambridge, UK.

Roger Dingledine, Nick Mathewson, and Paul Syverson. 2004. Tor: The second-generation onion router. In *Proceedings of the 13th Conference on USENIX Security Symposium, Volume 13 (SSYM'04).* USENIX Association, 303–320.

Roger Dingledine and Steven J. Murdoch. 2009. *Performance Improvements on Tor or, Why Tor is slow and what we're going to do about it.* Technical Report. The Tor Project. Retrieved from https://research.torproject.org/techreports/performance-2009-11-09.pdf.

Roger Dingledine and Paul Syverson. 2002. Reliable MIX cascade networks through reputation. In *Financial Cryptography*, Matt Blaze (Ed.). Lecture Notes in Computer Science, Vol. 2357. Springer, Berlin, 253–268.

Shlomi Dolev and Rafail Ostrobsky. 2000. Xor-trees for efficient anonymous multicast and reception. *ACM Trans. Info. Syst. Secur.* 3, 2 (may 2000), 63–84.

John R. Douceur. 2002. The sybil attack. In *Peer-to-Peer Systems*, Peter Druschel, Frans Kaashoek, and Antony Rowstron (Eds.). Lecture Notes in Computer Science, Vol. 2429. Springer, Berlin, 251–260.

Matthew Edman and Paul Syverson. 2009. As-awareness in Tor path selection. In *Proceedings of the 16th ACM Conference on Computer and Communications Security (CCS'09).* ACM, New York, NY, 380–389.

Matthew Edman and Bülent Yener. 2009. On anonymity in an electronic society: A survey of anonymous communication systems. *ACM Comput. Surveys (CSUR)* 42, 1, Article 5 (December 2009), 35 pages.

Christoph Egger, Johannes Schlumberger, Christopher Kruegel, and Giovanni Vigna. 2013. Practical attacks against the I2P network. In *Proceedings of the 16th International Symposium on Research in Attacks, Intrusions and Defenses (RAID'13).*

E. Erdin, C. Zachor, and M. H. Gunes. 2015. How to find hidden users: A survey of attacks on anonymity networks. *IEEE Commun. Surveys Tutor.* PP, 99 (2015), 1–1.

Nick Feamster and Roger Dingledine. 2004. Location diversity in anonymity networks. In *Proceedings of the 2004 ACM Workshop on Privacy in the Electronic Society (WPES'04).* ACM, New York, NY, 66–76.

Laura Marie Feeney. 1999. A taxonomy for routing protocols in mobile ad hoc networks. SICS Report, Technical Report, ISRN:SICS T-99/07 SE, 20 pages.

Michael J. Freedman and Robert Morris. 2002. Tarzan: A peer-to-peer anonymizing network layer. In *Proceedings of the 9th ACM Conference on Computer and Communications Security (CCS'02).* ACM, 193–206.

Michael J. Freedman, Emil Sit, Josh Cates, and Robert Morris. 2002. Introducing Tarzan, a peer-to-peer anonymizing network layer. In *Proceedings of the 1st International Workshop on Peer-to-Peer Systems (IPTPS'02).* 121–129.

John Geddes, Rob Jansen, and Nicholas Hopper. 2014. IMUX: Managing Tor connections from two to infinity, and beyond. In *Proceedings of the 13th Workshop on Privacy in the Electronic Society (WPES'14).* 181–190.

Sharad Goel, Mark Robson, Milo Polte, and Emin Sirer. 2003. *Herbivore: A Scalable and Efficient Protocol for Anonymous Communication.* Technical Report. Cornell University.

David M. Goldschlag, Michael G. Reed, and Paul F. Syverson. 1996. Hiding routing information. In *Information Hiding (Lecture Notes in Computer Science)*, Ross Anderson (Ed.), Vol. 1174. Springer, Berlin, 137–150.

Philippe Golle and Ari Juels. 2004. Dining cryptographers revisited. In *Advances in Cryptology—EUROCRYPT'04*, Christian Cachin and JanL. Camenisch (Eds.). Lecture Notes in Computer Science, Vol. 3027. Springer, Berlin, 456–473.

