Private Web Browsing

Paul F. Syverson, Michael G. Reed, and David M. Goldschlag *
Naval Research Laboratory

June 2, 1997

Abstract

This paper describes a communications primitive, *anonymous connections*, that supports bidirectional and near real-time channels that are resistant to both eavesdropping and traffic analysis. The connections are made anonymous, although communication need not be. These anonymous connections are versatile and support private use of many different Internet services. For our purposes, privacy means maintaining the confidentiality of both the data stream and the identity of communicating parties. These are both kept confidential from network elements as well as external observers. Private Web browsing is achieved by unmodified Web browsers using anonymous connections by means of HTTP proxies. Private Web browsing may be made anonymous too by a specialized proxy that removes identifying information from the HTTP data stream. This article specifies anonymous connections, describes our implementation, and discusses its application to Web browsing via HTTP proxies.

Keywords: Security, privacy, anonymity, traffic analysis.

1 Introduction

As the Internet rapidly becomes an important vehicle for moving data, as well as the foundation for many virtual private networks, it is becoming apparent that data moving over the Internet must be protected from eavesdropping. However, the simple fact that two parties are communicating is a source of information, and this information is not hidden by encrypting the data stream. We mean to expand the notion of privacy to include confidentiality of both the data stream and the identity of communicating parties. Ordinarily, privacy is taken to mean only confidentiality of the data stream. Furthermore, we must protect this (expanded) privacy from both network elements and external observers. This

*Address: Naval Research Laboratory, Center For High Assurance Computer Systems, Washington, D.C. 20375-5337, USA, phone: +1 202.767.2389, fax: +1 202.404.7942, e-mail: {*last name*}@itd.nrl.navy.mil.

requires protecting against traffic analysis. Outside the military, the threat of traffic analysis has been largely ignored. However, traffic analysis is becoming a significant threat to privacy:

- The browsing behavior of Web users is increasingly subject to public observation both by observers of the Internet and the servers that hold Web information. As Web based commerce becomes more prevalent, this behavior will include individual's shopping habits and spending patterns, as well as other personal data that people have traditionally considered private. Where one conducts one's shopping should not automatically be available to network observers, and the identity of a shopper should not be automatically revealed to the store. Unless steps are taken to protect against both eavesdropping and traffic analysis, this information is available.

- Certain electronic money protocols are supposed to allow secure transactions over the Web while preserving the untraceability that cash allows. However, if the electronic cash moves over a channel that identifies the purchaser, transactions are no longer anonymous.

- The Web is becoming an important source for information gathering. In a competitive environment, a company may wish to protect its current research interests. However, monitoring Web requests may indicate the company's focus. By keeping Web browsing private, the company's interests are protected while allowing completely transparent Web use.

This article describes a communications primitive, *anonymous connections,* which provides the function of TCP/IP socket connections but is also strongly resistant to both active and passive eavesdropping and traffic analysis attacks from observers both outside and inside the communications network. The connections are made anonymous, although communication need not be. Socket connections are near real-time bidirectional communication channels. The principles behind these anonymous connections can be applied to many circuit based communication and can be adopted to hide location information too (e.g., in cellular phone systems).

We also describe an implementation of anonymous connections, called *onion routing* and describe how it may be used by a variety of unmodified Internet services by means of *proxies*. For example, most Web browsers are proxy aware (to communicate through a firewall) and can use onion routing by means of a simple onion routing HTTP proxy. Private Web browsing may be made anonymous too by a specialized proxy that removes identifying information from the HTTP data stream. Both the onion routing system and proxies for Web browsers, remote login, and electronic mail have been implemented for Sun Solaris, and the code is available to the public.

This paper is organized in the following way: Section 2 presents an overview of the onion routing system and discusses our threat model and the system's vulnerabilities. Section 3 describes proxies for Web browsing. Section 4 describes related work. Section 5 presents concluding remarks.

2 Onion Routing

In a basic onion routing network configuration, an onion router might sit on the firewall of a protected site. This onion router serves as an interface between machines behind the firewall and the rest of the network. To complicate tracking of traffic originating or terminating within the protected site, this onion router should also route data between other onion routers.

