A Survey on Routing in Anonymous Communication Protocols

Article *in* ACM Computing Surveys · August 2016

DOI: 10.1145/3182658

Fatemeh Shirazi
KU Leuven
16 PUBLICATIONS **183** CITATIONS

SEE PROFILE

Milivoj Simeonovski
Universität des Saarlandes
13 PUBLICATIONS **121** CITATIONS

SEE PROFILE

Muhammad Rizwan Asghar
University of Auckland
100 PUBLICATIONS **1,005** CITATIONS

SEE PROFILE

Claudia Diaz
Universidad Nacional Autónoma de México
56 PUBLICATIONS **1,882** CITATIONS

SEE PROFILE

A Survey on Routing in Anonymous Communication Protocols

Fatemeh Shirazi
KU Leuven
ESAT/COSIC and iMinds

Milivoj Simeonovski
CISPA, Saarland University
Saarland Informatics Campus

Muhammad Rizwan Asghar
Department of Computer Science
The University of Auckland

Michael Backes
CISPA, Saarland University & MPI-SWS
Saarland Informatics Campus

Claudia Diaz
KU Leuven
ESAT/COSIC and iMinds

Abstract—**The Internet has undergone dramatic changes in the past 15 years, 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, ranging from profiling of users for monetizing personal information to nearly omnipotent governmental surveillance. As a result, public interest in systems for anonymous communication has drastically increased. Several such systems have been proposed in the literature, each of which offers anonymity guarantees in different scenarios and under different assumptions, reflecting the plurality of approaches for how messages can be anonymously routed to their destination. Understanding this space of competing approaches with their different guarantees and assumptions is vital for users to understand the consequences of different design options.**

In this work, we survey previous research on designing, developing, and deploying systems for anonymous communication. To this end, we provide a taxonomy for clustering all prevalently considered approaches (including Mixnets, DC-nets, onion routing, and DHT-based protocols) with respect to their unique routing characteristics, deployability, and performance. This, in particular, encompasses the topological structure of the underlying network; the routing information that has to be made available to the initiator of the conversation; the underlying communication model; and performance-related indicators such as latency and communication layer. Our taxonomy and comparative assessment provide important insights about the differences between the existing classes of anonymous communication protocols, and it also helps to clarify the relationship between the routing characteristics of these protocols, and their performance and scalability.

Keywords—Anonymous Communication; Routing; Privacy.

I. 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.

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 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 mitigate the threats of communications surveillance. The concept of AC systems was introduced by Chaum [1] 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 last decades has seen it applied to a variety of problems and scenarios, such as anonymous voting [2], [3], Private Information Retrieval (PIR) [4], censorship-resistance [5], [6], anonymous web browsing [7], hidden web services [8], and many others.

Public interest in AC systems has strikingly increased in the last 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 people [9].

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 constitutes the central element of the AC design, determining to a large extent both security and performance of the system. These 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 [10]–[18], 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 examine how their features determine the anonymity guarantees offered by those systems, as well as its 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 with respect to their routing characteristics, deployability, and performance. Then, we apply the taxonomy to the extensive scope of existing AC systems, in particular including Mixnets, DC-nets, onion routing systems, and DHT-based protocols. Finally, we discuss the relationship between the different routing decisions, and how they affect performance and scalability.

Outline. Section II provides our taxonomy for anonymous routing and describes the various routing features and dimensions that we are considering for our evaluation. Section III 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, substantiating the compact overview provided in the previous section. Section IV discusses the relationship between routing decisions and security and anonymity goals, and shares some lessons learned. Section V concludes the paper.

II. Anonymous Routing Protocol Characteristics

This section first introduces the routing characteristics considered in our taxonomy, and then discusses deployability, and performance metrics for AC networks.

A. 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 [19]–[21]. 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. Our classification (see Tables II and III) is an adaptation from Feeney's taxonomy [20], which classifies the routing characteristics of mobile ad hoc networks into four categories:

1) *Communication model* describes whether the communication is based on single-channels or multi-channels.
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 in overlay networks, which constitutes a standard design choice for many AC networks.

We redefine Feeney's criteria to account for design choices that are relevant to anonymous routing protocols. 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.

[1] https://wikileaks.org/

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

1) **Network Structure***:* We consider first the network features that are relevant to anonymous routing. These are, specifically, features relating 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 that control an Autonomous System (AS) [22], [23].

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 those two 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 [24].

Restricted routing proposals [25] have shown that for high-latency applications, 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 (our rule of thumb is that a node on average should be able to connect to at least 95% of the other nodes; this allows us to include systems that only exclude a small number of connections in order to prevent certain special cases from occurring).
- We call a network *mostly connected* (□) if its nodes can potentially connect to at least half the other 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 [24].

On the other hand, 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, but 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. In the first case, short-lived session keys can be setup to achieve forward secrecy properties; however, in non-interactive applications, such as email, forward secrecy is harder to achieve.

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 and response are visible to all nodes in the path. However, note that in 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* (≠) if the establishment of connections and relaying of messages is initiated by a user without any timing coordination with other participants.
- Connections are *synchronous* (≅) 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 [26] or perform end-to-end correlation attacks [27]–[29].

Synchronous systems are often more difficult to engineer and come with a performance or usability penalty; moreover, secure and reliable time becomes

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

TABLE I. 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)
	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)

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 [30], 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 [31]. 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 on 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 [24]. 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 in handling 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 routing. We call the resulting feature the *topology*:

- A network has a *flat* (···) 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 towards 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* (⊙) 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 placed on these entities, which constitute more of a point of failure than a simple relay.
- A network is *fully decentralized* (○) 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) ***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 to her.
   - The decision maker has a *partial view* (◐) of the system if the routing information available to her only covers 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 [32], [33] 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 in predefined time intervals.
   - Routing information is updated *event-based* (↯) if the updates are triggered by events in the network other than timeouts.
   - No updating mechanism is in place (✗).

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.

   Source-routing enables the initiator to pick nodes she 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 [32]. 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.*, [34]) and link-state routing (*i.e.*, [35]). 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 [36].

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, 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*. Uniform selection typically offers better anonymity levels, while weighted selection often improves performance.

B. Performance and Deployability

In addition to the routing characteristics identified before, we finally 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 two 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, respectively. In both cases, the answer is either yes (✓) or no (✗).

III. Routing Classification of AC Protocols

In this section, we present a categorization of AC protocols. We have classified these protocols into four main families: (1) Mixnet-based protocols, (2) Onion Routing-based protocols, (3) Random Walk and Distributed Hash Table (DHT)-based protocols, and (4) DCNet-based protocols (5) Miscellaneous, containing a few protocols that do not fit into the aforementioned categories. A few protocols are presented in the most representative category, albeit they can technically fall under other categories as well, *e.g.,* Octopus and Torsk are DHT-based, but they also use onion routing. We summarize our classification of the routing aspects in two comparative tables (namely Table II and Table III).

We now discuss the AC protocols individually, starting with Mixnet-based protocols (from Section III-A to Section III-D), and then proceeding with Onion Routing-based protocols (Section III-E and Section III-F), DHT-based protocols (Sections III-G), DCNet-based protocols (Section III-H), and finally the class of miscellaneous protocols (Section III-I).

A. Mixes

The idea of anonymous communication was originally proposed by David Chaum in 1981 [1] and initiated a new field of privacy research. The central concept proposed by Chaum is the use of *mix nodes*, or *mixes* in 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 ordered lexicographically.

B. Mix Selection Strategies

In order 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).

