Proceedings of the third DARPA Information Survivability
Conference and Exposition (DISCEX 3), April 2003

Design and Analysis of an IP-Layer Anonymizing Infrastructure

H.T. Kung, Chen-Mou Cheng, Koan-Sin Tan, and Scott Bradner
Harvard University
kung@harvard.edu, doug@eecs.harvard.edu, freedom@acm.org, sob@harvard.edu

Abstract

This paper describes an IP-layer anonymizing infrastructure, called ANON, which allows server addresses to be hidden from clients and vice versa. In providing address anonymity, ANON uses a network resident set of IP-layer anonymizing forwarders that can forward IP packets with nested encryption and decryption applied to their source and destination addresses. To prevent adversaries from compromising the anonymity by learning the forwarding path, ANON incorporates a suite of countermeasures, including non-malleable, semantically secure link encryption and link padding. To lower the bandwidth cost of padding traffic, two novel algorithms are suggested: on-demand link padding and probabilistic link padding. To prevent inband denial of service (DoS) attacks through the anonymizing infrastructure itself, ANON uses rate limiting. Finally, ANON makes use of fault-tolerant transport networks to enhance its resilience against failures and outband attacks.

1. Introduction

Over the current Internet, when a client acquires services from an application server, called a *target server* subsequently, packets sent and received by the client reveal the server's IP address in the packet headers. There are a number of situations where it would be useful for applications to be able to communicate with each other without revealing the IP address of the destination to the source, the IP address of the source to the destination, or both. The IP address of a destination may also want to be hidden from the public, beyond just clients. For example, a web site may want to hide its IP addresses to reduce the risk of denial of service (DoS) attacks aimed at these addresses, or an organization may want to ensure its anonymity by not revealing its IP addresses. See [13] for discussion on supporting anonymity at the IP layer.

One way to achieve this anonymity, as described in this paper, is to use an overlay network connecting a network resident set of IP-layer servers that can forward IP packets, with nested encryption and decryption applied to their source and destination addresses. We will call these network resident IP-layer servers *anonymizing forwarders*, or simply *forwarders*, and an IP anonymizing infrastructure based on these anonymizing forwarders an *forwarding infrastructure*, or simply *ANON*.

Using ANON, a client can send and receive packets to and from target servers by using *server handles* rather than server IP addresses, where a server handle, or simply a *handle*, is an information string from which a sequence of forwarders in the ANON infrastructure can translate into the IP address of the target server. This is analogous to an organization sending and receiving U.S. mail using its P.O. Box number rather than street address, where the P.O. Box number is the handle to the organization, and postal offices correspond to ANON forwarders.

In addition to performing address encryption and decryption, ANON employs a set of countermeasures to protect against adversaries from learning forwarding paths that could lead to the discovery of addresses that the infrastructure intends to hide. This includes protocol camouflaging, link encryption, link padding, and use of fault-tolerant overlay networks to forward packets, such as Chord [16], that can guard against path discovery using congestion-based attacks or DoS attacks. To defend against possible "inband" DoS attacks by adversaries on forwarders or target servers through the ANON infrastructure itself, ANON uses per-source and per-destination rate limiting rate limiting. Finally, to defend against "outband" DoS attacks by adversaries through networking paths external to the ANON infrastructure, ANON uses redundant forwarders, forwarders with anycast addresses [8,12], or a fault-tolerant overlay network mentioned above.

For link padding, ANON offers new ways of lowering its bandwidth cost. Note that link padding may degrade network's ability in serving normal traffic. Since padding traffic is supposed to be indistinguishable to routers, it would be inappropriate to assume that routers can give normal traffic preferential treatments to padding traffic. Consider those overplay links which pass through one or more intermediate routers. Padding traffic on such links will increase the load on these intermediate routers and

their adjacent links and thus decrease their bandwidth available in serving normal traffic. To reduce the bandwidth cost of link padding, we have developed two new approaches in ANON. The first is *on-demand link padding* which generates padding traffic only in the presence of real traffic and only at the rate just above that of real traffic. The second is *probabilistic link padding* which generates padding traffic probabilistically so that the total traffic load satisfies a heavy-tail probability distribution, rather than the usual uniform distribution. This allows increased bursts of real traffic without increasing padding.

The design of ANON assumes that it will be used mainly for low- to medium-bandwidth signaling and data applications, not data transfer that may require high bandwidth. As will be explained later in the paper, this assumption makes our countermeasures such as rate limiting effective. There are many applications that fit the model defined here, including signaling protocols such as connection setup and termination, user authentication and authorization, service discovery and registration, and instant messaging.

Consider, for example, the use of ANON to protect authentication servers against DoS attacks. By definition, an authentication server needs to process requests from unknown users. An adversary can exploit this fact to mount DoS attacks on the authentication server by sending it a large number of fake authentication requests. The risk of such DoS attacks can be significant for sophisticated authentication that uses substantial resources. ANON provides a solution to this problem by hiding the IP address of the authentication server from the public and thus from adversaries.

We have implemented a laboratory testbed for ANON at Harvard. The testbed currently incorporates a number of countermeasures mentioned above, including protocol camouflaging, link encryption, on-demand link padding, and rate limiting. For ease of use, the testbed also includes NAT gateways that allow existing client and server applications to use ANON directly.

The rest of this paper is organized as follows. We first describe our threat model in Section 2. We state our design objectives in Section 3 and give an overview and a few usage examples of the ANON infrastructure in Section 4. We then describe in Section 5 ANON's countermeasures in defense against threats in our threat model. In Section 6, we turn our focus to the new link padding schemes, namely, on-demand link padding and probabilistic link padding. In Section 7, we sketch an approach of using a fault-tolerant transport network to enhance ANON's defense against outband DoS attacks. We briefly describe our laboratory testbed implementation in Section 8. We compare ANON with previous approaches in related areas in Section 9 and conclude this paper in Section 10.

To be complete, we include in this paper some material which appeared previously in an overview paper [11].

2. Threat Model

The ANON forwarding infrastructure provides countermeasures to the following three types of threats:

- Type 1 threat (unauthorized address discovery). The forwarding infrastructure may leak address information that it is supposed to hide.
- Type 2 threat (inband DoS attacks). The forwarding infrastructure may be used as a conduit to launch DoS attacks on forwarders or target servers.
- Type 3 threat (outband DoS attacks). Forwarders and the underlying transport network in the infrastructure may themselves be subject to DoS attacks through external network paths.

We assume throughout this paper that forwarders are managed by trusted third parties and that they cannot be compromised. In addition, we assume that the location and addresses of forwarders are not publicized. This means that adversaries will not know the addresses of forwarders beyond the first hop, without tracing the forwarding infrastructure.

However, after having located a forwarder, we assume that adversaries can monitor traffic and observe the content of packets on links in and out of the forwarder. We feel that this strong adversary model is justified in view of the fact that it is possible to monitor a particular set of links even without physical wiretapping. For example, by tricking routers to think that there is a shorter path to or from a forwarder, an adversary will be able to direct the forwarder's traffic to his own networks for monitoring and logging purposes.

An example attack related to type 1 threat works as follows. Acting as a legitimate client, an adversary sends probe packets to a target server whose address he intends to discover. By specially marking his packets or transmitting them according to certain timing patterns, referred to as *packet tagging* or *traffic tagging* [14], respectively, and by using link monitoring, the adversary attempts to identify these packets on links of the forwarding infrastructure and trace through these links to discover the address of the target server. ANON uses techniques such as link encryption and link padding to defend against these attacks.