In onion routing, instead of making a socket connection directly to a responding machine (the *responder*), an initiating application (the *initiator*) makes a connection through a sequence of machines called onion routers. Onion routers are essentially bidirectional near real time mixes [3] (see section 4). Onion routers in the network are connected by longstanding socket connections. Anonymous connections through the network are multiplexed over the longstanding connections. For any anonymous connection, the sequence of onion routers in a route is strictly defined. However, each onion router can only identify the previous and next hops along a route. Data passed along the anonymous connection appears different at each onion router, so data cannot be tracked en route and compromised onion routers cannot cooperate.

The onion routing network is accessed via *proxies*. An initiating application makes a socket connection to an application specific proxy. That proxy defines a (perhaps random) route through the onion routing network by constructing a layered data structure called an *onion* and sending that onion through the network. Each layer of the onion in public key encrypted for the intended onion router and defines the next hop in a route. An onion router that receives an onion peels off its layer, identifies the next hop, and sends the remaining onion to that onion router. An onion's size is fixed, and each onion router adds random padding to replace the removed layer. After sending the onion, the initiator's proxy sends data through the anonymous connection.

The last onion router forwards data to another type of proxy, called the responder's proxy, whose job is to pass data between the onion network and the responder. An example onion routing network and with an anonymous connection from an initiator to a responder through onion routers W, X, Y, and Z is illustrated in figure 1.

In addition to carrying next hop information, each onion layer contains key seed material from which keys are generated for crypting[1] data sent forward or backward in the anonymous connection. (We define *forward* to be the direction

[1]We define the verb *crypt* to mean the application of a cryptographic operation, be it encryption or decryption.

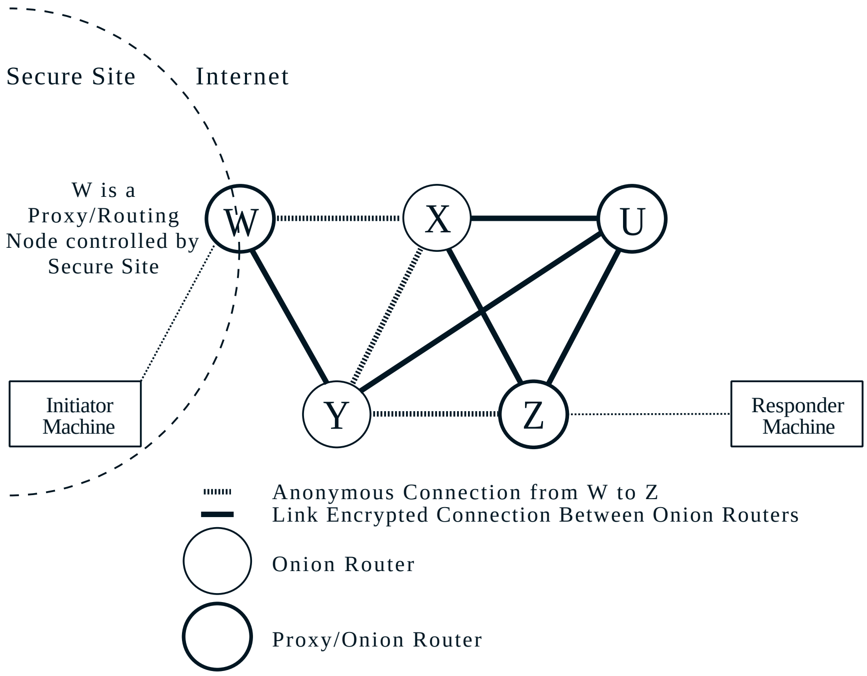

Figure 1: Routing Topology.

in which the onion travels and *backward* as the opposite direction.) Figure 2 is an onion that the initiator's proxy living on onion router W would create to build a route to the responder's proxy living on onion router Z through onion routers X and Y. Onion routers must recognize onions that they processed (until they expire) to prevent replay of onions. The expiration time is specified in the *exp_time* field. Clocks on different onion routers need not be closely synchronized if the expiration time is adequately conservative. (A longer expiration time requires more storage at an onion router, however.)