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

TABLE II. Routing Classification of Anonymous Communication Protocols: Mixnet and Onion Routing 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
Mixnet-based Protocols	Chaum's Mix Cascades	[1]	⊠	→	≠	●··●	···	✗	●	✗	●···	≡	✓	Ⓐ	◎	H	✉	✗	✗
	ISDN and Real-time	[31] [37]	⊏	↔	≅	●··●	···	⊙	◐	✗	●···	≡	✓	Ⓐ	◎	L	●–●	✗	✗
	Babel	[38]	⊠	→	≠	●··●	···	⊙	●	✗	●···/··●··	≡	✗	Ⓐ	⊛	H	✉	✗	✗
	Stop-and-Go-MIXes	[39]	⊠	→	≅	●··●	···	⊙	●	🕒	●···	≡	✗	Ⓐ	⊛	H	✉	✗	✗
	Webmixes	[40], [41]	⊏	↔	≅	●··●	···	⊙	●	🕒	●···	≡	✓	☺	◎	L	●–●	✓	✓
	A Reputation System for Mixnets	[42]	⊠	→	≅	●··●	···	⊙	●	🕒	●···/··●··	≡	✗	🛑	❋	H	✉	✗	✗
	Reliable Mix cascades	[43]	⊠	→	≅	●··●	···	⊙	●	🕒	●···	≡	✗	Ⓐ	❋	H	✉	✗	✗
	Mixmaster	[44]	⊠	→	≠	●··●	···	⊙	◐	✗	●···	≡	✗	Ⓐ	⊛	H	✉	✓	✓
	Mixminion	[45]	⊠	→	≠	●··●	···	⊙	●	🕒	●···	≡	✗	Ⓐ	⊛	H	✉	✓	✓
	Mix-Networks with Restricted Routes	[25]	⊏	→	≠	●··●	···	⊙	◐	🕒	●···	≡	✗	🌍	◎	H	✉	✗	✗
Onion Routing Protocols	Tor	[8]	□	↔	≠	●··⊙··●	···	⊙	●	🕒	●···	≡	✗	🌍🛑	◎	L	●–●	✓	✓
	AS-Aware Node Selection	[23]	□	↔	≠	●··⊙··●	···	⊙	●	🕒	●···	≡	✗	🌍🛑	◎	L	●–●	✓	✓
	LASTor	[46]	□	↔	≠	●··⊙··●	···	⊙	●	🕒	●···	≡	✗	🌍🛑	◎	L	●–●	✓	✗
	Coordinate Node Selection	[47]	□	↔	≠	●··⊙··●	···	⊙	●	🕒	●···	≡	✗	🌍🛑	◎	L	●–●	✓	✓
	Tuneup for Node Selection	[48]	□	↔	≠	●··⊙··●	···	⊙	●	🕒	●···	≡	✗	🌍🛑	❋	L	●–●	✓	✗
	Congestion-aware Node Selection	[49]	□	↔	≠	●··⊙··●	···	⊙	●	🕒	●···	≡	✗	🌍🛑	❋	L	●–●	✓	✗
	Panchenko Node Selection and Mator	[50] [51]	□	↔	≠	●··⊙··●	···	⊙	●	🕒	●···	≡	✗	🌍🛑	◎	L	●–●	✓	✗
	Torchestra, PCTCP, IMUX	[52] [53] [54]	□	↔	≠	●··⊙··●	···	⊙	●	🕒	●···	≡	✗	🌍🛑	◎	L	●–●	✓	✗
	Conflux	[55]	□	↔	≠	●··⊙··●	···	⊙	●	🕒	●···	≡	✗	🌍🛑	❋	L	●–●	✓	✗
	Prioritized Scheduling/ DiffTor	[56] [57]	□	↔	≠	●··⊙··●	···	⊙	●	🕒	●···	◇	✗	🌍🛑	◎	L	●–●	✓	✗
	PIR-Tor	[58]	□	↔	≠	●··⊙··●	···	⊙	◐	🕒	●···	≡	✗	🌍🛑	◎	L	●–●	✓	✗

TABLE III. Routing Classification of Anonymous Communication Protocols: DHT-based, DCNets, and Miscellaneous 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
DHT-based Protocols	Crowds	[59]	☒	↔	≠	●··●··●	···	⊙	●	↯	··●··	≡	✗	Ⓐ	⊛	L	✉	✓	✗
	MorphMix	[60], [61]	⊏	↔	≠	●··●··●	···	⊙	◐	↯	··●··	≡	✗	🌍	❋	L	●–●	✓	✓
	Torsk	[62]	⊏	↔	≠	●··⊙··●	···	⊙	◐	↯	●···	≡	✗	🌍	⊛	L	●–●	✓	✗
	NISAN	[63]	⊏	↔	≠	●··●··●	···	○	◐	↯	●···	≡	✗	Ⓐ	⊛	L	✉	✓	✗
	AP3	[64]	⊏	↔	≠	●··●··●	···	○	◐	↯	··●··	≡	✗	Ⓐ	⊛	L	●–●	✗	✗
	Salsa	[65]	⊏	↔	≠	●··●··● or ●··⊙··●	···	○	◐	↯	●···	≡	✗	Ⓐ	⊛	L	●–●	✓	✗
	Octopus	[66]	⊏	↔	≠	●··●··●	···	⊙	◐	↯	●···	≡	✗	Ⓐ	⊛	M	✉	✓	✗
	Freenet Opennet	[67]	⊏	↔	≠	●··●··●	···	⊙	◐	🕒	··●··	≡	✓	Ⓐ	❋	L	✉	✓	✓
	Freenet Darknet	[68]	⊏	↔	≠	●··●··●	···	○	◐	🕒	··●··	≡	✓	☺	❋	L	✉	✓	✓
	GAP from GNUnet	[69] [70]	⊏	↔	≠	●··●··●	···	○	◐	↯	··●··	⟐	✓	Ⓐ	❋	M	✉	✓	✓
DCNets	Chaum's DCNet, Revisited DCnet	[71] [72] [73]	☒	→	≠	●··●··●	···	✗	●	↯	●···	≡	✓	Ⓐ	◎	H	✉	✗	✗
	Herbivore	[74]	⊏	→	≠	●··●··●	⌘	⊙	◐	↯	●···	≡	✓	🌍	◎	H	✉	✓	✗
	Dissent	[75]	☒	→	≠	●··●··●	···	⊙	●	↯	●···	≡	✓	Ⓐ	◎	H	✉	✓	✓
	Dissent in Numbers	[76], [77]	⊏	→	≠	●··●	⌘	⊙	◐	↯	●···	≡	✓	🌍	◎	H	✉	✓	✓
Misc.	Tarzan	[78], [79]	□	↔	≠	●··●··●	···	○	●	↯	●···	≡	✗	🛑	⊛	L	●–●	✓	✓
	I2P	[80] [81]	□	→	≠	●··●··●	···	○	●	🕒	●···	⟐	✗	🌍🛑	❋	L	●–●	✓	✓

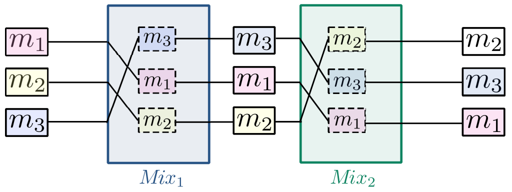

Fig. 1. A mix cascade with two mixes

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 message to that destination. This procedure is repeated until the last mix delivers the data to its final destination. In order to receive replies for messages while staying untraceable (to obtain recipient anonymity [82]), 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 her 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.

In order to overcome a single point of failure in availability of mix cascades, *free-route* mix networks have been proposed. 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 in mix cascades and free-route mix networks design is how mixes are selected. 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 [25] proposed the *restricted routes* mix networks that leverage the mix cascade model (*i.e.,* being less vulnerable to intersection attacks and being secure against global adversaries) and free-route mix networks (*i.e.,* being scalable). 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.* [42] proposed to identify mixes that fail and use a reputation system for mix selection leading to more 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, 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 towards 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 [43] proposed a mix cascade protocol with distributed trust. The system they propose uses a reputation mechanism for rearranging mix cascades in order to obtain more reliable 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 of mix cascade rearrangement. For each cascade, routing relevant information such as available bandwidth and expected waiting time are 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.

C. *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 in order 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 II-A. 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*) [39] 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 [83]. 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 a 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 [17], [83]. For example, the two prominent *remailers*, namely Mixmaster [44] and Mixminion [45], use *timed dynamic pool mixes* as flushing strategies [84], 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 [85]. We review these remailer protocols in Section III-D.

Next, we review some mix protocols from the literature that have been suggested for applications such as ISDN telephone, web browsing, and anonymous emails. In order to anonymize ISDN telephone communication with its intrinsic requirements on low-latency, Pfitzmann *et al.* [31] 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 use threshold mixes. To obtain sender and receiver anonymity, ISDN mixes use two mix cascades, each built by the sender and receiver, respectively, which are connected either by a connecting mix; 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 II, we disregard the setup broadcast for exchanging connected information for ISDN mixes. The basic design of ISDN mixes was later generalized by Jerichow *et al.* [37] to a system that enables low-latency, real-time communication.

A real-world realization built on ISDN mixes are *Webmixes* (also known as JAP) [40], [41] designed for real-time Internet applications, passing the traffic to several available mix cascades. In Webmixes, the mixes transform the messages cryptographically and re-shuffle 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 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. ISDN mixes, real-time mixes, and Webmixes have a deterministic node selection to build the mix cascade, where nodes selection for the cascades relies on the network state.