Deepika Gopal and Nadia Heninger. 2012. Torchestra: Reducing interactive traffic delays over Tor. In *Proceedings of the 2012 ACM Workshop on Privacy in the Electronic Society (WPES'12)*. ACM, New York, NY, 31–42.

Christian Grothoff. 2003. An excess-based economic model for resource allocation in peer-to-peer networks. *Wirtschaftsinformatik* 3-2003. Retrieved from http://grothoff.org/christian/ebe.pdf.

Ceki Gülcü and Gene Tsudik. 1996. Mixing email with babel. In *Proceedings of the 1996 Symposium on Network and Distributed System Security (NDSS'96)*. 2–16.

Muthumanickam Gunasekaran and Kandhasamy Premalatha. 2013. TEAP: Trust-enhanced anonymous on-demand routing protocol for mobile ad hoc networks. *IET Info. Secur.* 7, 3 (Sept 2013), 203–211.

Zygmunt J. Haas, Joseph Y. Halpern, and Li Li. 2006. Gossip-based ad hoc routing. *IEEE/ACM Trans. Netw.* 14, 3 (Jun 2006), 479–491.

Mor Harchol-Balter, Frank Thomson Leighton, and Daniel Lewin. 1999. Resource discovery in distributed networks. In *Proceedings of the 18th Annual ACM Symposium on Principles of Distributed Computing (PODC'99)*. 229–237.

Hsu-Chun Hsiao, T. H.-J. Kim, A. Perrig, A. Yamada, S. C. Nelson, M. Gruteser, and Wei Meng. 2012. LAP: Lightweight anonymity and privacy. In *Proceedings of the IEEE Symposium on Security and Privacy (SP'12)*. 506–520.

Markus Jakobsson, Ari Juels, and Ronald L. Rivest. 2002. Making mix nets robust for electronic voting by randomized partial checking. In *Proceedings of the 11th USENIX Security Symposium*. 339–353.

Anja Jerichow, Jan Müller, Andreas Pfitzmann, Birgit Pfitzmann, and Michael Waidner. 1998. Real-time mixes: A bandwidth-efficient anonymity protocol. *IEEE J. Select. Areas Commun.* 16, 4 (1998), 495–509.

Aaron Johnson, Chris Wacek, Rob Jansen, Micah Sherr, and Paul F. Syverson. 2013. Users get routed: Traffic correlation on Tor by realistic adversaries. In *Proceedings of the ACM SIGSAC Conference on Computer and Communications Security (CCS'13)*. 337–348.

Dogan Kesdogan, Jan Egner, and Roland Büschkes. 1998. Stop-and-go-MIXes providing probabilistic anonymity in an open system. In *Proceedings of the 2nd International Workshop on Information Hiding*. 83–98.

Jon Kleinberg. 2000. The small-world phenomenon: An algorithmic perspective. In *Proceedings of the 32nd Annual ACM Symposium on Theory of Computing (STOC'00)*. ACM, 163–170.

Albert Kwon, Mashael AlSabah, David Lazar, Marc Dacier, and Srinivas Devadas. 2015. Circuit fingerprinting attacks: Passive deanonymization of Tor hidden services. In *Proceedings of the 24th USENIX Security Symposium (USENIXSecurity'15)*. USENIX Association, Washington, D.C., 287–302.

David Lazar and Nickolai Zeldovich. 2016. Alpenhorn: Bootstrapping secure communication without leaking metadata. In *Proceedings of the 12th USENIX Symposium on Operating Systems Design and Implementation (OSDI'16)*. USENIX Association, GA, 571–586.

Brian N. Levine, Michael K. Reiter, Chenxi Wang, and Matthew Wright. 2004. Timing attacks in low-latency mix systems. In *Financial Cryptography*, Ari Juels (Ed.). Lecture Notes in Computer Science, Vol. 3110. Springer, Berlin, 251–265.