In addition, type 1 threat includes congestion-based attacks such as Wei Dai' attack [4] and DoS attacks, in which an adversary attempts to determine whether or not a given link or a node is on the forwarding path by

congesting the link or attacking the node, while sending probe requests to the target server. If the attack does not affect the return of replies from the server, the adversary concludes that the link or the node is not on the forwarding path; otherwise, it is. As described in Section 7, ANON uses fault-tolerant transport networks to defend against these types of attacks.

In type 2 threat, an adversary, again acting as a legitimate client, sends a large number of packets to a target server with the intention of swamping the server or forwarders on the forwarding path. Because the attack uses the infrastructure itself, we call it an *inband DoS attack*. ANON uses per-source and per-destination rate limiting to curtail these attacks by having upstream forwarders drop excessive traffic.

In type 3 threat, an adversary, after having discovered the address of a forwarder, sends DoS attack packets to the address, or paths to it, using network paths external to the forwarding infrastructure. Since the attack does not use the forwarding infrastructure, we call it an *outband DoS attack*. To address type 3 threat, ANON forwarders need to be resilient to DoS attacks, which can be achieved through the use of redundant forwarders, forwarders with anycast-style addresses, or the use of a fault-tolerant transport network such as Chord.

Our assumption that forwarders can not be compromised simplifies our threat model. For example, this assumption has allowed us not to be concerned with attacks originating from compromised forwarders and other forwarder-facilitated attacks. We are looking into a revised threat model where some of the forwarders might have been compromised.

3. ANON Design Objectives

There are two main objectives for the design of the ANON forwarding infrastructure:

- Stateless, real-time forwarding of IP packets with nested encryption and decryption applied to their source and destination addresses. The forwarding infrastructure should be able to support real-time traffic with small delays in each hop. Furthermore, for ease of management, forwarders should be stateless in packet forwarding. In particular, forwarders should not keep any connection or session state or any NAT-like mapping tables, and should be oblivious to the number of connections or sessions. However, a forwarder may keep statistics on traffic to specific destinations and from specific sources so that per-source or per-destination rate limiting can be implemented.
- Providing defense mechanisms against the three types of threats described in Section 2.

4. The ANON Infrastructure

The ANON infrastructure consists of a set of anonymizing forwarders and some number of initialization servers, as depicted in Figure 1. Forwarders will encrypt and decrypt IP addresses, whereas initialization servers will provide clients with handles to target servers, which consist of nestedly encrypted addresses of target servers and addresses of entry forwarders. An overlay network connects the forwarders. That is, a pair of forwarders may be connected using a path involving multiple IP routers. Request packets from a client to a target server will be forwarded over a forwarding path consisting of a subset of these forwarders. Reply packets from the target server to the client will use the same path in the reverse direction. For different reply-request sessions, different forwarding paths may be used.

Some trusted third parties will operate the anonymizing forwarding infrastructure. Since forwarders may decrypt IP addresses and thereby have access to the IP addresses that ANON intends to hide, it is important that forwarders are properly protected from being compromised. As mentioned earlier, we assume in the analysis of this paper that forwarders cannot be compromised. To increase availability, forwarders may use anycast-style addressing [8,12], so that any of a number of forwarders using the same anycast address may forward a packet sent to it.

The role of initialization servers is to provide clients with handles to target servers. Thus, initialization servers and server handles provided by them need to be properly authenticated, using, e.g., digital certificates, to ensure that these server handles will be trustworthy. In addition, initialization servers may need to be replicated in various locations to ensure their high availability.

Consider, for example, the case of hiding the IP address of a target server from clients. In this case, the use of ANON will involve three usage steps:

- Server registration. A target server whose IP address

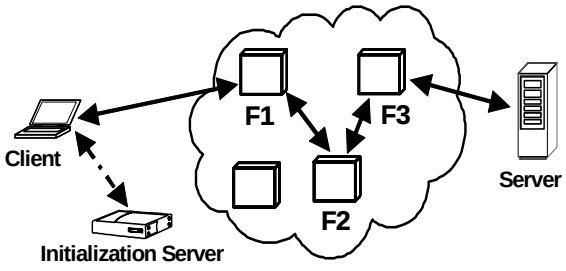

Figure 1. The ANON infrastructure. F1, F2, and F3 are anonymizing forwarders, and the solid arrows indicate an instance of a packet forwarding path.

needs to be hidden will invoke a process that constructs handles for the target server by selecting sequences of forwarders and using these forwarders to compute nestedly encrypted addresses for the target server, and finally registers the constructed handles at initialization servers. For a given server handle, the sequence of forwarders can be selected manually or automatically, as well as statically or dynamically. Note that initialization servers store handles, rather than IP addresses of target servers, so compromising an initialization server does not lead to compromise of target servers' address anonymity.

- Client initialization. Given a target server to which a client wishes to access, the client obtains a handle to the target server from an initialization server. The client may use alternative handles, should the forwarding paths corresponding to current handles fail to operate.
- Packet forwarding. Based on the information obtained from the client initialization, ANON forwards packets to and from the target server over the selected sequence of forwarders.

4.1. Notations and Assumptions

C: Client

S: Application Server
- S has generated an asymmetric key pair of public and private keys and S holds the private key.

I: Initialization Server

F: Anonymizing Forwarder
- F is assumed to be outside firewalls or NATs of C, S and I, should these firewalls or NATs exist.
- F holds a symmetric key for all its forwarding operations. Each F has its own symmetric key not known to others. Fs using the same anycast address share the same symmetric key.

[X]: IP address of X
- If X is a client behind a firewall or NAT, [X] is the IP address as seen from the outside of the organization.
- If X is a forwarder, [X] may be its unicast or anycast address.

[X]{payload}[Y]
- A packet with its source and destination IP addresses being [X] and [Y], respectively.

(z)r, where r is a symmetric key
- It is the content z encrypted in r.
- When r is the lower case letter of the name of a forwarder or server, r denotes the symmetric key of the forwarder or server. For example, (z)f means z encrypted in the symmetric key of forwarder F.

(z)A, where A is the name of a forwarder or a server
- It is the content z signed in A's private key or encrypted in A's public key. In the former case, A did the signing, whereas in the latter case another entity did the encryption.

X->Y: [X]{payload}[Y]
- X sends packet, [X]{payload}[Y], to Y.

X: operation
- X performs operation.

4.2. Forwarding Operations

Depending on the application, a forwarder may perform one of the forwarding operations listed below. Subsequent usage examples will illustrate the usage of these operations.

FWD-INC ("forward and include"):
Input packet: [X]{msg, [Y]}[F]
Output packet: [F]{msg, [X]}[Y]

FWD-CLR ("forward and clear"):
Input packet: [X]{msg, [Y]}[F]
Output packet: [F]{msg}[Y]

FWD-ENC ("forward and encrypt"):
Input packet: [X]{msg, [Y]}[F]
Output packet: [F]{msg, ([X])f}[Y]

DEC-FWD-INC ("decrypt, forward and include"):
Input packet: [X]{msg, ([Y])f}[F]
Output packet: [F]{msg, [X]}[Y]

DEC-FWD-CLR ("decrypt, forward and clear"):
Input packet: [X]{msg, ([Y])f}[F]
Output packet: [F]{msg}[Y]

DEC-FWD-ENC ("decrypt, forward and encrypt"):
Input packet: [X]{msg, ([Y])f}[F]
Output packet: [F]{msg, ([X])f}[Y]

In addition to these forwarding operations, a forwarder may also support management operations such as target servers' registration.