D. 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 [38] 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 [86]. 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 how routing information is communicated to users.

Mixmaster [44] 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, 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, the node selection is not specified by the protocol, though statistics on the reliability of mixes can be used to bias node selection [25]. Though the Mixmaster protocol did not specify details about maintaining mix information, later implementations of Mixmaster adopted an ad hoc scheme for distributing routing information [45]. One 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) [45] 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 II. 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 the other 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.

E. Onion Routing

Onion routing [7] [87] is designed for anonymizing connections for applications with low-latency constraints, such as web browsing.

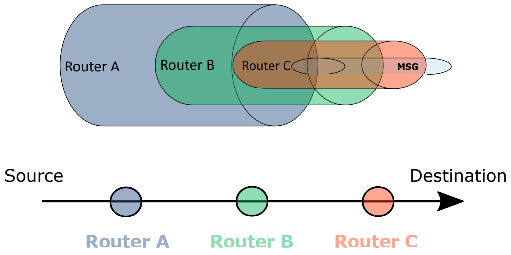

Fig. 2. The concept of onion routing

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 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 the adversary may identify and correlate traffic patterns at the initiator and receiver [14], [16], thus de-anonymizing the connection. Nonetheless, onion routing is a promising design to provide a low-latency AC network, and many currently used systems to build upon this design.

F. Onion Routing-based Protocols

Onion routing is used in Tor [8], which 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 free to use the framework, consists of a large set of volunteering routers (at the time of writing, there exist more than 7000 routers [9]). 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 are not possible. Tor's services are used daily by approximately 2,000,000 users [9]. 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 [88] on onion routing. To defend against this and related attacks, selecting a small set of nodes was introduced for Tor [89]. Previously, each user maintained a list of 3 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 [90].

In the early onion routing design, it was suggested to select the nodes uniformly random [91]. 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 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.* [51], [92].

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 [93]. Bauer *et al.* have proposed a traffic analysis attack aim at decreasing the anonymity of Tor [28]. 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 [50] and [51]. Involving AS-level information in the node selection has been proposed by [23] and [46]. Moreover, offering the user a tuneup option between uniformly random node selection (for high anonymity) and weighted random node selection with a bias towards high bandwidth nodes (for better performance) has been suggested by Snader and Borisov [48].

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 [13], [94], 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 [52]–[55], which can be used to separate interactive and bulk traffic into different connections. Another solution is to prioritize interactive traffic in the scheduling phase [56] [57]. 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 [50]. Furthermore, another solution to Tor's congestion problem is to enforce avoiding congested nodes at the node selection phase [49]. Another reason for Tor's high latency is circuitous paths [46]. To solve this problem, node selection strategies have been proposed that take the destination between chosen nodes into account [46], [47], [50].

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 [62], or by using private node retrieval [58].

G. 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 [59] is one of the early AC systems designed for anonymous web browsing. The key design feature of Crowds is a random peer selection. In Crowds, all nodes are grouped into so-called *crowds;* all nodes within a crowd might connect to each other for relaying a communication. Each node in the crowd is called a *jondo*. A so-called *blender* is responsible for managing and administrating nodes. Crowds has a peer-to-peer structure since all users of the system are nodes themselves. The 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. 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 [88], [95]. In order to prevent such type of attacks, Crowds suggested 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 jondo [59]. However, even keeping routes static for a day is not enough to prevent predecessor attacks [86].

MorphMix [60], [61] 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. In order to avoid repeated connections with the same set of nodes, a node has to forget about nodes it has not been connected and constantly require new node information. After an initiator selects the first node, she selects randomly a witness for each hop thereafter, randomly chosen from the nodes in her local database. She asks the next hop to extend the route with the assistance of the witness she has chosen, where nodes propose a set of candidate nodes for the next hop and the witness chooses one of them. To prevent path compromise, nodes can only propose nodes with different IP prefix to her 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. In order to mitigate guessing whether a node was initiator by the next hop, the initiator adds random delays to her communication before forwarding them in the tunnel establishment phase.

Efficiency is one of the main problems in random walk protocols. In the next section, we review DHT-based protocols, which aim at efficient node lookup and selection. Random walk protocols can employ DHT lookups to gain better efficiency (*e.g.,* AP3 protocol [64]).

1) DHT-based Protocols: In distributed systems, where there are network administrators, a challenge is to locate a node. One solution is to use Distributed Hash Tables (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 3). 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 [96] (locating the nodes based on their estimated distance using an XOR metric), Chord [97] (using a clockwise circle metric, where at each hop of the lookup, the distance to the node is decreased, at least half), and Pastry [98] (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 besides the classical lookup method. For example, node selection is carried out by selecting a random key and by then using a classical lookup method (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* [64], a random walk protocol aiming at providing 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.

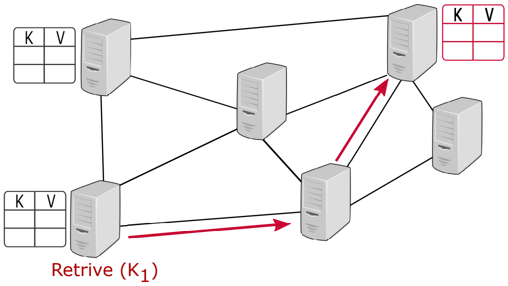

Fig. 3. The concept of distributed hash tables

Next, we review two protocols that aim at replacing node selection of source-routed protocols such as onion routing with structured DHT systems making the suitable to be combined with onion routing. *Salsa* [65], proposed by Nambiar *et al.*, aims at providing scalability and preventing malicious colluding nodes to be able to bias 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 in order to avoid malicious nodes returning wrong addresses. The lookup queries are carried out similar to the Chord lookup in a recursive fashion. qIn 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* [62], a peer-to-peer AC protocol, replacing Tor's node selection and directory service with a DHT design. It aims at providing 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 [99] and randomly selected secret "secret buddies."

Panchenko *et al.* proposed *NISAN* [63], an AC protocol that aims at achieving high scalability and preventing adversaries to bias routing. NISAN uses iterative search to select nodes randomly for constructing anonymous paths. It uses an adaptation of Chord, where the node lookups are aggregated. 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 [66] aims at providing security by preventing malicious nodes to be able to bias routing. It also aims at providing 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. In the first phase, nodes are selected by the path initiator (user). 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* [67], a peer-to-peer censorship-resistant system for sharing storage space. Freenet offers strong decentralization in order to provide privacy and robustness against attacks. The key design feature of Freenet is based on storage replication and plausible deniability. Files are stored multiple times at the nodes, are indexed by binary file keys, and can be looked up by their corresponding key. Each node has a dynamic routing table including the node information with the stored keys. The original design uses a heuristic deterministic routing using potentially all Freenet participating nodes choosing mostly neighborhood nodes (currently called Opennet mode). Freenet uses an adaptive routing using DHTs with keys that are location-independent. Three methods are used for key construction: keyword-signed key, signed-subspace key, and content-hash key (for more details see [67]). The routing table is updated periodically to achieve better performance. The replication of files provides resilience against node failure and node overloads. In the Opennet mode, a heuristic-based deterministic routing approach (a distance-directed depth-first search with backtracking) is used [68], [100]. 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, 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 in order to optimize routing for future requests. If the node that is contacted is not responding, the node sends the request to the node with the second-nearest key. 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 [67]. 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 the Darknet mode, such shortcomings are addressed. In 2010, Freenet has been extended by a membership-concealing Darknet mode, where trusted connection are used for routing [68]. In the Darknet mode, the user chooses the nodes from her trusted nodes [68]. The routing table is consisting of nodes derived from a fixed graph, which is the social graph of the node. In the Darknet mode, the routing table is not optimized during time and cannot include nodes that are not derived from the social graph of the node. 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 [101]. The relaying nodes only know their predecessor and the successor in order provide privacy. In Freenet, the data is encrypted using symmetric encryption. The key is transferred either with the address or in the header of the file request [67].

GNUnet [69] 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 [70]. 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 credit rating scheme, where relaying requests and replies increases 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 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"). Unlike Freenet, GAP uses a time-live restriction to avoid routing loops; 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 timed and threshold mixes for flushing batches of messages, where the time restriction is selected randomly.

H. DC Networks

The idea of *DCnets* (Dining Cryptographers Networks) was first proposed by Chaum [71] and later revisited [72], [73]. 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 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 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 (see Figure 4).

