Analysis of an Anonymity Network for Web Browsing

Marc Rennhard*, Sandro Rafaeli†, Laurent Mathy†, Bernhard Plattner* and David Hutchison†
* Swiss Federal Institute of Technology, Computer Engineering and Networks Laboratory; Zurich, Switzerland
† Lancaster University, Faculty of Applied Sciences; Lancaster, UK
{rennhard,plattner}@tik.ee.ethz.ch, {rafaeli,laurent,dh}@comp.lancs.ac.uk

Abstract

Various systems offering anonymity for near real-time Internet traffic have been operational. However, they did not deliver many quantitative results about performance, bandwidth overhead, or other issues that arise when implementing or operating such a system. Consequently, the problem of designing and operating these systems in a way that they provide a good balance between usability, protection from attacks, and overhead is not well understood. In this paper, we present the analysis of an anonymity network for web browsing that offers a high level of anonymity against a sophisticated attacker and good end-to-end performance at a reasonable bandwidth overhead. We describe a novel way of operating the system that maximizes the protection from traffic analysis attacks while minimizing the bandwidth overhead. We deliver quantitative results about the performance of our system, which should help to give a better understanding of anonymity networks.

Keywords: *anonymity, anonymous web browsing, MIX networks*

1 Introduction

At last year's WETICE workshop, we have presented an architecture for a MIX-based [4] anonymity network [11]. We have explained the details about its functionality and its protocol. The main motivation for the project was that although various MIX-based systems have been operational, none of them delivered many quantitative results about performance, bandwidth overhead, or other issues that arise when implementing or operating such a system. Particularly, it was not clear how to operate an anonymity network efficiently such that it provides a good balance between the level of anonymity if offers and the costs in terms of end-to-end performance and bandwidth overhead this implies. The goal of this project was to overcome this problem, find ways to efficiently operate a MIX-based system, and to analyze it by examining the performance it offers.

In this paper, we present the second part of our work. We present a novel way of operating the core components of an anonymity network – the MIXes – that maximizes the protection from traffic analysis attacks while minimizing the bandwidth overhead. We also deliver quantitative results about the performance of our system, which show that it indeed provides a good balance between usability, protection from attacks, and overhead.

This paper is organized as follows: we first recall the basic design of our system. In section 3, we discuss related work. We describe the mixing component of our system in section 4. In section 5, we present the results of our performance analysis and conclude this work in section 6.

2 The Anonymity Network

We briefly recall the basic design of our system. For a much more detailed description, check out our last year's WETICE paper [11] or the technical report [10]. The anonymity network (AN) is a MIX network to enable anonymous web browsing. It offers *sender anonymity* and *relationship anonymity* [7]. This means that the receiver (or server) should not find out who the sender (or client) is and an eavesdropper should not be able to detect that there is a communication relationship between the two parties. Figure 1 gives an overview of the AN.

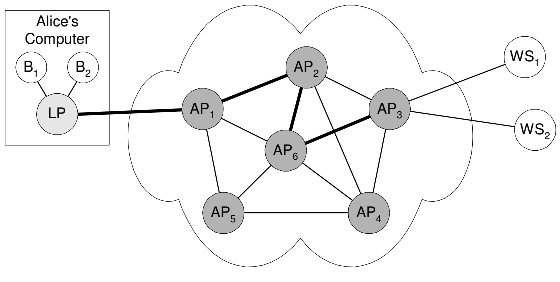


Figure 1. Anonymity Network overview.

The core of the AN consists of the *Anonymity Proxies* (APs). The APs are distributed in the Internet and play the role of the MIXes [4] in our system. For trust reasons, the APs should be operated by different institutions. A user Alice accesses the AN via a *Local Proxy* (LP), which is a program running on her computer. When Alice wants to browse the Web anonymously, she selects a subset of the APs she trusts, e.g. AP_1, AP_2, AP_6 and AP_3. The LP then sets up an *Anonymous Tunnel* from LP to AP_3 via AP_1, AP_2 and AP_6. The anonymous tunnel is longstanding and remains active until Alice decides to tear it down. Alice can now use her web browser to access the anonymous tunnel in exactly the same way as a normal web proxy. When Alice's browser B_1 want to fetch data from the web server WS_1, it sends the request to LP. LP sets up an *Anonymous Connection* within the anonymous tunnel from LP to AP_3 and tells AP_3 to connect to WS_1. When this is done, the actual request can be handled and the reply is sent from WS_1 back to B_1, via the anonymous connection. Note that multiple anonymous connections can be transported within one anonymous tunnel, so if Alice opens a second browser B_2 to access a web server WS_2, another anonymous connection is set up. The same holds when the browser requests the embedded objects in a web page in parallel: the LP simply establishes multiple anonymous connections. The LP also sanitizes the requests it receives from the browser by removing data that could possibly identify Alice.

