Slicing the Onion: Anonymous Routing Without PKI

Sachin Katti
skatti@mit.edu

Dina Katabi
dk@mit.edu

Katarzyna Puchala
kpuchala@mit.edu

Abstract– Recent years have witnessed many proposals for anonymous routing in overlay peer-to-peer networks. To provide both sender and receiver anonymity, the proposed protocols require the overlay nodes to have public-private key pairs, with the public keys known to everyone. In practice, however, key distribution and management are well-known difficult problems that have crippled any widespread deployment of anonymous routing. In this paper, we propose a novel protocol that uses a combination of information slicing and source routing to provide anonymous communication similar to Onion Routing but without a public key infrastructure.

1 INTRODUCTION

Anonymous routing plays a central role in private communication. Its applications range from file sharing to military communication, and include anonymous email, private web browsing and online voting. Traditionally, anonymous routing has required the help of a trusted third party, which either acts as a centralized proxy [1, 3], or provides the sender with the public keys of selected willing relays [2, 8]. However, the recent success of peer-to-peer systems has evoked interest in using them as anonymizing networks. Indeed, the large number of nodes (a few million [14]) and the heterogeneity of their location, communication patterns, political background and local jurisdiction make these networks ideal environments for hiding anonymous traffic. Many systems have been designed to exploit peer-to-peer overlays in anonymous communication, including Tarzan [10], AP3 [15], MorphMix [17] and Cashmere [19]. But to provide sender and receiver anonymity, these systems require the overlay nodes to have public-private keys obtained through a trusted authority; i.e., they require a public key infrastructure (PKI).[1]

But why is PKI problematic for peer-to-peer anonymizing networks? Key distribution and management are well-known difficult problems [4]. In particular, prior work assumes the sender knows a priori the public keys of all relay nodes [10, 15, 19]. The underlying assumption is that a trusted third party generates all keys and distributes them to the nodes. But such an approach is problematic for many reasons. First, in a large peer-to-peer network, the trust model may differ from one node to another (e.g., nodes in Cuba may not trust the same PKI as nodes in the US). Second, it opens up the system to attacks on the key distribution procedure and compulsion attacks that force the key originator to disclose the keys under the threat of force or if required by a court order [12, 13]. Indeed, some countries have provisions that allow them to legally request the decryption of material or the handing over of cryptographic keys [6, 9]. Additionally, PKI makes anonymous multicast difficult as all recipients of a multicast message have to share the same public private key pair. Finally, with time, an increasing fraction of the keys can get stolen off compromised machines. This necessitates the existence of key management and update protocols, complicating the problem further.

This paper shows how to perform anonymous *Onion Routing* without PKI. Onion routing [11] is at the heart of most prior work on peer-to-peer anonymizing networks [8, 10, 15, 19]. It uses a form of source routing, in which the IP address of each node along the path is encrypted with the public key of its previous hop. This creates layers of encryption–layers of an onion. To send a message, each node decrypts one layer, discovers its next hop, and forwards the message. Thus, each relay node knows only its previous and next hops; it cannot tell the sender, the receiver, the path, or the content of the message. Our scheme provides similar anonymity but without PKI.

Our approach is based on the simple but powerful idea of *Information Slicing*. To provide anonymous communication, each node along the path, the destination included, needs a particular piece of information, which should be hidden from other nodes in the network. For example, the destination needs to learn the content of the message without revealing that content to other nodes, while each intermediate relay needs to learn its next hop without other nodes in the network knowing that information. We divide the information needed by a particular node into many small random pieces. These information pieces are then delivered along disjoint paths that meet only at the intended node.Thus, only the intended node has enough bits to decode the information

[1] A few systems (e.g., Crowds [16]) do no require PKI, but they expose the receiver and message content.

content. We call this approach information slicing because it splits the information traditionally contained in an onion peel (i.e., the ID of the next hop) into multiple pieces/slices.