Dong Lin, Micah Sherr, and Boon Thau Loo. 2016. Scalable and anonymous group communication with MTor. *Proceedings on Privacy Enhancing Technologies (PoPETS'16)*.

Petar Maymounkov and David Mazières. 2002. Kademlia: A peer-to-peer information system based on the XOR metric. In *Proceedings of the 1st International Workshop on Peer-to-Peer Systems (IPTPS'01)*. Springer-Verlag, London, 53–65.

Jon McLachlan, Andrew Tran, Nicholas Hopper, and Yongdae Kim. 2009. Scalable onion routing with torsk. In *Proceedings of the 16th ACM Conference on Computer and Communications Security (CCS'09)*. ACM, New York, NY, 590–599.

Alan Mislove, Gaurav Oberoi, Ansley Post, Charles Reis, Peter Druschel, and Dan S. Wallach. 2004. AP3: Cooperative, decentralized anonymous communication. In *Proceedings of the 11th ACM SIGOPS European Workshop*. 30.

Prateek Mittal, Ahmed Khurshid, Joshua Juen, Matthew Caesar, and Nikita Borisov. 2011a. Stealthy traffic analysis of low-latency anonymous communication using throughput fingerprinting. In *Proceedings of the 18th ACM Conference on Computer and Communications Security (CCS'11)*. ACM, New York, NY, 215–226.

Prateek Mittal, Femi Olumofin, Carmela Troncoso, Nikita Borisov, and Ian Goldberg. 2011b. PIR-Tor: Scalable anonymous communication using private information retrieval. In *Proceedings of the 20th USENIX Conference on Security (SEC'11)*. USENIX Association, Berkeley, CA, 31–31.

Ulf Möller, Lance Cottrell, Peter Palfrader, and Len Sassaman. 2003. Mixmaster protocol - version 3. IETF Internet Draft.

John Moy. 1998. OSPF: Anatomy of an Internet Routing Protocol. Addison-Wesley Longman Publishing Co., Inc. Boston, MA, USA.

S. J. Murdoch and G. Danezis. 2005. Low-cost traffic analysis of Tor. In *Proceedings of the IEEE Symposium on Security and Privacy*. 183–195.

Steven J. Murdoch and Piotr Zielinski. 2007. Sampled traffic analysis by internet-exchange-level adversaries. In *Proceedings of the 7th International Symposium on Privacy Enhancing Technologies (PET'07)*. 167–183.

Arjun Nambiar and Matthew Wright. 2006. Salsa: A structured approach to large-scale anonymity. In *Proceedings of the 13th ACM Conference on Computer and Communications Security (CCS'06)*. ACM, New York, NY, 17–26.

Christian Grothoff, Nathan S. Evans, and Roger Dingledine. 2009. A practical congestion attack on Tor using long paths. In *Proceedings of the 18th USENIX Security Symposium (USENIXSecurity'09)*. USENIX, Montreal, Canada.

Gavin O'Gorman and Stephen Blott. 2009. Improving stream correlation attacks on anonymous networks. In *Proceedings of the 2009 ACM Symposium on Applied Computing (SAC'09)*. 2024–2028.

Andriy Panchenko, Fabian Lanze, and Thomas Engel. 2012. Improving performance and anonymity in the Tor network. In *Proceedings of the 31st IEEE International Performance Computing and Communications Conference (IPCCC'12)*. 1–10.

Andriy Panchenko, Stefan Richter, and Arne Rache. 2009. NISAN: Network information service for anonymization networks. In *Proceedings of the 2009 ACM Conference on Computer and Communications Security (CCS09)*. 141–150.

Charles E. Perkins and Elizabeth M. Royer. 1997. Ad hoc on-demand distance vector routing. In *Proceedings of the 2nd IEEE Workshop on Mobile Computing Systems and Applications*. 90–100.

