A Survey of Anonymous Peer-to-Peer File-Sharing

Tom Chothia and Konstantinos Chatzikokolakis

Laboratoire d'Informatique, École Polytechnique, 91128 Palaiseau Cedex, France
{tomc,kostas}@lix.polytechnique.fr

Abstract. This paper provides a survey of searchable, peer-to-peer file-sharing systems that offer the user some form of anonymity. We start this survey by giving a brief description of the most popular methods of providing anonymous communication. These include the Ants protocol, Onion routing, Multicasting, MIXes and UDP address spoofing. We then describe a number of implemented systems based on one, or a combination of, these methods. Finally, we discuss possible attacks on the anonymity of these systems and give examples of particular attacks and defences used by the systems we describe.

1 Introduction

File-sharing is a hot topic in academic circles, in the open source community and in the media, but there has been very little exchange of ideas between these groups. There are a number of peer-to-peer systems that offer some kind of anonymity. However, when talking about anonymous systems it is vital to be precise about what is anonymous, from whom, under what conditions, and exactly how anonymous. In this paper we discuss both the theory of anonymity and the implemented file-sharing systems.

The majority of anonymous peer-to-peer systems are "friend-to-friend" networks. These are peer-to-peer networks in which each peer (node) only connects to a small number of other, known nodes. Only the direct neighbours of a node know its IP address. Communication with remote nodes is provided by sending messages hop-to-hop across this overlay network. Routing messages in this way allows these networks to trade efficient routing for anonymity. There is no way to find the IP address of a remote node, and direct neighbours can achieve a level of anonymity by claiming that they are just forwarding requests and files for other nodes. These systems offer anonymity against an attacker that is a single node inside the system and that is looking for the IP address of someone who is searching for, or offering, a file.

In these systems every node has a pseudo address that can be used for communication. It is easy to find the pseudo address of any node that is sending or receiving a file, but it is hard to link these pseudo addresses with the node's real IP addresses. Furthermore, a node may stop using a pseudo address at anytime, independently make a new one or even have more than one.

There is a danger that the attacker will be able to link the pseudo address and the IP address of their direct neighbours, and thus find out which files the neighbour is requesting or offering. Some systems contain faults that leak this information while others allow an attacker to be up to 50% certain of their neighbour's pseudo address. This is more of a problem than it seems because, in the interests of growing the network, most systems make it easy for anyone to join at any point. It would be possible for an attacker to try random IP addresses until it finds someone running the protocol, and then negotiate a connection. None of the current systems try to make it hard for an attacker to work out whether or not someone is running the file-sharing software.

The level of anonymity a system offers usually degrades as more attackers join the network. Successful attacks are much easier if the attackers can choose where they join a system, particularly if they can surround a node. If the attacker knows how the peers are connected to each other, the systems offer very poor protection. There is also poor protection against attacks based on measuring the time a node takes to respond. Most systems offer no protection against an attacker that can observe all network communications, although it is possible that a MIX [Cha81] based file-sharing system could be effective.

In Section 2, we detail a number of theoretical ideas and protocols that form the basis for most types of anonymity. Section 3 catalogues implemented peer-to-peer systems that offer some kind of anonymity. We pay particular attention to systems that allow for anonymous searches and file downloading. We present a number of possible attacks against anonymous file-sharing systems in Section 4. The two most potent attacks against the current batch of implemented systems seem to be the use of multiple attackers - especially when used to surround a node - and time-based, statistical analysis attacks.

There are many interesting social, legal and economic issues related to anonymous peer-to-peer systems; too many to be covered in this survey. Many of these issues also lie outside the field of computer science. A collection of papers on these issues can be found at `http://www.inf.tu-dresden.de/∼hf2/anon/`. A longer, more detailed version of this paper is available online.

