Proxies for Anonymous Routing

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

Center for High Assurance Computer Systems
Washington, DC 20375-5337
Phone: +1 202.767.2389 (voice)
Fax: +1 202.404.7942 (fax)
e-mail: {reed, syverson, goldschlag}@itd.nrl.navy.mil

Abstract

Using traffic analysis, it is possible to infer who is talking to whom over a public network. This paper describes a flexible communications infrastructure, onion routing, which is resistant to traffic analysis. Onion routing lives just beneath the application layer, and is designed to interface with a wide variety of unmodified Internet services by means of proxies. Onion routing has been implemented on Sun Solaris 2.4; in addition, proxies for World Wide Web browsing (HTTP), remote logins (RLOGIN), e-mail (SMTP), and file transfers (FTP) have been implemented.

Onion routing provides application independent, real-time, and bi-directional anonymous connections that are resistant to both eavesdropping and traffic analysis. Applications making use of onion routing's anonymous connections may (and usually should) identify their users over the anonymous connection. User anonymity may be layered on top of the anonymous connections by removing identifying information from the data stream. Our goal here is anonymous connections, not anonymous communication. The use of a packet switched public network should not automatically reveal who is talking to whom. This is the traffic analysis that onion routing complicates.

1. Introduction

1.1 The Problem

Using traffic analysis, it is possible to infer who is talking to whom over a public network (Figure 1). For example, in a packet switched network [11], packets have a header used for routing, and a payload that carries the data. The header, which must be visible to the network (and to observers of the network), reveals the source and destination of the packet. Even if the header were obscured in some way, the packet could still be tracked as it moves through the network. Encrypting the payload is similarly ineffective, because the goal of traffic analysis is to identify who is talking to whom and not (to identify directly) the content of that conversation.

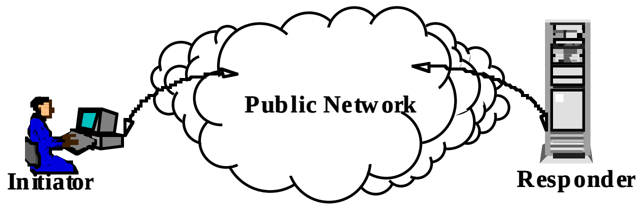

Figure 1. Communication over a Public Network

The efficiencies of the public Internet are strong motivation for companies to use it instead of private intranets. However, these companies may want to protect their interests. For example, a researcher using the World Wide Web (Web) may expect his particular focus to remain private, and inter-company collaborations should be confidential. Individuals may wish to protect their privacy as well. For example, the sending of e-mail should keep the identities of the sender and recipient hidden from observers. Also, a person shopping online may not want his visits tracked. Certainly someone spending anonymous e-cash would expect that the source of the e-cash be untraceable.

The use of a packet switched public network should not require revealing who is talking to whom. This paper presents a flexible communications infrastructure, *onion routing*, which is resistant to traffic analysis.

1.2 Objective

Onion routing is an infrastructure that

- complicates traffic analysis,
- separates identification from routing,
- supports many different applications.

Without dedicated links between every node and full utilization of each link, traffic analysis can, in principle, always be effective. But traffic analysis can be made more costly. Onion routing accomplishes this goal by separating identification from routing. Onion routing provides *anonymous connections* that are resistant to both eavesdropping and traffic analysis. Instead of containing source and destination information, packets moving along an anonymous connection contain only next hop and previous hop information. These anonymous connections can replace socket connections. Since socket connections are commonly used to support applications running over the Internet (like Web browsers, remote login, and e-mail) onion routing's anonymous connections can support a wide variety of unmodified applications using *proxies* that interface between applications and the onion routing network.

1.3 Overview of the Solution

Onion routing works in the following way: An application, instead of making a (socket) connection directly to a destination machine, makes a connection to an *onion routing proxy* on some remote machine. That onion routing proxy builds an anonymous connection through several other *onion routers* to the destination. Each onion router can only identify adjacent onion routers along the route. When the connection is broken, even this limited information about the connection is cleared at each onion router. Data passed along the anonymous connection appears different *at* and *to* each onion router, so data cannot be tracked en route and compromised onion routers cannot cooperate. An onion routing network can exist in several configurations that permit efficient usage by both large institutions and individuals.