The APs operate at the application layer and a TCP-connection is used to connect a pair of APs or an LP and an AP. Note that not every AP has to be connected to every other AP, but if they are connected, then there is exactly one TCP-connection between them. This means that the anonymous tunnels of different users are transported within a single TCP-connection between two APs. As a result, the anonymous tunnels are not visible as unique connections from the outside.

The goal of an APs is to hide the correlation of incoming and outgoing messages such that an attacker that performs traffic analysis to break the anonymity cannot follow a message through the network. To do so, the following measures are used: (1) all messages used in the system have exactly the same length (we use 1000 bytes in our system), which defeats traffic analysis based on correlating packets by their length. (2) The encoding of a message changes between entering and exiting an AP, which prevents traffic analysis by looking at the content of packets. This is usually achieved by encrypting or decrypting a message as it passes an AP. (3) An AP delays and reorders incoming packets from different users before forwarding them such that the outgoing sequence of packets is not related to the incoming sequence. This makes traffic analysis based on timing very difficult because for an eavesdropper, each of the outgoing packets can correspond to each of the incoming packets with the same probability. (4) Dummy messages are exchanged between APs whenever this is necessary to further complicate traffic analysis.

Messages are not simply encrypted between a pair of proxies, as this would disclose the message to each AP. As a result, the first AP would know the web server the user is communicating with. Furthermore, the other APs could derive information about anonymous tunnels by comparing the messages they receive. To prevent this, the user encrypts a message repeatedly for each AP, starting with the last in the anonymous tunnel. When traveling through the AN, each AP removes one layer of encryption until the last AP sees the message in the clear and forwards it to the web server. It works vice-versa when the reply from the web server is sent back to LP.

Our prototype is implemented in Java 1.3 and provides support for HTTP and HTTPS. It should be noted that although we currently only support web traffic, the LP can be extended such that other protocols can use the anonymous tunnel as well. Using Sun's Java HotSpot Server virtual machine, the system provides adequate performance for quantitative and qualitative evaluation.

3 Related Work

Anonymizing web traffic has been tried before. Simple services (e.g. The Anonymizer [5]) with only a proxy between user and web server are efficient but very vulnerable to traffic analysis attacks. In addition, the end-to-end relationship is not anonymous with regard to the single proxy itself, which render such systems useless if a high level of anonymity is needed. Another system offerning a relatively low level of anonymity is Crowds [9], where users forward each others web requests to conceal the real origin of a request.

More sophisticated MIX-based systems include Onion Routing [8, 12], the commercial Freedom system [3] (which has been shut down as of October 22nd, 2001), and Web MIXes [2]. All of them employ uniform message length and layered encryption to complicate traffic analysis, but so far, none of them has come up with an efficient solution to make use of dummy traffic to significantly increase the protection from traffic analysis attacks. As a result, these systems do not employ dummy traffic, which implies that they are not very resistant against a powerful adversary.

4 Mixing and Dummy Traffic

The most critical components in a MIX network that supports near real-time traffic are the delaying and reordering of messages and the use of dummy traffic. One has to find the balance between making it hard for an attacker to correlate incoming and outgoing messages at an AP without delaying the data so long that the end-to-end performance

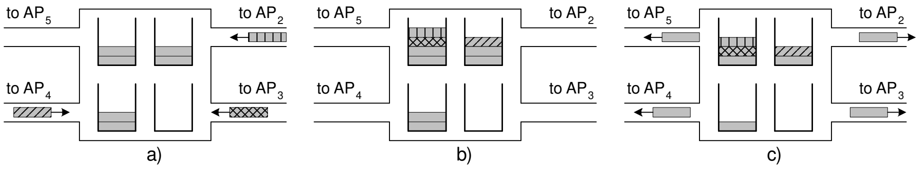


Figure 2. Mixing and dummy messages.

suffers noticeably and without introducing so many dummy messages that the bandwidth consumption increases drastically. In this section, we present a novel way of operating these MIXes that maximizes the protection from traffic analysis attacks while minimizing the bandwidth overhead.