2 Theoretical Background

When we design or analyse an anonymous system we must define what we mean by "anonymous". Generally speaking, our purpose is to hide the relationship between an observable action (for example, a message sent across a public network) and the identity of the users involved with this action. Some questions that immediately arise are "Which identity do we want to hide?", "From whom?" and "To what extent?". The answers to these questions lead to different notions of anonymity.

The main agents involved in file-sharing are the *sender*, who initiates a search for a file, and the *responder* or *receiver* who answers the search query and provides the file. In peer-to-peer networks these agents are communicating through a number of *nodes* which forward the request and possibly the search data. A *global attacker* is considered to have access to all messages that are sent over the network. These definitions lead to the following kinds of anonymity.

– Sender anonymity to any node, the responder or a global attacker.
– Responder anonymity to any node, the sender or a globel attacker.
– Sender-responder unlinkability to any node or a global attacker.

It may also be useful to consider an attacker that is a combination of a global attacker, sender, receiver and any number of nodes inside the system. Pfitzmann and Hanse [PK04] provide an extended discussion on this topic. Considering the level of anonymity provided by a system, Reiter and Rubin [RR98] provide the following useful classification:

Beyond suspicion (BS) From the attacker's point of view, the detected user appears no more likely to have originated the action than any other node.
Probable innocence (ProbI) From the attacker's point of view, the detected user appears no more likely to have originated the action than to not to have.
Possible innocence (PossI) From the attacker's point of view, there is a nontrivial probability that the detected user did not originated the action.

The following table gives a general idea of the anonymity provided by the methods discussed in the rest of this section. In many cases only a rough estimate of the anonymity guarantees is given in the corresponding papers, and the notions examined are not always the same.

Anonymity theories	Ants	Mixes	Crowds	Onion Routing	DC-nets	Multi -cast	Spoofed UDP	Freenet
S. anon. to G.A.	No	No	No	No/BS	BS	No	No	No
R. anon. to G.A.	No	No	No	No	BS	BS	No	No
S. anon. to R.	ProbI	BS	BS	BS	BS	No	ProbI	ProbI
S. anon. to N.	ProbI	No	ProbI	No/BS	BS	No	ProbI	ProbI
R. anon. to S	ProbI	No	No	No	BS	BS	No	No
R. anon. to N.	ProbI	No	No	No	BS	BS	No	No
S.-R. unlink. to N.	ProbI	BS	ProbI	BS	BS	BS	ProbI	ProbI
S.-R. unlink. to G.A.	No	BS	No	BS	BS	BS	No	No

The **Ants** protocol [GSB02] was designed for ad-hoc networks, in which nodes do not have fixed positions. In this setting, each node has a pseudo identity which can be used to send messages to a node, but does not give any information about its true identity. In order to search the network, a node broadcasts a search message with its own pseudo identity, a unique message identifier and a time-to-live counter. The search message is sent to all of the node's neighbours, which in turn send the message to all of their neighbours until the time-to-live counter runs out. Upon receiving a message, a node records the connection on which the message was received and the pseudo address of the sender. Each node dynamically builds and maintains a routing table for all the pseudo identities it sees. This table routes messages addressed to a pseudo identity along the connection over which the node has received the most messages from that pseudo identity. To send a message to a particular pseudo identity, a node sends a message with the pseudo identity as a "to" address. If a node has that pseudo address in its table, it forwards the message along the most used connection. Otherwise, it forwards the message to all its neighbours.

There exists no published anonymity analysis of the Ants protocol. It is widely believed that the Ants protocol provides probable innocence if a proper probabilistic time-to-live counter is used.

Onion routing is a general-purpose protocol [SGR97] that allows anonymous connection over public networks on condition that the sender knows the public keys of all the other nodes. Messages are randomly routed through a number of nodes called *Core Onion Routers (CORs).* In order to establish a connection, the initiator selects a random path through the CORs and creates an onion, a recursively layered data structure containing the necessary information for the route. Each layer is encrypted with the key of the corresponding COR. When a COR receives an onion, a layer is "unwrapped" by decrypting it with the COR's private key. This reveals the identity of the next router in the path and a new onion to forward to that router. Since inner layers are encrypted with different keys, each router obtains no information about the path, other than the identity of the following router.