1.4 Traffic Analysis

Traffic analysis makes inferences from three sources of information:

- Routing information
- Coincidences
- Load

Routing information is available in many forms: packet headers, phone touch-tones, and envelope addresses. This is the most obvious source that needs protecting. Coincidences, like similar traffic entering or leaving a node, or connections opening or closing at roughly the same time, are more difficult to hide. Finally, the very presence of communication over some link may reveal sensitive information. But load is very difficult to obscure if one is unwilling to use a constant amount of capacity all the time.

1.5 Organization of Paper

This paper is organized in the following way: Section 2 presents background information. Section 3 presents our goals and threat model. Section 4 presents our solution, and sections 5 and 6 provide more details. Section 7 describes the implemented prototype. Section 8 discusses vulnerabilities, costs, and variants of onion routing. Section 9 presents some concluding remarks.

2. Background

Chaum [1,2] defines a mechanism for routing data through intermediate nodes, called *mixes*. These intermediate nodes may reorder, delay, and pad traffic to complicate traffic analysis. Our onion routers are based upon mixes.

Anonymous Remailers [4,6] use mixes to provide anonymous e-mail services and also invent an address through which mail can be forwarded back to the original sender. Remailers work in a store-and-forward manner at the mail application layer by stripping off headers at each mix and forwarding the mail message to the next mix. Some remailers provide confirmation of delivery.

In [8,9], mixes are used to provide untraceable communication in an ISDN network. In the described 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 within each switch between the caller and the long distance network, which is obtained by routing calls through a predefined series of mixes. 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. Unlike anonymous remailers, anonymous connections are application independent and are meant to be used by a wide variety of Internet applications. The data carried by anonymous connections is varied, with real-time constraints often more severe than mail, but usually somewhat looser than voice. Both Web and ISDN connections are bi-directional, but, unlike ISDN, Web connections are likely to be small requests followed by short bursts of returned data. 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, because of the topology of the Internet, mixing has to be dispersed throughout the Internet, so hiding is greatly improved.

3. Objectives

3.1 Applications

Onion routing's anonymous connections are designed to replace TCP/IP socket connections [3] and to be able to work with unmodified applications. A socket connection is a reliable bi-directional connection carrying a stream of data between two machines. Socket connections provide the abstraction that shields an application from the unreliable and unordered communication that is provided by lower levels of the IP stack.

Many applications use socket connections:

- Web requests (HTTP)
- Remote logins (RLOGIN)
- e-mail (SMTP)
- File transfer (FTP)
- Internet Relay Chat (IRC)
- Encrypted IP Tunnel

These applications can connect to onion routing's anonymous connections using *proxies*. A proxy [11] is usually a relay between an initiating and responding application. In onion routing, anonymous connections are terminated by application specific proxies that relay information between the connection and the unmodified applications. Many applications are already *proxy aware* because proxies are commonly used to communicate through firewalls. For example, a Web browser on a network with a firewall will reach sites outside the firewall through an HTTP proxy on the firewall machine. In that way, direct connections are never made between internal and external machines.

3.2 Threat Model: Active and Passive Attacks

Onion routing's design is very conservative since it assumes that the public network is very vulnerable. In particular, we assume that:

- All traffic is visible.
- All traffic can be modified.
- Onion routers may be compromised.
- Compromised onion routers may cooperate.

In addition, a sophisticated adversary may be able to detect timing coincidences such as the near simultaneous opening of connections. Timing coincidences are very difficult to overcome, especially when real-time communication is important. But, if connections are routed over an unpredictable path in a busy network, this sort of attack is also very expensive.

The first four vulnerabilities, however, directly motivate certain design decisions in onion routing. Because traffic is visible, the headers and payloads of all traffic are essentially link encrypted between onion routers so the same data looks different when traveling between routers. Because traffic can be modified, stream ciphers [10] are used for encryption. Inserting, deleting, or modifying traffic en route will disrupt the stream and produce random bits downstream. Because onion routers may be compromised, anonymous connections span several onion routers, even though a single "perfect" mix is adequate to provide privacy. Because compromised onion routers may cooperate, data is encrypted in a layered fashion so it appears different to each onion router, not only between onion routers.

4. The Solution: Onion Routing

Onion routing has two parts: A network infrastructure that carries anonymous connections, and a proxy interfaces that mate these connections to unmodified applications.

