Abridged version appears in: *The Journal of Computer Security, Vol. 21, No. 2, pp 233-281, March 2013*

Spread Identity : A New Dynamic Address Remapping Mechanism for Anonymity and DDoS Defense

January 20, 2011, Revised April 6, 2012 and October 21, 2012

Dhananjay Phatak {phatak@umbc.edu}, Alan T. Sherman,[1]
Nikhil Joshi, Bhushan Sonawane, Vivek Relan, and Amol Dawalbhakta

Cyber Defense Lab, Department of CSEE
University of Maryland, Baltimore County (UMBC)
1000 Hilltop Circle, Baltimore, MD 21250 USA

Abstract. We present and experimentally evaluate *Spread Identity (SI)*—a new dynamic network address remapping mechanism that provides anonymity and DDoS defense capabilities for Internet communications. For each session between a source and destination host, the trusted source gateway dynamically and randomly assigns an IP address for the source host from the pool of all routable IP addresses allocated to the source organization (by the IANA). Similarly, in response to a name resolution query from the source gateway, the trusted authoritative DNS server (i.e., the ADNS) for the destination organization dynamically assigns an IP address for the destination host from the pool of all routable IP addresses allocated to the destination organization. These assignments depend upon the state of the server (including load, residual capacity, time of day) and policy. Different hosts can share the same IP address when communicating with distinct peers. Each gateway creates a NAT entry, valid for the communication session, based on the dynamic assignment by its organization.

An eavesdropper listening to packets flowing through the Internet between the source and destination gateways learns only the source and destination domains; the eavesdropper cannot see the actual complete IP addresses of the source and destination hosts. In addition, SI enhances DDoS defense capabilities by enabling packet filtering based on *destination* addresses. With multiple IP addresses for the same destination, filtering based on destination addresses can block attackers without necessarily blocking legitimate users. Deploying SI requires changes to organizational gateways and, possibly, to the edge-routers that interface with organizational gateways; but network mechanisms farther upstream, including the core routers in the Internet remain unchanged. Likewise, the installed base of operating systems running individual hosts in the internal network, together with the end-user application suites they support, remain untouched; thereby illustrating that the SI mechanisms are backward compatible, incrementally deployable, and robustly scalable. A naïve implementation of SI can increase the DNS traffic; however, when SI is implemented at both the source and the destination ends, it is possible for SI to *reduce* DNS traffic.

Ns-2 simulations and experiments on the DeterLab test bed corroborate the main hypotheses and demonstrate advantages of the SI paradigm. Ns-2 simulations demonstrate that file transfer success rates for our SI DDoS protection mechanism are similar to those of filter-based and capability-based approaches, with lower file transfer times than those for filter-based approaches. DeterLab trials demonstrate that SI consumes similar resources (connection establishment time, network address translation table size, packet forwarding rate, and memory) to those of a typical single NAT system; but with higher name resolution times.

---

[1] Sherman was supported in part by the Department of Defense under IASP Grants H98230-09-1-0404, H98230-10-1-0359, and H98230-11-1-0473.

Keywords. Address-hopping, address pooling, anonymity, applied cryptography, *Distributed Denial of Service (DDoS)* attacks, Domain Name Server (DNS), Internet Protocol (IP), Network Address Translation (NAT), network security, spread identity, statistical address multiplexing.

1 Introduction

Static mappings of identities create egregious vulnerabilities that undermine privacy and facilitate identity theft. For example, static passwords, static credit card numbers, static RFID tags, and static IP addresses simplify the tasks of a malicious adversary who wishes to track individuals or misuse their credentials. By contrast, dynamic passwords and credit card numbers [43]], dynamic RFID tags (*e.g.,* as realized by hash chains), and dynamic *Internet Protocol (IP)* addresses can substantially complicate the adversary's job. In this paper, we propose and analyze a new and powerful approach for dynamically "spreading the identities" (remapping the IP addresses) of source and destination hosts among a pool of identities (respectively, within a source and a destination organization) for Internet communications. This approach, which we call *Spread Identity (SI),* enhances network anonymity and enables new effective responses to many *Distributed Denial of Service (DDoS)* attacks. SI also improves trace-back capability (*i.e.,* ability to trace back toward the origin of a packet), by exploiting the sender's need to receive the dynamic IP address of the destination.

In common practice today, when a client establishes a session with a server over the Internet, the connection is established using a static mapping of IP addresses and host names. Consequently, an eavesdropper can discover which client is communicating with which server, violating the privacy of the communicants. Furthermore, a malicious adversary can easily concentrate a DDoS attack against a specific server by directing her evil accomplices to target a particular static IP address. We show how SI mechanisms applied to Internet communications (henceforth referred to simply as SI) provide a solution to these two problems (along with solutions to several other problems)

We describe how to implement SI at an organizational level (*e.g*., for a domain such as www.umbc.edu). In our model, each organization has a gateway which performs dynamic address translations valid for one session (this assumption can be relaxed and the granularity of dynamic address mapping can be set to any desired level; from host, session, … all the way to the other extreme where the addresses are changed in every packet in the same flow, for further details, please see Section 4.5) . For example, the sender's organizational gateway spreads the identity of each of its clients among its pool of routable IP addresses assigned to it by its *Internet Service Provider (ISP)*. Similarly, the destination's organizational gateway spreads the identity of each of its hosts among its pool of routable IP addresses. At the source gateway, the translation is performed by gateway routers using *Network Address Translation (NATing)*. At the destination, the translation is performed by the destination gateway using the NAT entries that are created at the beginning of every session, in cooperation with the authoritative DNS (ADNS) server for the destination.

In 2005, Phatak [47]] proposed the concept of SI . Soon thereafter, the initial concept was substantially extended and improved by pooling all routable IP address allocated to each organization at that organization's gateway and by reusing IP addresses via controlled NATing. These improvements mitigate restrictions in the original concept, which were caused by a limited set of (externally) routable IP addresses available under IPv4. This version with pooled addresses was first prototyped and tested by Dawalbhakta [18]]. This paper further extends the original 2005 work by

(i) illustrating that SI can substantially enhance the "anonymity"; simultaneously with all other benefits without compromising or adversely affecting any of the other unique capabilities it enables (ex: filtering flows based on the destination address; multi-level muti-pronged, and robust DDoS defenses as well as offenses; and substantially enhanced traceback), and by

(ii) presenting the results from additional implementations and experiments using real and simulated networks, including DeterLab testbed results by Joshi [31] [] and simulation results by Sonawane [55]].

As a defense against DDoS attacks, SI offers advantages over the three main existing defenses: capabilities-based approaches [3 ,6, 44, 66] [, filters-based methods [39] , and overlay strategies [52, 56, 57, 58] . In comparison with capabilities- and filters-based approaches, SI requires fewer changes to current infrastructure: SI requires changes only to organizational gateways and possibly to the edge routers (which are the last-hop routers in the ISP's infrastructure) that connect to the organizational gateways. Nothing farther upstream beyond the last hop needs to change. Thus, routers and other infrastructure in the core of the Internet are left untouched. Furthermore, the installed base of operating systems running end hosts in the internal organizational networks, together with the end-user application suites they support, are unchanged. Thus, all modifications required by SI are restricted to the routers on either side of the last-hop link. Overlay strategies also require minimal infrastructure changes but are potentially vulnerable to compromise or bypassing of the secret servlets (*e.g.*, an attacker might learn the true IP address of the destination and send traffic directly there).

We assume ISPs are willing to cooperate, either installing filters on their last-hop routers that connect directly with organizational gateways, or allowing use of proxy routers in-series-with and immediately after their last-hop router to install filters to protect the last-hop link from the ISP to the organization (which typically has the smallest bandwidth) against bandwidth-clogging DDoS attacks.

Using the ns-2 network simulator [42] and DeterLab test bed [19] , we demonstrate the feasibility, performance, and effectiveness of SI. With ns-2, we assess SI as a tool for facilitating destination filtering in response to DDoS attacks, and we measure some of the important parameters that are indicative of performance of a gateway. We measured file transfer success ratios and file transfer times for different numbers of DDoS attackers and for various bottleneck link capacities. Our results show that the file transfer success ratios for our SI DDoS protection mechanism are similar to those of filter and capability-based approaches (which are the other competing approaches), with lower file transfer times than those required for filter-based approaches.

With DeterLab, we compare a new implementation of SI with a baseline single-NAT scheme extensively deployed today. SI incurred modest overhead in the initial connection setup-stage, but thereafter overall delays were similar (and the survival rate for SI under DDoS attacks was higher).

It is possible to implement SI *only for inbound traffic flows arriving at the destination gateway (and nowhere else)*. Such a partial implementation still provides substantial DDoS defense capabilities and improved trace-back capability, but without sender anonymity. These properties provide an incentive for any organization to implement SI even if other organizations do not.

Contributions of this paper include: (1) System architecture and a connection-establishment protocol for SI for Internet communications. (2) Explanation of DDoS defense capabilities and network-level anonymity effected by SI. (3) Experimental demonstration and evaluation of SI's DDoS defense capabilities and performance.

The rest of the paper is organized as follows. Section 2 overviews the SI concept. Section 3 explains our SI architecture for Internet communications, including our assumptions and protocols. Section 4 discusses design and performance issues. Section 5 reviews related work. Section 6 describes advantages of SI and recommends strategies for mitigating DDoS attacks. Section 7 presents our experimental results using the ns-2 simulator and performance results obtained from DDoS attack experiments carried out on the DeterLab test bed. Section 8 discusses scalability and novel defenses against strategic attacks against the SI infrastructure. Finally, Section 9 summarizes our conclusions. Appendix A lists acronyms and abbreviations used in this paper. Appendix B provides details for our DeterLab experiments. We assume the reader is familiar with the basics of computer network security, as presented by Kaufman, Perlman, and Speciner [33] , for example.

2 Overview of Spread Identity

In this section we give a brief overview of SI, focusing on its architecture, dynamic address remapping mechanisms at the source and destination organization gateways, benefits of network communication anonymity and DDoS defense, and engineering challenges to its deployment.

As shown in Figure 1, we assume a model in which a client in some organization *X* wishes to communicate with a server in another organization *Y*. A trusted gateway for organization *X* dynamically assigns an IP address to the source client, from among the pool of all routable IP addresses within *X*.

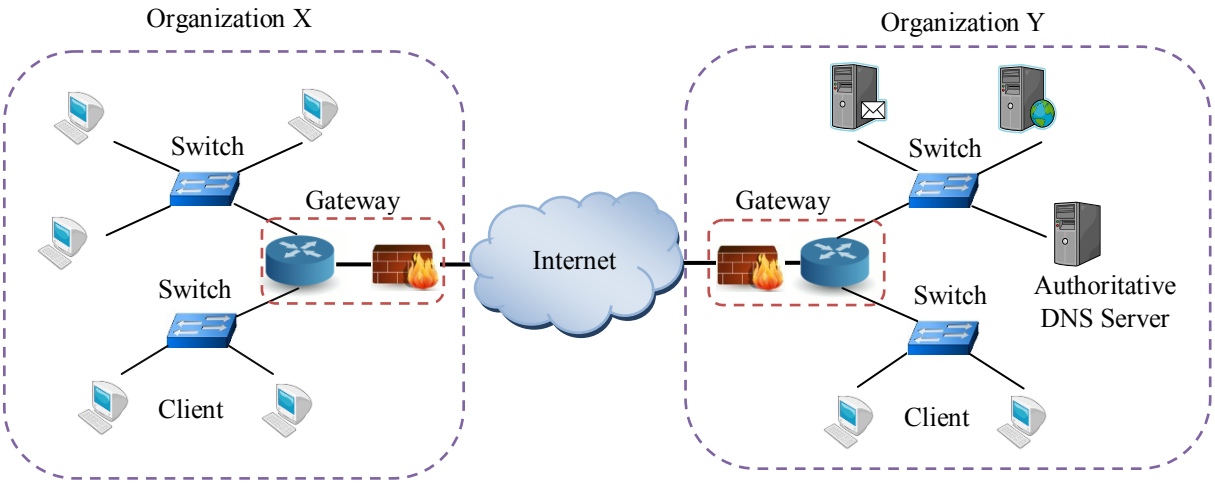


Figure 1: System architecture of spread identity for Internet communications. When a client in organization *X* initiates a communication session with a server in organization *Y*, the gateway for *X* dynamically and randomly assigns an IP address for the client from the pool of all routable IP addresses within *X*. In response to a name resolution query from *X*'s gateway, the authoritative DNS server for *Y* dynamically assigns an IP address for the destination server from the pool of all routable IP addresses within *Y*. Each gateway creates a NAT entry, valid for the communication session, based on the dynamic assignment by its organization.

Similarly, a trusted DNS server for organization *Y* dynamically assigns an IP address to the destination server, from among the pool of all routable IP addresses within *Y*. These translations are used only for the duration of the communication session. Unlike traditional NATing in which source IP addresses are assigned in a predictable way, SI assigns source IP addresses in an unpredictable manner (including random assignments). Unlike traditional static DNS assignments of hostnames to IP addresses, SI dynamically assigns hostnames to destination IP addresses in an unpredictable manner.

For simplicity, Figure 1 depicts organization *X* only as a source organization, and organization *Y* only as a destination organization. Typically, however, each organization would implement both source_SI (to process outgoing communications that originate from within the organization) and destination_SI (to handle inbound communications destined for hosts in the organization). It is also possible to implement SI at the destination gateway only (to manage only inbound communications), albeit doing so would not reap its sender anonymity benefits. Similarly, it is possible to implement SI at the source gateway only (to handle only outbound communications), though doing so would forfeit robust DoS protection and enhanced traceback capabilities.

SI works by dynamically translating addresses at the source gateway and destination DNS server. At the source gateway, *Network Address Translation (NAT)* table entries are created to enable the gateway to route packets to particular hosts and to discard packets without NAT table entries. In other words, the

NAT entries are leveraged as dynamic access control tokens (for this reason, we refer to NAT entries as "tokens"). The destination organization has an authoritative DNS server, which maps host names to IP addresses for that organization. With SI, each session must be preceded by a DNS query during which the source client discovers an IP address for the destination host, as dynamically assigned by the destination's authoritative DNS server. This DNS server stores triples of source IP address, destination host name, and dynamically assigned IP address of destination host. These dynamic NAT associations allow the same IP address to be assigned simultaneously to multiple concurrent inbound connections to the same or distinct destination hosts as long as the sources are distinct (so that each source-destination address pair is unique). Likewise, the same source IP address can be used simultaneously to support multiple concurrent outbound connections from sources to distinct destinations. Furthermore, since incoming and outgoing flows are handled separately, the same routable IP address can simultaneously support multiple concurrent incoming and outgoing connections, subject to a few distinctness constraints. The DNS server also communicates its mappings of

hostnames-to-IP_addresses to its organizational gateway, to enable routing and filtering.