There are two possible configurations for an end-user. They can either run their own COR (local-COR configuration) or use one of the existing ones (remote-COR). The first requires more resources, but the second provides better anonymity.

Onion routing has also been adaped to a number of different settings. The table gives the values for original Syverson, Goldschlag and Reeds version discuss here, the two values of sender anonymity correspond to the remote-COR (left) and local-COR (right) configurations.

Freenet [CSWH01] is a searchable peer-to-peer system for censorship resistant document storage. It is both an original design for anonymity and an implemented system. While it does not aim to hide the provider of a particular file it does aim to make it impossible for an attacker to find all copies of a particular file. A key feature of the Freenet system is that each node will store all the files that pass across it, deleting the least used if necessary. A hash of the title (and other key words) identifies the files. Each node maintains a list of the hashes corresponding to the files on immediately surrounding nodes. A search is carried out by first hashing the title of the file being searched for, and then forwarding the request to the neighbouring node that has the file with the most similar hash value. The node receiving the request forwards it in the same way. If a file is found, it is sent back along the path of the request. This unusual search method implements a node-to-node broadcast search one step at a time. Over time it will group files with similar title hash values, making the search more efficient.

Return Address Spoofing can be used to hide the identity of the sender. The headers of messages passed across the Internet include the IP address of the sender. This address is not used by routers, so it does not have to be correct. The Transmission Control Protocol (TCP) uses this return address to send acknowledgements and control signals, but the User Datagram Protocol (UDP) does not require these controls. Simply by using the UDP protocol and entering a random return address, a sender can effectively send data and hide their identity from the receiver. Without the controls of TCP, packets are liable to loss or congestion. However, if the receiver has an anonymous back channel to communicate with the sender, it can use this to send control signals. A problem with UDP-spoofing is that such behaviour is associated with wrongdoing, and so it is often prohibited by ISPs.

A **Broadcast** can be used to provide receiver anonymity by ensuring that enough other people receive the message to obscure the intended recipient. A broadcast can be performed in an overlay network by having each node send a message to all of its neighbours, which in turn send it to all of their neighbours, and so on. If a unique identity is added to the message, nodes can delete reoccurrences of the same message and stop loops from forming. In large networks it may be necessary to include some kind of time-to-live counter to stop the message flooding the network. In anonymous systems this counter is usually probabilistic. One of the most useful methods of broadcasting is Multicasting [Dee89].

Crowds is an anonymous protocol for web-transactions proposed by Reiter and Rubin [RR98]. This protocol involves a group of users, called a "crowd", each of whom wants to communicate with a corresponding web server but without revealing his identity. The idea is to randomly route each message through the crowd until one member of the crowd decides to pass it to the server. This ensures that neither the receiver nor the nodes in the system can tell who sent the message. This system requires all nodes to be connected to all other nodes, and so it scales badly to larger networks.

MIXes [Cha81] provide anonymity by forwarding messages from node to node, but instead of forwarding each message as it arrives, the nodes wait until they have received a number of messages and then forward them mixed up. When done correctly this can hide the sender and the receiver, as well as sender-receiver linkablity from an attacker that can see the whole network. This can be done without requiring all of the nodes to consistently broadcast packets. One draw back is that each node has to hold a message until it has enough messages to properly mix them up, which might be difficult in a file-sharing system due to the asymmetrical nature of downloading files. There are many different kinds of MIX systems. The values given in the table are for the classical Mixes described here, in which end-users do not perform mixing themselves, but they communicate with a mix node in order to send a message. Protection between the mix nodes is better.