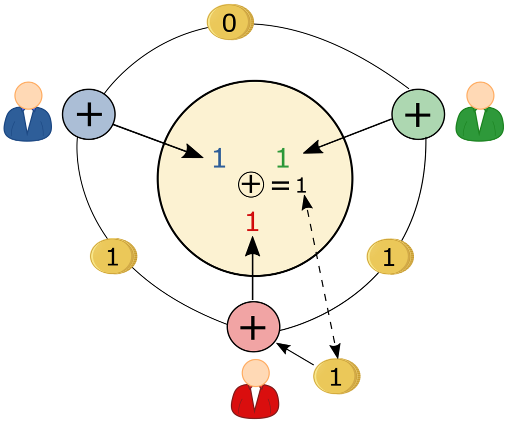

Fig. 4. The concept of DC network

The DCnet concept can be generalized, to transmit large messages simply by repeating the protocol as desired [73]. 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, 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 [102], [103]. Furthermore, recovering from a fault is only possible by re-broadcasting the messages.

Chaum proposed in his DCnet 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 the problem of detecting the disruptions because malicious participants can adapt their answers to other participants 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 [104] or star topologies [105]. 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 [106].

Herbivore [74] 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 [97]. In order 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 low-latency, we decided to classify the protocols as being *high 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) [75] is a latency-tolerant protocol for AC. 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. 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 her 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, where each participant receives the set of message descriptors (which were encrypted in a layered fashion) and shuffles them and passes them 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 a non-member node. Hence, Dissent primarily only guarantees 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 [76]. 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.

I. Miscellaneous Protocols

Tarzan [78], [79] 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 [78], where the peers only needed to know about a random subset of nodes, the final version [79] introduces a *gossip-based* protocol based on the Name-Dropper protocol [107], 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 [32]. Node information is stored in a ring model and lookups are carried out using the Chord algorithm [97]. 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.

I2P [108] 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 [109]. Currently, the number of I2P routers is estimated to be between 40,000 and 50,000 [110].

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 [111].

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 [81].

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 standard tier.

The communication through I2P is protected using *garlic encryption*. Garlic encryption is very similar to onion encryption, with the difference that multiple data messages may be contained in a single *garlic message*.

IV. Discussion

A. Routing Features: Commonalities and Differences

In this section, we discuss commonalities and differences between the investigated classes of AC protocols with respect to their routing characteristics. The discussion is grounded on our classification presented in Tables II and III, and strives to provide a deeper understanding of the relationships of individual routing characteristics.

Mixnet-based protocols, as classified in Table II, show the most heterogeneous routing design among the four investigated protocol classes. The main reason for this is they demonstrate routing diversity on multiple routing building blocks, such as proposing disparate flushing strategies, differentiating node selection strategies, 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 and only Webmixes and Restricted routed mix networks as partially connected.

Generally speaking, mix cascade networks employ rather synchronized connection because messages are sent in batches and mostly depend on their flushing algorithms in 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. Dingledine *et al.*'s proposal for a reputation system for mixnets [42] also uses a synchronized message relaying to enable verifying the correctness of the routing process.

In the mix protocols, node management has not been 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 [45]. 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 protocols in Tables II and III 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 III-A, 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 Stop-and-go mixes, where the delays are randomly selected from a restricted time range.

Onion routing protocols, as classified in Table II, are all Tor related schemes and hence, exhibit the most homogeneous routing design among the four investigated protocol classes. On a conceptual level, all these protocols are equally characterized by their routing features. However, there are three exceptions that affect: 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 [112]. Also, differences in the routing policy, which do not change the routing feature on a conceptual level such as changing node selection probabilities [51] and [50], are equally classified in the table, though node selection probabilities could be different.

One inherent routing feature of onion routing 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 onion routing protocols have a client-server model, which improves their usability and leads to a higher number of users, thus contributing to better anonymity for onion routing protocols [113]. They are characterized as complete network view due to a central authority, which distributes the list of Tor routers. One exception is [58], 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 in order to prevent the decision maker to choose 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 [56], [57], where they suggest a prioritization at the scheduling phase in favor of interactive traffic in order to reduce delays that interactive users might experience.

Node selection in all onion routing-based protocols is non-deterministic. This is important since the Tor network consists of volunteers and it 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. Since the adversary is assumed to be local, a non-deterministic node selection makes targeted surveillance harder.

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 [48] and for avoiding congestion [49], [55]. Onion routing 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, known as *bridges*, are not part of this view.

Next, we discuss random walk protocols and DHT-based protocols. Crowds are 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.

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 herself, 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 [114]; 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 onion routing 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 [32], and route bridging attacks [33]. 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 [115].

Most of the DHT-based protocols are characterized with 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, again here 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 providing unlinkability [82] 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.

Next, we discuss the DCNets protocols. *DCNet-based protocols*, as classified in Table III, 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-based protocols incur high latency and have message-based communication models. In DC-nets, we regard avoiding collusion and shuffling as routing relevant aspects of these protocols. The first designs of DCnets [71]–[73] and Dissent [75] are 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.

In order to improve efficiency and performance, some DCNet-based protocols [74], [76], [77] 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-based protocols enforce network restrictions to the selection set in order to increase efficiency and performance.

We conclude this part of the discussion with miscellaneous protocols. *Tarzan* protocol 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 III. The connectivity of *I2P* is similar to onion routing 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 [81], which could enable adversaries to carry out eclipse attacks targeting floodfill nodes [111]. 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 [116]. 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 [117].

B. Correlation, Conflicts, Trade-offs, and Applications

In this section, we address correlation (*i.e.,* dependencies and conflicts), and trade-offs between routing characteristics of AC networks. First, we review direct and indirect correlations of routing features by comparing them with each other. We conclude this section with a discussion about the relevance of specific routing characteristics for certain applications.

We have defined the topology only based on connectivity of relaying routers (see Section II-A). This is the reason why users and administrative entities are not taken into account. Hence, based on our definition, there is no correlation between topology and the two of the routing features, namely roles and decentralization.

There is an evident correlation between hierarchy and topology of AC networks. A hierarchical AC network does not have a fully connected network structure. For example, Herbivore, which has a hierarchical routing strategy, has a partially connected topology. 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. This holds for random walk and DHT-based protocols. One exception is demonstrated by PIR-TOR, which uses PIR to keep the network view minimal, albeit the topology is fully connected. Therefore, the correlation between topology and the network view depends on further factors. For example, if the topology is partially connected, it might be that the routing decision maker has a partial view, but it also might be due to some other routing restrictions.

We also observe a correlation between topology and selection set. Namely, restrictions in the selection set lead to reduced connectivity of the network topology. For example, in Restricted Route mix networks, the network view is complete; however, connectivity is restricted due to restrictions in the selection set, which leads to a partially connected network topology.

Although the synchronization of connections is not directly correlated to scheduling, it depends on the forwarding strategy of the particular nodes. As mentioned in Section III-A, the flushing algorithms influence synchronization when timed mixes are used.

AC networks with a hierarchical structure have partially connected network structure (*i.e.,* Herbivore and the extended version of Dissent). By definition, hierarchical organization of nodes restricts the selection set.

Node management is more challenging in fully decentralized AC networks. Therefore, obtaining a complete view and a periodic updating of routing information is more difficult. When the network view of the routing decision maker is partial, often source-routing has the advantage to prevent the bias of malicious nodes and partitioning attacks. Thus, AC protocols that use this combination need to employ a secure node selection policy in their protocol. Examples of such protocols are Octopus and Morphmix. Octopus uses bound checking and proactively identifies malicious nodes; while, the latter one randomly selects witnesses to prevent bias in node selection. A partial network view also restricts

TABLE IV. Overview of the adversary definitions, focus of routing feature, and challenges that our four routing classes face

Routing Class	Adversary Type	Routing Feature in Focus	Challenges
Mixnet	Global & active	Forwarding (scheduling) & node management (topology)	Traffic analysis attacks, such as flooding attacks
Onion routing	Local & active	Node selection	Traffic analysis attacks, *i.e.,* timing attacks
Random Walks (DHT)	Local & active	Node selection & transfer of routing information	Partitioning attacks & biasing node selection
DCnet	Global & passive	Forwarding	Collision and disruption

the selection set because the routing decision maker can select only nodes that it is aware of.

Clearly, flushing algorithms also influence scheduling. For example, pool mixes can be defined to induce prioritized scheduling. There is also a correlation between scheduling and latency because in a prioritized scheduling algorithm, some type of traffic is delayed.