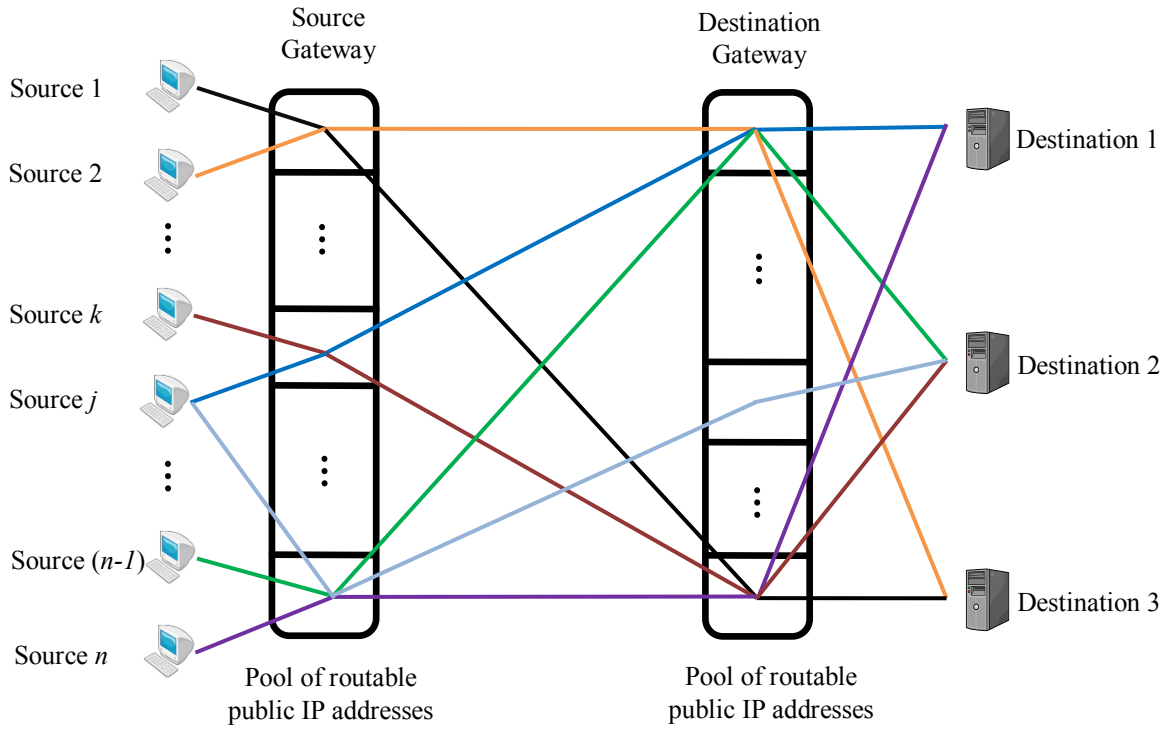


Figure 2: Network address translation tables of the source and destination gateways using spread identity. For each session, each gateway dynamically assigns to its communicant host an IP address from the pool of all routable IP addresses within its organization. Moreover, different hosts can share the same IP address when communicating with distinct peers. For example, in this figure, Source 1 and Source 2 share the same IP address. These dynamic address translations, augmented by address multiplexing, provide anonymity against network eavesdroppers. In addition, using multiple IP addresses for the same destination enables simple and powerful DDoS protections that block attackers without necessarily blocking legitimate users.

Figure 2 shows how the source and destination gateways assign IP addresses to hosts within their organizations. For example, for each session, the source gateway dynamically assigns to the source host

an IP address randomly selected from the pool of all routable IP addresses within the source organization. Different IP addresses can be assigned to the same host when simultaneously communicating with different parties. In addition, transport layer port numbers can be used to classify and route flows (this is a common practice today). Because SI does not use port numbers, port numbers can be used for any purpose.

SI for Internet communications achieves two major benefits: network-level anonymity and enhanced DDoS defense capabilities. Network-level anonymity is achieved through pseudonyms assigned by a trusted gateway at the source organization and by a trusted DNS server at the destination organization; as such, SI is similar to a one-layer mixnet [11] . For each session between a source host and a destination host, the source gateway dynamically assigns a temporary pseudonym to the source host chosen as one of the routable IP addresses assigned to the source organization. Similarly, the destination DNS server assigns a temporary pseudonym to the destination host chosen as one of the routable IP addresses assigned to the destination organization. Thus, an eavesdropper listening to packets flowing through the Internet between the source and destination gateways learns only the source and destination domains; the eavesdropper cannot see the actual complete IP addresses of the source and destination hosts. Although the eavesdropper can link source and destination packets within any session, she cannot link packets between different sessions. Encrypting DNS requests and responses at gateways would protect the established bindings of pseudonyms from the Internet eavesdropper.

Destination filtering can mitigate adverse impact on legitimate traffic as follows: A typical DDoS attack attempts to clog the network bandwidth of a particular target IP address with traffic emanating from many evil hosts. Without SI, blocking all packets sent to the target would block both malicious and legitimate traffic. With SI, however, each session with the target uses a different destination IP address. Therefore, packets with suspicious destination addresses (*e.g.,* destination addresses that occur too frequently) can be filtered without blocking legitimate packets sent to the target from other sessions. Furthermore, if other gateways and routers cooperate, destination filtering can be carried out inexpensively and closer to the source, lessening collateral bandwidth consumption from the attack. The SI concept can be deployed at the destination-side alone, at the source-side alone, or at both ends. Moreover, an organization can choose to implement SI for inbound traffic only, or for outbound traffic only, or for traffic flows in both directions. Thus, SI is incrementally deployable at any organization and for flows in either direction, because the changes it requires are restricted to the last-hop (which is controlled by the end-user organization in cooperation with the ISP).

Even if SI is deployed only at the destination organization and only for inbound flows, it substantially enhances DDoS defense capabilities by facilitating packet filtering on the destination address. The destination address, which is dynamically assigned for each source host, serves as a dynamic "flow marker," making it easier to track suspect flows. Filtering based on the destination address has several intrinsic advantages, when compared with filtering based on source-addresses:

(a) The destination address cannot be spoofed or modified in transit—otherwise the payload will not reach the intended victim host which is the destination. By contrast, source addresses can be spoofed.

(b) Typically, the number of destination addresses over which destination hosts are spread is drastically smaller than the number of all possible IP addresses, so the filter does not need to match against an unduly large number of possible destination addresses. In contrast, the source address can be any IP address.

(c) Because a source host must learn the dynamically assigned destination IP address, the destination gateway knows the IP address of the machine to which it sent the requested destination address. That is, with SI, the attacker must expose at least one of its bots to learn the dynamically issued destination address. While possibly incomplete, this IP address is a useful starting point for a trace-back analysis, even if the attacker shares the target address with her conspirators and even if this IP address is the last hop in an anonymizing network such as Tor [14, 21].

If SI is implemented at both the source and destination organizations, filtering of flows becomes easier because the source address is also dynamic and therefore can be used as a flow-marker token, together with the destination address.

When SI is implemented at both the source and destination organizations, the following enhanced variation is possible. Using a method analogous to spread-spectrum radio broadcasts, the source and destination gateways can frequently change their address translations within a session, following a cryptographically-secure pseudorandom pattern derived from a shared secret key. We refer to this enhanced version as the *Fully Dynamic Identity Spreading (FDIS)* method. It enables novel defenses against DDoS attacks targeting all addresses of an organization.

Implementing SI raises several engineering challenges: orchestrating DNS caching at hierarchical DNS servers and host machines, handling loads on DNS servers, and scaling gateways for larger organizations. In the next two sections we propose solutions to these challenges.

3. Spread Identity Architecture for Internet Communications

We describe our SI architecture for Internet communications in terms of its components, assumptions, protocols, and design and performance issues.

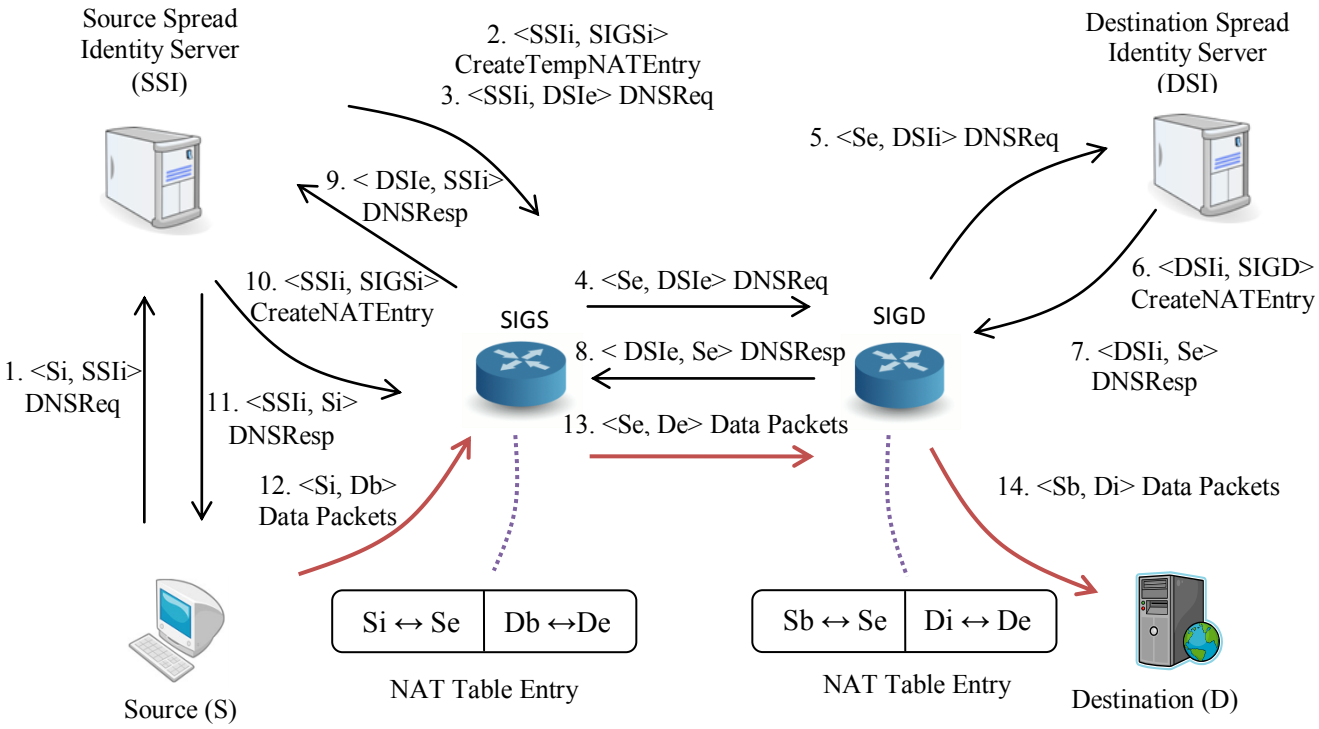


Figure 3: Connection establishment and data transfer protocol for spread identity. When a *Source (S)* establishes a connection with a *Destination (D)*, the *Source Spread Identity Server (SSI)* dynamically assigns an external IP address S*e* to S, and the *Destination Spread Identity Server (DSI)* dynamically assigns an external IP address D*e* to D. The DSI also creates a NAT entry for D*e* at the *Destination Spread Identity Gateway (SIGD)*. After receiving a DNS response from DSI, SSI creates an NAT entry for S*e* at the *Source Spread Identity Gateway (SIGS)* and forwards the DNS response to the source. This figure shows the eleven steps (shown in black) that precede the main data transfer (shown in red). For each step, the notation <*s, d*> denotes the original source and ultimate destination IP address of the message. For example, in Step 12, the source sends data packets from its internal IP address S*i* to the destination's external IP address D*e* via the SIGS and SIGD. An network eavesdropper listening to packets between SIGS and SIGD sees only the dynamically assigned IP addresses S*e* and D*e*.

3.1 Components

As shown in Figure 3, SI is implemented by trusted SI servers associated with the gateways of the source and destination organizations. More specifically, a *Source SI Server (SSI)* determines the associations of host internal (private) and external (public, routable) IP addresses within the source organization. The SSI includes the functionality of a DNS resolver for the source organization. Traffic flowing in or out of the source organization is processed by a *Source SI Gateway (SIGS)*, which includes a router and firewall whose NAT entries come from the SSI.

Similarly, a *Destination SI Server (DSI)* determines the associations of host names and routable IP addresses within the destination organization. The DSI includes the functionality of the authoritative DNS server for the destination organization. Traffic flowing in or out of the destination organization is processed by a *Destination SI Gateway (SIGD)*, which includes a router and firewall whose NAT entries come from the DSI.

The SSI is a single trusted logical component which includes a modified DNS resolver, and the DSI is a single trusted logical component which includes a modified authoritative DNS server. In designs with multiple gateways per organization, the SSI and DSI coordinate the gateway routers and firewalls.

3.2 Assumptions

To ensure appropriate performance and reliability, we assume the following. (1) Each organizational gateway can perform network translations at link speed [12, 41] . This capability enables each gateway to maintain per-flow state and to translate addresses for each incoming and outgoing packet. (2) Gateways are replicated for high availability. (3) To avoid single points of failure, organizations employ multi-homing [64] , with multiple addresses for network gateways and multiple links.

3.3 Protocols

Figure 3 shows how a source S establishes a connection with a destination D, and transfers data, using SI. Upon receiving a DNS query from the source, the SSI dynamically assigns an identity (external IP address) S*e* to S and forwards a DNS request to DSI. The DSI dynamically assigns an identity D*e* to D's hostname for the DNS response, sends a corresponding NAT entry to SIGD, and sends the DNS response (including D*e*) to SIGS. Based on the DNS response from DSI, SSI sends a NAT entry to SIGS and forwards the DNS response to S. Once the NAT entries at both ends are established, S sends data packets to D via SIGS and SIGD. An eavesdropper listening to packets sent between SIGS and SIGD sees only the dynamically assigned IP addresses S*e* and D*e*.

We now explain each step of the Connection Establishment and Data Transfer Protocols in more detail. We also explain their use of blinded addresses. Throughout, we use the following convention:

"[S, *s*] $\rightarrow$ [D, *d*]: msg" means message msg is sent from S (where *s* is the original source IP address of msg) to D (where *d* is the ultimate IP address for msg). Figure 3 abbreviates this notation as "<*s*, *d*> msg".

We also use the following notation to describe the internal (private) and external (public) IP addresses of various entities. For any entity E, E*i* denotes the internal IP address of E, and E*e* denotes the external IP address of E. Similarly, E*b* is an internal blinding address for E. With this notation, SSI*i* denotes the internal IP address of SSI, and DSI*e* denotes the external IP address of DSI.

3.3.1 Connection Establishment Protocol

1. [S, S*i*] → [SSI, SSI*i*]: DNS request (to resolve the hostname of D)
   To communicate with the destination host, the source host needs to resolve the hostname of the destination to obtain the dynamic identity (external IP address) D*e* of the destination.

2. [SSI, SSI*i*] → [SIGS, SIGS*i*]: Request to create temporary NAT entry
   First, the SSI randomly selects a public IP address S*e* for S from the pool of all public IP addresses for its organization that are not currently communicating with the same destination D. Second, the SIGS creates the temporary NAT entry (SSIi ↔ S*e*, DSI*e* ↔ DSI*e*) to be used for the DNS request and response.

3. [SSI, SSI*i*] → [SIGS, DSI*e*]: DNS request
   First, the SSI obtains the IP address DSI*e* of the destination domain authoritative DNS server using its local DNS cache or by making an iterative DNS query to higher level DNS servers. Second, the SSI sends the DNS request to DSI via SIGS.

4. [SIGS, S*e*] → [SIGD, DSI*e*]:
   Using the temporary NAT entry, the SIGS replaces the source IP address SSI*i* with S*e* in the DNS request. Then, acting as an edge router, the SIGS forwards the modified DNS request to the DSI.

5. [SIGD, S*e*] → [DSI, DSI*i*]: Forwarded DNS request
   The SIGD forwards the DNS request to DSI.

6. [DSI, DSI*i*] → [SIGD, SIGD*i*]: Request to create NAT entry
   First, the DSI randomly selects a public IP address D*e* for D from the pool of all public IP addresses for its organization that are not currently communicating with the same source S. Second, the DSI selects an internal "blinding" address S*b* for S to hide the relationship between S*e* and D*i* within D's private network (for more details, see below). Third, the SIGD creates the NAT entry (D*i* ↔ D*e*, S*b* ↔ S*e*) to be used for communications between S and D.

7. [DSI, DSI*i*] → [SIGD, S*e*]: DNS response
   The DSI sends its DNS response (of the resolved IP address D*e*) to S via SIGD.

8. [SIGD, DSI*e*] → [SIGS, S*e*]: Forwarded DNS response
   Using the NAT entries, the SIGD replaces the source IP address from DSI*i* to DSI*e* in the DNS response and forwards the modified DNS response to S*e*.