DC-nets [Cha88] and **XOR-trees** [DO00] use the XOR of combinations of messages. These methods provide prefect anonymity by requiring all members of the system to broadcast in every time slice.

3 Implemented Anonymous Systems

This section discusses implemented and publicly available systems for searchable, anonymous peer-to-peer file-sharing. The following table summarises the systems discussed in the section.

The development of real anonymous peer-to-peer systems can be more troublesome than one would at first suspect. A case in point was an anonymous peer-to-peer system known as "Winny". The author of this system pushed it as a truly anonymous file-sharing system and file-sharers who wished to swap movies quickly picked it up. While the specification of the system was never fully released, there was soon firm evidence that the system did not really guarantee anonymity, as police arrested two of the system's users and charged them with copyright theft. Shortly after this, the author of the software, who was a researcher in the Computer Science Department of Tokyo University, was also arrested and charged with aiding and abetting copyright theft [Ley04].

Name	Based on	Website or Paper
Ants	Ants	http://antsp2p.sourceforge.net
AP3	Crowds	[MOP$^+$04]
APFS	Onion routing	[SLS01]
Entropy	Freenet	http://entropy.stop1984.com
Free Haven	Secret sharing and MIXes	[DFM00]
Freenet	Freenet	[CSWH01]
GNUnet	MIXes	http://gnunet.org/
I2P	Onion routing	http://www.i2p.net/
Mantis	Ants and UDP spoofing	[BASM04]
Mute	Ants	http://mute-net.sourceforge.net
Nodezilla	Freenet	http://www.nodezilla.net
Napshare	Ants	http://napshare.sourceforge.net
Tor	Onion routing	[DMS04]
SSMP	Secret sharing and onion routing	[HLX$^+$05]
Waste	Friend-to-Friend	http://waste.sourceforge.net

Mute, **ANTS** and **Mantis** [BASM04] implement the Ants protocol. Mute uses a three-stage probabilistic time-to-live counter to avoid flooding the network. The time-to-live is reduced by a value proportional to the number of results found at a node and the number of connections the search message is forwarded to. This stops a node being flooded with too many responses, while allowing searches for less common files to go further. However, analysis of the counter may allow a statistical attack. Another point of interest in Mute is that all the probabilistic choices are fixed when a node starts running. This protects against statistical attacks by ensuring that the repetition of the same action yields no new information to the attacker.

Ants does not use a time-to-live counter. Instead, there is a chance that a packet will be dropped at any time. The Ants routing algorithm may send responding packets along more than one path, which leads to faster file downloads.

Mantis allows the searcher to exchange anonymity for download efficiency. Anonymous communication is used to search for files and to send control signals, while the data can be sent directly from the server to the client using return address spoofed UDP. To protect against attackers surrounding a node, Mantis uses a “Blender” to control access to the system.

Anonymous Peer-to-peer File-Sharing (APFS) [SLS01] is a searchable system with responder and sender anonymity. It is based on Onion Routing, with the addition of volunteer nodes that will act as proxies. Centralised servers are necessary to handle the searches. When a node is willing to share a file, it first picks an anonymous proxy and sends that proxy an onion route and a random identity. The node then sends the server the names of the files it is willing to offer, the name of the proxy and the random identity used to identify the connection.

Free Haven [DFM00] is an anonymous publishing system. It is made up of a number of servers - known as servnets - which agree to store and provide documents for anyone. The identities of these servnets are publicly known. All communications are made over an external MIX-based communication layer. When a publisher wants to publish a file on Free Haven it breaks it into a number of parts using Rabin’s information dispersal algorithm and sends each part to a different servnet. When a reader wants to download a file it must first find the hash of the file it is searching for and send this to a servnet. The servnet broadcasts the request to the other servnets, which then sends the pieces of the file to the reader.

The **WASTE** peer-to-peer system carefully controls which nodes may join the network. It is aimed at networks of 10-50 nodes and provides strong anonymity guarantees on condition that no one allows an attacker to join the network. Messages sent between nodes are encrypted, and idle nodes send dummy traffic, making traffic analysis attacks harder.