Anonymity via information slicing is not as straightforward as it sounds. To send a particular node the identity of its next hop along different anonymous paths, one needs to anonymously tell each node along these paths about its own next hop. Without careful design, this may need an exponential number of paths. Our keyless onion routing algorithm provides efficient information slicing using *a small constant* number of paths.

The rest of the paper describes the details of our information slicing protocol, and shows how to construct forwarding graphs that deliver anonymous messages using a small number of paths. It also presents our preliminary implementation results showing that the latency of setting up anonymous routes in our scheme is low enough to be practical.

2 Goals & Model

The objective of this work is to enable large and fully distributed peer-to-peer anonymizing networks. We focus on pragmatic anonymity for non-military applications, such as file sharing, private email and the communication of medical records. These applications strive for privacy but can deal with low probability of information leakage.

We assume an adversary who can observe some fraction of network traffic, operate relay nodes of his own, and can compromise some fraction of the relays. We do not protect against a global attacker who can snoop on all links. Though such an adversary is usually assumed when analyzing theoretical anonymity designs, all practical low-latency anonymizing systems, ours included, do not protect against such an adversary [8, 10, 15, 17, 19]. Also, similar to prior work [8, 10, 15, 19], we generate enough cover traffic to prevent simple traffic analysis attacks.

We also assume the sender can send from multiple IP addresses, and a secure channel like `ssh` is available between them. Many people have Internet access both at home and at work/school, and thus, can send from different IP addresses. Alternatively, the sender may have both DSL and cable connectivity. Or, he may belong to a multi-homed organization. For example, each of the authors has Internet access at home, as well as at school and on Planetlab machines. We believe that a large number of Internet users can send from multiple accounts with different IP addresses. An attacker may try to correlate IP addresses belonging to the same sender. However, in all of the examples above the IP addresses used belong to different domains. Additionally, most broadband providers and companies utilize NAT, preventing the association of an IP address with a particular user.

Last, we assume the attacker cannot snoop on all links

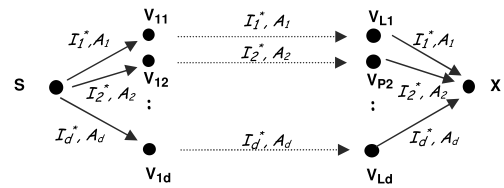

Figure 1—Node S sends a confidential message $\vec{m}$ to X by first multiplying the message with a random matrix $\vec{I}^* = A\vec{m}$, then splitting the resulting information content into multiple pieces, each follows a disjoint path to X. Only X receives enough information bits to decode the original message as $\vec{m} = A^{-1}I^*$.

leading to the receiver. This assumption can be ignored if the sender knows the receiver's key, which guarantees message confidentiality even if the attacker can collect all information slices sent to the receiver.[2]

3 Anonymous Communication Without PKI

Our approach to anonymity without PKI stems from a simple observation: anonymity can be built out of confidentiality. In particular, for anonymous communication, the source needs to send every relay node along the path its routing information (i.e., its next hop) in a confidential message, accessible only to the intended relay. One can send confidential messages without keys using a simple building block: information slicing.

Consider the scenario in Fig. 1, where sender, S, wants to send message m to node X. The sender divides the message into d blocks m_i, $\forall i \in \{1, \ldots, d\}$, such that the original message can be recovered only when a node has access to all d blocks. Sending a message block m_i in the clear may expose partial information to intermediate nodes. Thus, the sender multiplies the message vector $\vec{m} = (m_1, ..., m_d)$ with a *random but invertible* matrix A and generates d slices which constitute a random version of the message:

$$\vec{I}^* = A\vec{m}.$$

Then, the sender picks d disjoint paths to node X. It sends on path i both the slice I_i^* and A_i, where A_i is row i of A. An intermediate node sees only some random values I_i^* and A_i, and thus cannot tell the content of the message. Once the receiver receives all slices, it decodes the original message as:

$$\vec{m} = A^{-1}\vec{I}^*.$$

This slicing mechanism could be considered as a variation on the concept of secret sharing [18] customized for the our problem (see §6).