One approach could be to send messages on a link between two APs all the time. If real messages are available, these real messages are sent, otherwise dummy messages are used. Dummy messages consist of random bits and an eavesdropper cannot distinguish them from encrypted real messages. The advantage of this scenario is that it is virtually impossible for an observer to find out when real messages are actually exchanged between APs and when not. The disadvantage is that this results in a vast amount of dummy data consuming bandwidth. Our goal of operating an AP is to minimize the overhead imposed by the use of dummy messages while providing a high level of resistance against traffic analysis attacks.

The idea is to have at each time an AP connected to some other APs. Assume that AP_1 is connected to AP_2, AP_3, AP_4, and AP_5. When AP_1 gets a message that must be forwarded to AP_2, AP_1 also sends a dummy message to AP_3, AP_4, and AP_5. Since messages change their encoding while traversing an AP, an external attacker cannot relate any of the outgoing messages to the incoming one. The problem of this approach is that we are sending many dummy messages to hide a real one – only one out four messages is a real one in the example just described which means that only 25% of all messages are real messages. However, by modifying this approach a bit, it can be made much more efficient, as Figure 2 illustrates. When a message arrives at an AP, we do not forward it right away but buffer it in the output queue of the link it is supposed to be forwarded to.

Since we buffer the messages, there are usually some messages waiting in an AP to be forwarded to their respective next hop AP. In figure 2a, there are two messages waiting in each output queue to AP_2, AP_4, and AP_5, and none in the output queue to AP_3. Three more messages are just arriving, they are processed by the AP and put at the end of the appropriate output queues (figure 2b). Eventually, the AP decides to forward messages and sends the first message in each queue to the APs it is connected to (figure 2c). Since no message is currently in the output queue to AP_3, a dummy message is sent there.

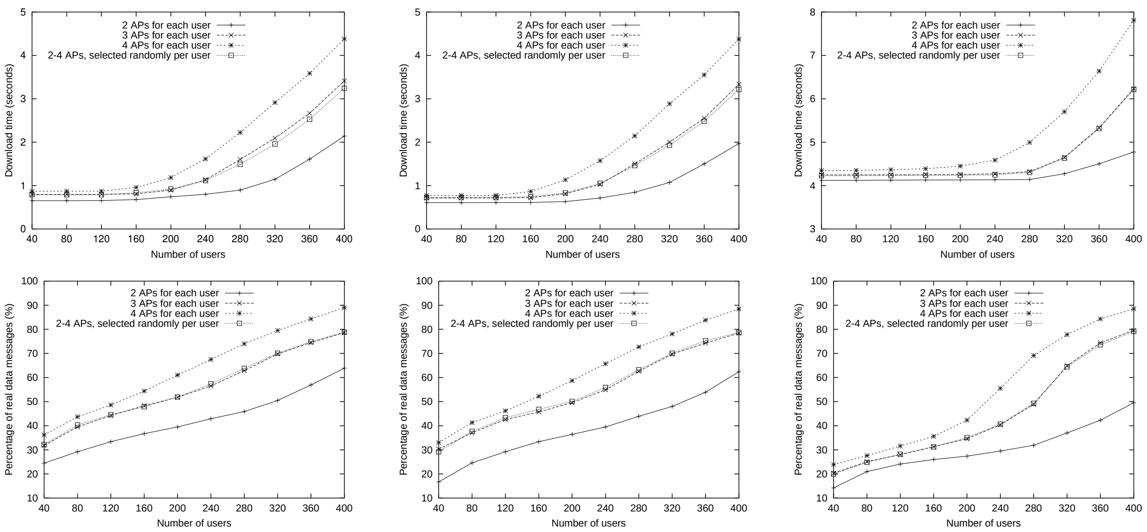
It is straightforward to see what we have achieved by buffering the messages for a short time: instead of using many dummy messages, an AP now uses the other real messages to hide a particular message. On the outgoing links where no packet is waiting, we still have to send dummy messages, but in general we can expect that much fewer dummy messages have to be used compared to the basic approach where a message is forwarded right away.

5 Performance Analysis

In this section we analyze how well the AN performs. There are two questions we want to answer with the performance measurements: (1) what end-to-end performance a user can expect, i.e. how long does it takes on average to download a document from the Web and (2) how many dummy messages the APs produce to protect the real messages. The test AN consists of six fully interconnected APs, which means there is a TCP-connection between each pair of APs. We simulate different numbers of users (40–400) and different numbers of APs (2–4) the users use in their anonymous tunnels. We believe that 2, 3, or 4 APs are realistic numbers of APs one would choose in a real-life scenario. Each user sends an HTTP request through the anonymous tunnel to a web server, receives the corresponding HTTP reply, waits for an arbitrary time between 0 and 10 seconds, issues the next request and so on.