Nodezilla is an anonymous transport layer that uses the Everlink protocol. This protocol implements a version of Freenet in which both nodes and objects have identities, and requests are routed to nodes with the closest identity to the requested object. Unlike Freenet, when a node receives a packet and cannot forward that packet to another node, it assumes that it is the packet’s intended recipient. As with Freenet, nodes cache copies of the data sent across them. So, after a node forwards a file to its neighbour, the neighbour will know that the node is offering that file for download.

GNUnet is a searchable peer-to-peer file-sharing network whose transport protocol, called GAP is based on MIXes. In GAP, a user can choose to trade anonymity for efficiency by stopping the MIX nodes from rewriting the reply address of each message. GAP also uses an economic system where each request counts towards a node’s allocated credit. An analysis of GNUnet together with some possible attacks can be found in [Küg03].

I2P is a network layer that allows applications to communicate anonymously. I2P uses a technique called *garlic routing*, based on onion routing, in which the sender defines the path for outgoing messages and the recipient defines the path for incoming messages. Any of the intermediate nodes can inject a number of hops before forwarding the message to the next peer. Another feature of I2P is that applications can select a tradeoff between latency and anonymity by adjusting some parameters of the protocol, such as the number of hops in their tunnels.

AP3 [MOP$^+$04] is a peer-to-peer overlay network which provides anonymous message delivery, anonymous channels and secure pseudonyms. It uses the same technique as Crowds but it is built on top of the Pastry [RD01] network. In Pastry random delivery is possible without knowledge of the whole network. A user can choose a random key and the network can deliver the message to the node which is "closest" to the key. AP3 does not have a built-in search, but as it supports anonymous multicasting one could be implemented.

Tor [DMS04] is an unsearchable transport layer system that uses Onion Routing. With care, it is possible to make standard file-sharing applications anonymous by running them over the Tor network. The Azureus BitTorrent client, for instance, provides the option of running over Tor. Responder anonymity is not part of the basic system, but it is made possible by "rendezvous points".

Han et al. [HLX$^+$05] have designed a file-sharing system based on Shamir's secret sharing scheme called the **Secret-Sharing-based Mutual anonymity Protocol (SSMP)**. It requires a search to be broadcast to all nodes, and for some messages to be forwarded randomly until they reach their goal, so the system scales badly to larger networks.

There are also a number of other anonymous peer-to-peer systems such as **Share**, **Rodi** or **UDPP2P**. Unfortunately, they lack proper documentation, so it is hard to assess their merit.

4 Possible Attacks Against Anonymous P2P Networks

The attacker in a peer-to-peer system may be the sender, the responder, any node in the system or an outsider. The usual goal is to find out who the sender or responder is, or what they are transferring. Alternatively, we can consider a global attacker who can see the whole network, with the additional goal of linking the sender and receiver. Attacks can become much more effective if there are a number of attackers working together. If attackers can mostly surround a node, in any system, they can usually degrade that node's anonymity. Knowledge of the network topology is also useful for an attacker. Just as many implemented systems combine a number of the basic methods for anonymity, real attacks on these systems may combine a number of the attack methods outlined below.

Time-to-Live Attacks: Time-to-live counters determine the maximum number of hops for a message and are used in most peer-to-peer networks to avoid flooding. If an attacker can send a request to a node with such a low time-to-live counter that the packet will probably not be forwarded, any response relieves that note as the responder. To avoid this problem, Mute, Ants, Freenet and Mantis use probabilistic time-to-live counters. In these systems there is always a chance that a request will be forwarded to another node.

Nodes in closely connected networks will receive copies of the same packet over each of its connections. If the attacker knows how the nodes are connected, it may be able to work out the pseudo address of certain other nodes from the difference in the time-to-live counters of each of these copies.