Flushing algorithms also influence latency. Timed mixes by themselves do not necessarily influence latency. However, they might induce latency if long time restrictions have been selected. Same with the threshold mixes, when the incoming traffic is low compared to the threshold that has been set up. There is also a correlation between latency and communication mode. High latency AC networks usually use a message based communication mode and vice versa. This is because connections are not going to be used further (*e.g.,* replies are not going to be sent in a short time); therefore, setting up a circuit is unnecessary.

Next, we compare our four main groups by discussing their applications. Mixnets are designed to be secure against traffic analysis and global adversaries by aggregating messages into batches. However, they are vulnerable to the collusion of mixnets and flooding attacks [84], in case if there are not enough (honest) users. Moreover, Mixnets' resilience against traffic analysis comes with a price and makes them more appropriate for high latency applications, such as emails and electronic voting.

Onion routing protocols, such as Tor, are more efficient (in particular faster) and have little computational overhead, making them suitable for low-latency applications, such as web browsing. Tor also leverages a large number of volunteer nodes. Almost all of these nodes are known to the routing decision maker. However, the complete network structure for the routing decision maker can limit scalability. Moreover, Tor is considered to be only secure against local adversaries and it is vulnerable to traffic analysis attacks [93], [118]–[122], in particular if the adversary can access both ends of the communication.

Random walk protocols and protocols using DHT are designed rather for fully peer-to-peer networks, where the network view is incomplete. Having a fully peer-to-peer network motivates the growth of the network and helps scalability. Therefore, they 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 is considered to affect the usability of the protocol. Unfortunately, this might lead to a decrease in the number of users of such systems and in turn reduce anonymity. Last but not least, classic DCnets provide information-theoretic anonymity but some of them require a restricted setting, where all users or nodes need to be honest. The classic DCnets were also not resilient against DoS attacks. Moreover, DCnets tend do have a large communication overhead and do not scale well. Even Dissent, which employs a client-server approach for better scalability, can only scale up to a few thousand clients [76]. Therefore, they are more suitable for applications, such as micro-blogging, but at a small scale.

In Table IV, we summarized the type of adversaries defined for the particular routing class. Moreover, the table shows the routing feature in focus for the particular class. Finally, the table lists challenges that our four routing classes face in terms of routing features and achieving anonymity and security.

V. Concluding Remarks

In this work, we classified anonymous routing characteristics. We identified main criteria groups, each with several routing features and dimensions tackling various aspects routing decisions in AC protocols. Moreover, we shortly described and then carefully evaluated the bulk of existing AC protocols under our classification. Furthermore, we discussed the relevance between routing decisions that are made in such networks and their influence on anonymity and security. We have learned several lessons from conducting our survey. On the one hand, security, anonymity, scalability, and performance goals that are favored for anonymous communication are very hard to reach altogether, simply because the routing decisions, which support each of these goals, often contradict each other. This is especially true for achieving 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. Therefore, it is very hard to separate the various routing aspects from one to another protocol. 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.

Acknowledgment

This work was supported in part by the Research Council KU Leuven: GOA TENSE (GOA/11/007) and the Flemish Government FWO G.0360.11N Location Privacy, FWO G.068611N Data mining and by the European Commission through H2020- DS-2014-653497 PANORAMIX and H2020-ICT-2014-644371 WITDOM. This work was also supported by Microsoft Research through its PhD 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).

References

[1] D. Chaum, "Untraceable electronic mail, return addresses, and digital pseudonyms," *Commun. ACM*, vol. 24, no. 2, pp. 84–88, 1981.

[2] K. Sako and J. Kilian, "Receipt-free Mix-Type voting scheme - A practical solution to the implementation of a voting booth," in *Advances in Cryptology - EUROCRYPT '95, International Conference on the Theory and Application of Cryptographic Techniques, Saint-Malo, France, May 21-25, 1995*, pp. 393–403, 1995.

[3] M. Jakobsson, A. Juels, and R. L. Rivest, "Making mix nets robust for electronic voting by randomized partial checking," in *Proceedings of the 11th USENIX Security Symposium, San Francisco, CA, USA, August 5-9, 2002*, pp. 339–353, 2002.

[4] R. Dingledine, M. J. Freedman, and D. Molnar, "The free haven project: Distributed anonymous storage service," in *Designing Privacy Enhancing Technologies, International Workshop on Design Issues in Anonymity and Unobservability, Berkeley, CA, USA, July 25-26, 2000*, pp. 67–95, 2000.

[5] M. Waldman, A. D. Rubin, and L. F. Cranor, "Publius: A robust, tamper-evident, censorship-resistant, and source-anonymous web publishing system," in *9th USENIX Security Symposium, Denver, Colorado, USA, August 14-17*, 2000.

[6] M. Waldman and D. Mazières, "Tangler: a censorship-resistant publishing system based on document entanglements," in *CCS 2001, Proceedings of the 8th ACM Conference on Computer and Communications Security, Philadelphia, Pennsylvania, USA, November 6-8, 2001*, pp. 126–135, 2001.

[7] D. Goldschlag, M. Reed, and P. Syverson, "Hiding routing information," in *Information Hiding* (R. Anderson, ed.), vol. 1174 of *Lecture Notes in Computer Science*, pp. 137–150, Springer Berlin Heidelberg, 1996.

[8] R. Dingledine, N. Mathewson, and P. Syverson, "Tor: The second-generation onion router," in *Proceedings of the 13th Conference on USENIX Security Symposium - Volume 13*, SSYM '04, pp. 303–320, USENIX Association, 2004.

[9] The Tor Project, "Tor metrics." https://metrics.torproject.org/. Last accessed: August 05, 2015.

[10] E. Erdin, C. Zachor, and M. Gunes, "How to find hidden users: A survey of attacks on anonymity networks," *Communications Surveys Tutorials, IEEE*, vol. PP, no. 99, pp. 1–1, 2015.

[11] K. Sampigethaya and R. Poovendran, "A survey on mix networks and their secure applications," *Proceedings of the IEEE*, vol. 94, pp. 2142–2181, December 2006.

[12] B. Conrad and F. Shirazi, "A survey on Tor and I2P," in *Ninth International Conference on Internet Monitoring and Protection (ICIMP2014)*, pp. 22–28, July 2014.

[13] M. AlSabah and I. Goldberg, "Performance and security improvements for Tor: A survey." Cryptology ePrint Archive, Report 2015/235, 2015.

[14] J. Ren and J. Wu, "Survey on anonymous communications in computer networks," *Computer Communications*, vol. 33, no. 4, pp. 420–431, 2010.

[15] M. Edman and B. Yener, "On anonymity in an electronic society: A survey of anonymous communication systems," *ACM Computing Surveys (CSUR)*, vol. 42, pp. 5:1–5:35, December 2009.

[16] G. Danezis and C. Díaz, "A survey of anonymous communication channels," tech. rep., Microsoft Research, 2008.

[17] A. Serjantov, "On the anonymity of anonymity systems," tech. rep., University of Cambridge, Computer Laboratory, October 2004.

[18] J. Raymond, "Traffic analysis: Protocols, attacks, design issues, and open problems," in *Designing Privacy Enhancing Technologies, International Workshop on Design Issues in Anonymity and Unobservability, Berkeley, CA, USA, July 25-26, 2000*, pp. 10–29, 2000.

[19] P. Bell and K. Jabbour, "Review of point-to-point network routing algorithms," *Communications Magazine, IEEE*, vol. 24, pp. 34–38, January 1986.

[20] L. M. Feeney, "A Taxonomy for Routing Protocols in Mobile Ad Hoc Networks," 1999.

[21] X. Zou, B. Ramamurthy, and S. Magliveras, "Routing techniques in wireless ad hoc networks - classification and comparison," in *Proceedings of the Sixth World Multiconference on Systemics, Cybernetics, and Informatics (SCI 2002*, 2002.

[22] N. Feamster and R. Dingledine, "Location diversity in anonymity networks," in *Proceedings of the 2004 ACM Workshop on Privacy in the Electronic Society*, WPES '04, (New York, NY, USA), pp. 66–76, ACM, 2004.

[23] M. Edman and P. Syverson, "As-awareness in Tor path selection," in *Proceedings of the 16th ACM Conference on Computer and Communications Security*, CCS '09, (New York, NY, USA), pp. 380–389, ACM, 2009.