4.3. Baseline Usage Example B1: Hide Target Server's Address

This example illustrates the use of ANON to achieve the following two objectives:

- A client C sends a request to a target server S without knowing S's address.
- C receives a reply from S without knowing S's address.

As described earlier, client C first interacts with an initialization server I. In its message to I, C expresses its wish to access a target server S. Then the initialization server I securely sends C a server handle, via, e.g., SSL, containing the following two items:

- [F], the unicast address of a forwarder F, or the anycast

address of a set of forwarders, also denoted by F.
- ([S])f

When C wishes to send a request to S, it builds a request packet containing the following content and sends it to [F]:
- (req, ck)S, where req is C's request to S, and ck is a cookie associated with the packet. (req, ck)S is (req, ck) encrypted by S's public key. The purpose of ck is to identify the request. After the packet is sent, C will keep ck around, so that it can be used later to associate the reply received from S with the request.
- ([S])f

Upon receiving the request packet, F decrypts the packet and forwards it to [S], with the source address of the original packet, [C], as seen by F included in the packet payload. That is, F performs the operation DEC-FWD-INC.

When S receives the request packet, it builds a reply packet containing the following content and sends it to [F]:
- (rep, ck)S: reply and cookie signed by S with its private key.
- [C]: the source address of the original packet as seen by F.

Upon receiving the reply packet, F forwards it to [C] without including [S] in the packet payload, so [S] will not be revealed. That is, F performs the operation FWD-CLR. When C receives the packet, it verifies S's signature on the received reply and cookie using S's public key. By comparing the received cookie with the original cookie stored at C, C determines whether the received reply is for the original request.

We summarize usage B1 for hiding [S] as follows:

C -> F: [C]{(req, ck)S, ([S])f}[F]
F: DEC-FWD-INC
F -> S: [F]{(req, ck)S, [C]}[S]
S: reply
S -> F: [S]{(rep, ck)S, [C]}[F]
F: FWD-CLR
F -> C: [F]{(rep, ck)S}[C]
C: verify S's signature and compare cookies

4.4. Baseline Usage Example B2: Hide Client's Address

This example illustrates the use of ANON to achieve the following two objectives:
- S receives request from C without knowing C's address.
- S sends reply to C without knowing C's address.

In this case, C will obtain [F] and [S] from an initialization server. We summarize usage B2 for hiding [C] as follows:

C -> F: [C]{req, [S]}[F]
F: FWD-ENC
F -> S: [F]{req, ([C])f}[S]
S: reply
S -> F: [S]{rep, ([C])f}[F]
F: DEC-FWD-INC
F -> C: [F]{rep, [S]}[C]
C: receive reply and [S]

Note that usage B1 and B2 can be combined to yield a scheme that will hide both [C] and [S].

4.5. Enhanced Two-hop Usage Example: Hide Target Server's Address

This example is an enhanced version of baseline usage example B1 above. It is designed to defend against an attack related to the type 1 threat described in Section 2. In this attack, an adversary, acting as C, repetitively submits request:

C -> F: [C]{(req, ck)S, ([S])f}[F]

while monitoring packet contents on links that could be on the path from F to S and vice versa. If those packets on the links that result from these repeated request submissions are identifiable by link monitoring, then the adversary will be able to learn [S] by examining destination or source addresses of these packets. It is thus important to avoid invariant bit strings in packet load, such as [C], (req, ck)S and (rep, ck)S in usage example B1 above, or any invariant tags the adversary may insert in his packets, that could be used to identify these packets.

ANON has provided mechanisms to protect itself against this type of attack. In particular, the infrastructure satisfies the "non-malleable, semantically secure packet encryption" property [5]. That is, *N* repeated submissions of the same packet will yield *N* different encrypted packet payloads on a link, as depicted in Figure 2. In addition, in the event when an adversary can change a packet by e.g., flipping a bit, the receiving forwarder can detect and discard the altered packet. The property, referred to as non-malleability, can prevent these packet tagging based attacks.

Below is an enhanced version of usage example B1 satisfying this non-malleable, semantically secure packet encryption property. It is a two-hop example, involving forwarders F1 and F2. Client C gets as a handle to the target server, (([S], s)f2, [F2])f1 and [F1], from an initialization server, where s is a symmetric key of S.

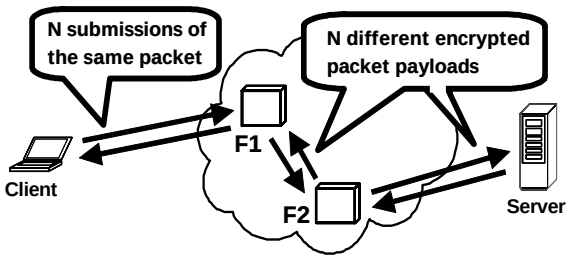

Figure 2. Semantically secure packet encryption

C -> F1: [C]{req, ck, (([S], s)f2, [F2])f1}[F1]
F1 -> F2: [F1]{req, ck, ([S], s)f2, [C]}[F2]
F2 -> S: [F2]{(req, ck, [C])r1, (r1)s, {[C], [F1]}r2, (r1, r2)f2}[S]
S -> F2: [S]{(rep, ck)r1, ([C], [F1])r2, (r1, r2)f2}[F2]
F2 -> F1: [F2]{rep, ck, [C]}[F1]
F1 -> C: [F1]{rep, ck}[C]

In ANON, the IPsec Encapsulating Security Payload (ESP) [9] is used for the encryption of packet transport between forwarders F1 and F2. Note that IPsec ESP has built-in support for non-malleable, semantically secure packet encryption. For each packet to be forwarded between F2 and S, F2 randomly selects symmetric packet keys r1 and r2 to implement non-malleable, semantically secure packet encryption. One can verify that for the tw0-hop example above, *N* submissions of the same packet by C will yield *N* different encrypted packet payloads on path segments from F2 to S and S to F2. For packet transport between C and F1, there is no need to apply packet encryption, since [F1] is known to the client and the public anyway.

It is straightforward to extend this two-hop enhanced scheme to hide [C], or both [C] and [S], and to allow additional hops.

5. ANON'S Countermeasures Against Threats

The ANON infrastructure provides countermeasures in defense against threats described in Section 2.

5.1. Defense Against Type 1 Threats

As illustrated in the enhanced two-hop usage example above, ANON uses multi-hop forwarding in defending against Type 1 threats where an adversary attempts to discover the address of a target server. Note that the adversary's objective is to identify the *exit forwarder*, the one that will have access to the decrypted IP address of the target server. Starting from an *entry forwarder*, the adversary would need to follow the forwarding path in order to discover the exit forwarder. (For the illustrative scenario in Figure 1, F1 is an entry forwarder and F2 is an exit forwarder.) The longer the forwarding chain, the harder must the adversary work. This is especially true if the forwarders are under different administrative authorities, since in this case the attacker will need to monitor links belonging to all the involved authorities in order to succeed.

To prevent packet content from revealing forwarding information, ANON uses non-malleable, semantically secure link encryption, as we have described in Section 4. Furthermore, to prevent packet headers from revealing forwarding information, ANON uses protocol camouflaging so that packets forwarded by forwarders have indistinguishable headers.

Note that it is also possible for adversaries to deduce a forwarding path by studying arrival and departure times of packets at forwarders without even having to examine the content of packets or their headers. For example, if the mixing of traffic is not enough, e.g., when the network is only lightly loaded, by tracing traffic with special temporal patterns, also known as *traffic tags*, an adversary can trace the forwarding path to reach the exit forwarder. Similar techniques have been reported in the field of intrusion detection to trace intruders who login through a chain of machines (called *stepping stones*) to hide where they come from [17].