There is also a time-to-live counter in the underlying Internet Protocol; In systems based on UDP spoofing, such as Mantis, the IP time-to-live counter could reveal some information about the sender.

Multiple Attackers or Identities: If attackers can make repeated connections to a single node with different identities, the anonymity of that node is usually lost. Some theoretical systems have complicated joining procedures to prevent attackers surrounding a single node. However, most implemented systems do not. This is one of the biggest security issues in up and running networks such as Mute or Ants. Attackers can make repeated connections to the same node and send that node a request for a file with an expired time-to-live counter. If the request is forwarded it will most likely go to one of the attackers, so if the attackers get any responses to their request they know it came from the node under attack. Douceur [Dou02] points out that it is almost always possible for a single attacker to assume a number of different identities, which he terms a "Sybil" attack. Another example of the use of multiple attackers is the locking or $n-1$ attack [GT96] on a MIX that mixes n messages. Here $n-1$ attackers send a messages to a MIX node. The next message sent to the node then triggers the MIX and can be traced, as it is the only message not known to the attacker.

One way to protect against this attack is to strictly control where a new node may join a network. A different approach is to have all nodes connected to all others, as in Crowds for instance. However, this tends to scale poorly to large networks. Even with these preconditions most systems will loose anonymity if the ratio of attackers to honest nodes gets too high.

Statistical Attacks: Any attacker can gather statistical data over time. Systems that are provably safe for a single run may reveal information about the identities of their participants when all the observable messages of a longer run are analysed for patterns.

An incorrect implementation of the Ants protocol search phase might call for a node to make a probabilistic choice to forward or drop a packet every time it receives one. While this would guarantee anonymity for a single run of the protocol, over time a node would be more likely to see requests with the identities of its direct neighbours. Mute defends against these kinds of attacks by fixing the probabilities of forwarding or dropping a packet when a node starts up. Hence, a repeated request will result in exactly the same action every time and provide no extra statistical information to the attacker. Another way to deal with this attack is to ensure that probabilities are used in such a way that all actions are equal likely. The nodes in Crowds forward messages to any other node in the system rather than just their nearest neighbours. When a node repeats an action, the message observable by the outsider is equally likely to come from any node.

If an attack can force a sender and a receiver to repeatedly re-establish a connection, or can identify the sender or receiver from the contents of a data stream, it may be able to gain the extra information it needs for a statistical attack. This is the case in both Crowds and Onion Routing, as shown by Wright et al. [WALS01]. Shmatikov shows the same result for Crowds [Shm02] used the Prism model checker. Back, Möller and Stiglic [BMS01] describe a traffic analysis attack in the Freedom system. Raymond gives a good, but somewhat dated, review of this area [Ray00].

Time-based attack: The time a responder takes to respond to a request may indicate how far away the responder is from the attacker, or how many steps the system took. Especially when combined with a statistical attack, this can provide some information on the responder and help to compromise their anonymity. Adding in artificial delays will help to mask these information leaks but at the cost of performance (although, if implemented correctly the overall delay when downloading a large file can be negligible). Levine et al. have looked at time-based attacks and defences for Mix-based networks [LRWW04], much of which could also be applied to file-sharing systems. Serjantov and Sewell [SS03] look at time-based attacks for connection based systems.

Attacks as Nodes Leave or Join: If a node leaves a network it can no longer send a signal. So, in a system using pseudo identities, if a node leaves the network and the attacker still observes requests using a certain identifier, then the attacker knows that the node that left cannot have had that identity. As more nodes leave a network, the possible pool of nodes for a given long-lived identity will become smaller. In the same way, if an attacker sees n identities in a network with n nodes, then observes a node join and searches messages with a new identity, it is highly probable that the new identity belongs to the new node. The best defence against this sort of attack is for the nodes to regularly change their identities. As long as the life time of an identity is shorter than the average time a node is a member of the system, the system should be safe. Wright et al. [WALS03] describe passive logging and intersection attacks against onion routing when nodes leave and join the system.