[2] Note that knowing the receiver's key is a much weaker constraint that knowing the keys of the overlay nodes, as in many instances of private communication the sender and receiver know each other.

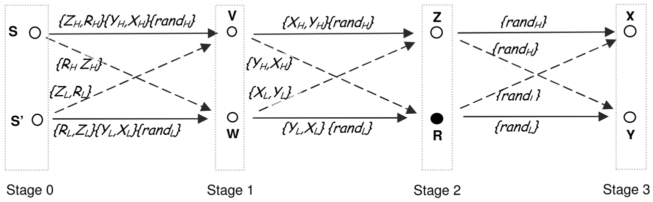

Figure 2—An example of anonymous routing with information slicing. Nodes S and S' are controlled by the sender. A message like $\{Z_l, R_l\}$ refers to the low-order words of the IDs of nodes Z and R, *rand* refers to random bits.

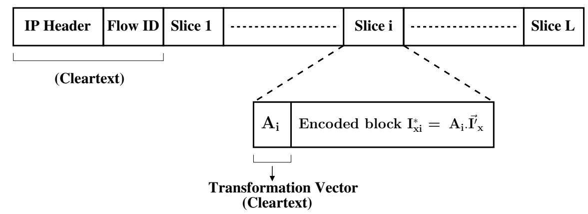

Figure 3—Packet Format. Each packet contains L information slices.

However, using information slicing to provide anonymity is tricky. To send a particular node the identity of its next hop along different anonymous paths, one needs to anonymously tell each node along these paths about its own next hop. This will need an exponential number of paths. To avoid exponential blow up, it is essential that the sender construct efficient forwarding graphs and divide the information smartly. The construction of these graphs in the general case is fairly complex. Thus, we first explain a simplified example in §3.1, while leaving the details of our routing protocol to §4.

3.1 Example of Anonymous Routing via Information Slicing

We use a simple scenario to show how to provide anonymity through information slicing. In onion routing, a node learns its next hop from its parent. Though the parent delivers this information to its child, it cannot access it because the information is encrypted with the child's public key. In the absence of keys, the path cannot be included in the message as that allows any intermediate node to learn the whole path from itself to the receiver. We need an alternative method to tell a node about its next hop without revealing it to other nodes, particularly the parent node.

How to preserve anonymity without a PKI? Fig. 2 shows an example keyless anonymous routing graph. Assume the sender has access to two IP addresses S and S'. To send an anonymous message to node R, the sender, in Fig. 2, has picked a few relay nodes at random. It has arranged them, with the receiver, into 3 stages (path length $L = 3$), each containing 2 nodes (split factor $d = 2$). The 0'th stage is the source stage itself. Each node in this graph is connected to every node in its successive stage. Also, note that the receiver node (the solid node labeled R) is randomly assigned to one of the stages in the graph.

Next, the sender in Fig. 2 wants to send each relay the IP address of its next hop by splitting this information over 2 paths. The sender could have split each IP address to its most significant and least significant words. This however is undesirable as most significant word may indicate the owner of the IP prefix. Instead the sender transforms the IP addresses of the relay nodes by multiplying each address by an *invertible matrix* A of size $d \times d$ (2×2). For example, assume V_l and V_h are the the low and high words of the IP address of node V; the sender splits the IP address as follows:

$$\begin{pmatrix} V_L \\ V_H \end{pmatrix} = A \begin{pmatrix} V_l \\ V_h \end{pmatrix} \quad (1)$$

and sends V_L and V_H to V's parents along two different paths.