To defend against these *traffic analysis* based attacks such as traffic tagging attacks, ANON uses link padding, which will inject artificial padding traffic into a link so that timing statistics of packets on the link will look similar with or without real traffic on the link. Link padding will thus increase the difficulty for an adversary to succeed in conducting traffic analysis. We will describe our link padding algorithms in Section 6.

In addition, ANON supports dynamic re-selection of forwarders and forwarding paths. A target server can dynamically register forwarding paths so that a new forwarding path can be used before the old one is cracked. That is, target servers may change the forwarders they use from time to time through the registration process.

Finally, ANON can address black box attacks [14], in which an adversary treats the whole forwarding infrastructure as a black box in the sense that it would observe packets going into and coming out from the infrastructure and correlate them to determine who is talking to whom. ANON prevents this type of attack by extending the infrastructure to within the physically protected local area networks of the communicating parties, through, e.g., VPNs as described in Section 8. This is equivalent to the

effect that participating users of the infrastructure incorporates one or more forwarders within their trusted regions.

5.2. Defense Against Type 2 Threats

To defend against type 2 threats where adversaries may conduct inband DOS attacks, ANON can reject packets with spoofed source IPs and rate limit remaining traffic on a per-source or per-destination basis. To implement rate limiting, each forwarder alternates between two phases, *equalization* and *relaxation*. The forwarder can use the "push-back" technology [6] to send control signals to its upstream forwarders to regulate the rate of traffic from those nodes to the current forwarder. In the equalization phase, if its current total rate is above certain high-water mark threshold, a forwarder will increase its push-back signal to upstream forwarders that have relatively larger traffic usage at present. In contrast, in the relaxation phase, if its current total rate is below certain low-water mark threshold, a forwarder will reduce its push-back signal to upstream forwarders that have relatively small traffic usage at present. This rate-limiting scheme can keep the utilization of the ANON network at a reasonably high level and its packet loss rate due to congestion at a reasonably low level, while blocking large users, including DoS attackers, from taking away bandwidths that other small users may need.

5.3. Defense Against Type 3 Threats

To defend against type 3 threats where adversaries may conduct outband DoS attacks, ANON can use multiple entry forwarders for the same target server. For example, multiple handles to the server can be made available to clients, so that unless an adversary is able to bring down all the entry forwarders the server is still reachable. Redundant forwarders can also be made available through the use of anycast addresses, or multiple lookup hash functions when an overlay network for lookup services such as Chord is used (see Section 7). DoS attacks at forwarders other than entry forwarders will not be possible unless their addresses become known to the attacker. This leaking should not happen if countermeasures against type 1 threats work.

We note that the mechanisms to defend against type 1 and type 3 threats are mutually reinforcing. It is easy to see how the capability of hiding the addresses of forwarders can help defend against DoS attacks on them. Conversely, as described in Section 2, an adversary may launch congestion-based attacks on links or DoS attacks on forwarders to deduce path information. For example, if an adversary can disable forwarders at will by DoS attacks, then by continually sending requests to a particular target server while DoS attacking forwarders one by one, he can deduce which forwarders are on the path. This is because once the adversary brings down any forwarder on the path, he will notice the absence of the replies from the target server. Hence, improving forwarders' capabilities of defending against type 3 threats will further improve their capabilities of defending against type 1 threats.

6. Link Padding

Link padding [14] concerns the problem of hiding the presence of real traffic from adversaries. As we have pointed out in Section 5.1, link padding is a necessary countermeasure to defend against traffic analysis based attacks when the load is light.

In link padding, the source node of a link will pad the link with artificial padding traffic, which we will also call *cover traffic* in this section, with the objective that the link will exhibit the same traffic statistics no matter whether or not real traffic is present. By using cover traffic on a superset of those links where real traffic traverses, an adversary will not be able to use traffic analysis to find out which subset of these links actually carries real traffic.

There are various types of link padding. The traditional one, which we call *flat link padding*, inserts padding traffic to a link so that the total link usage is constant [14]. This type of link padding is simple but can be bandwidth costly. In contrast, as proposed in this section, *on-demand link padding* inserts cover traffic only when real traffic is present, whereas *probabilistic link padding* inserts cover traffic in a stochastic manner to allow certain bursts in the application traffic to pass through without raising the average padding load.

6.1. A Model Problem

We describe a model problem that we will use in this section to study link padding. Although it is simple, this model problem captures essential issues of link padding.

In this model, we consider the scenario where a network path between a client C and a target server S needs to be hidden, such as a forwarding path in the ANON infrastructure. The basic protection strategy is that whenever C and S communicate, cover traffic will be generated over multiple network paths, so the *total traffic* on all these paths looks statistically the same, where the total traffic is defined to be the sum of both cover and real traffic.

We assume that a balanced tree network, as depicted in Figure 3, is used for hiding a path between C and S, with C being at the root and S being at a leaf. Given a path to be hidden, we further assume that the tree is fixed for a period of time in order to guard against intersection attacks [14].

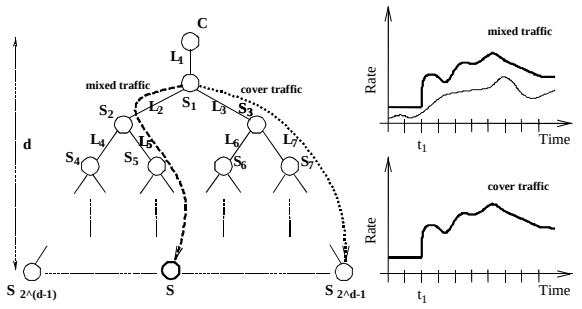

Figure 3. The cover-traffic tree for our model problem of Section 6.1. C is the client and S is the target server. The diagram on the left illustrates that the total traffic on the path from C and S is the sum of real and cover traffic, while that on any other link is just the cover traffic. The two diagrams on the right indicate that at any given time the total traffic over any link looks the same with or without real traffic.

6.2. On-Demand Link Padding

The basic concept of on-demand link padding is to add padding traffic based on the bandwidth usage observed from real traffic. This allows padding to be applied only when real traffic is present. Moreover, this allows the amount of the padding to be limited at a level just sufficient to cover real traffic.

6.2.1. On-Demand Link Padding with Delay

The first of the two implementations of on-demand link padding we will describe is *on-demand link padding with delay.*

Consider first real traffic from C to S in our model problem of Section 6.1. We break the traffic sent by C into consecutive data segments, each lasting for one RTT, the round-trip time for the path between C and S in the tree network of Figure 3. When a segment departs from C, it will stay in a delay buffer at the output port of C for 2RTT. During the first RTT, an averaging bandwidth usage of the segment is measured. During the second RTT, a signaling message is sent by C to all the tree nodes requesting them to generate padding traffic so that the total traffic sent over each link will be sent at a rate equal to the measured bandwidth. At the end of the second RTT, if C has received sufficiently many acknowledgements from the tree nodes, it sends out the segment at the rate of the measured bandwidth over a period of RTT.

For real traffic from S to C, an approach similar to the one described above is used. There is a delay buffer in the output port of S. When the bandwidth of the current segment sent by S has been measured, S will send a notification about the measured bandwidth to the root C so that C can signal all tree nodes to generate proper padding traffic. The delay buffer in this case needs to be extended to reflect the additional delay for S to send the bandwidth notification to C.