Denial of Service Attacks: A peer-to-peer system cannot be used for anonymous downloading if it cannot be used at all. Denial of service (DoS) attacks can be particularly awkward when nodes can act anonymously, as this could mean that the node performing a DoS attack could not be identified and removed from the system. While anonymous systems cannot stop all DoS attacks, care should be taken to ensure that their design does not make DoS attacks particularly easy. Dumitriu et al. [DKK$^+$05] have carried out an in depth study of DoS attacks in non-anonymous peer-to-peer networks. An attacker may carry out a DoS attack on another node in the Ants protocol by repeatedly broadcasting search messages using the other node's pseudo identity. If the attacker sends more search messages than the node under attack, all messages and files sent to the pseudo identity will be routed to the attacker. This could be avoided by using an authentication key as a pseudo address and including a signature of the message identity in the message. This way only the original user can generate new messages.

An attacker can gain a considerable advantage if it can use a targeted DoS attack to knock out a single node. The attacker could then observe the effect on communication of removing a node from the system, or it could remove a number of nodes in order to reshape the network.

5 Conclusion

The implementations described here are all in the early stage of development, so none of them provide the strong guarantees promised by the theoretical models. Currently, the advantage is on the side of the attacker. However, once the systems mature, and provided there is sufficient interaction between theory and implementation, real anonymity will be possible in a practical peer-to-peer file-sharing system.

Interest in the field of anonymity and anonymous peer-to-peer systems has grown rapidly. Papers on this subject are being published continuously; the Free Haven project keeps a useful list of these at `http://freehaven.net/anonbib`.

References

[BASM04] Steve Bono, Christopher A, Soghoian, and Fabian Monrose. Mantis: A high-performance, anonymity preserving, p2p network, 2004. Johns Hopkins University Information Security Institute Technical Report TR-2004-01-B-ISI-JHU.

[BMS01] Adam Back, Ulf Möller, and Anton Stiglic. Traffic analysis attacks and trade-offs in anonymity providing systems. *Lecture Notes in Computer Science*, 2137:245–??, 2001.

[Cha81] David Chaum. Untraceable electronic mail, return addresses, and digital pseudonyms. *Communications of the ACM*, 4(2), February 1981.

[Cha88] David Chaum. The dining cryptographers problem: Unconditional sender and recipient untranceability. *Communications of the ACM*, 24(2), 1988.

[CSWH01] Ian Clarke, Oskar Sandberg, Brandon Wiley, and Theodore W. Hong. Freenet: A distributed anonymous information storage and retrieval system. *Lecture Notes in Computer Science*, 2009:46+, 2001.

[Dee89] Steve Deering. Rfc 1112 "host extensions for ip multicasting", August 1989.

[DFM00] Roger Dingledine, Michael J. Freedman, and David Molnar. The free haven project: Distributed anonymous storage service. In *In Proceedings of the Workshop on Design Issues in Anonymity and Unobservability,*, July 2000.

[DKK$^+$05] D. Dumitriu, E. Knightly, A. Kuzmanovic, I. Stoica, and W. Zwaenepoel. Denial-of-service resilience in peer-to-peer file sharing systems. *SIGMETRICS Perform. Eval. Rev.*, 33(1):38–49, 2005.

[DMS04] R. Dingledine, N. Mathewson, and P. Syverson. Tor: The second-generation onion router. In *Proceedings of the 13th USENIX Security Symposium*, 2004.

[DO00] Shlomi Dolev and Rafail Ostrovsky. Xor-trees for efficient anonymous multicast and reception. *ACM Transactions on Information and System Security*, 3(2):63–84, May 2000.

[Dou02] J. Douceur. The sybil attack, 2002. In Proceedings of the IPTPS02 Workshop, Cambridge, MA (USA), March 2002.