Andreas Pfitzmann and Marit Köhntopp. 2000. Anonymity, unobservability, and pseudonymity—A proposal for terminology. In *Proceedings of the International Workshop on Design Issues in Anonymity and Unobservability: Designing Privacy Enhancing Technologies*. 1–9.

Andreas Pfitzmann, Birgit Pfitzmann, and Michael Waidner. 1991. ISDN-mixes: Untraceable communication with very small bandwidth overhead. In *Kommunikation in Verteilten Systemen*. Informatik-Fachberichte, Vol. 267. Springer, Berlin, 451–463.

Andreas Pfitzmann and Michael Waidner. 1986. Networks without user observability—Design options. In *Advances in Cryptology—EUROCRYPT'85*, Franz Pichler (Ed.). Lecture Notes in Computer Science, Vol. 219. Springer, Berlin, 245–253.

I2P Project. 2016a. I2P Peer Profiling and Selection. Retrieved from https://geti2p.net/en/docs/how/peer-selection.

I2P Project. 2016b. I2P Statistics. Retrieved from http://stats.i2p.re/.

I2P Project. 2016c. I2P Transport Overview. Retrieved from https://geti2p.net/en/docs/transport.

Jean-François Raymond. 2000. Traffic analysis: Protocols, attacks, design issues, and open problems. In *Proceedings of the International Workshop on Design Issues in Anonymity and Unobservability: Designing Privacy Enhancing Technologies*. 10–29.

Jean-François Raymond. 2001. Traffic analysis: Protocols, attacks, design issues, and open problems. In *Designing Privacy Enhancing Technologies*, Hannes Federrath (Ed.). Lecture Notes in Computer Science, Vol. 2009. Springer, Berlin, 10–29.

M. G. Reed, P. F. Syverson, and D. M. Goldschlag. 1998. Anonymous connections and onion routing. *IEEE J. Select. Areas Commun.* 16, 4 (May 1998), 482–494.

Michael K. Reiter and Aviel D. Rubin. 1998. Crowds: Anonymity for web transactions. *ACM Trans. Info. Syst. Secur.* 1, 1 (1998), 66–92. DOI: http://dx.doi.org/10.1145/290163.290168

Jian Ren and Jie Wu. 2010. Survey on anonymous communications in computer networks. *Comput. Commun.* 33, 4 (2010), 420–431.

Marc Rennhard and Bernhard Plattner. 2002. Introducing morphmix: Peer-to-peer based anonymous internet usage with collusion detection. In *Proceedings of the 2002 ACM Workshop on Privacy in the Electronic Society (WPES'02)*. ACM, New York, NY, 91–102.

Marc Rennhard and Bernhard Plattner. 2004. Practical anonymity for the masses with morphmix. In *Financial Cryptography*, Ari Juels (Ed.). Lecture Notes in Computer Science, Vol. 3110. Springer, Berlin, 233–250.

Stefanie Roos. 2016. *Analyzing and Enhancing Routing Protocols for Friend-to-Friend Overlays*. Ph.D. Dissertation. Dissertation, Dresden, Technische Universität Dresden, 2016.

Stefanie Roos, Benjamin Schiller, Stefan Hacker, and Thorsten Strufe. 2014. Measuring freenet in the wild: Censorship-resilience under observation. In *Proceedings of the 14th International Symposium on Privacy Enhancing Technologies (PETS'14)*. 263–282.

Stefanie Roos and Thorsten Strufe. 2015. On the impossibility of efficient self-stabilization in virtual overlays with churn. In *Proceedings of the IEEE Conference on Computer Communications (INFOCOM'15)*. 298–306.

Antony I. T. Rowstron and Peter Druschel. 2001. Pastry: Scalable, decentralized object location, and routing for large-scale peer-to-peer systems. In *Proceedings of the IFIP/ACM International Conference on Distributed Systems Platforms (Middleware'01)*. Springer-Verlag, London, 329–350.

Kazue Sako and Joe Kilian. 1995. Receipt-free mix-type voting scheme—A practical solution to the implementation of a voting booth. In *Proceedings of the International Conference on the Theory and Application of Cryptographic Techniques: Advances in Cryptology—EUROCRYPT'95*. 393–403.

