

An Infrastructure for Anonymous Internet Services

Olaf Landsiedel, Heiko Niedermayer, Klaus Wehrle

Wilhelm-Schickard-Institute for Computer Science
University of Tübingen, Germany

firstname.lastname@uni-tuebingen.de

ABSTRACT

Although recent research provides many techniques for anonymous web-browsing, anonymous internet services, i.e. to run a web server or file server without revealing ones identity, have received little or no attention. In this paper, we present an approach for anonymous internet services, enabling anonymous web and file servers and to run services like anonymous instant messaging and secure shells. Furthermore, we show how both communication partners can communicate anonymously, thus without revealing neither their identities nor the fact that data is exchanged between them.

Similar to a Chaumian Mix, distributed traffic mixes, layered encryption and a novel path concatenation scheme enable anonymous internet services. We present address virtualization to abstract from the user's and system's identity and to provide transparent application support. Thus, no application level gateways or proxies are required to strip away the user's identity from the communication. As IP-datagram service our scheme supports a wide range of applications using TCP and UDP protocols. Finally, we discuss how the presented approach can operate on top of a dynamic P2P overlay network or a dedicated set of traffic mixes.

1. INTRODUCTION

Increasing censorship and the erosion of privacy in digital communication and particularly the internet challenges the freedom of speech. Individual communication and private information is becoming subject to monitoring and surveillance of various state and corporate institutions. Around the world governments have tried and often succeeded to block access to web sites and information considered subversive and not suitable. Even the actions of European and US governments raise questions about privacy and civil liberties. For example, the Federal Bureau of Investigations surveillance system Carnivore, the Patriot Act, and the European Union's "Convention on Cybercrime". These give the authorities in the US and the European Union a wide range of powers to intercept, log and record personal digital communications.

The existence of controversial information is important. However, it also needs to be ensured that controversial information can be accessed without personal consequences. Thus, individuals need to conceal their activities in the internet in order to protect themselves. Anonymous internet services provide a great step towards this.

While designing our scheme, we primarily focused on:

- Privacy and anonymity for information providers and users
- Transparent integration of existing applications ranging from web and file servers to instant messaging and streaming ap-

plications

- Service discovery system, which can base on a Peer-To-Peer infrastructure or a set of dedicated servers and mixes
- A highly scalable, flexible and redundant architecture to ensure high availability

Unlike confidentiality of communication, e.g. data encryption, anonymity cannot be created by the sender or receiver themselves. Alice, Bob and all other communication partners must trust an infrastructure or each other to provide protection, since nobody can decide himself that the message one is going to send is an anonymous one. Furthermore, a number of participants must trust an individual system. Only this ensures that enough traffic is generated to hide the communication of individuals in the masses. Thus, each participant provides traffic for others to hide in and in the same moment benefits from their cover traffic.

Our anonymous communication scheme resides on network layer and provides a service similar to the service of the Internet Protocol (IP). Thus, it only provides an infrastructure supporting internet services, not any internet service itself. Higher level protocols and applications run on top of the anonymous communication scheme and provide services like anonymous web and file servers. Since our design is transparent to them, existing services, e.g. applications and protocols, can be used without modifications.

The remainder of this paper is structured as follows. First, in section 2 we address the short comings of today's anonymous routing systems. Next, we present our design goals in section 3. Section 4 discusses the chosen architecture, including service discovery, path concatenation and address virtualization to provide anonymous internet services. Section 5 evaluates the approach and addresses threats to anonymity. The paper concludes in section 6.

2. RELATED WORK

In this section we address the short-comings of today's anonymous communication schemes. Today's anonymous communication schemes base on the Chaumian Mix [3]. The communication traffic is routed through a cascading set of mixes. The traffic enters one of the mixes and leaves this mix network at some random point. Thus, no relation between sender and receiver can be determined. To reduce the impact of malicious mixes, the route through the mixes is set up via layered encryption. Thus, each mix only can decrypt the information about its successor in the cascade. This design provides sender and relationship anonymity, but no receiver anonymity. Based on this design, various systems have been proposed enabling near real-time communication for services like web browsing: Onion Routing [10], Tor [5], Freedom [2] and Web Mixes [1]. They use a static set of mixes and therefore suffer from drawbacks like limited scalability and fault tolerance. Furthermore,

they are easy to attack via Denial of Service or legal approaches, as the set of mixes is well known. Systems like MorphMix [7] and Tarzan [6] address these issues by the use of Peer-To-Peer networks. However, they suffer from the dynamics of these overlay networks and the limited bandwidth each node can provide. Although Tor provides rendezvous points for anonymous internet services, it does not discuss techniques to prevent applications from revealing their identity in the message content.