9. [SIGS, DSI*e*] → [SSI, SSI*i*]: Forwarded DNS response
   Using the temporary NAT entry, the SIGS replaces S*e* with SSI*i* in the DNS response and forwards it to SSI.

10. [SSI, SSI*i*] → [SIGS, SIGS*i*]: Create NAT entry request
    First, the SSI selects a blinding address D*b* for D to hide the relationship between S*i* and D*e* within S's private network. Second, the SSI sends a request to SIGS to replace the temporary NAT entry with
    (S*i* ↔ S*e*, D*b* ↔D*e*) to enable communications between S and D.

11. [SSI, SSI*i*] → [S, S*i*]: Forwarded DNS response
    The SSI forwards the DNS response (of the resolved IP address D*e*) to S.

Once the connection has been established, S and D can communicate with each other as follows.

3.3.2 Data Transfer Protocol

12. [S, S*i*] → [SIGS, D*b*]: Data packets
    Source S sends data packets to D addressed to D*b*.

13. [SIGS, S*e*] → [SIGD, D*e*]: Data packets
    Using its NAT entries, the SIGS replaces S*i* with S*e* and D*b* with D*e*. Then, SIGS forwards the data packets to D*e*.

14. [SIGD, S*b*] → [D, D*i*]: Data packets
    Using its NAT entries, the SIGD replaces D*e* with D*i* and S*e* with S*b*. Then, SIGD forwards the data packets to D.

3.3.3 Advantages of Blinded Addresses

We now explain our use of the *blinded addresses* S*b* and D*b*, which yields three advantages: increased host anonymity, a method for dealing with application-level caching of addresses, and expanded organizational address space.

1. The blinded addresses S*b* and D*b* increase the anonymity of the end hosts against a "global adversary" (see Section 6.2) who is able to (passively) observe the source and destination addresses in the headers of IP packets in transit within both the source and destination organizations (in addition to observing the externally-routable addresses that are publicly visible when the packets are in the Internet). By contrast, against an "Internet adversary" who can eavesdrop only on packets flowing between the source and destination gateways, blinded addresses are not necessary, since the actual IP addresses of the source and destination are not revealed outside the source and destination organizations. In Step 6, the DSI selects a blinded address S*b* for the dynamic address S*e*. Similarly, in Step 10, the SSI selects a blinded address D*b* for the dynamic address D*e*. These blinded addresses are used in Steps 12 and 14 of the Data Transfer Protocol. The blinded address D*b* hides the relationship between S*i* and D*e* against a global adversary who can eavesdrop on communications within the source organization; doing so is helpful if the adversary also knows the dynamic remapping between D*i* and D*e*. Similarly, the blinded address S*b* hides the relationship between D*i* and S*e* against a global adversary who can eavesdrop on communications within the destination organization; doing so is helpful if the adversary also knows the dynamic remapping between S*i* and S*e*. If an adversary can sniff packets on both sides of a SI gateway, then correlation analysis might reveal the connections. However, if the gateway is processing even a moderate amount of traffic, correlation analysis is harder, because multiple packets (belonging to multiple flows) always appear on either side of the SI gateway. The greater the number of simultaneous connections being handled by the gateway, the harder it is to perform correlation analysis.
2. The blinded address D*b* on the source-side provides a method to deal with the problem posed by application-level caching of recently resolved DNS addresses. For this purpose, the source-side SI gateway SIGS simply records recently resolved DNS queries in a list of ordered tuples of the form (Source-address (internal), name of target destination D, D*b,* time when address was assigned, session-expiry-time). The last entry in each tuple is set to 0 if the session is still not finished; otherwise the last entry records the time when the session finished, i.e., the time when the NAT entry (aka the token) was deleted from the gateway router. Let Δ be the time interval for which applications can reuse resolved addresses. SIGS retains each entry in the list for a time longer than Δ after the last connection between the resolved address and application host is terminated. Whenever the SIGS receives a name-resolution request, it first checks its list of currently active connections (NAT table) and the connections in timeout state (Δ list). If a matching entry is found in any one of

them, then the same blinded address is returned. For instance, if the same source host S requests resolution of the same host name D" a second time, (possibly after a while past the expiry of the first session with the same destination D), then SIGS returns it the same blinded address D*b*. As a result, applications on host S that might have cached the IP address D*b* continue to see the same address and therefore work transparently.

3. Using blinded addresses enables each organization to use any IP addresses for its internal network (even those assigned to other organizations). Double-NATing at the edge gateway creates a strong separation between internal and external address spaces.

   For example, suppose that the addresses used in the destination organization are also used in the source organization. This does not cause any problems as outlined below: (a) Hosts within the source organization can communicate with each other using the assigned addresses (which are also used inside the destination organization, but the hosts within the source organization need not be aware of this double usage. (b) If a host in the source organization needs to communicate with a host in the destination organization, it sends a name-resolution query. Provided the SIGS uses a blinded address that is not in the set of addresses used within the source organization, packets sent to the blinded address will be deemed as going outside the source domain and will arrive at SIGS, where proper NATing makes everything work transparently. Thus, , blinded addresses allow unlimited reuse of routable addresses, thereby drastically mitigating the address space problem that has been limiting IPv4.

In summary, blinded addresses are necessary to exploit the full potential of indirection.

Among the implementation details that need to be addressed is a strategy by the source to decide: does a NAT table entry needed for a session exist at the destination, and should a DNS query be reissued?

4 Design and Performance Issues

In this section we briefly identify some important design and performance issues and outline strategies for dealing with them. These issues include NAT table size and time required to perform NAT translations, DNS caching, and DNS traffic, communication delays, and dynamically adjusting the spreading granularity. We defer our discussion of the important issue of scalability to Section 8.1, following our experimental results. Software components from existing RFCs are sufficient to implement SI. In particular, we implemented SI building on RFC 2663 [53] and 2694 [54] . RFC 2663 describes twice NAT. RFC 2694 describes approaches to handle DNS name query resolution when the server is behind a NATed gateway.

To understand the performance of SI, we compare it to that of a single-NAT system. It is common practice to perform NAT at the edge gateway to hide the internal structure of an organization's network. In such a single-NAT system, all internal source addresses in outbound flows are replaced by one single IP address, which is externally visible (port numbers are used to manage distinct flows). Section 7.2 experimentally compares performance parameters of our implementation of SI with those of a baseline single-NAT system.

4.1 NAT Table Size and Translation Time

As confirmed by our experiments (Section 7.2), NAT table size in SI is no larger than that in a typical single-NAT system. NAT table size is dominated by connections-tracking information, not by address-pairs.

In our baseline single-NAT system, the number of NAT entries in the edge gateway does not depend on the number of client hosts that are simultaneously communicating with the server in the destination organization. A single rule, which specifies that the source addresses in all outbound flows are to be replaced by one single externally visible address, is sufficient in principle. By contrast, in SI each communicating peer requires a separate NAT entry. Therefore it appears that the gateway NAT table

could grow more rapidly in SI than in the baseline system. This would be the case if address-pairs were the only item requiring memory in a gateway.

However, almost all connection oriented higher level protocols used in the Internet (including FTP, SCP, SSH, HTTP) typically open multiple TCP connections simultaneously, using multiple port numbers; therefore, these protocols need to track the state of each connection. Consequently, even if a single client host is communicating with the server, the amount of memory required for proper connection tracking, even in the simple baseline single-NAT, is far more than that required for an address pair.

In addition, edge-gateway firewalls typically use a significant amount of space per connection. Edge gateways are appropriate and logical places to implement firewalls. Therefore, edge gateways typically contain firewall rules in addition to NAT entries. For a non-trivial level of protection, the firewalls need to be state-full, which in requires some non-negligible amount of memory per connection (as opposed to per communicating host).

Regardless of whether the destination organization has deployed SI or only the baseline single-NAT system, the number of deployed connections remains the same because the number of connections is dictated by actions of the client hosts. Consequently, NAT table size will be about the same for SI and the baseline single-NAT system.

NAT translation time with SI is about the same as that of a baseline single-NAT configuration. This fact is expected because translation, i.e., replacing source/destination headers in the packet takes a relatively small amount of time compared with the time spent in maintaining connection tracking information. Our experiments corroborate this observation.

4.2 Caching of DNS Responses by Non-SI-Aware DNS Servers

Typically, DNS queries are iterative, and most of the responses to hierarchical DNS servers can be cached by anyone exactly the same way it currently happens. However, the last reply from the authoritative DNS server for the destination organization to the source gateway must not be cached by intermediate servers, nor by the source side gateway if it is not SI-aware. This policy can be achieved by setting the *Time To Live (TTL)* field in the response from destination authoritative DNS server to zero. If the other DNS servers are properly implemented as per the DNS protocol specification, they will honor TTL = 0 and therefore will not cache the response from destination authoritative DNS server. Section 3.3.3 explains how application-level caching can be easily handled by blinded addresses.

4.3 DNS Traffic

How SI affects DNS traffic depends on the implementation. As our experiments show, a straightforward software implementation of SI can increase DNS traffic. However, when implemented at both source and destination organizations, SI can reduce DNS traffic. For most scenarios, non-SI methods offer no advantage over SI with regard to DNS traffic. For example, provided at least one connection to a destination host is open, a NAT entry for the destination host exists in the source-side SI gateway; in this case, any number of additional connection requests for that same destination host can be serviced without the need for DNS queries. Similarly, for a “cold start” when the NAT entry does not exist, both SI and non-SI methods will cause exactly the same amount of DNS traffic, which is likely to be dominated by the iterative part of the DNS search.

The only scenario in which non-SI methods have an advantage over SI with regard to DNS traffic is when the source SI gateway deletes the NAT entry for the destination host and the gateway cannot cache the last reply because TTL = 0. If another request for connection to the same destination arrives after the deletion of a NAT entry from the SI gateway, then the source-side DNS resolver must make a new query for the destination host again. The repeat DNS query is likely to be non-iterative because the source SI DNS resolver can cache the IP addresses of all the intermediate hierarchical DNS servers as well as the IP address for the destination authoritative DNS server from previous replies, exactly the same way and for the same amount of time as for non-SI gateways. Consequently, the extra traffic caused by SI methods

consists predominantly of non-iterative queries. Such queries do constitute extra DNS traffic; however, the relative increase in traffic (relative to the traffic for non-SI methods) is likely to be modest, because the iterative part of DNS queries is likely to dominate total DNS traffic.

If SI is deployed at both source and destination organizations, then it can actually reduce DNS traffic. As part of the deployment of SI for outbound flows, each organization could be required to dedicate a modest amount of additional hardware including sufficient memory to store the IP addresses of all authoritative DNS name servers of domains with whom it frequently communicates (doing so is easily possible even with today's hardware technology).[2] With such hardware, the need for recursive DNS queries can be obviated, thereby reducing DNS traffic.

4.4 Communication Delays

With SI, connection setup delay can increase due to (i) the extra name resolution call that is needed in some cases, and (ii) increased name resolution time, which results from added complexity to select an address from the address pool, as well as the blinded address, depending upon the system load, time of day and other state parameters. With high-speed firewalls (*e.g.*, by Cisco [41]), NATing can be performed at link speeds at SI gateways.

Once a connection is established, however, the ensuing delays are about the same for SI and non-SI methods. Thus, a data flow using TCP would experience communication delays similar to those for non-SI. Section 7.2 presents experimental data supporting these observations.

4.5 Dynamically Adjusting the Granularity of the Spread The SI methods are flexible and can be implemented at various levels to provide different types of anonymity and security. Furthermore, it is possible to adjust the level of spreading dynamically, depending on a variety of factors such as traffic conditions, system state, objectives, and policies. … etc. Table 1 highlights four granularity levels at which SI can be implemented. Our experiments operated at the IP level of granularity.

Granularity Level	Explanation
Network (coarse)	From within a given network, all hosts that request communication with entities in another external domain receive the same dynamic IP address in response to name resolution queries.
IP level	Each IP that initiates communication with another external domain, receives a unique dynamic IP in response to a name resolution query.
Session	Each session uses a different destination IP for the same domain, even though the source IP is same.
Packet (fine)	Each packet is destined to a dynamically chosen IP, even if the source IP is same. This is an example of Fully Dynamic Identity Spreading (FDIS) mentioned at the very end of Section 2 above.

Table 1: Four examples of different granularity levels for identity spreading of the destination address.

[2] The total number of IP addresses in the IPv4 protocol is about $2^{32} \approx 4$ billion (G). The number of IP domains is expected to be smaller than the number of IP addresses. Indeed the web-site www.domaintools.com/internet-statistics/ tracks the number of registered and supported domain-names on a daily basis. The total number of domain names is about 136,048,930 ≈ 136 million (M). An exhaustive storage would have 136 M names and an IP address with each name. Assuming that domain names are restricted to 160 characters (≈ 2 lines assuming 80 characters per line), the total memory required is about (164 × 136 M) ≈ 22 GBytes, which is can be easily incorporated in high- or even mid-level servers today--many individual mobile devices have 32 Gbits of RAM today.

For simplicity, Table 1, considers only dynamic control of the destination IP address, without spreading of the source address. More generally, it is possible to dynamically control the level of spreading of the source IP address simultaneously with (and in most cases, independently of) the spreading of the destination address.

The finer the granularity, the higher the implementation complexity. The finest; for example, packet-level spreading implies that the IP address(es) in each packet can be different. To achieve this level of spread, the SI mechanisms must be implemented at both the source and destination organizations, and the source and destination SI gateways must cooperate. In this scenario, the peer gateways decide upon the first address to use as part of the DNS exchange.[3] Also included in the DNS exchange are cryptographic seeds that are used by each side to generate a pseudorandom sequence of IP addresses from the respective pools of IP addresses. That is, the source IP address cycles through the source pool of IP addresses in accordance with a pseudorandom sequence generated from the shared secret. Analogously, the destination address also assumes values from within the destination pool in a pseudo-random manner.[4]

5 Previous Work

In this section, we briefly review previous related work in the areas of *Distributed Denial of Service (DDoS)* defense, anonymity, and next generation Internet architectures.

5.1 DDoS Defense

There are three main approaches to DDoS defense: (a) overlay, (b) filter, and (c) capability. Designed using core infrastructure routers, filter-based and capability-based approaches require changes in core infrastructure routers and client software. By contrast, by delegating functionality to a richly-deployed large-scale overlay network, overlay approaches do not require any changes in the core infrastructure, but they can cause significant communication latency [2, 63] .

5.1.1 Overlay Approaches

Secure overlay service (SoS) [34] is among the first overlay-based DoS defense mechanisms proposed in the literature. In SoS, only authenticated source traffic is forwarded to the destination host through a series of overlay nodes. Despite the facts that overlay nodes are richly deployed and the destination host is accessible only through the overlay nodes, a DoS attack can be mounted by spoofing the identities of overlay nodes. WebSoS [56] is an overlay approach for Web servers, wherein sources are not authenticated. Instead, graphical Turing tests attempt to differentiate attack bots from humans. Stavrou, *et al.* [56, 58] and Wang, *et al.* [63] show that overlay networks increase overall communication network latency by a factor of five to ten, due to the underlying chord routing protocol [60] . Mayday[2] discusses various overlay design choices, allowing a performance-security tradeoff. Stavrou, *et al*. [57] propose a sweeping DoS attack against overlay networks, wherein the attacker follows legitimate source traffic and brings down all overlay nodes with which the source communicates. To defend against such attacks, the multipath overlay approach [57] randomly spreads traffic across multiple overlay nodes.

Stoica, *et al.* [36] propose a DoS-resilient architecture using the *Internet Indirection Infrastructure (I3)* based overlay network [59] , in which the destination host can dynamically install triggers at overlay

[3] We assume all communications among SI gateways are encrypted.