[GSB02] Mesut Gunes, Udo Sorges, and Imed Bouazzi. Ara – the ant-colony based routing algorithm for manets. In *Proceedings of the International Workshop on Ad Hoc Networking (IWAHN 2002)*, Vancouver, August 2002.

[GT96] Ceki Gulcu and Gene Tsudik. Mixing email with babel. In *SNDSS '96: Proceedings of the 1996 Symposium on Network and Distributed System Security (SNDSS '96)*, page 2. IEEE Computer Society, 1996.

[HLX$^+$05] Jinsong Han, Yunhao Liu, Li Xiao, Renyi Xiao, and Lionel M. Ni. A mutual anonymous peer-to-peer protocol design. In *ipdps, vol. 1, no. 1*, page 68. IEEE, 2005.

[Küg03] Dennis Kügler. An analysis of gnunet and the implications for anonymous, censorship-resistant networks. In *Privacy Enhancing Technologies*, Lecture Notes in Computer Science, pages 161–176. Springer, 2003.

[Ley04] John Leyden. Japanese p2p founder arrested. *The Register*, 10th May 2004. http://www.theregister.co.uk/2004/05/10/winny_founder_arrested/.

[LRWW04] Brian N. Levine, Michael K. Reiter, Chenxi Wang, and Matthew K. Wright. Timing attacks in low-latency mix-based systems. In *the Proceedings of Financial Cryptography (FC '04),*, February 2004.

[MOP$^+$04] Alan Mislove, Gaurav Oberoi, Ansley Post, Charles Reis, Peter Druschel, and Dan Wallach. Ap3: A cooperative, decentralized service providing anonymous communication. In *Proceedings of the 11th ACM SIGOPS European Workshop*, Leuven, Belgium, September 2004.

[PK04] Andreas Pfitzmann and Marit Khntopp. Anonymity, unobservability, and pseudonymity: A proposal for terminology, draft v0.21, September 2004.

[Ray00] Jean-François Raymond. Traffic Analysis: Protocols, Attacks, Design Issues, and Open Problems. In H. Federrath, editor, *Proceedings of Designing Privacy Enhancing Technologies: Workshop on Design Issues in Anonymity and Unobservability*, pages 10–29. Springer-Verlag, LNCS 2009, July 2000.

[RD01] Antony Rowstron and Peter Druschel. Pastry: Scalable, distributed object location and routing for large-scale peer-to-peer systems. In *IFIP/ACM International Conference on Distributed Systems Platforms (Middleware)*, pages 329–350, November 2001.

[RR98] M. Reiter and A. Rubin. Crowds: anonymity for web transactions. *ACM Transactions on Information and System Security*, 1(1):66–92, 1998.

[SGR97] P F Syverson, D M Goldschlag, and M G Reed. Anonymous connections and onion routing. In *IEEE Symposium on Security and Privacy*, 1997.

[Shm02] Vitaly Shmatikov. Probabilistic analysis of anonymity. In *IEEE Computer Security Foundations Workshop (CSFW)*, pages 119–128, 2002.

[SLS01] V. Scarlata, B. Levine, and C. Shields. Responder anonymity and anonymous peer-to-peer file sharing, 2001. In Proceedings of IEEE International Conference on Network Protocols (ICNP) 2001.

[SS03] Andrei Serjantov and Peter Sewell. Passive attack analysis for connection-based anonymity systems. In *Computer Security ESORICS 2003.* Springer-Verlag LNCS, October 2003.

[WALS01] M. Wright, M. Adler, B. Levine, and C. Shields. An analysis of the degradation of anonymous protocols. Technical report, University of Massachusetts, Amherst., April 2001.

[WALS03] M. Wright, M. Adler, B.N. Levine, and C. Shields. Defending anonymous communication against passive logging attacks. In *Proc. IEEE Symposium on Research in Security and Privacy*, Berkeley, CA, May 2003.