We simulate the behavior of HTTP 1.0 [1]: the browser first fetches the page itself and then each embedded object. Since HTTP 1.0 opens a dedicated TCP-connection for the page and each embedded object, this means that the LP sets up an anonymous connection within the anonymous tunnel for each of these connections. We also use realistic numbers for the page size, the number of embedded objects per page, and the size of each embedded object. All these numbers follow a Pareto distribution with an average file size of 12 KB and an average number of embedded objects of 4 [6].

Each AP is running on a dedicated PC with 256 MB RAM and an Intel Pentium III or AMD Athlon CPU running at speeds between 800 MHz and 1 GHz. The PCs run Linux with a 2.4.7 kernel. The users are simulated on several Sun workstations running different versions of Solaris. The Apache web servers are running on the same systems as the APs. All systems are located in three different 100-



a) Basic configuration b) Handling dummy messages c) Fixed bit-rates

Figure 3. Performance analysis.

Mbit/s Ethernets that are connected by a router. Sun Microsystems' Java 1.3.1 is used to run the LPs and APs.

In all performance measurements, we measure the complete average download time and the percentage of real messages exchanged between the APs. The download time is the time to completely download a page and all embedded objects. Figure 3a illustrates the results when operating the AN as described in sections 2 and 4.

Figure 3a shows that the download times get longer when (1) there are more users in the system and (2) when the users use more APs in their anonymous tunnels, both is not surprising. We can also see that the download time is nearly constant until a certain number of users is reached. The reason for this is that when there are only a few users, messages are held back in the APs for a short time in the hope that more messages arrive to keep the amount of dummy messages low. When a certain number of users is reached, the download time increases with the number of users. The reason for this is that the APs are approaching their limit in terms of messages they can handle.

Similarly, the number of dummy messages decreases if there are more users and if more APs are used in an anonymous tunnel (figure 3a). This is reasonable, since both means that an AP must handle more messages at the same time, which increases the probability that real messages are waiting in many of the outgoing queues of an AP. Although 70% of dummy messages when there are only few users looks like a lot of overhead, it is perfectly justifiable, since (1) the data generated by these few users are not enough to confuse an attacker and to provide strong anonymity, which means that a lot of cover traffic has to be generated and (2) since the APs do not have to handle many real messages, they have enough computing power to process the dummies.

The measurements show that our way of operating an AP indeed seems to minimize the amount of dummy messages that must be used to protect the anonymity while providing very acceptable download times. For instance, when there are 320 users and each user selects randomly 2, 3, or 4 APs in her anonymous tunnel, then the average download time is below 2 seconds and the amount of dummy messages is below 30%. On the other hand, figure 3a also shows that our test AN consisting of 6 APs eventually reaches its limits when 400 users are using 4 APs in each anonymous tunnel and browse the Web intensively (i.e. requesting a web page every 5 seconds on average).

5.1 Blocked TCP-Connections

We employ TCP-connections between APs, which bears a problem when one of these connections blocks for any reason. In section 4, we have defined that an AP always sends out one message on *each* link to other APs, which implies that one blocked connection causes the AP to stop sending messages on all other connections as well.

To cope with this problem, an AP has the option to simply ignore blocked TCP-connections. Whenever an AP decides to send a message out on each link, then this is done only on these links where the TCP send buffer is not full. As a result, only the anonymous tunnels that use the particular TCP-connection are blocked and not all tunnels that go through the AP. However, one must realize that this does probably make the AN less resistant against traffic analysis attacks, since the loads on the links going out from an AP are not the same anymore. On the other hand, it is reasonable to assume that one or more TCP-connections are temporarily blocked in a practical scenario with an AN consisting of many APs. To not degrade the performance of the whole AN too much, it is probably required that messages are sent even if some links are blocked. A compromise could be that messages are sent if a certain number of the links are not blocked, e.g. if at least 75% of the links are usable. If an AP is connected to 8 other APs, this means that messages are sent if at most 2 TCP-connections are blocked.

We have implemented this functionality in the AN and analyzed its effect. We do not include the performance measurements in this paper, as the results in our test environment are virtually identical to those in figure 3a.