Before sending data over an anonymous connection, the initiator's onion router adds a layer of encryption for each onion router in the route. As data moves through the anonymous connection, each onion router removes one layer of encryption, so it finally arrives as plaintext. (This layering occurs in the reverse order for data moving back to the initiator.) Stream ciphers are used for all symmetric cryptography both to prevent replay attacks and to complicate data tracking.[2] Unlike the peeling of onions themselves, removing a layer of encryption does not change the size of a data packet. Onion routers also randomly reorder the data they receive before forwarding it (but preserve the

[2]We assume that the underlying network, TCP/IP streams, provides in-order and uncorrupted data delivery.

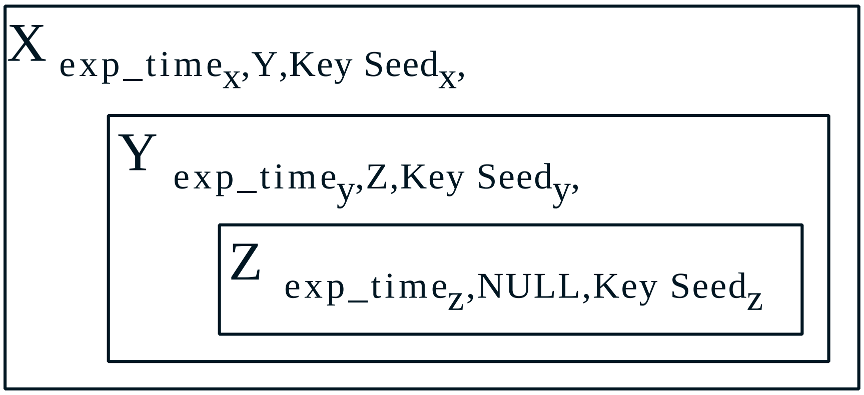

Figure 2: A Forward Onion.

order of data in each anonymous connection) so it is not possible to track data under a FIFO assumption, for example.

By layering cryptographic operations in this way we gain an advantage over simple link encryption. Even though the total cryptographic overhead for passing data is the same as for link encryption, the protection is better than link encryption. In link encryption the chain is as strong as the weakest link: one compromised node can reveal everything. In onion routing the chain is as strong as its strongest link: one honest node is enough to maintain the privacy of the connection. Even if link encryption is layered over end-to-end encryption, compromised nodes can cooperate to uncover route information. This is not possible in the onion routing network since data always appears different to different onion routers.

The current system does not manage the distribution or updating of long term public keys or network topology.

2.1 Threat Model

This section outlines our threat model. It does not intend to quantify the cost of attacks, but to define possible attacks. Future work will quantify the threat. First some vocabulary. A session is the data carried over a single anonymous connection. Data is carried in fixed length cells. Since these cells are multiply encrypted and change as they move through an anonymous connection, tracking cells is equivalent to tracking markers that indicate when cells begin. In a marker attack, the attacker identifies the set of outbound connections that some distinguished marker may have been forwarded upon. By intersecting these sets for a series of distinguished markers belonging to the same session, an attacker may determine, or at least narrow, the set of possible next hops. In a timing attack, the attacker records a timing signature for a session that correlates data rate over time. A session may have a very similar timing signature wherever it is measured over a route, so cooperating attackers may determine if they carry a particular session.

We assume that the network is subject to both passive and active attacks. Traffic may be monitored and modified by both external observers and internal network elements, including compromised onion routers. Attackers may cooperate and share information and inferences. We assume roving attackers that can monitor part, but not all, of the network at a time.

Our goal is to prevent traffic analysis, not traffic confirmation. If an attacker wants to confirm that two endpoints often communicate, and he observes that they each connect to an anonymous connection at roughly the same time more often than is statistically expected, it is reasonable to infer that the endpoints are indeed communicating. Notice that this attack is infeasible if endpoints live in protected networks behind onion routers.

If the onion routing infrastructure is uniformly busy, then passive external attacks are ineffective. Specifically, neither the marker nor timing attacks are feasible, since external observers cannot assign markers to sessions. Active attacks are possible since reducing the load on the system makes the network easier to analyze (and makes the system not uniformly busy).

Passive internal attacks require at least two compromised onion routers. Since onion routers can assign markers to a session, both the marker and timing attacks are possible. Specifically, timing signatures can be broadcast, and other compromised onion routers can attempt to find connections with matching timing signatures.