Fig. 2 shows how messages are forwarded such that each node knows no more than its direct parents and children. Consider an intermediate node in the graph, say V. It receives the message $\{Z_H, R_H\}\{X_H, Y_H\}\{rand_H\}$ from its first parent S. It receives $\{Z_L, R_L\}$ from its second parent S'. After receiving both messages, V can discover its children's IP addresses as follows:

$$\begin{pmatrix} Z_l & R_l \\ Z_h & R_h \end{pmatrix} = A^{-1} \begin{pmatrix} Z_L & R_L \\ Z_H & R_H \end{pmatrix} \quad (2)$$

But V cannot tell the children of its children (i.e., the children of nodes Z and R) because it misses half the bits in these addresses, nor does it know the rest of the graph. The same argument applies to other nodes in the graph.

You might be wondering how the graph in Fig. 2 will be used to send confidential messages to node R, without exposing the content of the messages to other nodes. Indeed, as it is, R does not even know it is the intended receiver. But this is easy to fix. In addition to sending each node its next hop IPs, we send it: (1) a key and (2) a flag indicating whether it is the receiver. Similar to the next hop, the key and the flag are also split along disjoint paths, and thus inaccessible to other nodes. Now every node in the graph shares a secret key with the sender. The sender can then use traditional onion routing to forward the message to the receiver, encrypting the message in layers using the secret keys of the nodes in the graph.

4 Information Slicing Protocol

We use the intuition from the previous section to construct an anonymous routing protocol based on information slicing.

4.1 Per Node Information

Let x be one of the nodes in the forwarding graph. I_x is the information the sender needs to *anonymously* deliver to node x. I_x consists of the following fields:

- *Nexthop IPs.* The IP addresses of the d children of node x.
- *Nexthop flow-ids.* These are d 64-bit ids whose values are picked randomly by the sender and are to be put in the clear in the packets going to the corresponding d next-hops. The sender ensures that different nodes sending to the same next hop put the same flow-id in the clear. This allows the next-hop to determine which packets belong to the same flow. The flow-id changes from one relay to another to prevent the attacker from detecting the path by matching flow-ids.
- *Receiver Flag.* This flag indicates whether the node is the intended receiver.
- *Secret Key.* The sender sends each node along the path a secret key which can be used to encrypt any further messages intended to this node.

4.2 Creating Information Slices

The node information I_x is chopped into d blocks of $\frac{|I_x|}{d}$ bits each and a d length vector $\vec{I'_x}$ is constructed. Further, $\vec{I'_x}$ is transformed into coded *information slices* using a full rank $d \times d$ random matrix A as follows:[3]

$$\vec{I^*_x} = \begin{pmatrix} A_1 \\ \vdots \\ A_d \end{pmatrix} \vec{I'_x} = A\vec{I'_x} \tag{3}$$

We call the elements in $\vec{I^*_x}$ *information slices*. We also add to information slice I^*_{xi} the row of the matrix A which created it i.e. A_i. The sender delivers the d slices to node x along disjoint paths.

4.3 Packet Format

Fig. 3 shows the format of a packet used in our system. In addition to the IP header, a packet has a flow id, which allows the node to identify packets from the same flow and decode them together. The packet also contains L slices. The first slice is always for this node (i.e., for the receiver of the packet). The other slices are for nodes downstream on the forwarding graph.

4.4 Constructing the Forwarding Graph

The sender constructs a forwarding graph which routes the information slices to the respective nodes along vertex disjoint paths, as explained in Algorithm 1. We demonstrate the algorithm by constructing such a graph in Fig. 4, where $L = 3$ and $d = 2$. We start with the 2 nodes in the last stage, X and Y. The sender assigns both the slices, I^*_{X1}, I^*_{X2} to X. Then it goes through the preceding stages, one by one, and distributes (I^*_{X1}, I^*_{X2}) among the 2 nodes at each stage; each node receives one of the slices. The path taken by slice I^*_{X1} to reach X can be constructed by tracing it through the graph. Slice I^*_{X1} traverses (S', W, Z, X), which is disjoint from the path taken by I^*_{X2}, i.e., (S, V, R, X). The source repeats the process for the slices of Y and every other node in every stage.