5.2 Handling of Dummy Messages

We have discussed when an AP sends dummy messages, but not what an AP does when it receives them. Until now, an AP simply ignored incoming dummy messages. The problem with this approach is that if an AP temporarily gets only dummy messages, it will not send messages any more. This gives an attacker additional information during a traffic analysis attack, since it tells him that only dummy messages are sent to this AP. To confuse an attacker, the AP has to send messages even if it only receives dummy messages.

The APs handle this problem as follows: when deciding to forward messages, an AP now also takes the incoming dummies into account. The effect is not very visible when there are many users and an AP receives mostly real messages. However, when the number of incoming real messages drops, the incoming dummies affect the APs decision to forward messages more and more. If an AP receives only dummy messages and there are no real messages to forward, it simply 'forwards' dummies such that an attacker cannot detect if an AP only receives dummy messages. Figure 3b shows how this affects the measurements.

By comparing figure 3b with figure 3a, we can see that especially when there are few users, the download times are shorter and the relative amount of dummy messages is bigger. The reason is that by taking the incoming dummies into account when deciding to forward messages, the condition to actually send messages is fulfilled earlier [10]. Consequently, messages are forwarded earlier and the download times get shorter. On the other hand, this also means that fewer real messages are in an AP at any time, which results in an increase of dummy messages that must be used. As the number of users grows, this effect diminishes as the relative number of incoming dummy messages gets smaller.

5.3 Fixed Bit-Rates on the LP–AP Link

So far, we have silently assumed that the strategy of an attacker is to observe an AP to correlate incoming and outgoing messages. However, if a powerful attacker can observe the whole network, then the AN basically collapses to one big node. The attacker can then try to correlate events at the endpoints of the AN. For example, if he sees data being sent over the connection from an LP to an AP, he can check at which AP data are sent to a web server a short time later. Similarly, he can try to correlate the events when the web server sends its response back to an LP.

To resist this attack, we must send cover traffic between the LP and the first AP. One approach could be that the LP sends dummy web requests from time to time and the first AP sends traffic back to the LP that looks like a reply from a web server. The problem here is that the attacker can still correlate the events at the endpoints since the real requests and replies are still visible between the LP and AP. The only way to remove any correlation is to send traffic between the LP and the first AP in both directions all the time. We have implemented this functionality in the AN and figure 3c shows the results when messages are exchanged at a constant rate of 128 Kbit/s on every LP–AP link.

The limited bandwidth on the LP–AP links results in significantly longer download times, as figure 3c shows. The percentage of dummy messages is much higher than in figures 3a and b when there are relatively few users. The reason is that the limited bandwidth on the LP–AP link slows down each anonymous tunnel, which means that there are fewer messages of each user in the system at any time [10].

Comparing figures 3b and c shows the balance one has to find when offering such an anonymity service: we have increased the system's resistance against an adversary that can observe the whole network at the price of much longer download times. For maximal resistance, every user should exchange data with the first AP in her anonymous tunnel at the same fixed bit-rate. We have used 128 Kbit/s which is still more than most home-users with dial-up connections get today. So for maximal resistance, we would have to use 64 Kbit/s or even less. This is certainly not acceptable for those that have much better connectivity. A compromise could be to offer different fixed bandwidths on the LP–AP links, e.g. 32, 64, 128, 256, 512, and 1024 Kbit/s. However, one must bear in mind that this again compromises the anonymity a little bit, since fast data exchange between the last AP and the web server can be correlated with users that use fast fixed bit-rates on the LP–AP link.

6 Conclusions and further work

We have designed, implemented, and analyzed an anonymity network for web browsing. We have presented a novel way of operating the MIXes to minimize the bandwidth overhead while providing a high level of anonymity. We have then delivered quantitative results which show that our system indeed produces relatively few dummy traffic while achieving very acceptable download times.

We have seen that compromises must be made between usability, overhead, and protection from attacks. If we had to decide now which of the options described in this paper to use, we would choose the system we evaluated in figure 3b, as it provides acceptable download times, small dummy message overhead, and good protection against a reasonable attacker that is able to observe several APs but maybe not the whole network. We also would employ the measures to cope with blocked TCP-connections (section 5.1), although we are aware that this means sacrificing some of the resistance against attacks for better usability and performance. It is questionable if the additional resistance gained by using fixed bit-rates on the LP–AP links is worth the performance penalty (section 5.3). If we assume a strong attacker that is indeed able to observe the whole network, then we should make use of the fixed bit-rates, but one and the same rate for all users seems too strict. So we would offer the users a set of different rates as proposed at the end of section 5.3, although we recognize again that this is a trade-off between performance and resistance against a global attacker.