4.1 Onion Routing: Network Infrastructure

The public network contains a set of onion routers. Each onion router has a single (socket) connection to each of a small set of neighboring onion routers. Onion routers only talk to their neighbors. Neighboring onion routers are neighbors for onion routing only. That is, communication between two neighboring onion routers is carried over a socket connection, and packets are routed (perhaps dynamically) through many hops by the IP protocol.

An anonymous *connection* is routed through a sequence of neighboring onion routers. Common segments of these routes are multiplexed over the single connection between neighbors. An onion router's obligation is to pass data from one connection to another after applying the appropriate cryptographic operations.

An anonymous connection from an initiator to a responder through four onion routers is illustrated in Figure 2.

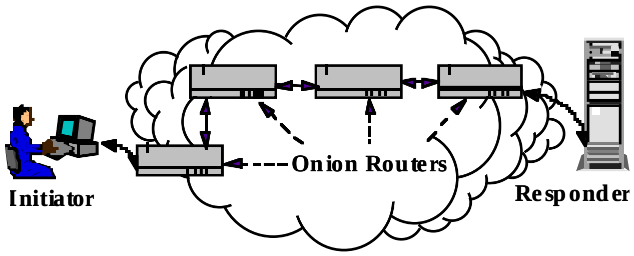

Figure 2. Onion Routing Network Infrastructure

4.2 Onion Routing: Proxy Interface

How are anonymous connections used? Proxies interface between applications and the network infrastructure. When a proxy is used on a firewall, it relays traffic between the protected site and the rest of the world. In onion routing, a proxy's functions are split into two: one part links the initiator to the anonymous connection and the other part links the anonymous connection to the responder. In this way, the initiating and responding applications need not be modified (although they do have to be able to use proxies).

Imagine an initiator sitting at her workstation using a Web browser. When she "clicks" on a URL link, the browser sends an HTTP request for that URL to some *onion routing proxy* instead of directly to the responder. In Figure 3, this is the onion routing proxy named W. W looks at the request and chooses a route through several other onion routers (e.g., W-X-Y-Z). W then sends an *onion* (see section 5.1) along that route; the onion is an instruction to those onion routers to construct an anonymous connection.

The last onion router in the route (Z) also functions as an onion routing proxy for the responder. Z passes data from the anonymous connection to the responder, and passes data from the responder back to the connection.

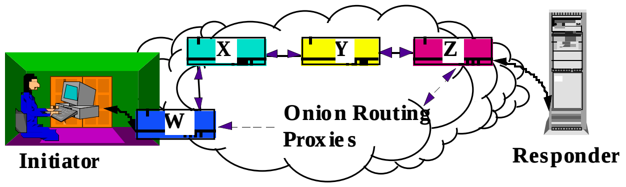

Figure 3: Onion Routing Proxy Interface

Instead of a single socket connection between an initiator and a responder, onion routing requires a socket connection between the initiator and his proxy, an anonymous connection between the initiator's proxy and the responder's proxy, and a socket connection between the responder's proxy and the responder. However, the three connections function as if they were a single (bi-directional and real-time) socket connection between the initiator and responder.

There are many configurations of an onion routing network. In one basic configuration, a site that is concerned about traffic analysis should control an onion routing proxy in order to protect communication between that proxy and its users. That onion routing proxy must also function as an intermediate onion router in other anonymous connections. If it is not used in this way, observers can monitor the load coming from onion routing proxy and trace it back to the sensitive site. However, if the onion routing proxy is also a busy intermediate onion router, observers cannot tell whether the sensitive site is consuming, producing, or relaying traffic.

Individuals may access an onion router through their Internet Services Provider (ISP), if the ISP controls an onion routing proxy. An individual could also make an encrypted connection to some public domain onion routing proxy. Finally, a user could run an onion routing proxy on his workstation, and route anonymous connections through other onion routers.

5. Using Onion Routing

After the initiator contacts his proxy, onion routing follows four stages:

1. Define the route.
2. Construct the anonymous connection.
3. Move data through the anonymous connection.
4. Destroy the anonymous connection.

The next four sections describe these stages in more detail. (The extra details in each *Details* subsection are independent of the rest of the paper.)