Active internal attacks amplify these risks, since individual onion routers can selectively limit traffic on particular connections. An onion router could, for example, force a particular timing signature on a connection, and advertise that signature.

The initiator's proxy, which builds the onion defining the route of the anonymous connection, is the most trusted element of the onion routing infrastructure. By moving this proxy to the initiator's machine [15], this trusted function may be placed under the control of the initiator. In this topology, the first onion router becomes a conduit to the rest of the network; that first onion router knows the source, but not the destination of a connection.

By layering end-to-end encryption over an anonymous connection, endpoints may identify themselves to one another without revealing the existence of their communication to the rest of the network. This emphasizes that our goal here is to prevent traffic analysis, not to enable anonymous communication.

3 Web Browsing

A proxy is a transparent service between two applications that would usually make a direct socket connection to each other but cannot. For example, a firewall might prevent direct socket connections between internal and external machines, and a proxy, running on the firewall, may enable the connection. Proxy aware applications are becoming quite common.

Our goal has been to design an architecture for private communication that would interface with *unmodified* applications, so we chose to use the proxy approach as the interface between applications and onion routing's anonymous connections. For applications that are designed to be proxy aware, (e.g., WWW browsers), we simply design appropriate interface proxies. Surprisingly, for certain applications that are not proxy aware (e.g., RLOGIN), we have also been able to design interface proxies.

In a simple case, for example, where a firewall lives between a trusted and untrusted network, the first onion router and its proxies live on the firewall. There are two classes of proxies: one that makes connections from initiating applications into the onion routing network, and the other that completes the connection from the onion routing network to responders. These two classes of proxies also move data from applications to the routing system and vice versa.

For the services we have considered, a nearly generic responder proxy is adequate. Its function is to read the data stream from the terminating onion router. The first datum identifies the desired service and responder's machine name. From that datum, the responder's proxy can determine the responder's IP address and port number. The responder's proxy makes a socket connection to that IP/port, and subsequently moves data between the onion network and the new socket. (For certain services, like rlogin, the responder's proxy also infers that the new socket must originate from a trusted port.)

Because an initiator's proxy bridges between applications and the onion routing network, it must understand both application protocols and onion routing protocols. Therefore, to simplify the design of application specific proxies, we partition this function into two, the *client proxy* and the *core proxy*. The client proxy bridges between a socket connection from an application and a socket connection to the core proxy in a one-to-one fashion. It is the obligation of the client proxy to massage the data stream so the responder's proxy can be application independent. For instance, the client proxy must prepend the data stream with a datum identifying the desired service and responder's machine name.

Proxying HTTP requests follows the IETF HTTP V1.0 Draft Specification [2]. An HTTP request from a client through an HTTP proxy is of the form:

```
GET http://www.server.com/file.html HTTP/1.0
```

followed by optional fields. Notice that an HTTP request from a client to a server is of the form:

```
GET file.html HTTP/1.0
```

also followed by optional fields. The server name and protocol type are missing, because the connection is made directly to the server.

As an example, a complete request from Netscape Navigator to an onion router HTTP proxy may look like this:

```
GET http://www.server.com/file.html HTTP/1.0
Referer: http://www.server.com/index.html
Proxy-Connection: Keep-Alive
User-Agent: Mozilla/3.0 (X11; I; SunOS 5.4 sun4m)
Host: www.server.com
Accept: image/gif, image/x-xbitmap, image/jpeg, image/pjpeg
```

The proxy must create an anonymous connection to `www.server.com`, and issue a request as if it were a client. Therefore, the request must be massaged to remove the server name and protocol type, and transmitted to `www.server.com` over the anonymous connection. Once this request is transmitted to the server, the proxy blindly forwards data in both directions between the client and server until the socket is broken by either side.

For the anonymizing proxy of HTTP, the proxy proceeds as outlined above but also sanitizes the optional fields that follow the `GET` command because they may contain identifying information. For example, we filter out cookies. Furthermore, the data stream during a connection must be monitored, to sanitize additional headers that might occur during the connection.