[4] Out of order packet delivery can be handled by expanding the size of a sliding window of IP addresses expected in the next window of packets. Instead of generating only the next address in the pseudorandom sequence, generate some *n* successive values and maintain them in a sliding window. Doing so is substantially more complex than IP-level spreading , but it is the only defense against a brute-force strategic DDoS attack against the entire address pool or against the authoritative DNS server and DNS infrastructure. See Section 8.2 for further details.

All these changes are still restricted to the edge gateway or the last-hop ISP router. Everything further upstream beyond the last hop is left untouched. Likewise, everything on the internal side of the SI Gateway is also left untouched.

nodes to enable communication with legitimate sources. These dynamic triggers are similar to the dynamic NAT entries at the destination in SI. Unlike SI, however, the I3-based approach uses a static destination identity, enabling an attacker to bypass the overlay network once the attacker discovers this static address.

To resist large DDoS attacks, overlay approaches require a rich deployment of the overlay network. In addition, overlay nodes are not managed by a single authority, complicating security management and increasing the risk that the attacker can compromise at least one overlay node. OverDoSe [52] addressed this issue by separating the source and destination hosts at the IP level. In OverDoSe, the destination host uses RSVP-TE [8] to establish tunnels with overlay nodes, which enable destination hosts to tear down connections with compromised overlay nodes dynamically. In addition, OverDoSe uses hash-based computation puzzles to enforce fairness in the request channel.

Akamai's SiteShield [1] is a commercial overlay-based DDoS protection mechanism. In SiteShield, the destination host is accessible only through Akamai's overlay nodes. Because of their large numbers and powerful hardware, overlay nodes in SiteShield can absorb large DDoS attacks. To reduce communication latency, SiteShield uses proprietary overlay routing protocols and Web caching. Nevertheless, Akamai's approach requires a very large DNS infrastructure.

Unlike overlay approaches, SI does not incur high communication latency.

5.1.2 Filter-Based Approaches

When attack traffic surpasses a specified threshold, filter-based approaches install source IP filters. Pushback [40] identifies attack traffic flow and recursively installs filters near the source. Pushback, however, suffers from *s*trategic filter-request spoofing attacks, wherein the attacker attempts to cause legitimate source traffic to be blocked. AITF [6] proposes a three-way handshake protocol to address filter-request spoofing. But AITF suffers from filter-exhaustion attacks, in which the attacker floods the source gateway with filter requests so that the source gateway cannot accept legitimate filter requests. StopIt [39] resists such filter-exhaustion attacks by verifying filter requests using a flow cache before filters are installed. Furthermore, StopIt uses two-level hierarchical fair queuing as a fail-safe method to mitigate DDoS attacks against its control channel for sending filter requests.

Filter-based approaches require changes to core infrastructure routers, and they require substantial state information since attacks can come from any source. By contrast, SI does not require any changes to routers (except possibly for edge routers), and by virtue of destination filtering requires less state information. Moreover, SI destination filtering is less likely to block legitimate traffic.

5.1.3 Capability-Based Approaches

Anderson, *et al.* [3] proposed the first capability-based approach for mitigating DoS attacks. In this two-step approach, the source first sends a request to the receiver seeking permission to send data. Second, if the receiver verifies the sender as a legitimate communicant, the receiver provides an authorization token. The sender includes this token in subsequent data packets, and routers verify the token. This design, however, does not stop DoS flooding attacks of capability requests. SiFF [65] prevents capability flooding attacks on bottleneck links by differentiating legitimate traffic from capability-request traffic. Yet in SiFF, the attacker can still flood the capabilities-request channel. TVA [66] addresses this problem using hierarchical fair queuing based on source path identifiers, but according to Portcullis [44] , path-identifier fair queuing scales poorly for large networks. Consequently, TVA proposes per-computation fairness using hashing puzzles.

Filter and capability defenses to DDoS attacks require changes to core infrastructure routers and end hosts. To perform at their best, they also require cooperation among different ISPs. These are among the reasons why, currently, Internet infrastructure providers commonly deploy overlay-based *Content Delivery Network (CDN)* approaches, such as Akamai's network, and simply enhance network bandwidth and server resources as a hedge against DDoS attacks.

For DoS protection, SI behaves much like a capability-based system, with authorization and communication phases. For authorization, SI uses the DNS mechanism. For communication, SI uses the dynamically revealed destination address like an authorization token, avoiding the need to send extra packet headers or to change functionality of core routers. Routers are designed to parse destination addresses. By leveraging the destination address as a flow marker, SI exploits fundamental capabilities of router hardware without adding any extra fields to headers or data. Consequently, SI works transparently, even if the payload (except the IP header) is encrypted.

5.2 Anonymity

A variety of approaches have been proposed for achieving anonymity in network applications. Chaum's [11] "mixnet" relays each message through a network of one or more mix nodes. At each stage, the current mix node encrypts the message using the next mix node's public key. Each mix node decrypts the received message, removes header information, appropriately pads the message, possibly batches messages, and forwards the message to the next stage.

Other relay-based anonymity techniques fall into two categories: those that introduce large and variable latency (*e.g.,* Babel [27] and Mixminion [17] , and those for interactive applications such as Web browsing and SSH that do not introduce significant latency (*e.g.,* Anonymizer [5], Tor [21] , and Crowds [48]).

Virtual Private Networks (VPNs) [26] [25] typically use tunneling protocols to virtually connect physically distinct locations within an organization, over the (public) Internet. A simple application of this idea is exemplified by the secure shell [67] protocol. In a VPN, the payload is encrypted so that no public observer can see the data. Any observer in the Internet, however, can easily determine the actual identity of the entities at the source and the destination, simply from the IP addresses in the headers of the packets in transit.

In comparison with SI, VPN is limited because an attacker will always be able to find the source and destination entities. In contrast, with SI, the difficulty to track the end entities increases as the depth of deployment increases. For example, suppose that two sub-organizations implement SI, and their parent organization also implements SI. In this case, a public attacker cannot discover identities any further than the top-level organization. If this top-level organization is sufficiently large (such as an ISP) then for all practical purposes, the attacker is unable to find identities of the source or destination. Nested NAT will also provide a similar effect; however, NAT is usually predictable and is not deliberately designed to make the mappings between host names and IP addresses appear many-to-many or random, as is SI. Moreover, SI can be implemented at finer granularity levels, making it possible to increase the level of anonymity further.

Anonymizer [5] removes user identifying information from HTTP requests, changes the source IP address, and forwards the request to the appropriate Web server. It is similar to a mixnet with one mix node, which enables a passive eavesdropper to link sender and receiver. Java Anon Proxy (JAP) [23] solves this problem using a cascade of mixes. PipeNet [15] is a theoretical model for accessing Web servers over the Internet. Like JAP, it uses a cascade of mixes. In PipeNet, all clients send legitimate or dummy traffic to the same cascade mix at the same time. PipeNet provides a strong level of anonymity and protects against traffic-analysis attacks, but PipeNet suffers from DDoS vulnerability and inefficiency.

Crowds [48] provides sender anonymity and sender-receiver unlinkability by probabilistically relaying Web requests to a randomly selected node in the crowd or to the final destination. Replies are sent through the established route. As suggested by Diaz, *et al.* [16] , anonymity in Crowds depends on the adversary being unable to observe all links. Hordes extends Crowds and improves its performance by using a UDP proxy and by using multicast replies instead of traversing the reverse path.

Tor [21] provides sender and receiver anonymity based on Chaum's mixnet. Tor does not make significant attempts to prevent global adversary or traffic analysis attacks. TARZAN [25] is similar to Tor for peer-to-peer anonymous IP overlay networks. Tunnel failures are more frequent in TARZAN because of peer failure or a peer leaving the overlay network. Tunnel failures result in significant computational overhead and latency.

SI provides network anonymity in a fashion similar to that of a mixnet with a fixed path of two nodes. The fixed path avoids the need to insert routing information into messages. As is true for ISDN mixnets [45] , the fixed path also protects SI from intersection attacks [10] . Unlike Tor, SI avoids the overhead of performing multiple encryptions and decryptions, but SI does reveal the source and destination organizations. We also note that SI can create Tor-like effects when two or more SI gateways are in-series in the path between the source and destination.

SI is transparent to the application; thus, there is no need to modify client software. Furthermore, applications with different transport protocols can share communication sessions. As discussed in Section 6.2, SI anonymity achieves forward secrecy because once the communication session terminates, the SIG gateways (the mix nodes) destroy NAT entries. Unlike Tor, because SI transparently anonymizes DNS requests, SI does not need to take special measures to prevent DNS leaks

5.3 Other Related Work

Kewley, *et al.* [35] describe a dynamic TCP/IP address translation mechanism called DYNAT for obstructing an eavesdropper from discerning the true network topology. Using a shared symmetric key between sender and receiver, the IP addresses and port are encrypted. As noted by Fink, *et al.* [24] , who subsequently developed a per-packet variant of DYNAT, this system could also be used for anonymity. Unlike SI, DYNAT must be implemented by both sender and receiver. Also, because DYNAT requires pairwise key exchanges, it is less scalable than is SI.

Proposed in 2000, TRIAD [13] is a next generation Internet architecture based on content routing that provides scalable content routing, caching, virtual private networking, policy-based routing, and load balancing, without relying on DNS servers. Both TRIAD and SI use NATing at source and destination edge routers to translate between external and internal IP addresses. SI works at the IP layer, whereas TRIAD works at the content layer. Unlike SI, TRIAD does not map identity (URL) to IP address dynamically.

Both VNAT [61] and SI use address blinding in network address translation. VNAT, however, uses address blinding to achieve host mobility, whereas SI uses it for anonymity (SI could additionally use address blinding for mobility too).

Amazon's Elastic Cloud (EC2) reportedly uses address pooling in its load-balancing service.

Following Phatak's [47] 2005 proposal of the SI concept, in 2008, Dawalbhakta [18] carried out some preliminary experiments on SI, measuring connection establishment delays and numbers of packets reaching their intended destinations, with and without SI. These experiments, however, did not compare SI with competing approaches. Also, we believe that, with appropriate high-speed routers, SI will not create any additional connection establishment delays, except possibly for increased DNS traffic. In 2010, Sonawane [55] reimplemented SI and, using the ns-2 simulator, compared its performance at mitigating DDoS attacks with several existing methods. In 2011, building on methods described in RFCs 2663 and 2694, Joshi [31] reimplemented SI and compared its performance with a single-NAT baseline system on a real network with DeterLab.

6 Advantages of Spread Identity

In this section, we explain the two major advantages of SI: enhanced defense against DDoS attacks using destination filtering, and network-level anonymity resulting from pseudonyms assigned by trusted gateways. We also discuss advantages of a restricted form of SI deployed at the destination organization only.

6.1 DDoS Defense Mechanisms

After summarizing our assumptions and threat model, we explain how SI provides a powerful destination-filtering mechanism for defending against host resource and bandwidth-clogging attacks. We also discuss strategies for mitigating attacks against the SI infrastructure. See Section 8.2 for an explanation of how fully dynamic identity spreading can counter a strategic attack launched simultaneously against all organizational addresses.

6.1.1 Assumptions and Threat Model

The attacker aims to consume network bandwidth and secondary resources (CPU time, memory) of the target destination host, DNS servers, and SI gateways. We assume that the adversary can launch synchronized DDoS attacks from millions of attack bots, using tools such as Trinoo [22] , and we assume the adversary can spoof any IP address in place of the true source address in IP packets. The destination address cannot be spoofed if the packets are to reach the intended victim.

We trust the source and destination SI servers (SSI, DSI) and gateways (SIGS, SIGD), and we assume they cannot be compromised. We assume that gateways have updated standard application specific defense mechanisms (*e.g.,* [36]). We assume the destination host or gateway can detect DDoS attacks, for example, using standard techniques described by Sekar, *et al*. [51] . Additionally we assume the following three conditions:

(i) The ISP cooperates either by allowing installation of filters on the last-hop ISP router that directly interfaces with the organizational gateway, or by allowing the destination organization to provision a proxy router immediately after the ISP's router in series with it. The organization can control the proxy router and install filters on it as required.

(ii) The ISP has the ability to handle a significant amount of traffic due to infrastructural advantages. Under this realistic assumption, any link beyond the last hop is part of the ISP's infrastructure and therefore has sufficiently high bandwidth to withstand any DDoS attack. That is, we assume links controlled by the ISP have bandwidths substantially higher than the last-hop bandwidth and therefore these links cannot be brought down by DDoS attacks. The destination gateway, DNS server, or last hop link is more likely to collapse under a DDoS attack before the high bandwidth links controlled by the ISP collapse.

(iii) The SI implementation uses different data links for DNS traffic and data. The data links can be further prioritized, assigning more important traffic flows to a physically separate link, and leaving all other lower priority data flows on another physical link.

As described by Phatak[47] , we identify the following four types of attacks, which aim to overload bandwidth or other resources at various targets.

1. *Attacks wherein the attackers do not get a valid token (i.e., do not make a DNS query).*

   For example one bot could make a DNS query and share the dynamically learned IP address of the destination with other bots.

2. *Attacks that are not distinguishable from genuine traffic overload.*

3. *Bandwidth clogging attacks*.

   Attackers know that their traffic will not make it past the destination gateway, so they try to clog the last-hop link by overloading it.

4. *Attacks against the SI DNS servers and other SI infrastructure.*

Next, we discuss defenses against these four attacks. For the rest of this section, we assume that SI is implemented only at the destination, and only for inbound flows, which is the hardest case for the defender. If SI is also implemented at the source organization, then the destination can send "traffic-

throttle" messages of various types as explained in more detail by Phatak [47] . Throttle messages make it possible to stop the unwanted flows at the source SI gateway without sacrificing any network bandwidth as a collateral damage.

6.1.2 Overview of Defenses

The main strategic advantage of SI is that it enables dynamic filtering on destination addresses (and more specifically on "source-destination" address pairs, aka, tokens) in a way that is unlikely to disrupt legitimate traffic. We explain defenses to the aforementioned attacks in three steps: defenses against attacks on destination hosts, defenses against input-bandwidth clogging attacks, and mitigating attacks on the SI infrastructure. Our defenses include a combination of four strategies: address filtering, rate limiting, reverse Turing tests, and redirection of attackers to honey pots.

6.1.3 Defenses against Attacks on Destination Host

Against Attack-1, where the attacker does not have a valid token: note that SI neutralizes the attack because flows are filtered at the destination gateway; without a valid token, a flow does not reach the destination host. Furthermore, bots cannot profitably reuse destination addresses given to them by other bots: if many bots used the same source-destination token, then the attack could be easily stopped by filtering only one destination address at edge routers. To mount an effective attack, each bot must make a fresh DNS query. SI's simple "token matching" (*i.e.,* matching both the source and destination addresses to an existing NAT entry) is highly effective against all Type-1 attacks, including TCP SYN floods and ``teardrop'' attacks.

Against Attack-2, where each bot requests and receives a valid token, a fundamental defense is to rate limit the number of DNS responses to a value less than the maximum number of simultaneous connections the server can support. This way the capacity of the server is never exceeded. Without a NAT entry, traffic cannot reach the destination host; it will be simply filtered off at the destination gateway. The NAT entry can be obtained only by making a DNS query. Therefore, any attack that wants to reach the destination must first manifest itself as a flood of DNS queries.

We also recommend deploying reverse Turing tests to prevent automatic attackers from crowding out legitimate users. Sources that participate in a DDoS attack are likely to be controlled by some automated mechanisms, whereas genuine users are more likely to be live human beings interacting with the server. This critical difference can be exploited to distinguish and isolate the attack flows as follows: If the destination gateway sees a flood of DNS queries and thinks that it is under a DDoS attack, then it can send all the sources that made the queries a CAPTCHA test (for example, as a pop-up window that simply says "Experiencing heavy load, to proceed please enter the plaintext corresponding to the distorted-text seen in the image."). If a human is at the source end, he or she will likely be able to solve the CAPTCHA, whereas attack bots controlled by automated scripts will likely fail the test. Once a source is identified as abnormal or as an attack bot, the corresponding NAT entry can be deleted, thereby taking away the attacker host's ability to reach the victim.