Algorithm 1 Information Slicing Algorithm

> Pick Ld nodes randomly including the destination
> Randomly organize the Ld nodes into L stages of d nodes each
> **for** Stage $l = L$ to $l = 0$ **do**
>   **for** Node x in stage l **do**
>   Assign to node x its own slices $I^*_{xk}, k \in (1, \ldots, d)$.
>   **for** Stages $m = l - 1$ to $m = 1$ **do**
>   Distribute slices $I^*_{xk}, k \in (1, \ldots, d)$ uniformly among the d nodes in stage m, assigning one slice per node
>   **end for**
>   **end for**
>   Connect every node in stage $l - 1$ to every node in stage l by a directed edge going towards l
>   **for** every edge e **do**
>   Assign the slices which are present at both the nodes at the endpoints of the edge e to the packet to be transmitted on e.
>   **end for**
> **end for**

Slices are delivered in packets transmitted between nodes in successive stages. The slices a node sends to its downstream neighbor are the intersection of the sets of slices assigned to both nodes by Algorithm 1. E.g., for edge (V, R), the slices (I^*_{R2}, I^*_{X2}) are present at both nodes V and R. These slices are contained in the packet transmitted from node V to node R. The source determines the packet contents for every edge in the graph. The algorithm thus ensures that slices belonging to a node take vertex disjoint paths to the node.

4.5 Decoding the Information Slices

A node can decode its information from the d slices it receives from its parents. The first slice in every packet node x receives is for itself. It consists of one of d-slices of x's information, I^*_{xi}, and the row of the transform matrix that helped create it, A_i. Node x constructs the $d \times 1$ vector $\vec{I^*_x}$ from the d slices it receives, and assembles a $d \times d$ matrix $A = [A_1; \ldots; A_d]$ from the d rows of the transform matrix sent in the slices. Then, node x computes its information vector, $\vec{I'_x}$, as $\vec{I'_x} = A^{-1}\vec{I^*_x}$. Node x lays out the elements of $\vec{I'_x}$ next to each other to reconstruct the original information sent to it by the source, as shown in Fig. 5. The node then recovers the IP addresses of each of its d next hops along with the flow id to be put on packets destined to that next hop.

[3]Elements of $\vec{I'_x}$ and A belong to a finite field F_{p^q} where p is a prime number and q is a positive integer. All operations are therefore defined in this field and differ from conventional arithmetic.

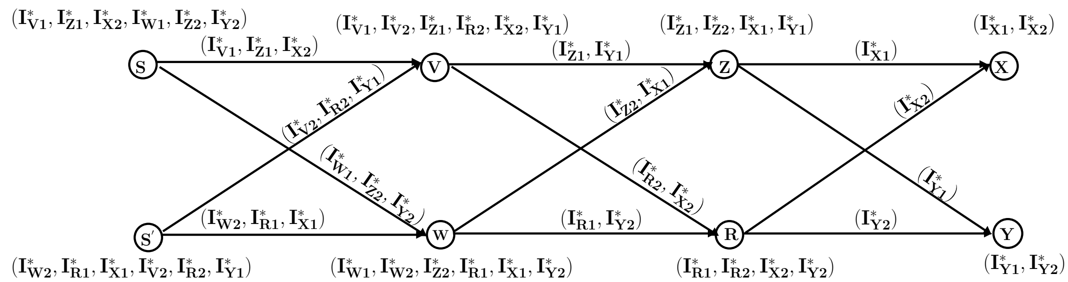

Figure 4—An example showing how to split information slices along disjoint paths. R is the receiver, S and S' are the senders.