As shown in Figure 4, we use a pipeline architecture to process three consecutive segments simultaneously at three pipeline stages. The three stages, each being RTT long, are for bandwidth measurement, signaling and data sending, respectively.

Since the bandwidth usage of a segment is measured before the target rate of link padding is set, on-demand link padding with delay can target the total link bandwidth usage to be exactly at the measured bandwidth. This means that, in on-demand link padding with delay, there is no need to waste network bandwidth on the path for cover traffic. However, this advantage is achieved at the expense of delaying traffic for 2RTT.

6.2.2. On-Demand Link Padding with Headroom

Our second implementation of on-demand link padding is *on-demand link padding with headroom.* This implemen-

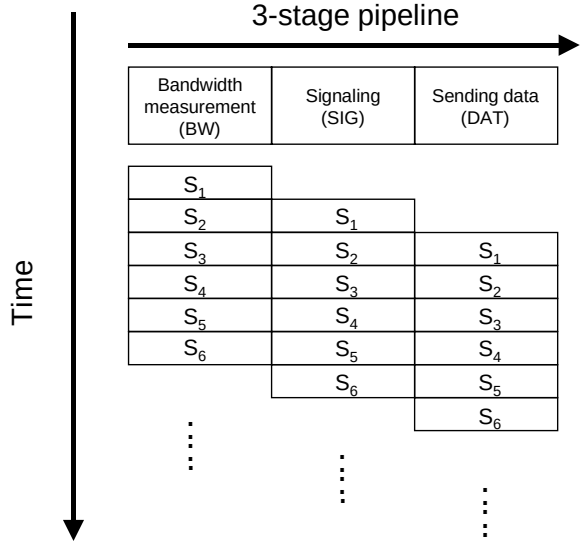

Figure 4. The pipeline architecture for data segment processing in on-demand link padding with delay. When the first segment S_{i-1} of three consecutive segments is being sent over the network links, the signal that carries the bandwidth requirement for the second segment S_i is being propagated from traffic source to all the nodes in the network. Meanwhile, the bandwidth requirement of the third segment S_{i+1} is being measured while the segment is being transmitting into a delay buffer.

tation is of interest because it does not require extra delay. We describe the idea by considering real traffic from C to S in Figure 3. By using a traffic shaper, as shown in Figure 5, we can ensure that any bandwidth increase in the current segment is limited to the allowed range. This means that the current segment can be sent out without delay.

However, this zero-delay link padding method suffers from the drawback that large headroom will be needed to accommodate real traffic that has large bandwidth usage fluctuations between consecutive segments. Large headroom results in large use of network bandwidth in link padding.

To address this link efficiency issue, the input queue in the traffic shaper of Figure 5 will be used to absorb bandwidth usage fluctuations between consecutive segments, thereby reducing the need of large headroom. The queueing required in the traffic shaper, however, could introduce delay. Thus when using on-demand link padding with headroom, one can trade off network utilization caused by the headroom for lowered queueing delay caused by the traffic shaper.

6.2.3. Illustrations

We illustrate the two implementations of on-demand link padding in Figure 6. In Figure 6(a) we show that the 2RTT delay is required by on-demand link padding with delay. For example, the first segment S_1 is sent using the bandwidth measured during S_1 BW and the signal delivered during S_1 SIG. Figure 6(b) shows that using on-demand link padding with headroom, we can send segments without delay. For example, the third segment S_3 is sent using the bandwidth measured during S_1 BW and the signal delivered during S_1 SIG. Since the method imposes a limit that the bandwidth actually used by S_3 will

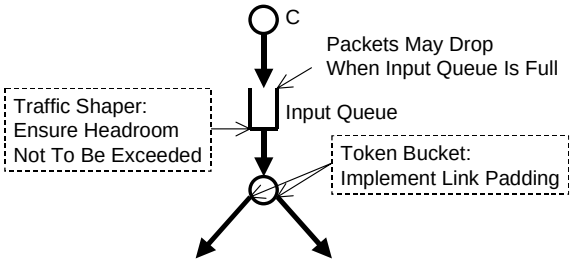

Figure 5. Traffic shaper installed at the client C when using on-demand link padding with headroom. The traffic shaper will shape the outgoing traffic from C and cap rate increases within the allowed range absorbable by the headroom size. If the traffic rate of C continues to increase, the input queue will build up, and eventually packets will be dropped after the queue fills up.

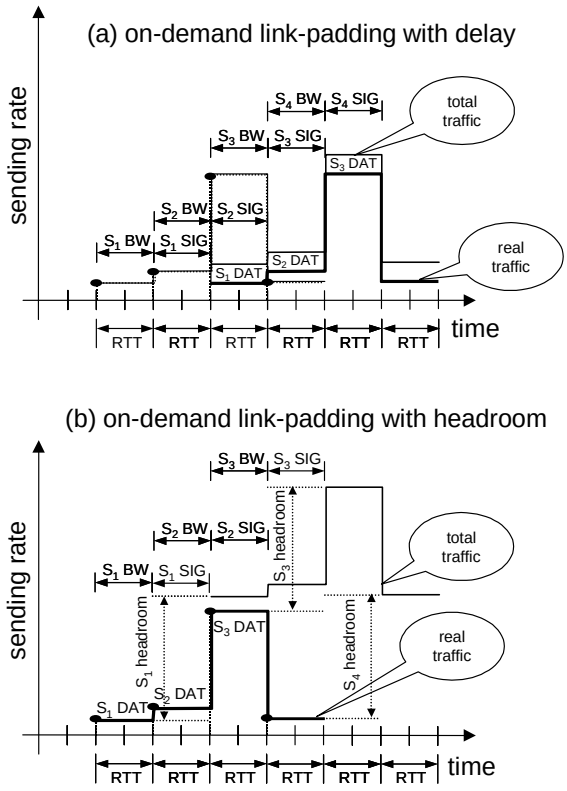

Figure 6. Sending rate as a function of time in on-demand link padding. Both (a) and (b) show the pipeline depicted in Figure 4. The upper diagram (a) shows the 2RTT delay. For example, S_1 is sent using the bandwidth measured during S_1 BW and the signal delivered during S_1 SIG. The lower diagram (b) shows that with sufficient headroom we can send segments without delay. For example, S_3 is sent using the bandwidth measured during S_1 BW and the signal delivered during S_1 SIG. Since the sum of the measured bandwidth and headroom is larger then the bandwidth required by S_3, we can send it without delay.

be less than the sum of the measured bandwidth and headroom, we can ensure that the traffic of S_3 will not stick out above the total traffic seen on other paths.

6.3. Probabilistic Link Padding

The basic concept of probabilistic link padding is to generate cover traffic in a probabilistic rather than deterministic way. The total traffic on all the links in the system including real and padding traffic will follow a given distribution communicated to all participating nodes beforehand. This allows variable-rate real traffic to be covered by a similar padding traffic, in order to decrease the required

padding traffic. As long as the total traffic on all links has the same statistical characteristics, an adversary will not be able to distinguish between a real forwarding path and a dummy path by monitoring links. This scheme would allow an increased level of bursts of real traffic without compromising path anonymity, as well as the use of real traffic from other applications that satisfies certain desired traffic distributions as cover traffic.

In probabilistic link padding, we discretize time into epochs, within each of which the traffic rate remains the same. The traffic rate of each epoch is drawn independently from a given distribution, such as a heavy-tail distribution. We call the rates in a sequence of consecutive epochs a *rate sequence* hereafter.