6.1.4 Defenses against Input Bandwidth-Clogging Attacks

We now consider defenses against attacks that aim to saturate the bandwidth of the destination organization, rather than reaching a destination host. We assume the last-hop link is full-duplex, which implies that the attackers can clog only the input bandwidth, not the output bandwidth, of the last-hop link. In such an attack, a master bot instructs slave bots to send traffic toward a victim host, with the goal of occluding the victim's network links.

For example, in one attack, the master first makes a DNS query to obtain the IP address of the victim, which it shares with its slaves. In this case, the defense is the same as for Attack-1. With SI, the SIGD sends the master bot a dynamic address for the victim. Once the destination organization detects the attack and identifies the targeted dynamic address, the SIGD can block in-coming traffic destined to the targeted dynamic address. ISP edge routers are also instructed to filter out packets with the specified destination address. A major advantage of this approach is that it does not necessarily block legitimate traffic to the victim, most of which would likely have a different dynamic destination addresses. Traffic

sent to the victim without a valid dynamic address would be blocked at the SIGD or edge routers. Because each source requires a valid dynamic address, the attacker cannot benefit from spoofing her source address in her DNS requests.

This defense also works against a variation of the attack in which the master skips the initial DNS request and instead finds a valid dynamic address pair by listening to network traffic and spoofing the source address.

The SIGD can send adaptive responses if it suspects that a DDoS attack is in progress. In this case, the following strategy can be adopted: intentionally reissue the same dynamic address for the victim in response to DNS requests from suspicious domains. This strategy herds all attackers into the same corner of the namespace (*i.e.,* all the attackers are forced to use the same IP address or at most a very small set of IP addresses for the victim destination). The herding of attackers into a corner of the name space simplifies filtering off unwanted flows by blocking the specified addresses. When under attack, another more active strategy is to redirect suspicious in-coming flows to high-volume honey pots (perhaps provided by the ISP). The goal would be to convince the attacker that she has connected to a target host, when the attacker is actually consuming her resources connecting to the honey pot. Whereas filtering simply stops packets from reaching the target without stopping the attack, redirection to a honey pot safely diverts the attack.

6.1.5 Mitigating Attacks on the Spread Identity Infrastructure

We now discuss strategies for mitigating attacks against the SI infrastructure. These include attacks to consume memory, CPU, and network resources of SI gateways and DNS servers. Note that only the SI gateways are directly visible to the external world; the SI DNS servers sit behind the gateway and firewall. See also Section 8.2.

Two types of attacks can be mounted against the SI gateways: (i) a resource consumption attack (*e.g.,* attempts to consume huge amounts of memory for NAT tables). (ii) a bandwidth-clogging attack that aims at clogging the edge bottleneck link.

First, we consider resource consuming attacks against the destination gateway SIGD. In such an attack, the attacker must send many DNS requests. Suppose that millions of bots send DNS requests to the SIGD, which the gateway will detect as a flurry of DNS requests on SIGD. In this case we recommend that the SIGD require the DNS requestor to solve computational challenges. Specifically, using OAuth [28] or a similar redirection mechanism, the SIGD will first redirect each request to a third party which deploys a pool of servers that generate puzzles (such as those suggested by Shi, *et al.* [52] and Parno, *et al.* [44]) or graphical Turing tests [62] . Only if the requestor solves the puzzle or passes the test, the query is directed back to SIGD for further processing.

Another variation of this resource consumption attack might be a variable rate attack that tries to mimic a normal crowd signature. To protect against such attacks, CAPTCHA-styled puzzles could be used to revalidate the end users of sessions corresponding to oldest NAT entries.

We assume that all other standard defensive mechanisms are in place to guard against non-SI related attacks (*i.e.* attacks that are not related to SI mechanisms).

Similar techniques can also be used for flooding attacks against the SIGS. Such attacks are easier to deal with because they must emanate from within the SIGS domain.

Second, we consider defenses to bandwidth-clogging attacks against the SIGD's connection to the Internet (*i.e.,* the last-hop links). Even if the adversary is unable to fill the SIGD's NAT table, she might still attempt to deny legitimate uses access to the SIGD by saturating bottleneck links. We suggest two ways to mitigate this attack: (a) Use a separate SIGD link for DNS traffic to separate data traffic from SI traffic. Filters on last-hop routers will stop unwanted data from entering the data links. As soon as a name query resolution is successful, a filter can be installed on the router to allow traffic for that pair of IPs.

(b) If the bandwidth clogging attack uses DNS traffic links, it will create a furry of DNS requests, when the SSI and SIGD can handle the attack as suggested above under resource consumption attacks.

We note that SI does not provide any special tools for dealing with "flash crowds"—a sudden increase in legitimate user traffic to a particular destination, resulting in increased packet loss and congestion. One option is to employ a CDN-based approach, as recommended by Krishnamurthy*, et al.* [32] .

Finally, we observe that the following active defensive strategies are available using only inbound filtering at the destination organization. When the destination gateway believes it is under attack, in response to DNS queries from a suspicious source host, it can return the address of another suspicious source. The goal would be to keep the attack bots busy attacking themselves. A less drastic option is "reflection:" to return to the source its own address, or a standard loopback-address, so that the bot keeps attacking itself, thereby reducing consumption of network resources. Although we do not recommend such nontraditional strategies, their availability illustrates the flexible and powerful capabilities of SI.

For an alternate approach to DNS-flooding attacks, see Portcullis [44] [42].

6.2 Anonymity

After summarizing our assumptions and threat model, we informally explain how SI provides sender and receiver anonymity, and we explain two notions of unlinkability of communicants. We also suggest techniques for mitigating selected privacy attacks.

6.2.1 Assumptions and Threat Model

Listening to network traffic between gateways, the attacker aims to determine the identities of the communicants. The attacker also aims to link sender to receiver, or to link packets from the same communicant. We assume that the adversary can compromise routers and some number of hosts in each organization, for example, to listen to, inject, modify, or delete traffic. Furthermore, the adversary can spoof any IP address.

We consider two different notions of linkability. We shall use the phrase *crowds-linkability* [48] to mean that an adversary can determine that a particular sender is communicating with a particular receiver, which is a more damaging violation of privacy than simply determining a superset of senders and a superset of receivers. By *Chaum-linkability* [11] , we mean the adversary can determine that some pair of packets came from the same sender (or was sent to the same receiver), even if she cannot determine the communicant's identity.

We trust the source and destination SI servers (SSI, DSI) and gateways (SIGS, SIGD). We assume that DNS queries are encrypted to obscure the destination name and IP address, which can be accomplished with DNSCrypt [29] (DNSsec [7] does not encrypt DNS queries). Encrypting DNS queries requires some effort, including the use of PKI. This encryption will also introduce some additional delays caused by public-key operations which are relatively slow.

We consider two classes of adversaries. An *Internet adversary* intercepts only traffic flowing between organizational gateways. By contrast, a more powerful *global adversary* can additionally intercept traffic within organizational boundaries. Each of these adversaries can be passive or active.

6.2.2 Anonymity with Spread Identity

For each session between a source host and a destination host, the source gateway dynamically assigns a temporary pseudonym to the source host chosen as one of the routable IP addresses assigned to the source organization. Similarly, the destination DNS server assigns a temporary pseudonym to the destination host chosen as one of the routable IP addresses assigned to the destination organization. Thus, an eavesdropper listening to packets flowing through the Internet between the source and destination gateways learns only the source and destination domains; the eavesdropper cannot see the actual complete IP addresses of the source and destination hosts. Encrypting DNS requests and responses at gateways protects the established bindings of pseudonyms from the Internet eavesdropper. Anonymity provided by our system is similar to that in a two-node mixnet [11] .

For similar reasons, SI also achieves crowds-unlinkability of sender and receiver against a passive Internet adversary. As for Chaum-unlinkability, although a passive Internet adversary can link source and destination packets within any session, she cannot link packets between different sessions because the pseudonyms for each session are chosen independently. The blinding mechanism explained in Section 3.3 also provides for both types of unlinkability against a passive global adversary.

6.2.3 Mitigating Selected Privacy Attacks

Implementation artifacts raise potential threats to anonymity. We now briefly identify some passive and active traffic-analysis attacks and suggest how to mitigate them. For these attacks, we assume the attacker can listen to traffic both between organizational gateways and within the source and destination organizations. Against such a global adversary, it is important that the links between sender and SSI, and between DSI and receiver be encrypted.

Passive traffic analysis attacks include monitoring message lengths, bit patterns of encrypted payloads, clock skew, and packet flow rates and volume. Using such information, an attacker might try to follow packets as they travel from source to SIGS to SIGD to destination. We recommend deploying standard cryptographic and traffic-shaping defenses, including message padding, re-encryption of payloads, timing delays, and insertion of dummy traffic. However, as discussed by Back, *et al*. [9] , traffic shaping can cause network delays and inefficiencies.

Active traffic-analysis attacks include tagging packets, varying the amount of traffic (*e.g.,* in a reply), maliciously leaking information through a compromised host, and replaying legitimate traffic. Integrity checking of packets can help prevent tagging. To mitigate the other attacks, we recommend using standard traffic-shaping defenses.

6.3 Spread Identity Implemented at Destination Only

A partial form of SI can be implemented at the destination organization only. This variation does not achieve sender anonymity, but it does achieve receiver anonymity and most of the benefits of DDoS defense mechanisms described above. If an organization deploys full SI, it can still use destination-only SI when communicating with source hosts from organizations without SI. Receiver anonymity and DDoS defense capabilities provide immediate incentives for organizations to deploy SI.

Additionally, because the attacker must provide the SIGD with a valid source address in her initial DNS request (to receive the dynamic destination address), SI enhances trace-back capability.

7 Experimental Evaluation of the Spread Identity DDoS Defense Mechanism

In this section we present results of experiments using the ns-simulator and DeterLab test bed to measure the performance and effectiveness of the SI DDoS defense mechanism.

7.1 Ns-2 Simulations

Using the ns-2 discrete network event simulator [42] , we experimentally compared the effectiveness of our SI DDoS defense mechanism against capability and filter based approaches, file transfer success ratios, and file transfer times of legitimate users under bandwidth flooding DDoS attacks. We now describe our experiments and present and interpret our results [55] .

In 2008, Dawalbhakta [18] carried out separate ns-2 simulations to explore how SI affects connection establishment time and other performance metrics in comparison with a baseline network without SI. He found that adding SI caused no appreciable increase in connection establishment time, even when the network was under DDoS attack, for all networks he tested, which had up to 4,000 nodes.

7.1.1 Purpose, Scope, and Test Mechanisms

The purpose of our experiments is to measure how well our DDoS defense mechanism performs in comparison with some leading alternatives from the recent literature. In the ns-2 experiments, we focused solely on the file transfer success ratios and file transfer times of legitimate users; we did not measure communication delay or DNS traffic. For an assessment of these and other performance metrics, see Section 7.2.

Our test set of DDoS defense mechanisms comprised two forms of SI (SI+, SI), the corresponding capability mechanisms (TVA+, TVA [66] , Portcullis [44]) and filter mechanisms (StopIt [39] , AITF [6] , Pushback [40]). TVA+ extends TVA by using the Passport [38] authentication mechanism to avoid source IP address spoofing.

In our simulations, we installed destination IP address filters in different locations for SI+ (source and destination SI) and SI (destination only SI). SI+ installs destination filters at organizational gateways, whereas SI installs destination filters at ISP edge routers. In SI, where there is no source filtering and where the destination trusts no other organization, the destination protects the bottleneck last-hop link by installing filters at the ISP edge router. In SI+, the source and destination gateways trust each other and communicate with each other securely with SSL or HTTPS. In SI+, the source gateway performs destination IP address filtering to stop suspicious packets at the source, and the destination gateway also filters for added protection.

We have not compared our approach with overlay approaches such as WebSoS and Akamai. Such comparisons would be useful but would raise additional complications, including the need to try a variety of overlay network sizes. Also, we did not simulate any NAT exhaustion attacks because they can be mitigated by using CAPTCHAs or hashing puzzles to limit the rate of DNS requests.

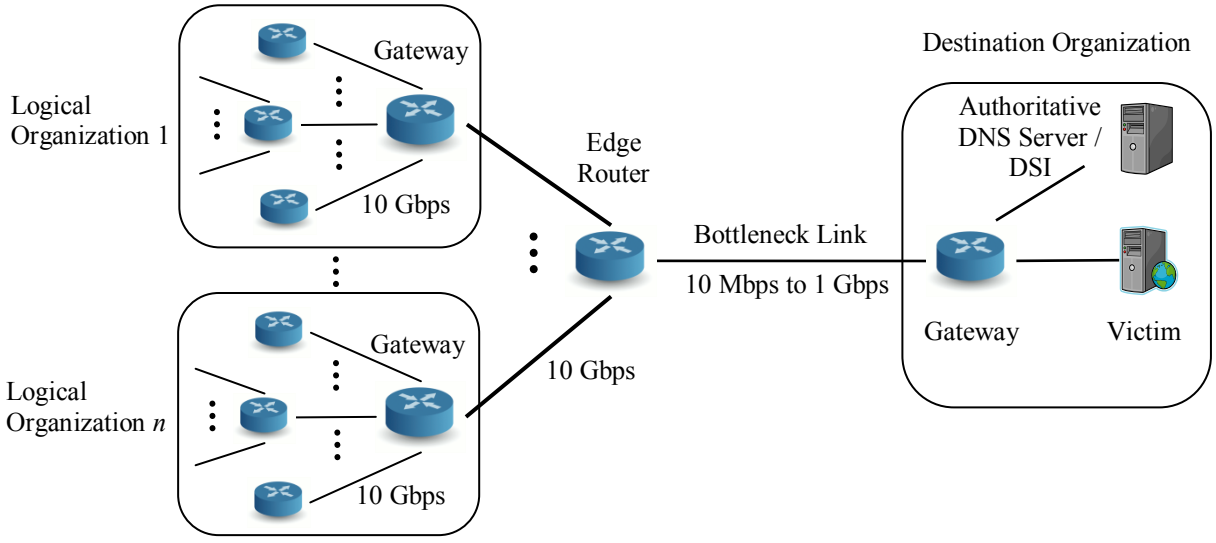


Figure 4: Experimental network topology for our ns-2 simulations. The destination organization contains an authoritative DNS server and the victim; it is connected to the Internet via a bottleneck link with capacity that can be varied from 10 Mbps to 1 Gbps. All routers that share the same gateway for connecting to the destination organization are viewed as one logical organization. Each router directly connected to the edge router acts as the organizational gateway of the corresponding logical organization.

7.1.2 Methods

Our experiments measured file transfer success ratios and file transfer times of legitimate users under bandwidth flooding DDoS attacks, which attempted to flood a bottleneck network link that connects an organization to the Internet. We varied three parameters: number of attackers (10^3 to 10^7), bandwidth of bottleneck link (10 Mbps to 1 Gbps), and file size (20 kb to 200 kb).[5] Each reported number represents the average of two trials.

For SI+ and SI, we implemented our methods in about 700 lines of C++ code. For the competing methods, we used code provided by Yang and Liu [39] .

Figure 4 shows our simulated network topology, wherein the destination organization is connected to the Internet using a bottleneck link of 10 Mbps. In some experiments, we alternatively used bottleneck bandwidths of 100 Mbps or 1 Gbps. We created logical organizations by grouping all routers that share the same gateway for connecting to the destination organization. Our experimental setup assumes that only the bottleneck link needs protection and that none of the routers are compromised. By contrast, some of the comparison strategies assume a stronger attack model.