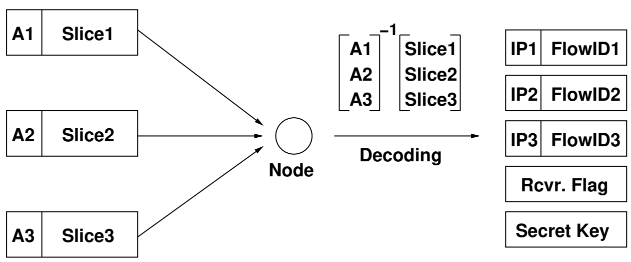

Figure 5—An example showing how a node decodes its information from its incoming slices. It uses the 3 incoming slices and reconstructs the original information by inverting the matrix A and gets the IP addresses of its next hops as well as the flows ids and its secret key.

It also recovers its key and the receiver flag, which tells it whether it is the intended receiver for the eventual message.

4.6 Data Transmission

After the forwarding graph has been set up and every node has a shared key with the source, the source uses traditional onion routing to transmit its data messages. In the graph we setup in the initial phase, each node has d next hops. The source picks one path within this graph to be used for delivering data messages using traditional onion routing. It encrypts each message with the receiver's key and then encrypts the encrypted message with the previous hops key and so on. Each node can then decrypt incoming packets with its own key and forward it to the next hop.

5 Performance

We have implemented our scheme in Python, and performed preliminary tests on a 100 Mbps switched network with the tested relay daemons running on 2.8 GHz Pentium boxes with 1 GB of RAM and a Linux 2.6.11 kernel. Table 1 shows route setup latency for different path lengths and split factor of 2. Setup latency is measured end-to-end from when the sender initiates route establishment, connects to the stage-1 relay processes, which process the next hop computation, store the forwarding information, and connect to their next hop relays, which repeat the procedure until all routing messages reach the receiver. On average, we incur a setup cost of 19 ms per hop. This figure suggests that the latency of the underlying network will dominate even during route setup. The table shows that the route setup latency incurred by our scheme is comparable to other anonymous routing protocols such as Tarzan [10], and is low enough to make it practical.

Route Length & Split factor	Setup Latency (ms)	Standard Deviation (ms)
L=1, D=2	11.59	1.88
L=2, D=2	39.05	4.20
L=3, D=2	61.14	9.33
L=4, D=2	89.86	7.56
L=5, D=2	109.12	11.09

Table 1—Setup latency in milliseconds and its variance for the construction of multi-hop routes through pre-defined relays.

6 Related Work

Related work can be divided into two areas: anonymous communication and secret sharing.

(a) Anonymous Communication: Research on anonymous communications started in 1981 with Chaum's seminal paper [7]. Early anonymous systems redirected traffic through a trusted centralized proxy [1, 3]. The services failed after a lawsuit attempted to force the administrator to reveal confidential information [12].

Modern anonymizing systems are based on the principle of onion routing [11]. The communication traverses a sequence of relays, which the sender picks from a set of willing nodes. The sender recursively encrypts the message and the address of the next hop with the public key of each relay on the path. Recently, there has been a number of proposals on peer-to-peer anonymizing networks based on onion routing. The large size and diversity of nodes in a peer-to-peer network makes collusion difficult, hence they are promising candidates for anonymous communication. Many such systems exist including Tarzan [10], MorphMix [17], AP3 [15], and Cashmere [19], all of which assume a centralized trusted PKI.

In contrast to the above, Crowds [16] does not rely on onion routing. In Crowds, a user relays her message by passing it to a randomly selected node in the crowd. Upon receiving a message, a relay node forwards the message to its final receiver with probability p and sends it to another randomly picked relay with probability $1 - p$. Thus, Crowds does not require PKI. However, Crowds does not provide the anonymity our system provides: it exposes the receiver and the message content, and a snooping attacker can easily correlate the clear-text of the message on various links.

(b) Secret Sharing: A secret sharing scheme is a method for distributing a secret among a group of participants, each of which is allotted a *share* of the secret. The secret can only be reconstructed when the shares are combined together, individual shares are of no use on their own. Research on secret sharing started with Shamir's seminal paper [18]. Information slices can be viewed as *shares* of a secret. Similar to secret sharing, a node can reconstruct the original information only when it gets all the slices. Indeed there are linear block code based secret sharing [5] schemes, but our work is the first to apply secret sharing to anonymous communication.