[24] R. Böhme, G. Danezis, C. Díaz, S. Köpsell, and A. Pfitzmann, "On the PET workshop panel mix cascades versus peer-to-peer: Is one concept superior?," in *Privacy Enhancing Technologies* (D. Martin and A. Serjantov, eds.), vol. 3424 of *Lecture Notes in Computer Science*, pp. 243–255, Springer Berlin Heidelberg, 2005.

[25] G. Danezis, "Mix-networks with restricted routes," in *Privacy Enhancing Technologies, Third International Workshop, PET 2003, Dresden, Germany, March 26-28, 2003, Revised Papers*, pp. 1–17, 2003.

[26] G. Danezis, "Statistical disclosure attacks," in *Security and Privacy in the Age of Uncertainty, IFIP TC11 18th International Conference on Information Security (SEC '03), May 26-28, 2003, Athens, Greece*, pp. 421–426, 2003.

[27] B. Levine, M. Reiter, C. Wang, and M. Wright, "Timing attacks in low-latency mix systems," in *Financial Cryptography* (A. Juels, ed.), vol. 3110 of *Lecture Notes in Computer Science*, pp. 251–265, Springer Berlin Heidelberg, 2004.

[28] K. Bauer, D. McCoy, D. Grunwald, T. Kohno, and D. Sicker, "Low-resource routing attacks against Tor," in *Proceedings of the 2007 ACM Workshop on Privacy in Electronic Society*, WPES '07, (New York, NY, USA), pp. 11–20, ACM, 2007.

[29] Y. Zhu, X. Fu, B. Graham, R. Bettati, and W. Zhao, "Correlation-based traffic analysis attacks on anonymity networks," *IEEE Trans. Parallel Distrib. Syst.*, vol. 21, no. 7, pp. 954–967, 2010.

[30] S. J. Murdoch and P. Zielinski, "Sampled traffic analysis by internet-exchange-level adversaries," in *Privacy Enhancing Technologies, 7th International Symposium, PET 2007 Ottawa, Canada, June 20-22, 2007, Revised Selected Papers*, pp. 167–183, 2007.

[31] A. Pfitzmann, B. Pfitzmann, and M. Waidner, "Isdn-mixes: Untraceable communication with very small bandwidth overhead," in *Kommunikation in verteilten Systemen*, vol. 267 of *Informatik-Fachberichte*, pp. 451–463, Springer Berlin Heidelberg, 1991.

[32] G. Danezis and R. Clayton, "Route fingerprinting in anonymous communications," in *Peer-to-Peer Computing, 2006. P2P 2006. Sixth IEEE International Conference on*, pp. 69–72, IEEE, 2006.

[33] G. Danezis and P. Syverson, "Bridging and fingerprinting: Epistemic attacks on route selection," in *Proceedings of the 8th International Symposium on Privacy Enhancing Technologies*, PETS '08, (Berlin, Heidelberg), pp. 151–166, Springer-Verlag, 2008.

[34] S. R. D. Charles E. Perkins, Elizabeth M. Belding-Royer, "RFC3561: Ad hoc On-Demand Distance Vector (AODV) Routing," 2003.

[35] J. Moy, "RFC2328: OSPF Version 2," 1998.

[36] C. Grothoff, "An excess-based economic model for resource allocation in peer-to-peer networks," *Wirtschaftsinformatik*, vol. 3-2003, June 2003.

[37] A. Jerichow, J. Müller, A. Pfitzmann, B. Pfitzmann, and M. Waidner, "Real-time mixes: a bandwidth-efficient anonymity protocol," *IEEE Journal on Selected Areas in Communications*, vol. 16, no. 4, pp. 495–509, 1998.

[38] C. Gülcü and G. Tsudik, "Mixing email with babel," in *1996 Symposium on Network and Distributed System Security, NDSS '96, San Diego, CA, February 22-23, 1996*, pp. 2–16, 1996.

[39] D. Kesdogan, J. Egner, and R. Büschkes, "Stop-and-Go-MIXes providing probabilistic anonymity in an open system," in *Information Hiding, Second International Workshop, Portland, Oregon, USA, April 14-17, 1998, Proceedings*, pp. 83–98, 1998.

[40] O. Berthold, H. Federrath, and S. Köpsell, "Web MIXes: A system for anonymous and unobservable internet access," in *Designing Privacy Enhancing Technologies, International Workshop on Design Issues in Anonymity and Unobservability, Berkeley, CA, USA, July 25-26, 2000*, pp. 115–129, 2000.

[41] O. Berthold, H. Federrath, and M. Köhntopp, "Project anonymity and unobservability in the internet," in *Proceedings of the Tenth Conference on Computers, Freedom and Privacy: Challenging the Assumptions*, CFP '00, (New York, NY, USA), pp. 57–65, ACM, 2000.

[42] R. Dingledine, M. Freedman, D. Hopwood, and D. Molnar, "A reputation system to increase MIX-Net reliability," in *Information Hiding* (I. Moskowitz, ed.), vol. 2137 of *Lecture Notes in Computer Science*, pp. 126–141, Springer Berlin Heidelberg, 2001.

[43] R. Dingledine and P. Syverson, "Reliable MIX cascade networks through reputation," in *Financial Cryptography* (M. Blaze, ed.), vol. 2357 of *Lecture Notes in Computer Science*, pp. 253–268, Springer Berlin Heidelberg, 2002.

[44] U. Möller, L. Cottrell, P. Palfrader, and L. Sassaman, "Mixmaster protocol - version 2," 2003.

[45] G. Danezis, R. Dingledine, and N. Mathewson, "Mixminion: Design of a type III anonymous remailer protocol," in *2003 IEEE Symposium on Security and Privacy (SP 2003), 11-14 May 2003, Berkeley, CA, USA*, pp. 2–15, 2003.

[46] M. Akhoondi, C. Yu, and H. Madhyastha, "Lastor: A low-latency as-aware Tor client," in *Security and Privacy (SP), 2012 IEEE Symposium on*, pp. 476–490, May 2012.

[47] M. Sherr, M. Blaze, and B. T. Loo, "Scalable link-based relay selection for anonymous routing," in *Proceedings of Privacy Enhancing Technologies, 9th International Symposium (PETS 2009)* (I. Goldberg and M. J. Atallah, eds.), vol. 5672 of *Lecture Notes in Computer Science*, pp. 73–93, Springer, August 2009.

[48] R. Snader and N. Borisov, "Improving security and performance in the Tor network through tunable path selection," *Dependable and Secure Computing, IEEE Transactions on*, vol. 8, pp. 728–741, September 2011.

[49] T. Wang, K. Bauer, C. Forero, and I. Goldberg, "Congestion-aware path selection for Tor," in *Financial Cryptography and Data Security* (A. Keromytis, ed.), vol. 7397 of *Lecture Notes in Computer Science*, pp. 98–113, Springer Berlin Heidelberg, 2012.

[50] A. Panchenko, F. Lanze, and T. Engel, "Improving performance and anonymity in the Tor network," in *31st IEEE International Performance Computing and Communications Conference, IPCCC 2012, Austin, TX, USA, December 1-3, 2012*, pp. 1–10, 2012.

[51] M. Backes, A. Kate, S. Meiser, and E. Mohammadi, "(Nothing else) MATor(s): Monitoring the Anonymity of Tor's Path Selection," in *Proceedings of the 2014 ACM SIGSAC Conference on Computer and Communications Security*, CCS '14, (New York, NY, USA), pp. 513–524, ACM, 2014.

[52] D. Gopal and N. Heninger, "Torchestra: Reducing interactive traffic delays over Tor," in *Proceedings of the 2012 ACM Workshop on Privacy in the Electronic Society*, WPES '12, (New York, NY, USA), pp. 31–42, ACM, 2012.

[53] M. AlSabah and I. Goldberg, "PCTCP: per-circuit TCP-over-IPsec transport for anonymous communication overlay networks," in *2013 ACM SIGSAC Conference on Computer and Communications Security, CCS'13, Berlin, Germany, November 4-8, 2013*, pp. 349–360, 2013.

[54] J. Geddes, R. Jansen, and N. Hopper, "IMUX: managing Tor connections from two to infinity, and beyond," in *Proceedings of the 13th Workshop on Privacy in the Electronic Society, WPES 2014, Scottsdale, AZ, USA, November 3, 2014*, pp. 181–190, 2014.

[55] M. AlSabah, K. S. Bauer, T. Elahi, and I. Goldberg, "The path less travelled: Overcoming Tor's bottlenecks with traffic splitting," in *Privacy Enhancing Technologies - 13th International Symposium, PETS 2013, Bloomington, IN, USA, July 10-12, 2013. Proceedings*, pp. 143–163, 2013.