Following a similar approach taken by Liu, *et al*. [39] , we created our network topology using BGP table dumps obtained from routeview servers [49] . For our simulations, each BGP dump contained 26 K *Autonomous Systems (AS)* and AS level paths in the Internet. Running on an Intel Core-2 duo CPU (2.0 GHz) laptop with 3.00 GB memory under the Ubuntu (Linux operating System), our ns-2 simulator handled up to about 2000 nodes. To create an AS-level topology, we randomly sampled 1505 AS-level paths from the BGP dumps.

We varied the number of attackers from 1000 to 10 million, assuming that 1 million attackers could completely occlude a 1 Gbps bottleneck link. To simulate more attackers than nodes supported by our ns-2 simulator, we used the same method deployed by Liu, *et al.* [39]: we modified attack packet interval times as follows

$$packet\ interval\ = attack\ packet\ size\ \cdot\ \frac{number\ of\ attacker\ nodes}{number\ of\ attackers \cdot\ bandwith\ per\ attacker}\ . \quad (1)$$

For example, with 60 attacker nodes out of 2000 network nodes, and assuming 10 kbps bandwidth per attacker, we simulated 1 million attackers sending attack packages of size 100 bytes. To do so, we set the attack packet interval to 4.8 microseconds.

As measures of effectiveness of each DDoS protection mechanism, we computed the success ratios of file transfers and file transfer times, where

$$success\ ratio\ = \frac{number\ of\ successful\ file\ transfers}{number\ of\ successful\ file\ transfers + number\ of\ file\ aborts}. \quad (2)$$

In our simulations, each of the 1455 legitimate users starts a TCP file transfer of a 20 Kb file. To limit the duration of the simulation, we set the file transfer timeout at 25 secs.

7.1.3 Results and Analysis

Figures 5a and 5b show observed success ratios and file transfer times for various DDoS protection mechanisms. In the first experiment, we set the bottleneck link at 1 Gbps, and legitimate users transferred 20 kb files. We varied the number of attackers from 1 K to 10 M.

[5] Throughout, K denotes 1000 ; M denotes 1 million; and G denotes 1 billion. Similarly, Kb denotes kilo-bits; KB denotes kilo-bytes; Gb denotes giga-bits; and GB denotes giga-Bytes.

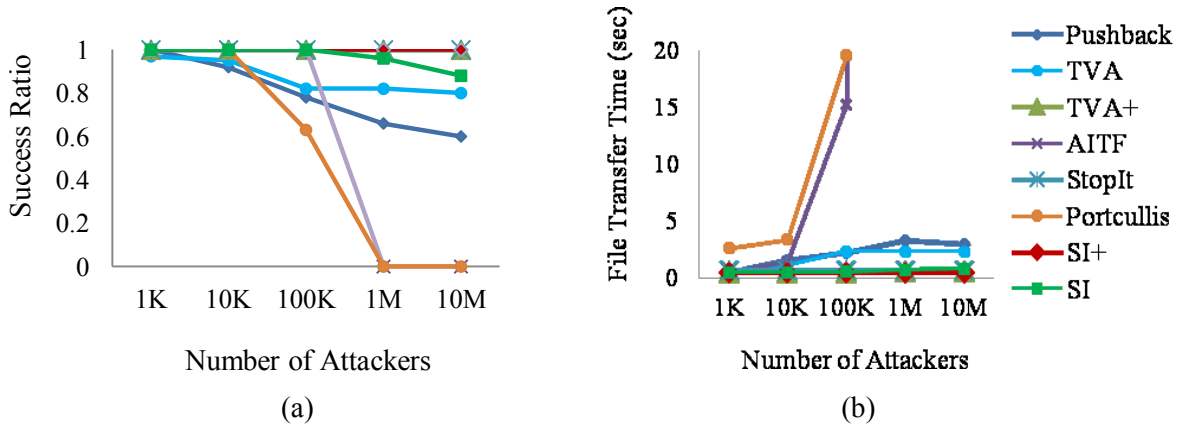


Figure 5: (a) File transfer success ratios and (b) file transfer times for *Spread Identity (SI, SI+)* and six other DDoS protection mechanisms under bandwidth colluding attacks, as the number of attackers varied from 1 K to 10 M. In these ns-2 simulations, each legitimate client attempted to transfer 20 Kb files with a 25 sec timeout. A 1 Gbps bottleneck link connected the destination organization to the Internet. Overall, our SI mechanisms performed similarly to StopIT and TVA+, and they outperformed these and all other defenses tested at 10 million attackers.

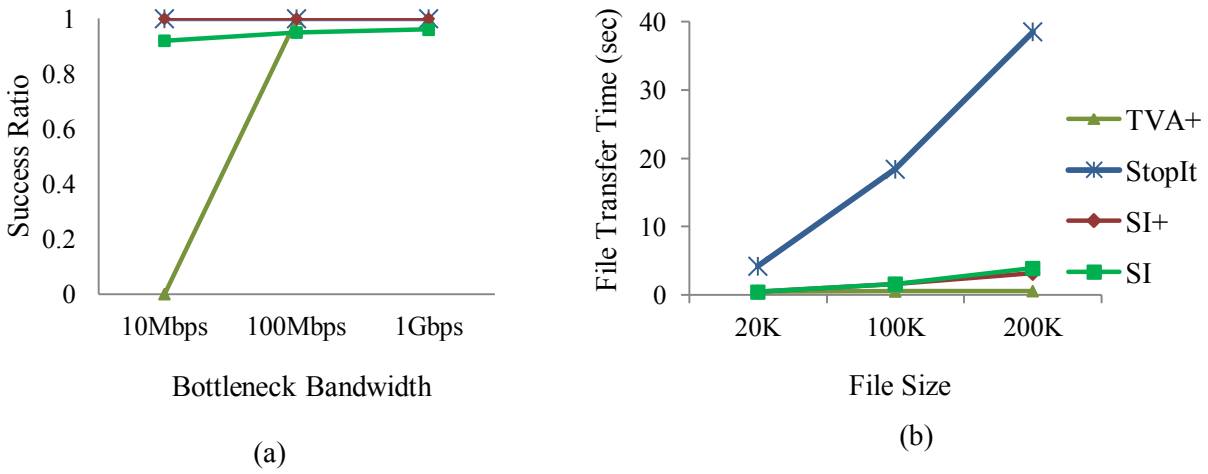


Figure 6: (a) File transfer success ratios and (b) file transfer times for *Spread Identity (SI, SI+)* and two other leading DDoS protection mechanisms under bandwidth colluding attacks, for various bottleneck capacities and file sizes. In these ns-2 simulations, 1 million attackers flooded a target organization connected to the Internet via a bottleneck link. Figure 6a shows the observed file transfer success ratios as the bottleneck capacity varied from 10 Mbps to 10 Gbps, and legitimate hosts attempted to transfer 20 Kb files with a 25 sec timeout. The success ratio for SI was at least as high as that for all other defenses tested. Figure 6b shows the observed file transfer times as the file size varied from 20 Kb to 200 Kb, with a 1 Gbps bottleneck link. SI had a slightly slower file transfer time than did TVA+ , but TVA+ had a significantly worse success ratio. SI had file transfer times at least as fast as those for all defenses other than TVA+.

In Figure 5a, the success ratio of Portcullis dropped suddenly after 100 K attackers with file transfer timeout at 25 sec. In Portcullis, however, legitimate users can complete their file transfers by waiting longer and solving difficult puzzles. As discussed by Liu, *et al.* [39], , AITF does not perform well for 1 million attackers because it uses a three-way handshake protocol for installing filters: under a large volume of attack traffic, SYN-ACK packets are lost. Success ratios of the TVA and pushback DDoS protection mechanisms were similar because they ensure per-path-fairness. TVA performs hierarchical fair queuing based on path identifier, and Pushback recursively installs filters near the source. Due to per-path-fairness, legitimate users suffer because they share the same path with the attackers. StopIt, TVA+, and our SI mechanism outperform the other DDoS protection methods.

In Figure 5a, the performance of SI dropped slightly below that of SI+ beginning at about 100 K simultaneous attackers. This experiment used a 1 Gbps bottleneck link connecting the destination to the ISP edge router. In SI, which installed destination filters at the ISP edge router, at about 100 K attackers the performance of the ISP edge router degraded under the heavy load, adversely affecting the file transfer rate of legitimate traffic. By contrast, in SI+, which filtered destination addresses at the source gateway, the ISP edge router did not encounter this heavy load.

Figure 6b compares the file transfer times for various file sizes. TVA+ was the most efficient method. The reason is that it does hierarchical fair queuing, thereby giving more preference to legitimate traffic than to attack traffic. With our SI architecture, destination IP address filtering stopped attack traffic immediately; hence, only legitimate traffic flowed through the bottleneck link.

In another set of experiments, we measured the file transfer success ratios of SI as we varied the bandwidth of bottleneck links connecting organizations to the Internet. In Figure 6a, StopIt performed well because attack traffic was completely blocked near the source. The success ratio of TVA+ was 0 for 10 Mbps of simulated bandwidth because attackers can flood the bottleneck links with capability request

packets. In these experiments, SI and SI+ performed similarly to StopIt, but better than the other competing mechanisms.

Overall, SI and SI+ performed at least as well as the leading alternatives StopIT and TVA+. Furthermore, SI and SI+ performed strictly better than TVA+ in the bottleneck experiment (Figure 6a), and SI and SI+ performed strictly better than StopIT in the file transfer time experiment (Figure 6b).

Limitations of our ns-2 experiments include limited network size and possible differences between a simulation and a real implementation. Also, we did not test overlay approaches and did not measure possible communication delays.

7.2 Experiments using the DeterLab Test Bed

Using the DeterLab test network [19] , we analyzed the performance of our implementation of SI and compared and contrasted it with a typical baseline single-NAT implementation. DeterLab is especially appropriate for our experiments because it is specifically designed to facilitate testing of large scale DDOS attacks.

We experimentally compared our SI implementation with a single-NAT baseline, with both systems using NetFilter for NATing. We generated 100,000 simultaneous connections (which is a typical load for medium-sized organizations) to test the baseline NAT and SI systems. Except for name resolution time, both systems perform similarly for all measured parameters. In particular, we found that there is no significant difference between end-to-end connection establishment time, and that there is up to about 90 μs difference in packet translation time at the gateways.[6] We also found that NAT table size cannot be considered in isolation: NAT table entries are a part of connection tracking entries which require 304 byte entries per connection for both our implemented baseline and SI systems.

Table 3 summarizes our experimental results.

[6] Throughout, ms denotes 1 millisecond = 10^{-3} sec, and μs denotes 1 microsecond = 10^{-6} sec.

Parameter	Single NAT Baseline	Spread Identity	Observation
input load (Connections handled per second)	90 maximum connections handled per second	35 maximum connections handled per second	Single NAT processed more connection requests per second than did SI because name resolution was slower in SI.
name resolution time	5 to 40 ms	90 to 2000 ms	Large difference in response times was result of serial nature of Netfilter NAT entry processing. Comparable lower bounds indicate that parallel NAT entry software with an optimized algorithm might result in comparable performance.
connection establishment time	1500 to 2500 us	1500 to 2500 us	After name resolution, TCP connection establishment time was similar.
TCP handshake packet #1 translation time	6 to 60 us	10 to 150 us	Almost double worst-case time difference between two setups. In both cases time varied more by input load, than by NAT table size.
TCP handshake packet #2 translation time	10 to 50 us	10 to 60 us	Similar translation time observed in both setups. Again, time varied more by input load, than by NAT table size. This time was representative of the rest of the IP packet flow.
memory requirement	equal to number of hosts / number of connections	min (private Host-blindfold IP pairs, pooled IP client pairs) or number of connections	In case of IP-tables[4], once the connection was established, NAT tables are not consulted, rather the connection tracking handles NAT data. As a result, NAT table processing was a part of connection. If connection tracking was used, each connection consumed 304 bytes of space per connection.

Table 3: Performance metrics for our implementations of Spread Identity and a baseline single-NAT system as measured on the DeterLab test bed using 100,000 simultaneous connections. Except for name resolution time, both systems performed similarly for all measured parameters.

As seen in Table 3, name resolution for SI took 90 to 2000 ms, compared to approximately 50 ms for the single-NAT baseline. The fastest name-resolution time of 90 ms occurred when there were few name resolution queries. We traced this performance degradation to a limited implementation of the Netfilter software. Specifically, Netfilter lacks the ability to install NAT entries in parallel. Consequently, when a

burst of queries arrived, the delay required to process the late arriving NAT entries included time to install all prior NAT entries from that burst.

Using NAT software (or hardware) that can install NAT entries in parallel would circumvent this unnecessary limitation imposed by Netfilter. Furthermore, such parallel processing will improve the performance of SI much more than it will help the single-NAT baseline. Therefore, we expect such parallel processing to yield name-resolution times for SI similar to those for the single-NAT baseline.

From these results, we conclude that the load patterns on the gateway from a typical single NAT system are similar to those resulting from the deployment of SI mechanisms.

For more details about our experimental methods, architecture, and results, see Appendix B and [31] .

8 Discussion

In this section we discuss the following issues: scalability of SI, defense against another strategic attack against the SI infrastructure, migrating to IPv6, and hardware NAT modules.

8.1 Scalability

We now examine the scalability of SI by considering the scalability of the SI infrastructure, how SI interacts with existing load-balancing mechanisms, and the scalability of organizational applications. We conclude that SI is highly scalable.

Scalability of the SI Infrastructure:

SI mechanisms have the following two attributes which support scalability: (a) SI mechanisms are based on existing, well understood and mature techniques such as double NAT and DNS services which are widely deployed today. Furthermore, the formats of the component mechanisms (*e.g.*, a DNS query) are left unchanged. Consequently, SI mechanisms are transparent to the existing infrastructure, backward-compatible, and incrementally deployable. (b) All the changes to the existing infrastructure that are required to implement SI are restricted to the routers and gateways at either end of the last-hop from the ISP to an organization. This means that the installed base of operating systems running the individual end-hosts, as well as the suites of end-user applications supported, are untouched. The routers and all other infrastructure (everything beyond the last-hop ISP router) in the core of the Internet is also untouched.

A factor negatively impacting scalability is degradation of certain performance metrics--especially connection establishment delay, overall communication delay, DNS traffic volume. Sections 4.1-4.4 and 7.2 give reasons and experimental data showing why these and other performance metrics should not be overly adversely affected when the SI mechanisms are properly introduced.

For these reasons, the SI infrastructure is highly scalable.

Interaction of SI with Existing Load-Balancing Mechanisms

SI can easily coexist with typical load-balancing mechanisms crucial to large services. Heavily used web services such as search engines (*e.g.,* Google, Yahoo, Bing) or news web sites (*e.g.,* CNN, C-Span, PBS, BBC) often deploy network services from vendors such as Akamai that specialize in providing load balancing, high bandwidth and high reliability and availability to their network customers. Typically, load balancing is achieved by physical replication of resources (such as web-servers), wherein a request for a web server can be dynamically directed to any physical server in the pool to balance the load.

SI mechanisms are independent of such load balancing mechanisms based on physical replications, and by themselves neither enhance nor degrade the performance of the load-balancing mechanisms. Therefore, SI can be implemented in various ways with existing load-balancing infrastructure. For example: (i) Load balancing might direct a request to the physically closest server. The SI mechanisms could be deployed within the domain of each such physical server, *i.e.,* in-series with and after the load

balancing. (ii) By contrast, SI could be inserted in-series with but before load balancing in an equally straightforward manner. For instance, Akamai could provide an arbitrary IP address under its control in response to a connection to a server and then decide to which physical server to send the request, provided Akamai's gateway creates and manages a proper NAT entry.

Scalability of Organizational Applications

SI infrastructure can easily adapt to increased demands created by organizational applications, such as web services, that may grow in size and complexity. Standard scalability mechanisms can be incorporated in the design of SI at any given site. For example, a load balancer can be implemented within the SSI's name resolution mechanism. SI algorithms are not affected by such a load balancer because SI's selection of an IP address does not depend on the application server's load.