Systems like Freenet [4] and Publius [11] allow to publish documents anonymously, but they store the data somewhere in their network. However, due to limited storage space documents can be overwritten. Furthermore, their design approach is not feasible to implement internet services and interactive communication between two nodes.

3. ARCHITECTURAL GOALS

We designed our approach to meet several architectural goals:

- **Sender and receiver anonymity:** The system enables sender and receiver anonymity against malicious nodes via the concatenation of anonymous paths. Thus, it allows anonymous internet services.
- **Relationship anonymity:** Sender and receiver anonymity also implies relationship anonymity.
- **Transparent application support:** Using a virtual network interface and IP-datagram service the system provides transparent application support, e.g. via address virtualization no changes to the applications' implementations are required.
- **Near real-time service:** Via traffic mixes we provide near real-time service.
- **Infrastructure independent:** For traffic mixing and service discovery a Peer-To-Peer network, e.g. benefiting from its high scalability and fault tolerance, or a set of dedicated machines, benefiting from high bandwidth per node and high level of trust, or even combination between these two systems can be used.
- **Practical anonymity:** We do not aim to provide anonymity against a global eavesdropper as our system does not use cover traffic and aims to provide real-time service. We provide practical anonymity against a weaker adversary, who can observe the network partially and participate actively.

4. SYSTEM ARCHITECTURE

In this section we discuss the architecture, focusing on service discovery, transparent application support and encryption. In the Chaumian Mix (see section 2) the sender communicates to the receiver through a number of randomly selected nodes, e.g. mixes. Each of the mixes forwards the data to its successor and thus hides the sender's identity and the relationship of sender and receiver. Compared to this, we additionally ensure receiver anonymity.

4.1 Service Discovery

To provide anonymous internet services, we need some point two communication partners can meet and initiate their communication. Thus, when Bob offers an anonymous service, i.e. a web service, he needs an anonymous ID over which his service can be reached. This ID does not reveal any information about Bob's identity or Bobs service. Furthermore, the design has to ensure, that nobody can impersonate an anonymous ID and so offer – maybe malicious

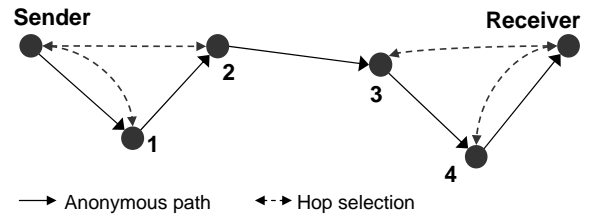


Figure 1: Sender and receiver independently select a number of hops in the anonymous communication path.

– services. We use a public/private key pair to provide the above described features. Thus, Bob computes the key pair and the public key becomes his anonymous ID. As public/private key encryption is considered secure, it is nearly impossible to compute the private key corresponding to Bob's private key and thus impersonate his anonymous identity. This anonymous ID can be published on web sites and other services on the traditional internet; or it can be propagated by out of band methods. This ID and some additional information (see next section) is stored in a service directory. This service can either be a highly scalable and redundant Peer-To-Peer network or a set of dedicated machines. When communicating with the service directory, Bob signs his messages to prove his anonymous identity.

Next to the service providers, the service directory stores the nodes which mix, e.g. relay, the communication traffic. These also have a public/private key pair, which forms their relay identity and protects against impersonation. One can assume that each participating node relays traffic for other nodes to ensure high scalability and to protect its own anonymous communication. However, one can also imagine that nodes may need to fulfill several criteria before they are allowed to relay traffic. This creates an infrastructure, consisting of a set of highly reliable nodes. Last but not least, one can assume a set of dedicated machines to relay traffic, too. These are design criteria and depend on the type of communication traffic to anonymize. However, the type of underlying infrastructure does not influence the communication scheme presented in this paper.

4.2 Anonymous Communication

Together with the anonymous ID, a service provider stores an encrypted communication path in the service directory. Thus, it randomly selects a number of nodes from the service directory to relay its traffic. When Alice wants to access a service, she also selects a number of random nodes to relay her traffic. Assume she chooses to access the service Bob anonymously provides. First, she retrieves the anonymous communication path Bob has published from the service directory. Next, Alice concatenates the nodes she selected and the communication path she retrieved. Thus, she selected the head and Bob the tail of the path (see Fig. 1).