K. Sampigethaya and R. Poovendran. 2006. A survey on mix networks and their secure applications. *Proc. IEEE* 94, 12 (December 2006), 2142–2181.

Jody Sankey and Matthew Wright. 2014. Dovetail: Stronger anonymity in next-generation internet routing. In *Privacy Enhancing Technologies*, Emiliano De Cristofaro and StevenJ. Murdoch (Eds.). Lecture Notes in Computer Science, Vol. 8555. Springer International Publishing, 283–303.

Lars Schimmer. 2009. Peer profiling and selection in the I2P anonymous network. In *Proceedings of the Privacy Enhancing Techniques Convention (PET-CON'09)*. 59–70.

Andrei Serjantov. 2004. *On the Anonymity of Anonymity Systems.* Technical Report. University of Cambridge, Computer Laboratory.

Andrei Serjantov, Roger Dingledine, and Paul Syverson. 2003. From a trickle to a flood: Active attacks on several mix types. In *Information Hiding*, Fabien A. P. Petitcolas (Ed.). Lecture Notes in Computer Science, Vol. 2578. Springer, Berlin, 36–52.

Micah Sherr, Matt Blaze, and Boon Thau Loo. 2009. Scalable link-based relay selection for anonymous routing. In *Proceedings of the 9th International Symposium on Privacy Enhancing Technologies (PETS'09) (Lecture Notes in Computer Science)*, Ian Goldberg and Mikhail J. Atallah (Eds.), Vol. 5672. Springer, 73–93.

Rob Sherwood, Bobby Bhattacharjee, and Aravind Srinivasan. 2002. P5: A protocol for scalable anonymous communication. *J. Comput. Secur.* 13 (Dec. 2002), 839–876.

Fatemeh Shirazi, Claudia Diaz, and Joss Wright. 2015. Towards measuring resilience in anonymous communication networks. In *Proceedings of the 14th ACM Workshop on Privacy in the Electronic Society (WPES'15)*. ACM, New York, NY, 95–99. DOI: http://dx.doi.org/10.1145/2808138.2808152

R. Shokri, N. Yazdani, and A. Khonsari. 2007. Chain-based anonymous routing for wireless ad hoc networks. In *Proceedings of the 4th IEEE Consumer Communications and Networking Conference*. 297–302.

R. Snader and N. Borisov. 2011. Improving security and performance in the Tor network through tunable path selection. *IEEE Trans. Depend. Secure Comput.* 8, 5 (Sep. 2011), 728–741.

Ion Stoica, Robert Morris, David Karger, M. Frans Kaashoek, and Hari Balakrishnan. 2001. Chord: A scalable peer-to-peer lookup service for internet applications. In *Proceedings of the 2001 Conference on Applications, Technologies, Architectures, and Protocols for Computer Communications (SIGCOMM'01)*. ACM, 149–160.

Paul Syverson, Gene Tsudik, Michael Reed, and Carl Landwehr. 2001. Towards an analysis of onion routing security. In *Designing Privacy Enhancing Technologies*, Hannes Federrath (Ed.). Lecture Notes in Computer Science, Vol. 2009. Springer, Berlin, 96–114.

Can Tang and Ian Goldberg. 2010. An improved algorithm for Tor circuit scheduling. In *Proceedings of the 17th ACM Conference on Computer and Communications Security (CCS'10)*. ACM, New York, NY, 329–339.

The Tor Project. 2017. Tor Metrics. (2017). Retrieved from https://metrics.torproject.org/.

Juan Pablo Timpanaro, Isabelle Chrisment, and Olivier Festor. 2012. I2P's usage characterization. In *Traffic Monitoring and Analysis*, Antonio Pescapè, Luca Salgarelli, and Xenofontas Dimitropoulos (Eds.). Lecture Notes in Computer Science, Vol. 7189. Springer, Berlin, 48–51.