5.1 Defining the Route

Consider Figure 3. The initiator's proxy, W, chooses to make an anonymous connection through (W-X-Y-Z). Therefore, W constructs a layered data structure called an *onion* (Figure 4):

Each layer of the onion is intended for a particular onion router and contains the identity of the next onion router in the anonymous connection, and the key that should be used when communicating with the previous onion router in the connection. The final layer of the onion is intended for Z. Since Z is the last onion router in the connection, its layer only contains a key.

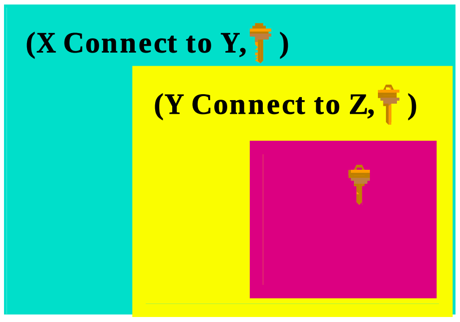

Figure 4: An Onion

Using public key cryptography [10], the onion is constructed so only the intended recipient can peel off the outermost layer, thereby revealing both his layer and the onion embedded inside. No recipient knows who created the onion. So, onion routers can identify only whom they received an onion from and to whom they are obliged to send the embedded onion. And, no recipient can determine what the other onions embedded in an onion look like.

The onion routing proxy that creates an onion keeps a copy of the keys in the onion until the anonymous connection is destroyed. We will see how these keys are used in sections 5.3 and 5.4.

5.1.1 Onion Details

The onion routing proxy routes the anonymous connection through neighboring onion routers. Therefore, it must know the topology of the onion routing network.

The size of an onion limits the length of a route. To prevent observers from inferring the length of a route, onions are padded to some fixed size. This padding becomes part of and is indistinguishable from the already embedded onion.

The key at each layer of the onion is used for bi-directional communication between an onion router and the previous onion router. Therefore, the key really specifies two stream ciphers, one for forward communication (in the direction the onion travels) and the other for backward communication (in the opposite direction).

Each layer of an onion also contains an expiration time. An onion router is to ignore an expired onion and is to ignore replayed onions. Therefore, onion routers must keep track of onions during their lifetimes.

For efficiency, the entire onion is not encrypted using a public key cryptosystem. Instead some prefix (corresponding to the block size of the public key cryptosystem) of the onion is encrypted using public key cryptography, and the rest of the onion is encrypted using an efficient stream cipher initialized with a key specified in the prefix [5,7,10].

5.2 Constructing the Anonymous Connection

After constructing the onion, W sends the onion to the first onion router in the anonymous connection. The onion moves between onion routers (Figure 5):

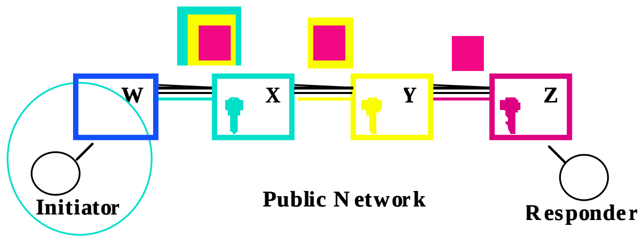

Figure 5: Use of an Onion

Each layer of the onion is intended for a particular onion router, and can be peeled off only by that onion router. The first layer of this onion is intended for X. When X peels off that layer, it obtains a key that it will use when communicating with W (from whom X received the onion), and notes that future traffic from that connection should be forwarded to Y. X also forwards the embedded onion to Y. In a similar way, Y peels off its layer of the onion, revealing the key that it should use to communicate with X (from whom Y received the onion), and notes that future traffic from that connection should be forwarded to Z. Z peels off its layer, revealing only the key that it will use to communicate with Y, and notes that it is the last onion router in the anonymous connection. The first data that Z receives along that anonymous connection will identify the intended responder.

5.2.1 Anonymous Connection Construction Details

To keep onion size constant, each onion router is obliged to add padding to the onion corresponding to the fixed size layer that was removed. Onion routers cannot distinguish padding from embedded onions. If an onion router fails to pad an onion, however, the next onion router will notice that the onion it received is too small and will not process the onion. Because of the padding, even onion routers themselves cannot tell how much of an anonymous connection has been constructed.