4 Related Work

Chaum [3] defines a layered object that routes data through intermediate nodes, called *mixes*. These intermediate nodes may reorder, delay, and pad traffic to complicate traffic analysis. In mixes, the assumption is that a single perfect mix adequately complicates traffic analysis, but a sequence of multiple mixes is typically used because real mixes are not ideal. Because of this, mix applications can use mixes in fixed order, and often do. Onion routers differ from mixes in at least two ways: onion routers are more limited in the extent to which they delay traffic at each node because of the real-time expectations that the applications demand of socket connections. Also, in a typical onion routing configuration, onion routers are also entry points to the onion routing network, and traffic entering or exiting at those nodes is not visible. This makes it hard to track packets, because they may drop out of the network at any node, and new packets may be introduced at each node. While onion routing cannot delay traffic to the extent that mixes can, traffic between onion routers is multiplexed over a single channel and is link encrypted with a stream cipher. This makes it hard to parse the stream.

Anonymous remailers like Penet [11] strip headers from received mail and forward it to the intended recipient. They may also replace the sender's address with some alias, permitting replies. These sorts of remailers store sensitive state: the mapping between the alias and the true return address. Also, mail forwarded through a chain of remailers may be tracked because it appears the same to each remailer.

Mix based remailers like [4, 10] use mixes to provide anonymous e-mail services. Essentially, the mail message is carried in the innermost layer of the onion data structure. Another onion type structure, used for a return address, can be contained in the message. This makes the return path self contained, and the remailer essentially stateless. Onion routing shares many structures with Babel [10] but it uses them to build (possibly long lived) application independent connections. This makes anonymous connections accessible to a wide variety of applications.

In [6], a structure similar to an onion is used to forward individual IP packets through a network. By maintaining tracking information at each router, ICMP error messages can be moved back along the hidden route. Essentially, a connection is built for each packet in a connectionless service. Although a followup paper [7] suggests that performance will be good, especially with hardware based public key cryptography, our experience suggests that both the cryptographic overhead of building onions and the tracking of onions against replay is not efficiently done on a packet-by-packet basis. However, it is easy to imagine an onion routing proxy that collects IP packets and forwards them over some anonymous connection. In this way, communication is anonymous at the IP layer, but connections need not be built for each IP packet. This anonymous IP communication may be more robust than our current architecture: it could survive a broken anonymous connection, since IP does not expect reliable delivery.

In [14], mixes are used to provide untraceable communication in an ISDN network. Here is a summary of that paper. In a phone system, each telephone line is assigned to a particular local switch (i.e., local exchange), and switches are interconnected by a (long distance) network. Anonymous calls in ISDN rely upon an anonymous connection between the caller and the long distance network. These connections are made anonymous by routing calls through a predefined series of mixes within each switch. The long distance endpoints of the connection are then mated to complete the call. (Notice that observers can tell which local switches are connected.) This approach relies upon two unique features of ISDN switches. Since each phone line has a subset of the switch's total capacity pre-allocated to it, there is no (real) cost associated with keeping a phone line active all the time, either by making calls to itself, to other phone lines on the same switch, or to the long distance network. Keeping phone lines active complicates traffic analysis because an observer cannot track coincidences.

Also, since each phone line has a control circuit connection to the switch, the switch can broadcast messages to each line using these control circuits. So, within a switch a truly anonymous connection can be established: A phone line makes an anonymous connection to some mix. That mix broadcasts a token identifying itself and the connection. A recipient of that token can make another anonymous connection to the specified mix, which mates the two connections to complete the call.

Our goal of anonymous connections over the Internet differs from anonymous remailers and anonymous ISDN. The data is different, with real-time constraints more severe than mail, but somewhat looser than voice. Both HTTP and ISDN connections are bidirectional, but, unlike ISDN, HTTP connections are likely to be small requests followed by short bursts of returned data. As described in [14], in a local switch, capacity is pre-allocated to each phone line, and broadcasting is efficient. But broadcasting over the Internet is not free, and defining broadcast domains is not trivial. Most importantly, the network topology of the Internet is more akin to the network topology of the long distance network between switches, where capacity is a shared resource. In anonymous ISDN, the mixes hide communication within the local switch, but connections between switches are not hidden. This implies that all calls between two businesses, each large enough to use an entire switch, reveal which businesses are communicating. In onion routing, mixing is dispersed throughout the Internet, which improves hiding.