Next to the anonymous path, encryption is needed to prevent that content, relationship and path information cannot be revealed by malicious nodes, eavesdroppers and the participants themselves. As pointed out, each relaying node has a public/private key pair (denoted k_{pub-R}). It is used to encrypt the route via layered encryption. Thus, Bob only publishes the first node in his path information as plaintext and allows via layered public key encryption that each hop in the path is able to decrypt its successor (see Fig. 2). This scheme is called onion routing, as – like an onion – each hop removes one layer of encryption and so can only determine its successor. Thus, when Bob publishes the information, as shown in figure 2, he encrypts his own ID with the public keys of mix 4 and 3. Furthermore, it encrypts the ID of node 4 with the public key of node 3. When concatenating the path section, Alice applies layered encryption with the public keys of the hops it selected. For example, the information about hop 3 in the path is now encrypted

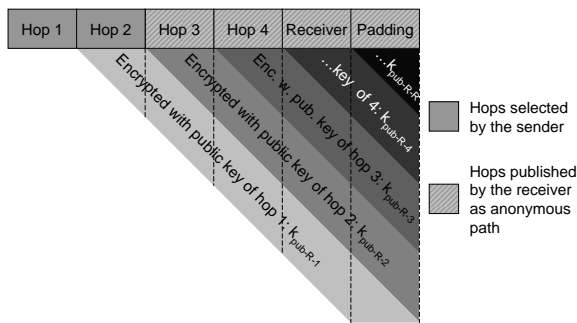


Figure 2: Path information: Similar to onion routing, the layered public key encryption ensures that each hop can only decrypt its successor.

with the public keys of hop 2 and 1. Each hop removes one layer of encryption and so hop 2 can decrypt the information about its successor in the path, e.g. hop 3. Additionally, random padding prevents nodes from identifying the length of the path and their position in it.

Users may change the anonymous paths over time to reduce the possibility of traffic analysis. As described, when publishing a new path section or other information in the service directory, a node signs this with its anonymous ID and use relaying nodes to store it in the directory. Thus, it ensures that nobody can impersonate its anonymous identity and that itself stays hidden.

4.3 Payload Encryption

In the previous paragraph we discussed how the path information is encrypted. Furthermore, the message to transmit, e.g. the payload, itself is encrypted with the anonymous ID of the receiver and each hop always encrypts the message with the ID k_{pub-R} of the next hop.

4.4 Transparent Application Support

Networking applications commonly use TCP and UDP protocols and need an underlying IP-protocol with IP-address. As applications, like `H.323` and `ftp`, tend to communicate their IP-address in the packet payload, the address shall not give any information about the identity of the node. Furthermore, many send control packets to some service point and then rely on this service point to contact them on another port and often different protocol to provide the requested data. Other anonymous communication schemes use application level gateways or proxies, but this requires the gateway or proxy to know the protocol formats of all used applications. With the increasing number of application level protocols ranging from traditional hypertext and file transfer to audio and video streaming, instant messaging and Peer-To-Peer file sharing, such an approach is not feasible. Furthermore, some protocols are proprietary and so their formats are not published. We address the above described problem by the use of an "anonymous communication stack" assigned an virtual IP address, for example `10.x.y.z` (see Fig. 3). The essence of this address virtualization technique is to abstract away any information about the identity of a node. The IP-address is derived from the node's anonymous ID using a hash. However, the address space is too small to neglect collisions, and this addresses an open question in this project. Alternatively, we are discussing the possibility to use the service directory as DHCP-Server.

To communicate the virtual network interface maps the virtual IP-address to the corresponding anonymous ID; next the control application computes an anonymous path as described in section 4.2.

The receiver may publish various anonymous paths tails in the service directory. Thus, a participants's request for the anonymous

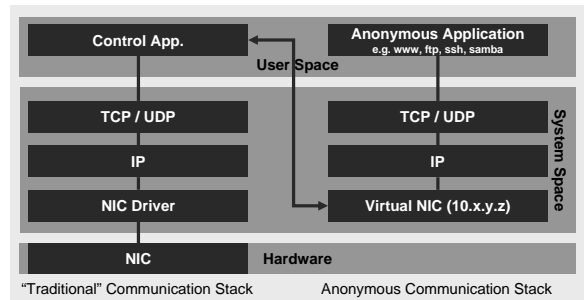


Figure 3: The virtual network interface enables transparent application support.

path tail will result in several path tails. The sender can now select a different path for each packet. As losses, e.g. when a node reached bandwidth limits, will result in retransmissions, the new packet will probably use a different path and so reach its destination. As result, the system adapts to the network dynamics transparently to the user. Additionally, one can imagine to send packets for applications which require real-time performance, e.g. streaming or even `H.323`, multiple times using a different path for each packet. As the anonymous application has its own protocol stack, its protocols will detect multiple packets and discard them. These two described techniques are essential when the mix network does not consist of a set of reliable dedicated machines, but of a highly dynamic Peer-To-Peer system, where nodes can leave the network without announcement.