Remember that all communication between neighboring onion routers is multiplexed in the data stream of a single socket connection. Therefore, all data travels in a series of fixed size cells. Each cell has a header that identifies the anonymous connection it is assigned to, as well as the type of payload it carries. For example, cells carrying onions will be labeled as `onion` cells, and will also contain the identifier of the new anonymous connection that is to be multiplexed over that socket connection. Notice that this identifier is chosen by the onion router relaying the onion, and in each socket connection carrying a segment of an anonymous connection, the anonymous connection may have a different identifier. Each onion router maintains a table that maps between the identifiers of incoming connections and outgoing connections, and the cryptographic keys that are to be applied to data moving along an anonymous connection.

Cells traveling over a socket connection between onion routers are link encrypted in a peculiar way: headers and payloads are encrypted separately, for efficiency. For example, headers are encrypted with some stream cipher negotiated between the neighboring onion routers. The payload of a cell of type `onion` need not be encrypted, since the onion was already encrypted for the next onion router by the onion routing proxy that created the onion.

Because of the link encryption, observers monitoring the data stream between onion routers cannot read cell headers. Therefore, observers cannot distinguish between onions and other types of cells.

5.3 Moving Data Forward

The anonymous connection moves data from the initiator's proxy to the responder's proxy and vice versa. In the forward direction, the initiator sends plaintext to his onion routing proxy. The onion routing proxy repeatedly *crypts*[1] the data using the inverse of the keys[2] specified in the onion, applying the keys innermost first. Each onion router along the route removes one layer of cryption. The responder's proxy forwards the plaintext to the responder.

This is illustrated in Figure 6.

[1] We define the verb *crypt* to mean the application of a cryptographic operation, be it encryption or decryption, where the two are logically interchangeable. For example, in a stream cipher using Output Feedback Mode (e.g., DES OFB), encryption and decryption are the same operation.

[2] Each key really specifies a stream cipher. The inverse of a key, therefore, is the inverse of the corresponding stream cipher.

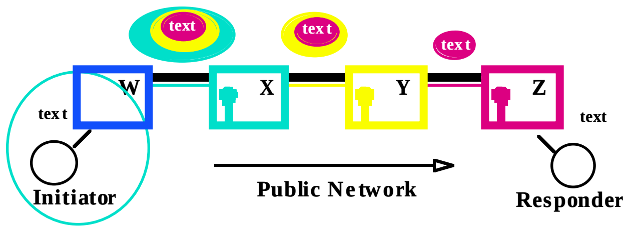

Figure 6: Moving Data Forward

The purpose of the pre-cryptions is to make the data look different as it travels through the anonymous connections, both to outside observers and to the onion routers. Notice that observers cannot match data along the route, and onion routers cannot predict what data will look like later.

Notice that each onion router does one cryption, while the initiator's onion routing proxy does one pre-cryption for each subsequent onion router in the connection.

5.3.1 Moving Data Forward Details

Data moving in the direction that the onion was sent is defined to be moving in the forward direction. Data moving in the reverse direction is defined to be moving backward. This distinction is important when discussing *reply onions* (section 6).

As with onions, data is carried in cells through the multiplexed socket connections. The cells have type `data` and are labeled with the identifier of the associated anonymous connection. Although the headers of `data` cells are link encrypted between onion routers, the payloads of `data` cells are not link encrypted, as the cryption operation done at each onion router is sufficient.

When a `data` cell arrives, the onion router looks up the cell's identifier in its tables and finds the corresponding outbound identifier. The appropriate cryptographic operation is applied and the crypted payload is formed and sent along the outbound connection.

As with the link encryption of the headers, the payloads of `data` cells are encrypted using stream ciphers.

5.4 Moving Data Backward

Moving data backward is just the reverse of sending data forward. The responder's onion routing proxy receives plaintext from the responder. It and each subsequent onion router adds one layer of cryption and sends the data to the next onion router. The initiator's onion routing proxy removes the layers of cryption by applying the inverse of the keys in the onion outermost first. The resulting plaintext is forwarded to the initiator.

This is illustrated in Figure 7.