[56] C. Tang and I. Goldberg, "An improved algorithm for Tor circuit scheduling," in *Proceedings of the 17th ACM Conference on Computer and Communications Security*, CCS '10, (New York, NY, USA), pp. 329–339, ACM, 2010.

[57] M. AlSabah, K. S. Bauer, and I. Goldberg, "Enhancing Tor's performance using real-time traffic classification," in *the ACM Conference on Computer and Communications Security, CCS'12, Raleigh, NC, USA, October 16-18, 2012*, pp. 73–84, 2012.

[58] P. Mittal, F. Olumofin, C. Troncoso, N. Borisov, and I. Goldberg, "PIR-Tor: Scalable anonymous communication using private information retrieval," in *Proceedings of the 20th USENIX Conference on Security*, SEC '11, (Berkeley, CA, USA), pp. 31–31, USENIX Association, 2011.

[59] M. K. Reiter and A. D. Rubin, "Crowds: Anonymity for web transactions," *ACM Trans. Inf. Syst. Secur.*, vol. 1, pp. 66–92, November 1998.

[60] M. Rennhard and B. Plattner, "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, (New York, NY, USA), pp. 91–102, ACM, 2002.

[61] M. Rennhard and B. Plattner, "Practical anonymity for the masses with morphmix," in *Financial Cryptography* (A. Juels, ed.), vol. 3110 of *Lecture Notes in Computer Science*, pp. 233–250, Springer Berlin Heidelberg, 2004.

[62] J. McLachlan, A. Tran, N. Hopper, and Y. Kim, "Scalable onion routing with Torsk," in *Proceedings of the 16th ACM Conference on Computer and Communications Security*, CCS '09, (New York, NY, USA), pp. 590–599, ACM, 2009.

[63] A. Panchenko, S. Richter, and A. Rache, "NISAN: network information service for anonymization networks," in *Proceedings of the 2009 ACM Conference on Computer and Communications Security, CCS 2009, Chicago, Illinois, USA, November 9-13, 2009*, pp. 141–150, 2009.

[64] A. Mislove, G. Oberoi, A. Post, C. Reis, P. Druschel, and D. S. Wallach, "AP3: cooperative, decentralized anonymous communication," in *Proceedings of the 11st ACM SIGOPS European Workshop, Leuven, Belgium, September 19-22, 2004*, p. 30, 2004.

[65] A. Nambiar and M. Wright, "Salsa: A structured approach to large-scale anonymity," in *Proceedings of the 13th ACM Conference on Computer and Communications Security*, CCS '06, (New York, NY, USA), pp. 17–26, ACM, 2006.

[66] Q. Wang and N. Borisov, "Octopus: A secure and anonymous DHT lookup," in *2012 IEEE 32nd International Conference on Distributed Computing Systems, Macau, China, June 18-21, 2012*, pp. 325–334, 2012.

[67] I. Clarke, O. Sandberg, B. Wiley, and T. W. Hong, "Freenet: A distributed anonymous information storage and retrieval system," in *International Workshop on Designing Privacy Enhancing Technologies: Design Issues in Anonymity and Unobservability*, pp. 46–66, Springer-Verlag New York, Inc., 2001.

[68] I. Clarke, O. Sandberg, M. Toseland, and V. Verendel, "Private communication through a network of trusted connections: The dark freenet," *Network*, 2010.

[69] K. Bennett, T. Stef, C. Grothoff, T. Horozov, and I. Patrascu, "The gnet whitepaper," June 2002.

[70] K. Bennett and C. Grothoff, "gap practical anonymous networking," in *Privacy Enhancing Technologies* (R. Dingledine, ed.), vol. 2760 of *Lecture Notes in Computer Science*, pp. 141–160, Springer Berlin Heidelberg, 2003.

[71] D. Chaum, "The dining cryptographers problem: Unconditional

sender and recipient untraceability," *Journal of Cryptology*, vol. 1, no. 1, pp. 65–75, 1988.

[72] M. Waidner and B. Pfitzmann, "The dining cryptographers in the disco: Unconditional sender and recipient untraceability with computationally secure serviceability," in *Advances in Cryptology - EUROCRYPT '89* (J.-J. Quisquater and J. Vandewalle, eds.), vol. 434 of *Lecture Notes in Computer Science*, pp. 690–690, Springer Berlin Heidelberg, 1990.

[73] P. Golle and A. Juels, "Dining cryptographers revisited," in *Advances in Cryptology - EUROCRYPT '04* (C. Cachin and J. Camenisch, eds.), vol. 3027 of *Lecture Notes in Computer Science*, pp. 456–473, Springer Berlin Heidelberg, 2004.

[74] S. Goel, M. Robson, M. Polte, and E. Sirer, "Herbivore: A scalable and efficient protocol for anonymous communication," tech. rep., Cornell University, 2003.

[75] H. Corrigan-Gibbs and B. Ford, "Dissent: Accountable anonymous group messaging," in *Proceedings of the 17th ACM Conference on Computer and Communications Security*, CCS '10, pp. 340–350, 2010.

[76] D. I. Wolinsky, H. Corrigan-Gibbs, B. Ford, and A. Johnson, "Dissent in numbers: Making strong anonymity scale," in *Proceedings of the 10th USENIX Conference on Operating Systems Design and Implementation*, OSDI '12, pp. 179–192, USENIX Association, 2012.

[77] D. I. Wolinsky, H. Corrigan-Gibbs, B. Ford, and A. Johnson, "Scalable anonymous group communication in the anytrust model," in *European Workshop on System Security (EuroSec)*, vol. 4, 2012.

[78] M. J. Freedman, E. Sit, J. Cates, and R. Morris, "Introducing tarzan, a peer-to-peer anonymizing network layer," in *Peer-to-Peer Systems, First International Workshop, IPTPS 2002, Cambridge, MA, USA, March 7-8, 2002, Revised Papers*, pp. 121–129, 2002.

[79] M. J. Freedman and R. Morris, "Tarzan: A peer-to-peer anonymizing network layer," in *Proceedings of the 9th ACM Conference on Computer and Communications Security*, CCS '02, pp. 193–206, ACM, 2002.

[80] "I2P documentation." https://geti2p.net/en/docs. Last accessed: August 05, 2015.

[81] L. Schimmer, "Peer profiling and selection in the I2P anonymous network," in *Proceedings of PET-CON 2009.1*, pp. 59–70, March 2009.

[82] A. Pfitzmann and M. Köhntopp, "Anonymity, unobservability, and pseudonymity - A proposal for terminology," in *Designing Privacy Enhancing Technologies, International Workshop on Design Issues in Anonymity and Unobservability, Berkeley, CA, USA, July 25-26, 2000, Proceedings*, pp. 1–9, 2000.

[83] C. Díaz and B. Preneel, "Taxonomy of mixes and dummy traffic," in *Information Security Management, Education and Privacy, IFIP 18th World Computer Congress, TC11 19th International Information Security Workshops, 22-27 August 2004, Toulouse, France*, pp. 215–230, 2004.

[84] A. Serjantov, R. Dingledine, and P. Syverson, "From a trickle to a flood: Active attacks on several mix types," in *Information Hiding* (F. Petitcolas, ed.), vol. 2578 of *Lecture Notes in Computer Science*, pp. 36–52, Springer Berlin Heidelberg, 2003.

[85] C. Díaz and A. Serjantov, "Generalising mixes," in *Privacy Enhancing Technologies* (R. Dingledine, ed.), vol. 2760 of *Lecture Notes in Computer Science*, pp. 18–31, Springer Berlin Heidelberg, 2003.

[86] G. Danezis, C. Diaz, and P. F. Syverson, "Systems for Anonymous Communication," in *CRC Handbook of Financial Cryptography and Security* (B. Rosenberg and D. Stinson, eds.), CRC Cryptography and Network Security Series, pp. 341–390, Chapman & Hall, August 2010.

[87] M. Reed, P. Syverson, and D. Goldschlag, "Anonymous connections and onion routing," *Selected Areas in Communications, IEEE Journal on*, vol. 16, pp. 482–494, May 1998.

[88] M. Wright, M. Adler, B. N. Levine, and C. Shields, "An analysis of the degradation of anonymous protocols," in *Proceedings of the Network and Distributed System Security Symposium, NDSS 2002, San Diego, California, USA*, The Internet Society, 2002.