To support scalability of architecture within an organization, we propose using standard mechanisms for handling increased infrastructure load. For instance, an organization might deploy multiple, physically separate gateways for added reliability. In this example, SI can be distributed across all such gateways using a method identical to the distributed firewall strategy proposed by Bellovin, *et al.* [30] .

As shown in Section 7.2, our experiments illustrate that load on a gateway (in terms of memory consumption and packet translation time) is largely unaffected by deployment of SI.

8.2 Defense against Another Strategic DDoS Attack

In this section we propose a novel defense against another strategic DDoS attack--one in which the adversary maliciously sends traffic to all IP addresses in the destination organization. Our defense assumes that SI is implemented at both source and destination organizations. It is based on *Fully Dynamic Identity Spreading (FDIS),* in which the source and destination gateways agree on a cryptographically-secure pseudorandom sequence of addresses to use for each packet in a session.

Such an attack is serious threat. With simple destination filtering alone, there is a risk of disconnecting the entire destination organization.

For the rest of this section, assume that SI is implemented at both source and destination organizations. For each session, the source and destination gateways agree on a shared secret session seed, exchanged as part of the payloads in the initial encrypted DNS query and response. From this seed, the gateways can generate a cryptographically-secure pseudorandom sequence of destination addresses to use in successive packets in the flow.

The destination gateway shares the seed with the last-hop ISP router (or proxy router). Address sequence filtering is then performed at the source gateway, last-hop ISP router, and destination gateway. Only flows with the correct pseudorandom sequence of addresses are permitted through. Out of order packet delivery is handled by a sliding window strategy.

A similar strategy can be used to defend against attacks on the Authoritative DNS name servers and SI gateways in each organization. Here the strategy is applied to the connectionless UPD protocol. Additionally, the source gateway first requires each host to solve a CAPTCHA puzzle. Only hosts passing the CAPTCHA puzzle will be allowed by the source gateway. Adversaries operating from a source organization without SI would be unable to learn a valid pseudorandom sequence of addresses and hence would be filtered out at the ISP last-hop routers.

All the changes are restricted to the gateways on either side of the last-hop links. The core routers in the Internet are left untouched. Likewise, the installed-base of end-host operating systems and user applications are also untouched. As a result our proposed address-hopping method is as scalable and incrementally deployable, as are all other SI methods that use a coarser granularity.

8.3 Migration to IPv6

In this section we discuss two issues pertaining to SI and the migration from Internet Protocol IPv4 to IPv6. First, because SI is based on well-known and reliable mechanisms available under IPv4, and because all of these mechanisms are also supported under IPv6, migration from IPv4 to to IPv6 will not present difficulties for SI implementations built under IPv4.

Second, we explore implications for SI of the huge 128-bit wide address space that comes with IPv6. Under IPv6, it might be possible to assign a new address for each communication. Some people might believe that SI is not needed if a new address can be allocated for each communication. We now explain why SI is still helpful, and we point out some dangers of large address spaces.

Unique addresses undermine anonymity. For strong anonymity, a large number of real entities should dynamically share a much smaller pool of apparent identities.

With a very large available address space, addresses will likely be assigned in a hierarchical fashion to reduce the size of the required routing tables. Maintaining a small pool of externally routable addresses within an organization helps keep routing tables small in core routers, is especially important for IPv6.

As a dynamic address indirection mechanism, SI provides an effective way to manage relationships between apparent and real addresses, which is especially needed for the huge address space of IPv6. The strength of SI stems in part from its use of dynamic indirection, which has proven enormously powerful in other contexts including virtual memory.

8.4 Hardware NAT Modules

The distinguishing feature of SI is that the NAT mapping is deliberately made to appear as *many-to-many* when viewed from either side of the entity performing the NAT (*i.e.*, the SI gateway). There would be significant security advantages to build and deploy special-purpose NATing hardware to support this core SI functionality. For anonymity, the most crucial information is the dynamic binding of addresses. This crucial information could be generated and used on special-purpose hardware, erased immediately after each session, and never divulged anywhere except as needed by the SI protocol in encrypted form to other such boxes. Such hardware can support Tor-like anonymity without the long delays associated with Tor [21] [20]. Using such hardware helps isolate and reduce the required trusted base for anonymous network communications. By contrast, today most organizations run NAT software (*i.e.* IPTable) on powerful commodity general-purpose hardware, on clusters of stateless devices, or in the cloud.

8.5 Limitations

Throughout the paper we have pointed out specific limitations of SI. We now summarize the main limitations.

The efficacy of SI depends on the number of real, externally-routable addresses available for pooling. If the number of addresses in the pool is small, the level of DoS protection, as well as the level of anonymity that can be achieved is limited. If SI is deployed at the ISP level, there will likely be a sufficiently large number of addresses available to achieve high levels of anonymity and a robust DoS defense. The attacker still learns the source and destination organizations. The severity of this anonymity loss depends on the context and organizations. When the organizations are ISPs, this loss will typically be rather small.

For simplicity, we have assumed (i) routers are not compromised, and (ii) only the link between the destination ISP and the destination organization may be congested by attackers and therefore it is the only link that needs DoS protection. These assumptions might make the significance of SI seem somewhat limited. Some alternate DoS strategies assume stronger attack models. It is possible, however, to filter

unwanted flows at any intermediate router and/or to protect any link. Specifically, SI makes it possible to filter flows based on source address only, destination address only, or (source, destination)-address pairs that are dynamically created.

There can be tension between the degree of anonymity of the source and the level of DoS protection at the destination. Selecting a random source address hides the real identity of a host, preventing the destination from remembering the addresses of past attackers. When source and destination SI gateways cooperate, this tension can be managed in reasonable ways. For example, when under attack, the destination SI gateway can signal the source SI gateways to issue CAPTCHA challenges to filter out automated traffic at the source organization. Doing so reduces attack damage caused by consuming bandwidth and other resources. In addition to filtering automated traffic, each source SI gateway can be instructed also to rate-limit the DNS queries it sends out to the destination.

Filtering traffic by using CAPTCHA challenges has several drawbacks. Some legitimate communications are automated (*e.g.,* web crawler for indexing). CAPTCHA challenges do not always block automated communications, and they increase latency by adding another round of packet exchanges. They also require software changes to edge routers and end hosts, and to communication protocols when a session involves intermediaries such as HTTPS proxies.

There are some implications of the choice of session length. Shorter sessions offer greater anonymity but require more DNS work.

A potential threat to SI is an attack that floods the DNS server with queries. Section 6.1.5 discusses defenses to this strategic attack, including rate-limiting strategies based on the capacities of the DNS server. These defenses, however, will likely interfere with some legitimate traffic.

Finally, another potential threat to SI is an attack that floods all addresses in the destination organization. Section 8.2 discusses defenses to this strategic attack, including FDIS. These defenses, however, will likely interfere with some legitimate traffic.

9. Conclusion

We have presented and experimentally evaluated a new *Spread Identity (SI)* architecture to provide network anonymity and enhanced DDoS defense based on filtering with low impact on legitimate traffic. It also provides a limited traceback capability. This perimeter defense technology leverages trusted organizational gateways and applies fundamental concepts of dynamic bindings, address pooling, indirection, and pseudonyms. The main deployment costs are modifying organizational gateways (and optionally edge routers) and increasing DNS traffic, but no changes are required to Internet routers. Furthermore, and unlike overlay approaches, SI does not slow down communication traffic appreciably.

Our simulations demonstrate that the approach is viable and that it works as well as existing approaches based on filtering- and capability-based DDoS mechanisms. Specifically, using the ns-2 simulator, we demonstrate that file transfer success ratios for our SI DDoS protection mechanism are similar to those of filter- and capability-based approaches, with lower file transfer times than for filter-based approaches. Experiments with real networks using DeterLab further confirm that our system is technically feasible and results in similar load patterns on the gateway, for all measured parameters except for name resolution time, to those from a typical single NAT system.

Future work includes: (i) investigating the effectiveness of SI at the ISP level, where large address spaces, high bandwidth links, and high-speed routers offer substantial potential, and (ii) developing standards for SI (*e.g.,* through Internet RFCs).

Although we focus on SI at Internet organizational gateways, the concept is applicable much more broadly to many applications with large name spaces. Moreover, the principle of identity spreading via many-to-many mappings of apparent names, is a powerful fundamental principle and technique for effecting anonymity and DDOS defense in communication networks. *Identity spreading*, together with the more commonly recognized principles of *dynamic indirection* and *end-to-end communication* [55] , form a three-legged foundation for robust anonymous network communications.

Acknowledgments

We are grateful to Xiaowei Yang and Xin Liu for providing their implementations of filter-based and capability-based DDoS protection mechanisms, which we used in our ns-2 simulations. Josiah Dykstra, Russ Fink, John Pinkston, and the referees offered helpful comments.

References

[1] Akamai SiteShield Module <*http://www.akamai.com/dl/feature_sheets/Akamai_Site_Shield.pdf*>.

[2] D. Andersen, Mayday, Distributed filtering for Internet services, *in: 4th Usenix Symposiumon Internet Technologies and Systems (*USITS 2003), pp. *pp. 31-42*

[3] T. Anderson, T. Roscoe, D. Wetherall, Preventing Internet denial-of-service with capabilities, *in: Proceedings of Hotnets-II* (November 2003), pp. 39–44.

[4] O. Andreasson, IPTables Tutorial, <*http://www.faqs.org/docs/iptables/tcpconnections.html*>.

[5] Anonymizer. <*http://www.anonymizer.com/*>.

[6] K. Argyraki and D. Cheriton; Active Internet Traffic Filtering: Real-Time Response to Denial-ofService Attacks, *in Proceedings of USENIX 2005, pp 135–148*

[7] G. Ateniese, S. Mangard, A new approach to DNS security (DNSSEC), in: *Proceedings of the 8th ACM Conference on Computer and Communications Security* (November 2001), pp. 86–95.

[8] D. O. Awduche, L. Berger, D. Gan, T. Li, V. Srinivasan, G. Swallow, RSVP-TE: Extensions to RSVP for LSP Tunnels, *Internet Draft draft-ietf-mpls-rsvp-lsp-tunnel-08.txt, IETF* (February 2001).

[9] A. Back, U. Möller, A. Stiglic, Traffic analysis attacks and trade-offs in anonymity providing systems, in: *Proceedings of the 4th international Workshop on Information Hiding* (April 2001), pp. 245–257.

[10] O. Berthold, A. Pfitzmann, R. Standtke, The disadvantages of free MIX routes and how to overcome them, in *International Workshop on Designing Privacy Enhancing Technologies: Design Issues in Anonymity and Unobservability* (2001), pp. 30–45.

[11] D. Chaum, Untraceable electronic mail, return addresses, and digital pseudonyms, *Communication of the ACM (24) (1981), pp. 84–90.*

[12] Y. Chen, A. Bilas, S. Damianakis, C. Dubnicki, K. Li, UTLB: A mechanism for address translation on network interfaces, in: *Proceedings of the eighth international conference on Architectural support for programming languages and operating systems* (1998), pp. 193–204.

[13] D. R. Cheriton, M. Gritter. TRIAD: A new next generation internet architecture.

<*http://www-dsg.stanford.edu/papers/triad/triad.html*>.

[14] J. Clark, P. Oorschot, C. Adams, Usability of anonymous Web browsing: An examination of Tor interfaces and deployability, in: *Proceedings of the 3rd Symposium on Usable Privacy and Security* (July 2007), pp. 41–51.

[15] W. Dai. Pipenet 1.1. *Usenet post* (August 1996). <*http://www.freehaven.net/anonbib/cache/pipenet10.html*>.

[16] G. Danezis, C. Diaz. A survey of anonymous communication channels. *Technical Report MSRTR-2008-35, Microsoft Research* (January 2008).

[17] G. Danezis, R. Dingledine, N. Mathewson, Mixminion: Design of a type III anonymous remailer protocol, *in: Proceedings of the 2003 IEEE Symposium on Security and Privacy* (May 2003), pp. 2–15.

[18]] A. Dawalbhakta, Demonstrate the application of the concept of Spread Identity to enhance security on Internet against DDoS attacks, *MS Thesis, Dept. of CSEE, University of Maryland, Baltimore County* (2008).

[19] DETER research project, http://www.isi.edu/deter/ or http://deter-project.org/

[20] C. Díaz, S. Seys, J. Claessens, B. Preneel, Towards measuring anonymity, in: *Proceedings of the 2nd International Conference on Privacy Enhancing Technologies* (April 2002), pp. 54–68.

[21] R. Dingledine, N. Mathewson, P. Syverson, Tor: The second-generation onion router, *in: Proceedings of the 13th USENIX Security Symposium* (2004), pp. 303–320.

[22] D. Dittrich, The DoS Project's 'trinoo' distributed denial of service attack tool, *Technical report, University of Washington,* 2000.<*http://staff.washington.edu/dittrich/misc/trinoo.analysis.txt*>.

[23] H. Federrath. JAP— Anonymity & privacy. <*http://anon.inf.tu-dresden.de/index_en.html*>.

[24] R. Fink, M. Brannigan, S. Evans, A. Almeida, S. Ferguson, Method and apparatus for providing adaptive self-synchronized dynamic address translation, US Patent 7,043,633 (May 9, 2006).

[25] M. Freedman, R. Morris, Tarzan: A Peer-to-Peer Anonymizing Network Layer, in: *Proceedings of the 9th ACM Conference on Computer and Communications Security* (2002), pp. 193–206.

[26] Gleeson, *et al.*, IP Based Virtual Private Networks, *RFC 2764*, February 2000.

http://trac.tools.ietf.org/html/rfc2764

[27] C. Gulcu, G. Tsudik, Mixing E-mail with Babel, *Proceedings of the Symposium on Network and Distributed System Security* (February 1996), pp. 2–16.

[28] E. Hammer-Lahav, Ed., The OAuth 1.0 Protocol, *RFC 5849, Internet Engineering Task Force,* April 2010. *http://tools.ietf.org/html/rfc5849*

[29] Introducing DNScrypt (Preview Release). <*http://www.opendns.com/technology/dnscrypt/*>

[30] S. Ioannidis, A. Keromytis, S. Bellovin, J. Smith, Implementing a Distributed Firewall, in: *Proceedings of the ACM Conference on Computer and Communications Security (CCS),* (November 2000) 190–199.

[31] N. Joshi, Experimental evaluation and implementation of the Spread Identity framework, *MS Thesis, Dept. of CSEE, University of Maryland, Baltimore County* (December 2011).

[32] J. Jung, B. Krishnamurthy, M. Rabinovich, Flash crowds and denial of service attacks: Characterization and implications for CDNs and Web sites, in: *Proceedings of the 11th International Conference on World Wide Web* (May 2002) 293–304.

[33] C. Kaufman, R. Perlman, M. Speciner, *Network Security: Private Communication in a Public World, Prentice-Hall,* New Jersey (1995).

[34] A. Keromytis,V. Misra, D. Rubenstein, SOS: An architecture for mitigating DDoS attacks, *IEEE Journal on Selected Areas in Communications* 22 (1) (January 2004), pp. 176–188.

[35] D. Kewley, R. Fink, J. Lowry, and M. Dean, Dynamic approaches to thwart adversary intelligence gathering, in: *Proceedings of DISCEX II, vol. 1, IEEE Computer Society* (2001), pp. 176–185 .

[36] K. Lakshminarayanan, D. Adkins, A. Perrig, I. Stoica, Taming IP packet flooding attacks, in: *Proceedings of SIGCOMM* (January 2004), pp. 45–50.

[37] J. Lemon, Resisting SYN flood DoS attacks with a SYN cache, in: *Proceedings of the USENIX BSD Conference 2002* (February 2002), *<http://www.usenix.org/publications/library/proceedings/bsdcon02/lemon.html>*

[38] X. Liu, A. Li,X. Yang, D. Wetherall, Passport: Secure and adoptable source authentication, in: *Proceedings of the 5th USENIX Symposium on Networked Systems Design and Implementation* (April 2008), pp. 365–378.