Next, we will show how typical widely-used protocols can benefit from the presented virtualization approach.

4.4.1 FTP

FTP uses TCP to reliably transfer files over the internet. FTP relies on multiple parallel TCP sessions to transfer a file. The file transfer is initiated by a client (Alice) via the well-known port 21 of the server. This is the so called control session in which the IP address of the initiator is transmitted in the packet payload. Thus, we need an virtual IP address, as we do not want to modify the communication payload. To transfer the file itself, a data session, initiated by the server (Bob) to port 20 of the client, is used. And this forms another dilemma, as Bob does not (and shall not) know the IP address of Alice. However, using a virtual IP, Bob now can retrieve the anonymous ID of this virtual IP and corresponding path information from the service directory. Now, Bob can compute an anonymous path to Alice and initiate the corresponding TCP data connection.

4.4.2 Real-Audio

Real-audio clients access a real-audio server via TCP port 7070 to set up an audio stream and to exchange control message during stream playback. The audio data then is send from the server to the client in UDP packets on a different port range. This results in a dilemma similar to the one described above.

4.4.3 H.323

`H.323` is a very complex protocol. We just want to mention that it uses dynamic ports and the packets contain control information like the IP addresses and ports of the communication partners. Information is encoded in ASN.1 and to make things worse, session information can be encrypted. This would forces proxies and application level gateways to man-in-the-middle attacks to fully support these applications. However, using a virtual IP-address, the client and server will embed the virtual IP-address in the bit-stream and not reveal their own identities.

4.5 Name Service

For a seemingness integration of existing applications, a name service is required. Existing Web browsers and other web applications do not provide ways to enter anonymous IDs. Furthermore, users do not want to type a public key into a browser window to access a service. Thus, we provide a name service, transparent to existing DNS, to map between DNS requests and our scheme. Thus, a user can enter `http://www.freespeech.anon`. Next, the system sends an DNS request in order to resolve this domain. Our scheme reroutes this request to the service directory and determines the virtual IP and corresponding communication paths from the service directory (see Fig. 4). To distinguish between traditional domain names and the anonymous ones, we introduce new domain names, like `.anon`, as used in the previous example.

5. ANALYZING THE ARCHITECTURE

In this section we analyze the presented system for anonymous internet services. We evaluate the communication overhead and delays introduced by service discovery and the cascading mixes. The architecture presented in this paper is ongoing work, thus we only provide an analytical analysis. Finally, we present our threat model and discuss threats to anonymity and privacy.

5.1 Communication Evaluation

Obviously communication via an anonymous path increases transmission latency. However, this is an inherent property of all anonymous communication schemes. The number of mixes and their position in the network determines the delay. To reduce delay, one can choose mixes close to each other and reduce their number – the price is reduced privacy. The access to the service directory results only in minimal overhead, compared to traditional DNS lookup. Instead to access a DNS server, the system contacts the service directory and retrieves the virtual IP-address and path information. Only when the anonymous server sends its response, we introduce little overhead, as the server also needs to contact the service directory to retrieve an anonymous path.

5.2 Threat Model

It would be nice to protect against a global passive adversary. However, like all practical anonymous communication schemes, that enable low-latency communication, we do not protect against such a strong adversary. Instead we assume a practical adversary who can:

- Observe some part of the network
- Participate actively by relaying traffic of other nodes, e.g. running one or more own instances of the anonymous communication scheme.
- Compromise a limited number of nodes
- Influence communications by generating, delaying and modifying traffic content and patterns.

5.3 Security Analysis

In this section we discuss possible attacks to our anonymous communication system. We address various active and passive attacks.

Passive attacks:

Content observation: As communication traffic is encrypted, its observation will not reveal content.

Source / destination observation: As traffic is relayed, its observation will not reveal source or destination. As nodes relay traffic for other nodes and messages are padded to constant length, it is

not possible to determine via observation whether a node is sender, relay or receiver of a message.