Since the bandwidth need of the real traffic may not coincide with the current rate in the rate sequence, congestion could happen. In this case the real traffic queue at the token bucket, such as the one in Figure 7, will grow beyond certain threshold. Probabilistic link padding deals with this problem by allowing some of the bursts to pass through to alleviate congestion, while minimizing disturbance to traffic statistics.

In order to make long-term statistics indistinguishable over all links with or without real traffic, when congestion is detected, we reorder rate sequences instead of generating new ones. Specifically, when the queue occupancy at the real-traffic token bucket, as depicted in Figure 7, exceeds certain threshold, we search in a pre-generated rate sequence for a rate larger than the current rate and schedule a swap after a *randomized swap delay.* This randomized delay prevents an adversary from detecting predictable responses resulting from load with high regularity such as regular traffic pulses. For similar reasons we impose a *randomized recovery time*, which is the number of epochs to wait before the next swapping can occur.

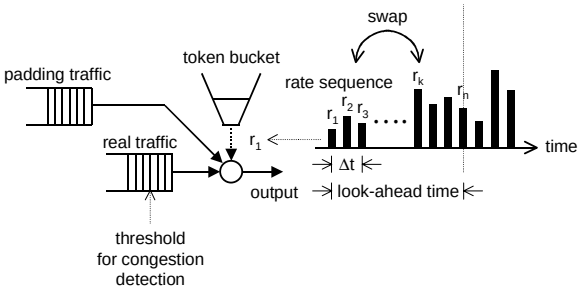

Figure 7. Probabilistic link padding. Time is discretized into epochs, within each of which the input rate of the token bucket remains constant. The amount of look-ahead time governs how far a forwarder searches forward for a large rate to satisfy current need, when the real traffic queue exceeds certain threshold due to congestion. To avoid temporal correlation along the forwarding path, the swapped rate is not immediately used; rather, a random delay is introduced, preventing such a mechanism from being triggered deterministically and thus leaking information about the forwarding path. Note that Δt denotes a time interval within which the randomized delay of a rate swapping may take

6.4. Discussions

We have used *ns-2* simulator [1] to study the two link padding algorithms. The simulation results show that communication paths can be hidden effectively and efficiently. However, due to space limitation of this paper, we will not report the detailed simulation results and figures here. Instead, we focus in this section on more qualitative aspects of our link padding algorithms.

Note that in a real-world situation, multiple clients can request for cover traffic on multiple trees at the same time. If a node receives multiple requests for cover traffic, the node can try to meet the sum of all the requests or a subset of them. As depicted in Figure 4, the node sends out replies in the signaling stage to notify each of the requesting clients whether or not their individual requests can be met. Replies or acknowledgements destined to a same client may be combined or piggybacked in data packets to save bandwidth when necessary.

We envision a hybrid scheme where both on-demand and probabilistic link padding are used at the same time. From our simulation results, we have found that for on-demand link padding to be effective, we need a reasonably-sized queue to absorb traffic bursts. Use of probabilistic link padding on top of on-demand padding would allow bursts to be cleared at a faster rate by swapping a larger rate in the future into near future from a rate sequence. Therefore probabilistic link padding may help on-demand link padding to reduce queue occupancy while maintaining the same packet drop rate. On the other hand, on-demand link padding will change the traffic rate dynamically so that an adversary can not observe for a long enough period in order to obtain the required statistical significance. By changing the rate frequently enough, the method will limit the adversary's observation period and hence reduce the effectiveness of statistical correlation based attacks, which use various statistical tests.

Finally, we note that if signaling packets are lost during an epoch, then those participating nodes that do not receive signaling will not be able to provide cover traffic. This means that these nodes that fail to provide cover traffic can not be the real clients (or the real target servers). As an

adversary observes for more and more epochs, eventually it might be able to find out the true IP address of the real client (or the real target server) by taking intersection. We note that this problem can be mitigated by letting participating nodes lie about whether they have received signaling packets. For example, false information can be given to the adversary if, with certain probability, the client (or the target server) will not send out real traffic for that epoch, thus pretending to be a cover traffic providing node that fails to receive signaling packets in that epoch. The adversary will then need to increase his statistics gathering in order to figure out the real client or target server.

7. Use of Fault-tolerant Transport in ANON

The ANON infrastructure uses a fault-tolerant overlay network to transport packets among its forwarders. This enhances ANON's resilience against outband DoS attacks aiming at forwarders and network paths connecting them, as well as possible infrastructure failures. We illustrate the idea by considering Chord [16], a fault-tolerant overlay network originally designed for peer-to-peer object sharing purposes.

We first give a brief overview of Chord. Each object, be it a participating node or a file to be stored, has a *Chord ID*, obtained by taking the hash value of whatever identifies the object. We call the range of the hash function the *ID space*, in which IDs are ordered on an *ID circle*, with the largest ID connected to the smallest. Every participating node in Chord are identified by its Chord ID, obtained by hashing the IP address of the node. Each node has two neighbors in the ID space; the one with a smaller ID is referred to as the *predecessor*, while the one with a larger ID as the *successor*. Every node is responsible for those objects whose IDs are between its own ID and that of its predecessor. Given an object's Chord ID, it is straightforward to route the requests for that object in Chord; each node simply forwards the requests to the node with the closest Chord ID that it knows. Further routing optimization is achieved by connecting the nodes in a way that is similar to the topology of a hypercube: each node has as its neighbors those nodes whose IDs are closest to its own ID plus all powers of two modulo the size of the ID circle.

It is straightforward to extend ANON to use a Chord network to transport packets among forwarders. This scheme, which we call *ANON over Chord*, uses nestedly encrypted Chord IDs instead of IP addresses, as depicted in Figures 8. A forwarder is reachable by its responsible Chord node, determined by the hash value of the forwarder's IP address. For example, node N53 is responsible for F2 since the hash value of F2's IP address, denoted by h([F2]), is 44, and node N53 is in charge of 44.

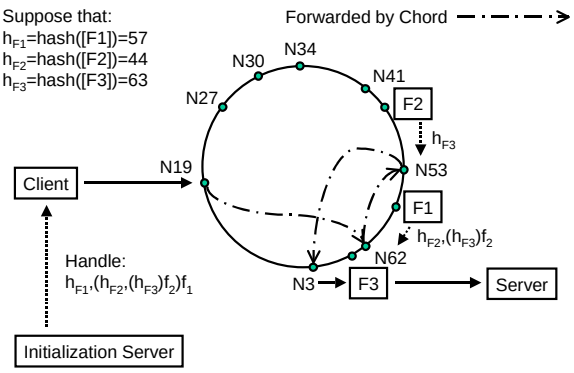

Figure 8. ANON with Chord as the underlying transport network, i.e., ANON over Chord. In contrast, nested encryption is now applied to Chord IDs instead of IP addresses. The IP address of a forwarder is used by a hash function to derive its Chord ID; Chord nodes will forward packets addressed to a forwarder as if it were a request for the object with the same Chord ID.

This is similar to how an object stored in Chord is found by using the hash value of the object.

With Chord being used to forward packets, outband DoS attacks related to type 3 threats of Section 2 now refer to attacks on the Chord network or forwarders themselves. Note that a Chord network is fault-tolerant in the sense that each node has redundant information about their neighbors' neighbors, so that when a neighbor is down, a node can reach the neighbor's neighbors to repair connectivity. Thus with this fault-tolerant capability, ANON can mitigate these outband DoS attacks of the Chord network, using the recovery process illustrated in Figure 9.