[89] M. Wright, M. Adler, B. Levine, and C. Shields, "Defending anonymous communications against passive logging attacks," in *Security and Privacy (SP), 2003. Proceedings. 2003 Symposium on*, pp. 28–41, May 2003.

[90] R. Dingledine, N. Hopper, G. Kadianakis, and N. Mathewson, "One fast guard for life (or 9 months)," *7th Workshop on Hot Topics in Privacy Enhancing Technologies (HotPETs 2014)*, 2014.

[91] P. Syverson, G. Tsudik, M. Reed, and C. Landwehr, "Towards an analysis of onion routing security," in *Designing Privacy Enhancing Technologies* (H. Federrath, ed.), vol. 2009 of *Lecture Notes in Computer Science*, pp. 96–114, Springer Berlin Heidelberg, 2001.

[92] M. Backes, A. Kate, P. Manoharan, S. Meiser, and E. Mohammadi, "AnoA: A framework for analyzing anonymous communication protocols," in *Computer Security Foundations Symposium (CSF), 2013 IEEE 26th*, pp. 163–178, June 2013.

[93] S. Murdoch and G. Danezis, "Low-cost traffic analysis of Tor," in *Security and Privacy, 2005 IEEE Symposium on*, pp. 183–195, May 2005.

[94] R. Dingledine and S. J. Murdoch, "Performance improvements on Tor or, why Tor is slow and what we're going to do about it," tech. rep., The Tor Project, November 2009.

[95] M. K. Wright, M. Adler, B. N. Levine, and C. Shields, "The predecessor attack: An analysis of a threat to anonymous communications systems," *ACM Trans. Inf. Syst. Secur.*, vol. 7, pp. 489–522, Novmeber 2004.

[96] P. Maymounkov and D. Mazières, "Kademlia: A peer-to-peer information system based on the xor metric," in *Revised Papers from the First International Workshop on Peer-to-Peer Systems*, IPTPS '01, (London, UK, UK), pp. 53–65, Springer-Verlag, 2002.

[97] I. Stoica, R. Morris, D. Karger, M. F. Kaashoek, and H. Balakrishnan, "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, pp. 149–160, ACM, 2001.

[98] A. I. T. Rowstron and P. Druschel, "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 Heidelberg*, Middleware '01, (London, UK, UK), pp. 329–350, Springer-Verlag, 2001.

[99] P. Wang, I. Osipkov, N. Hopper, and Y. Kim, "Myrmic: Provably secure and efficient DHT routing," tech. rep., DTC, 2006.

[100] S. Roos, B. Schiller, S. Hacker, and T. Strufe, "Measuring freenet in the wild: Censorship-resilience under observation," in *Privacy Enhancing Technologies - 14th International Symposium, PETS 2014, Amsterdam, The Netherlands, July 16-18, 2014. Proceedings*, pp. 263–282, 2014.

[101] J. Kleinberg, "The small-world phenomenon: An algorithmic perspective," in *Proceedings of the Thirty-second Annual ACM Symposium on Theory of Computing*, STOC '00, pp. 163–170, ACM, 2000.

[102] J. Bos and B. den Boer, "Detection of disrupters in the dc protocol," in *Advances in Cryptology - EUROCRYPT '89* (J.-J. Quisquater and J. Vandewalle, eds.), vol. 434 of *Lecture Notes in Computer Science*, pp. 320–327, Springer Berlin Heidelberg, 1990.

[103] M. Waidner, "Unconditional sender and recipient untraceability in spite of active attacks," in *Advances in Cryptology - EUROCRYPT '89* (J.-J. Quisquater and J. Vandewalle, eds.), vol. 434 of *Lecture Notes in Computer Science*, pp. 302–319, Springer Berlin Heidelberg, 1990.

[104] S. Dolev and R. Ostrobsky, "Xor-trees for efficient anonymous multicast and reception," *ACM Trans. Inf. Syst. Secur.*, vol. 3, pp. 63–84, May 2000.

[105] A. Pfitzmann and M. Waidner, "Networks without user observability - design options," in *Advances in Cryptology - EUROCRYPT '85* (F. Pichler, ed.), vol. 219 of *Lecture Notes in Computer Science*, pp. 245–253, Springer Berlin Heidelberg, 1986.

[106] J. Douceur, "The sybil attack," in *Peer-to-Peer Systems* (P. Druschel, F. Kaashoek, and A. Rowstron, eds.), vol. 2429 of *Lecture*

Notes in Computer Science, pp. 251–260, Springer Berlin Heidelberg, 2002.

[107] M. Harchol-Balter, F. T. Leighton, and D. Lewin, “Resource discovery in distributed networks,” in *Proceedings of the Eighteenth Annual ACM Symposium on Principles of Distributed Computing*, PODC ’99, pp. 229–237, 1999.

[108] J. P. Timpanaro, I. Chrisment, and O. Festor, “I2P’s usage characterization,” in *Traffic Monitoring and Analysis* (A. Pescapè, L. Salgarelli, and X. Dimitropoulos, eds.), vol. 7189 of *Lecture Notes in Computer Science*, pp. 48–51, Springer Berlin Heidelberg, 2012.

[109] J. P. Timpanaro, C. Isabelle, and F. Olivier, “Monitoring the I2P network,” tech. rep., October 2011.

[110] “I2P statistics.” http://stats.i2p.re/. Last accessed: January 25, 2016.

[111] C. Egger, J. Schlumberger, C. Kruegel, and G. Vigna, “Practical attacks against the i2p network,” in *Proceedings of the 16th International Symposium on Research in Attacks, Intrusions and Defenses (RAID 2013)*, October 2013.

[112] M. AlSabah, K. Bauer, I. Goldberg, D. Grunwald, D. McCoy, S. Savage, and G. Voelker, “Defenestrator: Throwing out windows in Tor,” in *Privacy Enhancing Technologies* (S. Fischer-Hübner and N. Hopper, eds.), vol. 6794 of *Lecture Notes in Computer Science*, pp. 134–154, Springer Berlin Heidelberg, 2011.

[113] R. Dingledine and N. Mathewson, “Anonymity loves company: Usability and the network effect,” in *Proceedings of the Fifth Workshop on the Economics of Information Security (WEIS 2006)* (R. Anderson, ed.), (Cambridge, UK), June 2006.

[114] G. Danezis, C. Diaz, E. Ksper, and C. Troncoso, “The wisdom of crowds: Attacks and optimal constructions,” in *Computer Security ESORICS 2009* (M. Backes and P. Ning, eds.), vol. 5789 of *Lecture Notes in Computer Science*, pp. 406–423, Springer Berlin Heidelberg, 2009.

[115] M. Castro, P. Druschel, A. Ganesh, A. Rowstron, and D. S. Wallach, “Secure routing for structured peer-to-peer overlay networks,” *SIGOPS Oper. Syst. Rev.*, vol. 36, pp. 299–314, Dec. 2002.

[116] “I2P peer profiling and selection.” https://geti2p.net/en/docs/how/peer-selection. Last accessed: January 25, 2016.

[117] “I2P transport overview.” https://geti2p.net/en/docs/transport. Last accessed: January 25, 2016.

[118] S. Chakravarty, A. Stavrou, and A. Keromytis, “Traffic analysis against low-latency anonymity networks using available bandwidth estimation,” in *Computer Security - ESORICS 2010* (D. Gritzalis, B. Preneel, and M. Theoharidou, eds.), vol. 6345 of *Lecture Notes in Computer Science*, pp. 249–267, Springer Berlin Heidelberg, 2010.

[119] C. G. Nathan S. Evans, Roger Dingledine, “A practical congestion attack on Tor using long paths,” in *Presented as part of the 18th USENIX Security Symposium (USENIX Security 09)*, (Montreal, Canada), USENIX, 2009.

[120] A. Johnson, C. Wacek, R. Jansen, M. Sherr, and P. F. Syverson, “Users get routed: traffic correlation on tor by realistic adversaries,” in *2013 ACM SIGSAC Conference on Computer and Communications Security, CCS’13, Berlin, Germany, November 4-8, 2013*, pp. 337–348, 2013.

[121] P. Mittal, A. Khurshid, J. Juen, M. Caesar, and N. Borisov, “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, (New York, NY, USA), pp. 215–226, ACM, 2011.

[122] G. O’Gorman and S. Blott, “Improving stream correlation attacks on anonymous networks,” in *Proceedings of the 2009 ACM Symposium on Applied Computing (SAC), Honolulu, Hawaii, USA, March 9-12, 2009*, pp. 2024–2028, 2009.