Pattern observation: Although traffic observation does neither reveal content nor sender and receiver, it may reveal patterns. However, such patterns can be the result of various applications and nodes using a particular node as relay at the same time and so do not reveal information.

End-to-end timing correlation: When an adversary has some idea about the sender and receiver identity, end-to-end timing correlation confirms such a relation with high probability. We do not provide much protection against such attacks, but it is likely that each participant also relays traffic of other users. Thus, a weak adversary cannot determine where the end points of a communication are and so this attack is only useful for strong adversaries.

End-to-end packet size correlation: Packet size correlation – like the timing correlation described above – is only feasible for strong adversaries. Using packets of constant length and delaying traffic at relaying nodes strengthens the architecture against powerful adversaries, but this will introduce more overhead and latency. Thus, we are still discussing this feature.

Active attacks:

Compromise multiple relays: An adversary may run own instances of the anonymous communication scheme or compromise others and so try to identify the hops of an anonymous communication route. As it takes time to compromise nodes, it is important that nodes change their overlay ID and so their encryption keys faster than an adversary is expected to propagate along an anonymous communication path. When a node computes a new ID, it deletes the old key set and so renders this attack useless. Additionally, the destination of a communication cannot be revealed without compromising the destination itself, as route information contains padding and so hides the end point. Only when an adversary runs a large number of relays, there are chances that a user may select an anonymous path only through these nodes and so reveal its identity (but not the message content). Furthermore, nodes can select their hops from different IP-address ranges as it is difficult for an adversary to obtain addresses in various IP-ranges [6].

Compromise keys: Of course, an adversary may try to compromise a key, but we consider such an attack not realistic as keys change over time and public / private key encryption itself is considered secure.

Offer own service: An adversary may offer anonymous internet services or access them itself and so identify or even produce traffic patterns. Thus, making end-to-end traffic analysis attacks easier. In this case the adversary also selects a part of the anonymous communication path and so already knows a number of hops, which can be under his control, too. However, it is one of our basic design principles, that each communication partner selects a number of nodes in the anonymous communication path. Thus, each communication partner needs to select a set of nodes to "hide" behind these.

Denial of service: An attacker may use DOS attacks against nodes not in its control to increase traffic through controlled nodes. Additionally, an adversary may attack the entry point of a published communication path and so disable communication to a certain service. However, publishing multiple entry points and changing them over time makes such an attack not practical.

Marking, tagging and replacing: End-to-end encryption and integrity checks prevent attacks on marking, tagging and replacing messages.

Replay attacks: Replay attacks are addressed with key changes as well as short time message hashing. For this, nodes store a hash

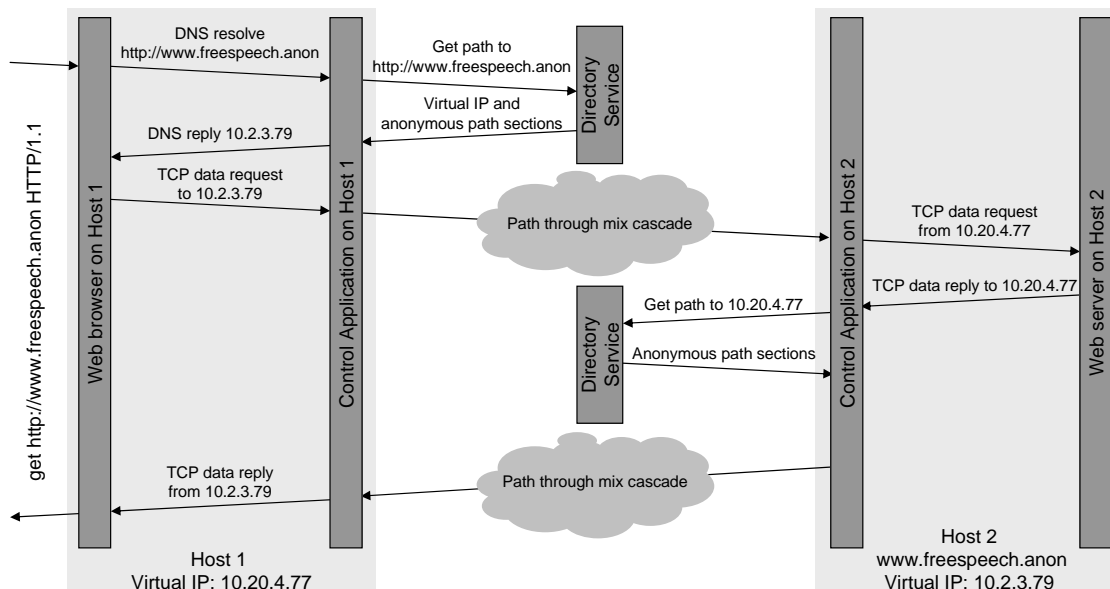


Figure 4: Seemingness integration of existing applications into the anonymous communication scheme: In this example a participant uses a web browser to request the site `http://www.freeseech.anon` from an anonymous web server.