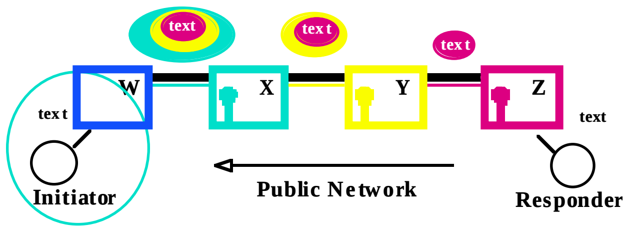

Figure 7: Moving Data Backward

5.4.1 Moving Data Backward Details

As with forward data, the initiator's onion routing proxy handles the bulk of the cryption burden.

5.5 Destroying the Anonymous Connection

Just as socket connections are torn down, anonymous connections need to be destroyed when the connection is broken. An onion router that decides to tear down a connection sends a destroy message forward and backward along the anonymous connection. It also cleans up its own tables. An onion router that receives a destroy message is obliged to clean up its own table and relay the message in the same direction.

This is illustrated in Figure 8.

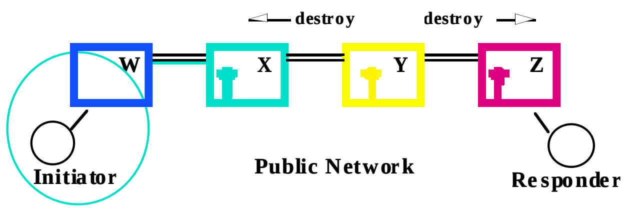

Figure 8: Destroying an Anonymous Connection

Notice that the multiplexed socket connections between neighboring onion routers remain active.

5.5.1 Destroy Details

Destroy messages are sent in cells of type `destroy`. The header identifies the anonymous connection that is to be destroyed. The payload is random and changes at each onion router.

6. Reply Onions

The (forward) onion described in section 5 is used by the initiator's onion routing proxy to construct an anonymous connection to some responder's onion routing proxy. What happens if an initiator expects a later reply from the responder? An obvious solution is to keep the anonymous connection open. This may not always be practical. Another solution is a *reply onion*.

An initiator's onion routing proxy can create a reply onion that defines a route back to him. For example,

Figure 9 illustrates a reply onion that will construct an anonymous connection back to W from Z through onion routers Y and X:

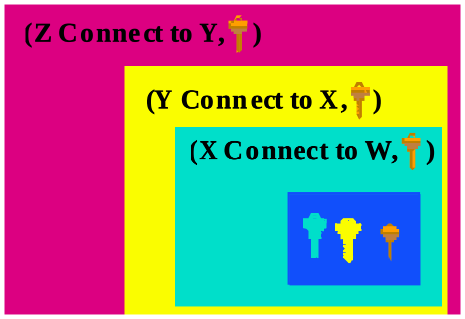

Figure 9: A Reply Onion

The reply onion is sent by the responder to onion routing proxy Z, who peels off its first layer, and sends the embedded reply onion on to onion router Y after extracting the key that Z will use when communicating with Y. Onion routers Y and X do the same operation. Onion routing proxy W receives a reply onion with a sequence of keys.

The anonymous connection established by this reply onion is illustrated in Figure 6, and is identical to the anonymous connection established by the (forward) onion illustrated in Figure 4. Once the anonymous connection is established, each onion router has the same role it has in a forward connection from the initiator to the responder: That is, the initiator's onion routing proxy repeatedly pre-crypts data and other onion routers crypt only once.

Reply onion's can also be used to allow anonymous replies back to some initiator. The initiator may publish a reply onion, which can be picked up and used by any responder. The responder forwards the onion to the designated onion routing proxy and an anonymous connection back to the initiator will be constructed.

6.1 Reply Onion Details

As with forward onions, reply onions contain expiration times to prevent replays. This means that published reply onions can only be used once. If an initiator expects several replies, he should publish many reply onions.

During connection construction, both the responder's and initiator's onion routing proxies know that they have a reply onion. Once the connection is established, however, this distinction is irrelevant. Intermediate onion routers can never distinguish between forward and reply onions. In fact, the only difference between anonymous connections formed by forward onions and those formed by reply onions is that the sets of keys used to crypt data in each direction are swapped: forward keys are used as backward keys and vice versa.

A reply onion may also be created by a third party to define an anonymous connection back to some initiator. Third party reply onions are unusual because both the third party and the initiator know all the onion keys.