The Anonymizer [1] is a Web proxy that filters the HTTP data stream to remove a user's identifying information. This makes Web browsing private in the absence of any eavesdropping or traffic analysis. The Anonymizer is vulnerable in three ways: It must be trusted. Second, traffic between a browser and the Anonymizer is sent in the clear, so that traffic identifies the true destination of a query, and includes the identifying information that the Anonymizer would filter. Third, even if the traffic between the browser and the Anonymizer were encrypted, traffic analysis could be used to match incoming (encrypted) data with outgoing data. The Anonymizer, however, is now readily available to everyone on the Web. It could be used together with onion routing as the HTTP proxy front end to provide a nice interface and good filtering for anonymity, with strong resistance to both eavesdropping and traffic analysis.

NetAngels [13] is similar to the Anonymizer, except that it builds personal profiles of its subscribers and targets advertisements to match the profile. However, the profile is not released to the advertiser and is deleted when a subscription is canceled. Subscribers must trust NetAngels, and connections to the service are subject to the same attacks as the Anonymizer.

LPWA [12, 8] (formerly known as Janus) is a "proxy server that generates consistent untraceable aliases for you that enable you to browse the Web, register at web sites and open accounts, and be 'recognized' upon returning to your accounts, all while still *preserving your privacy.*"

Pipe-net [5] is the proposal most similar to onion routing. It has not been implemented, however. Pipe-net's threat model is more paranoid than onion routing's: it attempts to resist active attacks by global observers. For example, Pipe-net's connections carry constant traffic (to resist timing signature attacks) and disruptions to any connection are propagated throughout the network.

Crowds [17] is essentially a distributed and chained Anonymizer, with encrypted links between crowd members. Web traffic is forwarded to a crowd member, who can either forward it to some other crowd member or to the destination. This makes communication resistant to local observers.

5 Conclusion

The behavior of Web users is increasingly subject to public observation by observers of the Internet, by servers that hold Web information, and by other network elements. As Web based commerce becomes more prevalent, this behavior will include individual's shopping habits and spending patterns, the identities of e-mail correspondents, as well as other personal data that people have traditionally considered private. A personal profile of one's behavior should be considered private in the same sense that one's credit card numbers and social security numbers are.

Onion routing's anonymous connections provide a communications infrastructure that is strongly resistant to both eavesdropping and traffic analysis. Onion routing accomplishes this goal by separating identification from routing. The connections are anonymous, although communication need not be. The onion routing infrastructure can be used for private Web browsing by means of proxies.

Privacy filters like the Anonymizer, LPWA, and NetAngels have two functions. One is to provide connections to the outside world that do not implicitly reveal the original source. The other is to manage and limit distribution of individual's private information. These privacy filters accomplish these goals by means of centralized public proxies. This creates a single point of vulnerability (and an attractive attack target). Furthermore, users must trust the centralized proxy.

We feel that services such as these are very important for managing private information, but the anonymous connections should be a separate function. The service these systems deliver could be improved (and simplified) by moving the filtering proxy to the user's workstation and connecting it to the outside network using anonymous connections. This places trusted function and information under the user's local control. Such partitioning works even when the filter provides the added function of profiling. The profile would also be collected locally, and the relevant subset of the profile may be shared, anonymously of course, with some other centralized service [19].

Onion routing has been implemented and its code is in the public domain. One can evaluate the performance of our implementation in several ways. Connection setup will always be relatively expensive, as long as public key cryptography is expensive. Implementations of public key cryptography in hardware would achieve a significant improvement. On our test machine, a 167Mhz Sun Ultrasparc, the public key cryptography necessary for onion processing takes roughly 0.5 seconds. Since Web response is quite variable, we rarely notice increased overhead for connection setup. Data throughput in our system depends on the speed of the symmetric cryptographic engine. On the Ultrasparc, we observe DES speeds of roughly 1MB/sec. Notice that the multiple encryptions, both at the initiator's proxy, and at the series of onion routers can be effectively pipelined. And, it is easy to implement engines that are faster than typical modem connections to the Internet.