of each relayed message over a certain amount of time. Incoming messages are hashed and only relayed when the hash has not been encountered before.

Impersonation: An adversary also may try to impersonate somebody else's anonymous identity, for example some popular anonymous internet service. As stated in 4.3 each message about the anonymous identity or publishing path sections is signed with the anonymous ID and so cannot be impersonated.

Attacks against the service directory:

An adversary may attack the service directory to render the mixes useless. As the amount of data stored in the service directory is small, a high degree of redundancy, e.g. multiple distributed servers, are the best answer to this threat. Furthermore, it is very feasible to use a Peer-To-Peer network to store this data. This provides the required high scalability, fault tolerance and redundancy [9].

Moreover, the service directory only stores anonymous path sections. Thus, when an adversary gains control of the server, no information can be revealed. To ensure privacy, access to the service directory is anonymous, too. Thus, it uses the relay nodes.

6. CONCLUSION

In this paper we present a novel approach to anonymous internet communication. Our approach enables anonymous internet services, ensuring sender, receiver and relationship anonymity. Its transparent architecture and name service allows a seemingness integration of existing web applications ranging from web and file servers to instant messaging and secure shell login without changes to the applications or communication protocols. Furthermore, we address questions and problems not identified by other approaches providing anonymous internet services [5]: we use address virtualization and an own communication stack to provide the transparent application support. Thus, applications cannot reveal the identify of a node in the message payload and so we guarantee receiver anonymity. Future work focuses on simulation to optimize various design parameters and to continue the ongoing implementation.

Concluding, the possibility not only to publish data anonymously, but to offer internet services without revealing ones identity, brings anonymous networking to a new level.

7. REFERENCES

- [1] O. Berthold, H. Federrath, and S. Köpsell. Web MIXes: A system for anonymous and unobservable Internet access. In *Proc. of Designing Privacy Enhancing Technologies: Workshop on Design Issues in Anonymity and Unobservability*, July 2000.
- [2] P. Boucher, A. Shostack, and I. Goldberg. Freedom systems 2.0 architecture. White paper, Zero Knowledge Systems, Inc., Dec. 2000.
- [3] David L. Chaum. Untraceable electronic mail, return addresses, and digital pseudonyms. *Communications of ACM*, Feb. 1981.
- [4] I. Clarke et al. Freenet: A distributed anonymous information storage and retrieval system. In *Proc. of Designing Privacy Enhancing Technologies: Workshop on Design Issues in Anonymity and Unobservability*, July 2000.
- [5] R. Dingledine, N. Mathewson, and P. Syverson. Tor: The second-generation onion router. In *Proc. of 13th USENIX Security Symposium*, August 2004.
- [6] M. J. Freedman and R. Morris. Tarzan: A peer-to-peer anonymizing network layer. In *Proc. of 9th ACM Conference on Computer and Communications Security (CCS)*, Nov. 2002.
- [7] M. Rennhard and B. Plattner. Introducing MorphMix: Peer-to-Peer based Anonymous Internet Usage with Collusion Detection. In *Proc. of Workshop on Privacy in the Electronic Society (WPES)*, Nov. 2002.
- [8] S. Rieche et al. Reliability of data in structured peer-to-peer systems. In *Proc. of HOT-P2P Workshop*, Oct. 2004.
- [9] I. Stoica et al. Chord: A scalable Peer-To-Peer lookup service for internet applications. In *Proc. of ACM SIGCOMM Conference*, Feb. 2001.
- [10] P. F. Syverson, D. M. Goldschlag, and M. G. Reed. Anonymous connections and onion routing. In *IEEE Symposium on Security and Privacy*, Feb. 1997.
- [11] M. Waldman, A. Rubin, and L. Cranor. Publius: A robust, tamper-evident, censorship-resistant and source-anonymous web publishing system. In *Proc. of 9th USENIX Security Symposium*, August 2000.