To defend against outband attacks on forwarders themselves, ANON can use redundant forwarders. That is, a forwarder can have multiple copies deployed at various locations, with each forwarder copy associated with a separate hash function h_i for some i. We use F to denote the group of a forwarder and all of its copies. For each forwarder copy in the group, the responsible Chord node is determined by $h_i(F)$, where h_i is the hash function associated with the forwarder copy, and F is the group ID. When a packet is unable to reach a given forwarder copy, a randomly selected h_i will be applied to F to determine a new Chord node to reach that could be responsible for another forward copy in the group.

Note that when Chord is used, link encryption and link padding will be applied to links of the Chord network. These provide some additional advantages. First, packets being forwarded no longer reveal the IP address of the next-hop forwarder. Second, bandwidth allocation for link

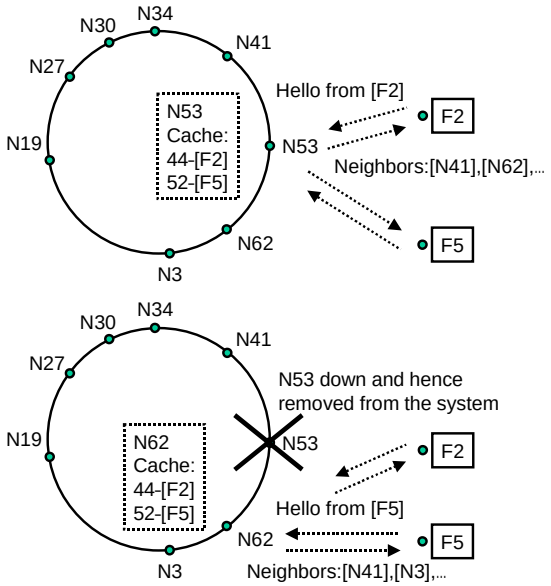

Figure 9. The recovery process: if a Chord node N53 is brought down by outband DoS attacks or some other reasons, N62 will take over the responsibilities of N53. Forwarders F2 and F5 will reassociate with N62 once they discover that N53 is no longer available. When a forwarder is associated with a Chord node, the forwarder learns the node's neighbors to support the fault-tolerant mechanism.

padding can be made efficient for a low-connectivity overlap network such as the Chord ring. That is, link padding on a Chord link can be provided at a rate just above the expected maximum aggregate bandwidth an adversary's attack may command, while being able to defend against congestion-based attacks.

The ANON forwarder's interface to a Chord node is general. It can work with other fault-tolerant low-connectivity overlay networks to achieve similar advantages described above.

8. Testbed Implementation

We have implemented a laboratory testbed for ANON at Harvard, which supports basic functions including protocol camouflaging, link encryption, link padding, rate limiting, and a simple fault-tolerant transport network. The testbed, depicted in Figure 10, uses nodes implemented on top of the FreeBSD operating system. The FreeBSD divert socket is used to implement various header processing operations at the user-level. For the symmetric key algorithm, the testbed uses the AES reference implementation from NIST.

The testbed includes NAT gateways, GWc for clients and GWs for servers. These gateways allow existing

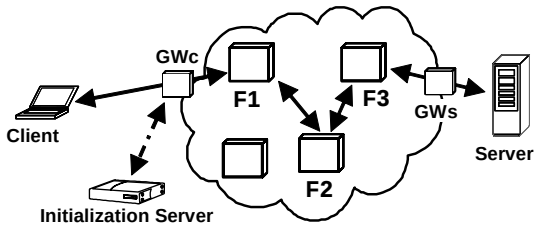

Figure 10. Experimental testbed at Harvard. GWc and GWs are NAT gateways that allow existing clients and servers to use ANON without modifications.

clients and servers to use the testbed without modifications. Clients and servers may connect to their respective gateways directly or via a VPN connection.

The current testbed implementation can achieve a throughput of 5 Mbps. This performance, adequate for signaling protocols and low-bandwidth data applications, is made possible mainly because we have managed to avoid using public key encryption and decryption in packet forwarding, as illustrated in usage examples earlier.

9. Related Work

As we have described in this paper, ANON is for the purpose of hiding addresses of communicating parties. The system achieves this objective by providing an overlay networking infrastructure where forwarders on a forwarding path can decode nestedly encrypted addresses for a target server. We compare ANON to other systems which have related objectives and approaches.

Chaum proposed to construct an untraceable email system MIX [2], in which an intermediate, trusted agent, called a MIX node, batches and relays in a randomized order collections of messages whose intended recipients are encrypted using the public key of the MIX node. The idea is general, and has served as the basis for many later anonymizing communication systems. MIX has a simple goal that even if an adversary is capable of observing all network traffic, he cannot link a particular message in its unencrypted form with the sender, who is hidden in the set of all participating senders, called an *anonymity set*. This includes the case where the recipient is the adversary trying to figure out the identity of the sender. An adversary can only obtain information that a participant may be communicating with one or more parties by noticing that participant is in the anonymity set, but he can not be sure with which party or parties the participant is communicating.

MIX is designed to handle single-message exchanges; it does not offer any mechanism that can defend against long-term analysis, which can be especially hard to deal with when an adversary has access to contextual information

about the anonymous communication sessions such as number of messages exchanged, active and inactive times of communication, etc. With the aids of such information, an adversary can launch what is generally termed as an *intersection attack* [14]. A typical scenario would be: if an adversary knows the times when a sender is communicating with a particular recipient, then he may be able to figure out who the sender is by taking the intersection of the sets of active senders during these times.

We could, in theory, use MIX as a solution to the problem of hiding addresses that ANON intends to solve, by using the MIX node as the anonymizing infrastructure. But such a solution would not be effective for several reasons. First, the solution will not be able to defend against long-term analysis as mentioned above. Second, the solution will not support real-time packet forwarding, since the MIX node may delay a message for an indefinite amount of time before it receives enough messages to form a batch. Third, we will face a dilemma in determining the size of the anonymity set required for the mix operation. On one hand, naturally we would like to have as large as possible an anonymity set better anonymity. However, once the size of the anonymity set is larger than the capacity of the MIX node, we open up opportunities for intersection attacks. On the other hand, if we keep the anonymity set reasonably sized so as not to exceed the capacity of the MIX node, it will take relatively a small amount of resources for the adversary to launch DoS attacks on nodes in the anonymity set to further shrink the set. In contrast, ANON attempts to hide a target server in a large candidate set of possible servers unknown to users and adversaries. By using a sufficiently large candidate set, ANON can ensure that indiscriminating DoS attacks on all these candidate servers will be infeasible. By using link encryption and link padding, as well as extending the infrastructure to the trusted regions such as physically protected local area networks near clients and target servers, ANON can defeat intersection attacks.

Onion routing [15] is an extension to MIX that supports real-time packet forwarding. Similar to ANON, onion routing uses nestedly encrypted addresses, called *onions,* to reach target servers. The corresponding sequences of forwarders, called *onion routers,* can decrypt these encrypted addresses. In order to avoid expensive public key operations on a per-packet basis, onion routers store session keys for a connection. The sender of a connection first chooses a sequence of onion routers to traverse and constructs an onion that consists of the addresses of the onion routers with nested encryption applied. (The sender's choice of a route is similar to that in Mixmaster [3].) Then a connection is set up for subsequent messages, and each message is forwarded along the circuit formed by the onion routers specified in the onion. In contrast, forwarders in ANON store no connection states. Because of this stateless property of ANON, redundant forwarders can readily take over the jobs of failed or overloaded forwarders.