7 Conclusion and Future Work

We have shown it is possible to design anonymizing peer-to-peer overlays that do not need a public key infrastructure (PKI). Our information slicing protocol can hide the source, the destination, the path, and the content of the message, even when the sender does not know the public keys of the nodes in the overlay. We believe this is an important step towards truly peer-to-peer anonymous communications; it obviates the need for a universal trusted PKI and avoids the difficulties of large scale key distribution in a global peer-to-peer network.

A number of avenues for future work suggest themselves. Node churn is quite significant in peer to peer overlay networks. We are investigating how to make our slicing protocol robust against node departures in the middle of a session. Snooping attackers can link similar incoming and outgoing packets at a relay node to detect the chain of forwarders, and eventually trace the source. We are investigating techniques to make packets belonging to the same flow bitwise unlinkable, thereby preventing such attacks. Finally, a true evaluation of a system happens only when it is deployed and used in the real Internet; we are building and deploying the Information Slicing protocol on the Planetlab overlay network.

8 Acknowledgments

We thank Jeremy Stribling, Vinod Vaikuntanathan, Hariharan Rahul, Maxwell Krohn and Frank Dabek for their comments on the paper. The authors acknowledge the support of NSF under NSF Career Award CNS-0448287. The opinions and findings in this paper are those of the authors and do not necessarily reflect the views of NSF.

References

[1] Anonymizer- Anonymous Web Surfing. http://www.anonymizer.com.

[2] MixMinion- Anonymous Remailer. http://www.mixminion.net.

[3] Safeweb- Anonymous Web Surfing. http://www.safeweb.com.

[4] M. Bellare and P. Rogaway. Entity authentication and key distribution. In *CRYPTO '93*.

[5] M. Bertilsson and I. Ingemarsson. A construction of practical secret sharing schemes using linear block codes. In *AUSCRYPT '92*.

[6] Bob Sullivan. FBI software cracks encryption wall. www.msnbc.com/news/660096.asp?cp1=1.

[7] D. L. Chaum. Untraceable Electronic Mail, Return Addresses and Digital Pseudonyms. Commun. ACM 21, 2(1981).

[8] R. Dingledine, N. Mathewson, and P. Syverson. Tor: The second-generation onion router. In *USENIX Security 2004*.

[9] Foundation for Information Policy Research. Regulation of Investigatory Powers Information Centre. www.fipr.org/rip/.

[10] M. J. Freedman and R. Morris. Tarzan: A peer-to-peer anonymizing network layer. In *Proceedings of ACM CCS 2002*.

[11] D. M. Goldschlag, M. G. Reed, and P. F. Syverson. Hiding Routing Information. In *Proceedings of Information Hiding: First International Workshop*, 1996.

[12] Johan Helsingius. anon.penet.fi is closed! www.penet.fi.

[13] John Markoff. New File-Sharing Techniques Are Likely to Test Court Decision. The New York Times, Aug 1.

[14] Kazaa. http://www.kazaa.com/.

[15] A. Mislove, G. Oberoi, A. Post, C. Reis, P. Druschel, and D. Wallach. Ap3: A cooperative, decentralized service providing anonymous communication. In *Proceedings of the 11th ACM SIGOPS European Workshop*, September 2004.

[16] M. Reiter and A. Rubin. Crowds: Anonymity for web transactions. *ACM Transactions on Information and System Security*, 1(1), June 1998.

[17] M. Rennhard and B. Plattner. Introducing MorphMix: Peer-to-Peer based Anonymous Internet Usage with Collusion Detection. In *WPES 2002*, Washington, DC, USA.

[18] A. Shamir. How to share a secret. In *Communications of the ACM, 1979*.

[19] L. Zhuang, F. Zhou, B. Y. Zhao, and A. Rowstron. Cashmere: Resilient anonymous routing. In *NSDI 2005*.