Juan Pablo Timpanaro, Chrisment Isabelle, and Festor Olivier. 2011. *Monitoring the I2P Network.* Technical Report. Retrieved from https://hal.inria.fr/hal-00653136.

Michael Waidner. 1990. Unconditional sender and recipient untraceability in spite of active attacks. In *Advances in Cryptology—EUROCRYPT '89*, Jean-Jacques Quisquater and Joos Vandewalle (Eds.). Lecture Notes in Computer Science, Vol. 434. Springer, Berlin, 302–319.

Michael Waidner and Birgit Pfitzmann. 1990. The dining cryptographers in the disco: Unconditional sender and recipient untraceability with computationally secure serviceability. In *Advances in Cryptology—EUROCRYPT'89*, Jean-Jacques Quisquater and Joos Vandewalle (Eds.). Lecture Notes in Computer Science, Vol. 434. Springer, Berlin, 690–690.

Marc Waldman and David Mazières. 2001. Tangler: A censorship-resistant publishing system based on document entanglements. In *Proceedings of the 8th ACM Conference on Computer and Communications Security (CCS'01)*. 126–135.

Marc Waldman, Aviel D. Rubin, and Lorrie Faith Cranor. 2000. Publius: A robust, tamper-evident, censorship-resistant, and source-anonymous web publishing system. In *Proceedings of the 9th USENIX Security Symposium.*

Peng Wang, Ivan Osipkov, Nicholas Hopper, and Yongdae Kim. 2006. *Myrmic: Provably Secure and Efficient DHT Routing.* Technical Report. DTC.

Qiyan Wang and Nikita Borisov. 2012. Octopus: A secure and anonymous DHT lookup. In *Proceedings of the IEEE 32nd International Conference on Distributed Computing Systems.* 325–334.

Tao Wang, Kevin Bauer, Clara Forero, and Ian Goldberg. 2012. Congestion-aware path selection for Tor. In *Financial Cryptography and Data Security*, Angelos D. Keromytis (Ed.). Lecture Notes in Computer Science, Vol. 7397. Springer, Berlin, 98–113.

David Isaac Wolinsky, Henry Corrigan-Gibbs, Bryan Ford, and Aaron Johnson. 2012a. Dissent in numbers: Making strong anonymity scale. In *Proceedings of the 10th USENIX Conference on Operating Systems Design and Implementation (OSDI'12)*. USENIX Association, 179–192.

David Isaac Wolinsky, Henry Corrigan-Gibbs, Bryan Ford, and Aaron Johnson. 2012b. Scalable anonymous group communication in the anytrust model. In *Proceedings of the European Workshop on System Security (EuroSec'12)*, Vol. 4.

M. Wright, M. Adler, B.N. Levine, and C. Shields. 2003. Defending anonymous communications against passive logging attacks. In *Proceedings of the Symposium on Security and Privacy (SP'03)*. 28–41.

Matthew Wright, Micah Adler, Brian Neil Levine, and Clay Shields. 2002. An analysis of the degradation of anonymous protocols. In *Proceedings of the Network and Distributed System Security Symposium (NDSS'02)*. The Internet Society.

Matthew K. Wright, Micah Adler, Brian Neil Levine, and Clay Shields. 2004. The predecessor attack: An analysis of a threat to anonymous communications systems. *ACM Trans. Inf. Syst. Secur.* 7, 4 (Nov. 2004), 489–522.

Ye Zhu, Xinwen Fu, Bryan Graham, Riccardo Bettati, and Wei Zhao. 2010. Correlation-based traffic analysis attacks on anonymity networks. *IEEE Trans. Parallel Distrib. Syst.* 21, 7 (2010), 954–967.

Xukai Zou, Byrav Ramamurthy, and Spyros Magliveras. 2002. Routing techniques in wireless ad hoc networks—Classification and comparison. In *Proceedings of the 6th World Multiconference on Systemics, Cybernetics, and Informatics (SCI'02).*

Received January 2017; revised November 2017; accepted January 2018