Moreover, unlike ANON, onion routing does not intend to hide the IP address of a target server; in fact, the sender must know this address in order to construct an onion for the target server. In ANON, the sender will be given a server handle to the target server, instead of building its own onion.

When facing an adversary capable of monitoring links, it is pointed out in [15] that link padding among onion routers is necessary; however, the details as how to pad the links are left unspecified. Note that link padding needs to be deployed on a superset of those links which carry anonymous traffic, but it is unclear in onion routing what this superset would be.

Many extensions to and improvements over onion routing have been reported recently. Tarzan [7] is an IP-layer anonymizing network with similar goals as onion routing, which are different from ANON's primary goal of providing address anonymity for target servers. Tarzan uses a peer-to-peer networking infrastructure as opposed to ANON's use of a network resident set of forwarders managed by trusted third parties. Tarzan does not address inband or outband DoS attacks on forwarding nodes as ANON does. In fact, inband DoS attacks could represent a serious threat in Tarzan. This is because, due to its peer-to-peer nature, it is fairly easy for an adversary to have his nodes join the network and then launch inband DoS attacks from the nodes, as described in Section 2.

SOS [10] is a Chord-based [16] overlay network that aims to provide survivability under outband DoS attacks. SOS uses the fault-tolerant property of a Chord network as the main mechanism to defend against such attacks. To protect a target server from being DoS attacked, the server first secretly recruits several servlets, whose addresses are to be kept secret. Only these chosen servlets can send messages to the server, which can be done by setting up appropriate filters on the edge routers located at egress points of the core network to the server. The addresses of the servlets are secretly communicated to a number of beacons determined by applying different hash functions to the IP address of the server. Only authenticated clients can send messages to the server. This implies that unlike ANON, SOS does not address inband DoS attacks, nor inband packet or traffic tagging attacks. Authenticated clients can then send messages to servers by first sending the messages to beacons via Chord, and the beacons will relay the messages to the server through servlets. If any of the nodes, including beacons and servlets, along a path is

brought down by an adversary, Chord will try to repair the network by connecting the neighbors of the failed node. Therefore, an adversary would now need increased resources to launch long lasting attacks in order to stop the communication between clients and servers, as short-term damage can be quickly repaired by Chord.

SOS makes a few assumptions that are different from ANON's. First, as mentioned above, SOS does not provide protection against inband attacks, in which an adversary launches attacks from within the system, e.g., by flooding the network with requests from legitimate clients. In contrast, ANON is designed with inband attacks in mind, so ANON is equipped with rate limiting to defend against inband DoS attacks, as well as link encryption and link padding to defend against packet and traffic tagging attacks, respectively. Second, SOS does not try to preserve the anonymity of the communicating parties, though it is preserved under a weaker adversary model that an adversary can not monitor links. ANON assumes a stronger and more realistic adversary model that allows adversaries to monitor links, as justified in Section 2.

10. Summary and Concluding Remarks

We have described an anonymizing infrastructure at the IP layer. The infrastructure is specially designed for low- to medium-bandwidth applications such as authentication, authorization and instant messaging. We have illustrated usage examples of hiding addresses of servers and clients. By employing a suite of countermeasures, such as non-malleable, semantically secure packet encryption, link padding and rate limiting, we have shown that even if an adversary is capable of monitoring links, it would be difficult for him to compromise the anonymity provided by the infrastructure. Likely, the only way an adversary can succeed is to take on the direct attack of compromising forwarders one by one. By using trusted third parties to manage the forwarding infrastructure, in a way similar to how current backbone routers are managed, we can make sure that compromising forwarders would be very difficult.

To lower the bandwidth cost of link padding, we have designed two novel algorithms that can generate padding traffic in an economical manner without sacrificing anonymity. We have also sketched an approach of using fault-tolerant overlay networks to enhance the resilience of the anonymizing infrastructure against attacks and failures. To demonstrate the implementation feasibility, we have developed a laboratory testbed.

An important next step that we plan to carry out is application trials of this anonymizing infrastructure. When sufficient experiences have been learned from these application experiments, we will consider the possibility of incorporating some of the anonymizing features into routers.

Acknowledgment

This work was supported in part by DARPA through AFRL/IFKD under contract F33615-01-C-1983.

References

[1] Breslau, L., D. Estrin, K. Fall, S. Floyd, J. Heidemann, A. Helmy, P. Huang, S. McCanne, K. Varadhan, Y. Xu, and H. Yu, "Advances in Network Simulation," *IEEE Computer*, vol. 33 no. 5, May 2000, pp. 59–67.

[2] Chaum, D., "Untraceable Electronic Mail, Return Addresses, and Digital Pseudonyms," *CACM*, February 1981, pp. 84– 88.

[3] Cottrell, L., "Frequently Asked Questions about Mixmaster Remailer," http://www.obscura.com/~loki/remailer/mixmaster-faq.html, 1996.

[4] Dai, W., "A Practical Attack Against ZKS Freedom," USENET Newsgroup Archive, http://www.packetstormsecurity.org/9905-exploits/zks.freedom.flaws.txt

[5] Dolev, O., C. Dwork, and M. Naor. "Non-malleable Cryptography," in *Proceedings of the 23rd Symposium on Theory of Computing*, ACM STOC, 1991.

[6] Ioannidis, J. and S. M. Bellovin, "Pushback: Router-Based Defense against DDoS Attacks," *NDSS*, February 2002.

[7] Freedman, M. J., E. Sit, J. Cates, and R. Morris, "Introducing Tarzan, A Peer-to-Peer Anonymizing Network Layer," in *Proceedings of 1st Intl. Workshop on Peer-to-Peer Systems*, Cambridge, MA, March 2002.

[8] Katabi, D. and J. Wroclawski, "A Framework for Global IP-Anycast (GIA)," in *Proceedings of ACM SIGCOMM 2000*, Stockholm, Sweden, 2000.

[9] Kent, S. and R. Atkinson, "IP Encapsulating Security Payload (ESP)," RFC 2406, November 1998.

[10] Keromytis, A., V. Misra, and D. Rubenstein, "SOS: Secure Overlay Services," in *Proceedings of ACM SIGCOMM 2002*, Pittsburgh, PA, 2002.

[11] Kung, H.T., S. Bradner, and K.-S. Tan, "An IP-layer Anonymizing Infrastructure," *MILCOM 2002*, Anaheim, CA, October 2002.

[12] Milliken, W., C. Partridge, and T. Mendez, "Host Anycasting Service," RFC 1546, November 1993.

[13] NymIP Effort, the, http://www.nymip.org/

[14] Raymond, J.-F., "Traffic Analysis: Protocols, Attacks, Design Issues and Open Problems," in *Proceedings of International Workshop on Design Issues in Anonymity and Unobservability*, LNCS, vol. 2009, Springer-Verlag, 2001, pp. 10–29.

[15] Reed, M., P. Syverson, and D. Goldschlag, "Anonymous Connections and Onion Routing," *IEEE Journal on Selected Areas in Communications*, vol. 16 no. 4, May 1998, pp. 482–494.

[16] Stoica, I., R. Morris, D. Karger, M. F. Kaashoek, and H. Balakrishnan, "Chord: A Scalable Peer-to-peer Lookup Service for Internet Applications," in *Proceedings of ACM SIGCOMM 2001*, San Diego, CA, 2001.

[17] Wang, X., D. S. Reeves, and S. F. Wu, "Inter-Packet Delay Based Correlation for Tracing Encrypted Connections Through Stepping Stones," in *Proceedings of ESORICS 2002*, Zurich, Switzerland, 2002.