7. Implementation

Onion routing has been implemented on Sun Solaris 2.4. Onion routing proxies for Web browsing (HTTP), RLOGIN, e-mail (SMTP), and FTP have been implemented also. Furthermore, versions of these proxies that anonymize the data stream have been implemented. These proxies allow anonymous communication that is resistant to both eavesdropping and traffic analysis.

An extension to this prototype must handle changes to the topology of the onion routing network. This includes, for example, new onion routers, different neighbors, and distribution of onion routers' public keys.

8. Discussion

To be effective, onion routing must be widely deployed and there must be significant use of all the onion routers. Furthermore, onion routing proxies must also be intermediate onion routers. Otherwise, it is easy to infer that traffic to and from a particular onion routing proxy is really to and from the sensitive site that controls the proxy.

8.1 Vulnerabilities

Onion routing is not invulnerable to traffic analysis attacks. With enough data, it is still possible to analyze usage patterns and make educated guesses about the routing of messages. Also, since our first application (Web requests) requires real-time communication, it may be possible to detect the near simultaneous opening of socket connections on the initiator's and responders' onion routing proxies, thereby revealing who is requesting what information. (Of course, even this attack is impossible if the initiator's onion routing proxy is controlled by his sensitive site.) However, these sorts of attacks require the collection and analysis of huge amounts of data by external observers.

One way to further complicate this sort of analysis is to pass dummy traffic through the network to make the traffic level fairly constant. There is an obvious tradeoff here between security and cost: Adding dummy traffic undermines the efficiencies of the Internet as a shared resource. It is difficult to calculate the value of this tradeoff. If traffic is very bursty and response time is important, smoothing out network traffic requires wasting capacity. If, however, traffic is relatively constant, additional smoothing may not be necessary. From a practical point of view, the Internet may not provide the control necessary to smooth out traffic: unlike ATM, users do not own capacity on shared connections. The important observation, however, is that onion routing provides an architecture within which these tradeoffs can be made and explored.

Other attacks depend upon compromised onion routers. If the initiator's onion routing proxy is compromised, then all information is revealed. In general, it is sufficient for a single onion router to be uncompromised to complicate traffic analysis.

Any compromised onion router can still destroy connections or stop forwarding messages, resulting in denial of service attacks. Although this appears to be akin to the denial of service problem in IP source routing, where the unreachability of any part of the route causes packet loss, the situation is closer to loose source routing where packets may be routed arbitrarily between the prespecified routers. Furthermore, in onion routing, if the connection is broken, the rest of the onion routers are informed via destroy messages.

Onion routing uses expiration times to prevent replay attacks. It is curious that, unlike other services that depend upon a common clock, the vulnerability due to poor synchronization here is a denial of service attack, instead of a replay attack. If an onion router's clock is too fast, otherwise timely onions will appear to have already expired. Also, since expiration times define the window during which onion routers must store used onions, an onion router with a slow clock will end up storing more information.

The data stream cannot be replayed, as stream ciphers are used for encryption. If the data stream is changed in any way, synchrony will be lost and the data stream will become irreversibly corrupted. Since TCP/IP socket connections are used to carry the data stream, we expect error free data delivery.

8.2 Cryptographic Overhead

In onion routing, the cryptographic overhead on intermediate onion routers is less than the burden of link encryption on routers. In link encryption, each packet is encrypted by each sender and decrypted by each recipient. In onion routing, only one cryptographic operation is applied between every two onion routers. This is because the initiator's onion routing proxy repeatedly pre-crypts data.

The total number of encryptions remains the same, however. It is interesting to note that shifting the encryption burden provides (for free):

- Link encryption.
- End to end encryption.
- Data hiding: the same data looks different to each onion router.

8.3 Infrastructure Variations

An interesting application of onion routing is a variation of IRC (Internet Relay Chat). Two sites can build anonymous connections through onion routers they each trust to meet at a designated onion routing proxy. That proxy mates the two connections. Privacy is guaranteed, and neither party needs to trust the other to hide his participation from outside observers.

Since connection setup is relatively expensive, it may be useful to delink sockets and anonymous connections. For example, when using a Web browser to view a particular Web page, several socket connections may be established to retrieve various parts of the document. There is no reason that those socket connections could not all use (either serially or in parallel) the same anonymous connection.