Furthermore, on the Web, reading a typical page requires several simultaneous or consecutive socket connections. There is no reason that these socket connections cannot all use the same anonymous connection, either by concurrent multiplexing or sequential reuse. This amortizes the setup cost. In fact, HTTP 1.1 does this with pipelined connections to reduce the cost and number of socket connections. So, an HTTP 1.1 compliant proxy would reduce the load on the onion routing infrastructure also.

This system depends upon heavy use by a wide audience to achieve effective privacy. Unless an organization's goal is only to hide its internal communication patterns, multiple organizations should share an onion routing network. Onion routers owned by individual organizations should also be intermediate onion routers for the rest of the network.

In the past, access to networked machine services was granted, in part, based on identifying information in packet headers. This approach has been a significant source of security vulnerabilities and should be abandoned for most services. In onion routing, the anonymous connection carries no identifying or authenticating information. Any authenticating information must be in the data stream where it belongs.

References

[1] http://www.anonymizer.com

[2] T. Berners-Lee, R. Fielding, and H. Frystyk. *Hypertext Transfer Protocol – HTTP/1.0,* ftp://ds.internic.net/rfc/rfc1945.txt

[3] D. Chaum. *Untraceable Electronic Mail, Return Addresses, and Digital Pseudonyms,* Communications of the ACM, v. 24, n. 2, Feb. 1981, pages 84-88.

[4] L. Cottrell. *Mixmaster and Remailer Attacks,* http://obscura.obscura.com/~loki/remailer/remailer-essay.html

[5] W. Dai. Pipe-net, February 1995, post to the `cypherpunks` mailing list.

[6] A. Fasbender, D. Kesdogan, O. Kubitz. *Variable and Scalable Security: Protection of Location Information in Mobile IP,* 46th IEEE Vehicular Technology Society Conference, Atlanta, March 1996.

[7] A. Fasbender, D. Kesdogan, O. Kubitz. *Analysis of Security and Privacy in Mobile IP,* 4th International Conference on Telecommunication Systems Modeling and Analysis, March 1996, Nashville, USA.

[8] E. Gabber, P. Gibbons, Y. Matias, and A. Mayer. *How to Make Personalized Web Browsing Simple, Secure, and Anonymous,* Financial Cryptography '97, February 1997.

[9] D. Goldschlag, M. Reed, P. Syverson. *Hiding Routing Information,* in Information Hiding, R. Anderson, ed., LNCS vol. 1174, Springer-Verlag, 1996, pp. 137–150.

[10] C. Gülcü and G. Tsudik. *Mixing Email with* Babel*,* 1996 Symposium on Network and Distributed System Security, San Diego, February 1996.

[11] J. Helsingius. www.penet.fi.

[12] http://www.bell-labs.com/project/lpwa

[13] http://www.netangels.com

[14] A. Pfitzmann, B. Pfitzmann, and M. Waidner. *ISDN-Mixes: Untraceable Communication with Very Small Bandwidth Overhead,* GI/ITG Conference: Communication in Distributed Systems, Mannheim Feb, 1991, Informatik-Fachberichte 267, Springer-Verlag, Heildelberg 1991, pages 451-463.

[15] M. G. Reed, P. F. Syverson, and D. M. Goldschlag. *Privacy on the Internet,* INET '97, Kuala Lumpur, Malaysia, June, 1997.

[16] M. G. Reed, P. F. Syverson, and D. M. Goldschlag. *Proxies for Anonymous Routing,* Proc. 12th Annual Computer Security Applications Conference, San Diego, CA, IEEE CS Press, December, 1996, pp. 95–104.

[17] M. Reiter and A. Rubin. *Crowds: Anonymity for Web Transactions (preliminary announcement),* DIMACS Technical Reports 97-15, April 1997.

[18] P. Syverson, D. Goldschlag, and M. Reed. *Anonymous Connections and Onion Routing,* Proceedings of the Symposium on Security and Privacy, Oakland, CA, May 1997.

[19] P. Syverson, S. Stubblebine, and D. Goldschlag. *Unlinkable Serial Transactions,* Financial Cryptography '97, February 1997.