We believe that the AN provides strong resistance against passive traffic analysis attacks. Eavesdropping at an AP does probably not enable an attacker to correlate incoming and outgoing messages. Fixed bit-rates on the LP–AP link makes attacks difficult even if the adversary is able to observe the whole network. The AN also resists some active attacks, as the protocol guarantees that inserting or replaying messages does not help the attacker to learn anything about anonymous tunnels [10]. If the adversary can delete or modify messages between two proxies, then the messages are retransmitted due to the underlying TCP-connection. However, by blocking the message stream on a link between two proxies, the attacker can check on which connections between APs and web servers the traffic flow also stops to derive information about anonymous tunnels.

More research has to be done to really understand the resistance of MIX networks against various kinds of attacks, especially when a subset of the MIXes is not honest. There are other challenging problems to solve, e.g. how to organize a big AN. Would it be good to have relatively few but powerful and fully interconnected APs or is a system of many small APs where each AP is connected only to a few other APs better? The second approach would limit a user's choice to freely select the APs she wants to use in her anonymous tunnel, since not all APs would be directly interconnected. One could also think of a pure peer-to-peer network, where each user is also an AP at the same time.

With our system, we have tried to find a good compromise between usability, overhead, and protection from attacks. Our next steps are to further optimize the system in terms of performance and stability and to start a public user trial (end of April 2002) to get feedback and to see how well it works in an Internet-wide scale.

Acknowledgement

The work presented here was done within ShopAware – a research project funded by the European Union in the Framework V IST Program (project 12361). Marc Rennhard would like to thank also the Swiss Federal Office for Education and Science for his sponsorship.

References

[1] T. Berners-Lee, R. Fielding, and H. Frystyk. Hypertext Transfer Protocol – HTTP/1.0. RFC1945, 1996.

[2] O. Berthold, H. Federrath, and S. Köpsell. Web MIXes: A System for Anonymous and Unobservable Internet access. In *Proceedings of the Workshop on Design Issues in Anonymity and Unobservability*, pages 101–115, Berkeley, CA, USA, July 25–26 2000.

[3] P. Boucher, A. Shostack, and I. Goldberg. Freedom Systems 2.0 Architecture. White Paper, `http://www.freedom.net/info/whitepapers`, December 18 2000.

[4] D. L. Chaum. Untraceable Electronic Mail, Return Adresses, and Digital Pseudonyms. *Communications of the ACM*, 24(2):84–88, February 1981.

[5] L. Cottrell. The Anonymizer. `http://www.anonymizer.com`.

[6] A. Feldmann, A. C. Gilbert, P. Huang, and W. Willinger. Dynamics of IP Traffic: A Study of the Role of Variability and the Impact of Control. In *Proceeding of SIGCOMM '99*, Massachusetts, USA, September 1999.

[7] A. Pfitzmann and M. Köhntopp. Anonymity, Unobservability, and Pseudonymity - A Proposal for Terminology; Draft v0.12. `http://www.koehntopp.de/marit/pub/anon/Anon_Terminology.pdf`, June 17 2001.

[8] M. Reed, P. Syverson, and D. Goldschlag. Anonymous Connections and Onion Routing. *IEEE Journal on Selected Areas in Communications*, 16(4):482–494, May 1998.

[9] M. K. Reiter and A. D. Rubin. Crowds: Anonymity for Web Transactions. *ACM Transactions on Information and System Security*, 1(1):66–92, November 1998.

[10] M. Rennhard, S. Rafaeli, and L. Mathy. Design, Implementation, and Analysis of an Anonymity Network for Web Browsing. TIK Technical Report Nr. 129, TIK, ETH Zurich, Zurich, CH, February 2002.

[11] M. Rennhard, S. Rafaeli, L. Mathy, B. Plattner, and D. Hutchison. An Architecture for an Anonymity Network. In *Proceedings of the IEEE 10th Intl. Workshop on Enabling Technologies: Infrastructure for Collaborative Enterprises (WET ICE 2001)*, pages 165–170, Boston, USA, June 20–22 2001.

[12] P. Syverson, G. Tsudik, M. Reed, and C. Landwehr. Towards an Analysis of Onion Routing Security. In *Proceedings of the Workshop on Design Issues in Anonymity and Unobservability*, pages 83–100, Berkeley, CA, USA, July 25–26 2000.