It is interesting to consider protocol encapsulation. Onions can be carried over the anonymous connections. This would enable extending connections and may enable using parts of connections or linking together parts of connections. This process allows the length of anonymous connection routes to be extended indefinitely, and permits the size of the onion routing network to grow arbitrarily.

Since real-time connections are inherently more vulnerable to traffic analysis than less time critical applications, it makes sense to tag connections with various service guarantees. A real-time connection will be less resistant to traffic analysis than a slow connection because intermediate onion routers have less flexibility with buffering its data stream.

Onion routing networks can exist in many configurations to accommodate the requirements of large institutions, ISPs, and individuals through a combination of institution or ISP controlled onion routing proxies, public domain onion routing proxies, and public domain onion routers. The combination of many sources of traffic enables the network to further complicate traffic analysis.

8.4 Lower Levels of the Stack

Can onion routing be implemented at lower levels of the communications stack? The obvious advantage is that this would eliminate the need for application specific onion routing proxies. The difficulty with pushing onion routing beneath the socket layer of the IP stack is that onion routing's connection setup is relatively expensive, and is impractical to use each time a packet is sent over a connectionless circuit.

Since ATM is connection based, however, it is perfectly reasonable to consider using onion routing's approach for connection setup to make anonymous ATM connections. In fact, in our prototype, we are modeling our cells based on ATM cells.

9. Conclusion

Onion routing provides real-time, bi-directional communication through anonymous connections that are resistant to both eavesdropping and traffic analysis. These anonymous connections can substitute for socket connections in a wide variety of unmodified Internet applications using proxies. Our prototype of onion routing includes proxies for Web browsers (HTTP), remote login, e-mail, and file transfer protocols as well as anonymizing versions of these protocols The anonymizing version of the e-mail proxy creates an anonymous connection between two *sendmail* daemons and removes identifying information from the headers of the mail message. This approach contrasts with Anonymous Remailers, where each remailer provides a single hop in a chain of mail forwarding. This highlights the difference between onion routing and other uses of Chaum mixes: Privacy and anonymity are moved beneath the application layer and made application independent.

Onion routing will only be effective in complicating traffic analysis if its infrastructure is widely deployed and widely used. This deployment is considerably simplified because applications need not be modified.

Our motivation here is not to provide anonymous communication, but to separate identification from routing. Authenticating information must be carried in the data stream. Applications can (and usually should) identify themselves to each other. But, the use of a public network should not automatically reveal the identities of communicating parties. The goal here is anonymous routing, not anonymity.

References

1. D. Chaum. *Untraceable Electronic Mail, Return Addresses, and Digital Pseudonyms,* Communications of the ACM, v. 24, n. 2, Feb. 1981, pages 84-88.
2. D. Chaum, *The Dining Cryptographers Problem: Unconditional Sender and Recipient Untraceability,* Journal of Cryptology, 1/1, 1988, pages 65-75.
3. D. E. Comer. *Internetworking with TCP/IP, Volume 1: Principles, Protocols, and Architecture,* Prentice--Hall, Engelwood Cliffs, New Jersey, 1995.
4. L. Cottrell. *Mixmaster and Remailer Attacks,* http://obscura.obscura.com/~loki/remailer/remailer-essay.html
5. D. Goldschlag, M. Reed, and P. Syverson. *Hiding Routing Information*. Workshop on Information Hiding, Cambridge, UK, May, 1996.
6. C. Gulcu and G. Tsudik. *Mixing Email with* Babel, 1996 Symposium on Network and Distributed System Security, San Diego, February 1996.
7. A. Pfitzmann and B. Pfitzmann. *How to Break the Direct RSA-implementation of MIXes,* Advances in Cryptology--EUROCRYPT '89 Proceedings, Springer-Verlag, Berlin, 1990, pages 373-381.
8. 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, Heidelberg 1991, pages 451-463.
9. A. Pfitzmann and M. Waidner. *Networks Without User Observability,* Computers & Security, 6/2 1987, pages 158-166.
10. B. Schneier. *Applied Cryptography: Protocols, Algorithms and Source Code in C,* 2nd edition, John Wiley and Sons, 1996, (the red one).
11. W. R. Stevens. *TCP/IP Illustrated, Volume 3: TCP for Transactions, HTTP, NNTP, and the UNIX Domain Protocols,* Addison--Wesley, Reading, Mass., 1996.