39] X. Liu, X. Yang, Y. Lu, To filter or to authorize: Network-layer DoS defense against multimillion-node botnets, *in: Proceedings of the ACM SIGCOMM 2008 Conference on Data Communication* (August 2008), pp. 195–206.

[40] R. Mahajan, S. Bellovin, S. Floyd, V. Paxson, S. Shenker, Controlling high bandwidth aggregates in the network, *ACM Computer Communications Review* 32 (3) (July 2002), pp. 62–73.

[41] M. Mazzola, T. Edsall, L. Cafiero, Address translation mechanism for a high-performance network switch, *U.S. Patent 5740171* (April 1998).

[42] S. McCanne, S. Floyd, The Network Simulator NS-2. <http://www.isi.edu/nsnam/ns/>.

[43] I. Molloy, J. Li, N. Li, Dynamic virtual credit card numbers, in: *Financial Cryptography 2007*, LNCS 4886 (2007), pp. 208–223.

[44] B. Parno, D. Wendlandt, E. Shi, A. Perrig, B. Maggs, Y.-C. Hu, Portcullis: Protecting connection setup from denial-of-capability attacks, *in: Proceedings of SIGCOMM* (2007), pp. 289–300.

[45] A. Pfitzmann, B. Pfitzmann, M. Waidner, ISDN mixes: Untraceable communication with very small bandwidth overhead, in: *GI/ITG Conference on Communication in Distributed Systems* (February 1991), pp. 451–463.

[46] A. Pfitzmann, M. Waidner, Networks without user observability--Design options, *in: Proceedings of EUROCRYPT 1985, Springer-Verlag, LNCS 219* (1985). *http://www.semper.org/sirene/publ/PfWa_86anonyNetze.html*

[47] D. S. Phatak, Spread-Identity mechanisms for DOS resilience and security, in: *Proc. of the IEEE International Conference on Security and Privacy for Emerging Areas in Communications Networks* (*Sept. 2005, Athens, Grerece*), pp. 23–34.

[48] M. Reiter, A. Rubin, Anonymous Web transactions with Crowds, *Communication of the ACM* 42(2) (February 1999), pp. 32–48.

[49] Route Views Archive. *<http://archive.routeviews.org/oix-route-views/>*.

[50] J. H. Saltzer, D. P. Reed, and D. D. Clark, End-to-end arguments in system design, *ACM Transactions on Computer Systems (TOCS),* vol. 2, issue 4 (Nov. 1984), pp. 277–288

[51] V. Sekar, N. Duffield, O. Spatscheck, J. Merwe, H. Zhang, LADS: Large-Scale automated DDOS detection system, *in Proceedings of the Annual USENIX Technical Conference* (June 2006), pp. 171–184 .

[52] E. Shi, I. Stoica, D. Andersen, A. Perrig, OverDoSe: A generic DDoS protection service using an overlay network, *Technical Report CMU-CS-06-114, Carnegie Mellon University* (2006).

[53] P. Srisuresh, IP network address translator (NAT) terminology and considerations, *RFC 2663, Internet Engineering Task Force, August 1999. ftp://ftp.ietf.org/rfc/rfc2663.txt*

[54] P. Srisuresh, DNS extensions to network address translators (DNS_ALG), *RFC 2694, Internet Engineering Task Force, September 1999.* *ftp://ftp.ietf.org/rfc/rfc2694.txt*

[55] B. Sonawane, Spread Identity: A new dynamic address remapping mechanism for anonymity and DDoS Defense, *MS Thesis, Dept. of CSEE, University of Maryland, Baltimore County* (December 2010).

[56] A. Stavrou, D. Cook, W. Morein, A. Keromytis, V. Misra, D. Rubenstein, WebSOS: An overlay-based system for protecting Web servers from denial of service attacks, *Computer Networks: The International Journal of Computer and Telecommunications Networking* 46 (5) (2005), pp. 781–807.

[57] A. Stavrou, A. Keromytis, Countering DoS attacks with stateless multipath overlays, in: *Proceedings of the 12th ACM Conference on Computer and Communications Security* (2005), pp. 249–259.

[58] A. Stavrou, A. D. Keromytis, J. Nieh, V. Misra, D. Rubenstein, MOVE: An end-to-end solution to network denial of service, in: *Proceedings of the ISOC Symposium on Network and Distributed System Security (SNDSS)*, (February 2005), pp. 81–96.

[59] I. Stoica, D. Adkins, S. Zhuang, S. Shenker, S. Surana, Internet indirection infrastructure, *in: Proceedings of SIGCOMM* (August 2002), pp. 73–86.

[60] I. Stoica, R. Morris, D. Karger, M. Kaashoek, H. Balakrishnan, Chord: A scalable peer-to-peer lookup service for internet applications, *in: Proceedings of the 2001 Conference on Applications, Technologies, Architectures, and Protocols For Computer Communications, SIGCOMM '01* (2001), pp. 149–160.

[61] G. Su, J. Nieh, Mobile Communication with virtual network address translation, *Technical Report CUCS-003-02, Department of Computer Science, Columbia University* (February 2002).

[62] L. von Ahn, M. Blum, N. Hopper, J. Langford, CAPTCHA: Using hard AI problems for security,*in: Proceedings of Eurocrypt 2003*, *LNCS 2656* (2003), pp. 294–311.

[63] J. Wang, X. Liu, A. A. Chien, Empirical study of tolerating denial-of-service attacks with a proxy network, in: *Proceedings of the 14th USENIX Security Symposium* (August 2005), pp. 51–64.

[64] H. Wang, H. Xie, L. Qiu, A. Silberschatz, Y.R. Yang, Optimal ISP subscription for Internet multihoming: Algorithm design and implication analysis, in: *Proceedings of the 24th Annual*

Joint Conference of the IEEE Computer and Communications Societies 4 (March 2005), pp. 2360–2371.

[65] A. Yaar, A. Perrig, D. Song, SIFF: A stateless Internet flow filter to mitigate DDoS flooding attacks, in: *IEEE Symposium on Security and Privacy* (May 2004), pp. 130–143.

[66] X. Yang, D. Wetherall, T. Anderson, TVA: A DoS-limiting network architecture, *IEEE/ACM Transactions on Networking* 16(6) (December 2008), pp. 1267–1280.

[67] Ylonen and Lonvick, SSH Protocol Architecture, *RFC 4251, Internet Engineering Task Force,* January 2006. *http://tools.ietf.org/html/rfc2767*

Appendix A: List of Abbreviations and Acronyms

AS	Autonomous Systems
BGP	Border Gateway Protocol
CDN	Content Delivery Network
DDoS	Distributed Denial of Service
DNS	Domain Name Server
DSI	Destination Spread Identity Server
DSI*e*	external IP address of the DSI
DSI*i*	internal IP address of the DSI
FDIS	Fully Dynamic Identity Spreading
IP	Internet Protocol
IPv4	Internet Protocol version 4
IPv6	Internet Protocol version 6
ISP	Internet Service Provider
NAT	Network Address Translation
PKI	Public Key Infrastructure
SI	Spread Identity
SIGD	Destination Spread Identity Gateway
SIGS	Source Spread Identity Gateway
SSI	Source Spread Identity Server
SSI*e*	external IP address of the SSI
SSI*i*	internal IP address of the SSI
TCP	Transmission Control Protocol
TTL	Time To Live
VPN	Virtual Private Network

Appendix B: Performance Comparison of Spread Identity vs. Single NAT

This section presents more detailed methods and results from our DeterLab experiments that we carried out comparing the performance of our *Spread Identity (SI)* gateway with that of a baseline single-NAT gateway. We found, that for all parameters measured except name resolution time (see Section 7.2), the SI gateway showed similar load patterns as did the single-NAT gateway. For more details, see Joshi [31] .

For SI, DNS name resolution took considerably more time than it did for the baseline gateway. First, with SI, a single name resolution requires two name queries (one for resolving IP of SSI/DSI, one for resolving the IP of the actual host). Second, resolving the actual host is complicated by the need to find appropriate blindfolded and public IP pairs that are eligible for the IPs of the given private host and public client. We implemented name resolution as a single query to the DSI of the SI gateway.

As explained in Section 7.2, the increased time of name resolution for SI (approximately 90 to 2000 ms compared with 50 ms for the single-NAT baseline) is due primarily to limitations of the existing Netfilter implementation. A parallel implementation of Netfilter should yield similar name-resolution times for SI and the baseline.

Figures B1-B3 illustrate selected results from our DeterLab experiments measuring connection establishment time, TCP handshake packet #1 forwarding time, and TCP handshake packet #2 forwarding rate. Figures B4-B5 describe our SI gateway and connection topology, and pseudocode for Algorithm IPClient documents part of our experimental procedures.

Data underlying these figures were generated as follows. We created 100 K connections. Each subnet was a single machine with aliased IPs. Each machine had up to 15 threads running (each acting as a single client). Each of these threads looped on a set of IPs and executed Algorithm IPClient below. We ran this experiment on the SI gateway and on the single-NAT baseline. Values reported in Section 7.2 were also computed from the same data.

Connection Establishment Time

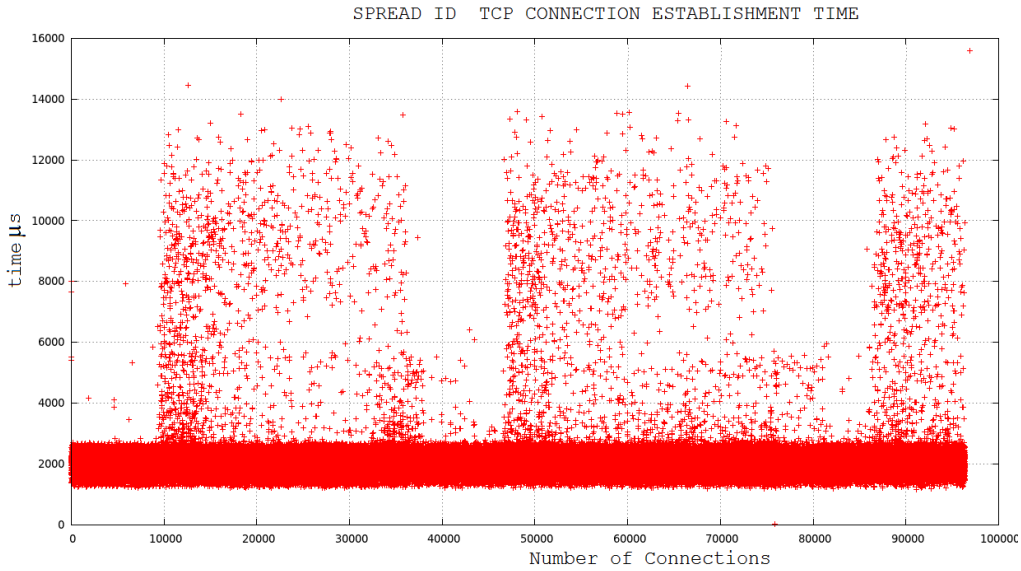


Figure B1a: Spread identity connection establishment time. On average the connection establishment time was between 1000 to 3000 microseconds. Each of the three bands marks the beginning of a subnet: increasing the number of connection requests per unit time increased connection establishment time. NAT did not increase connection setup time: increasing the number of connections did not increase connection establishment time.

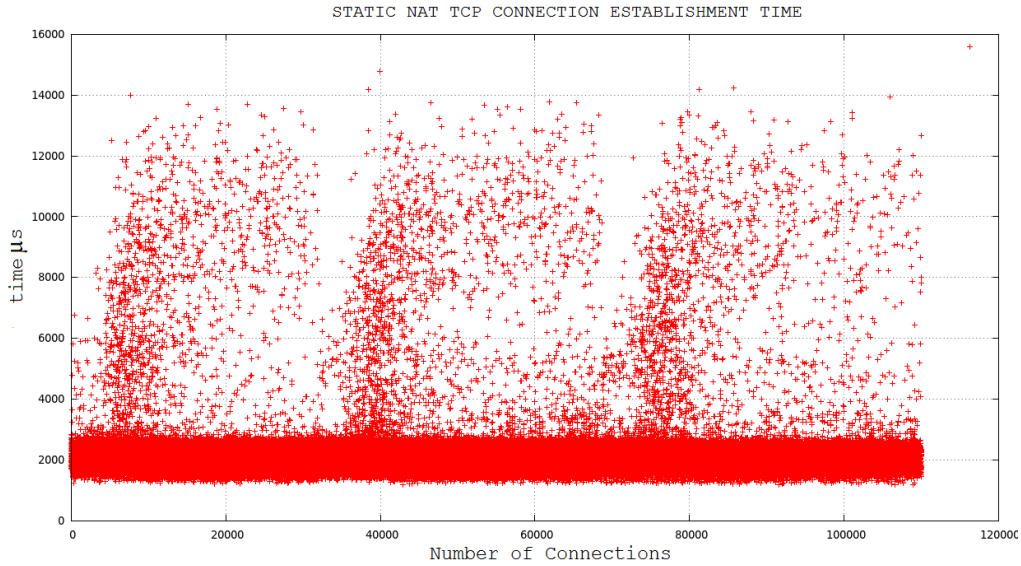


Figure B1b: Baseline Connection establishment time with single NAT. The average the connection establishment time was between 1000 to 3000 microseconds, the same as that in Figure B1a. Figure B1b is similar to Figure B1a. The bands mark the beginning of the subnets. In comparison with Figure B1a, the bands are shifted left slightly, because the subnets started earlier with the baseline than they did with SI.

TCP Handshake Packet #1 Forwarding Time

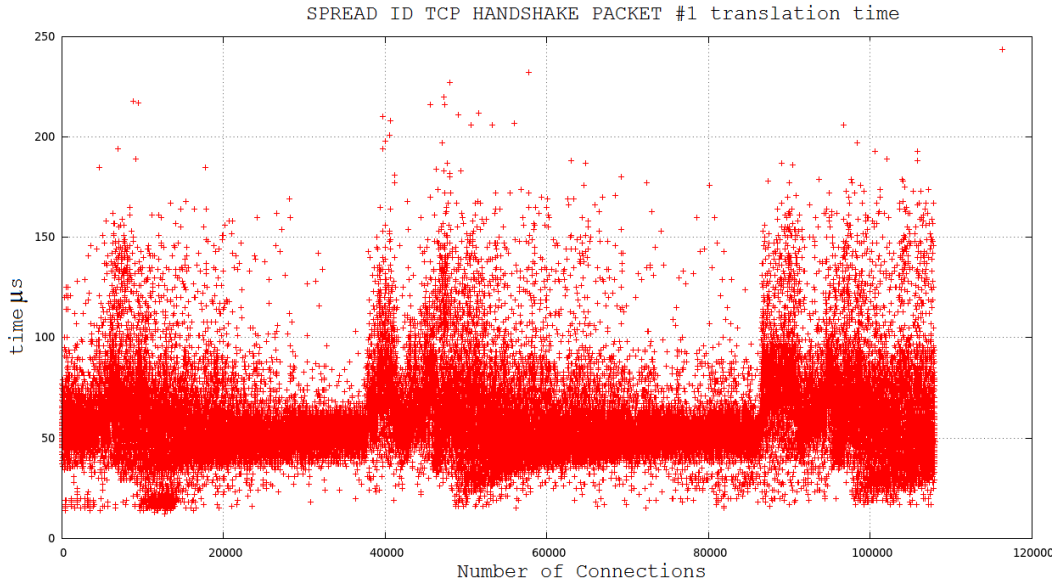


Figure B2a: Time taken by the first TCP handshake packet to traverse through the SI Gateway. Each band marks the beginning of a subnet, when a burst increase in connection requests slows down the system. The traversal time did not increase appreciably with more connections.

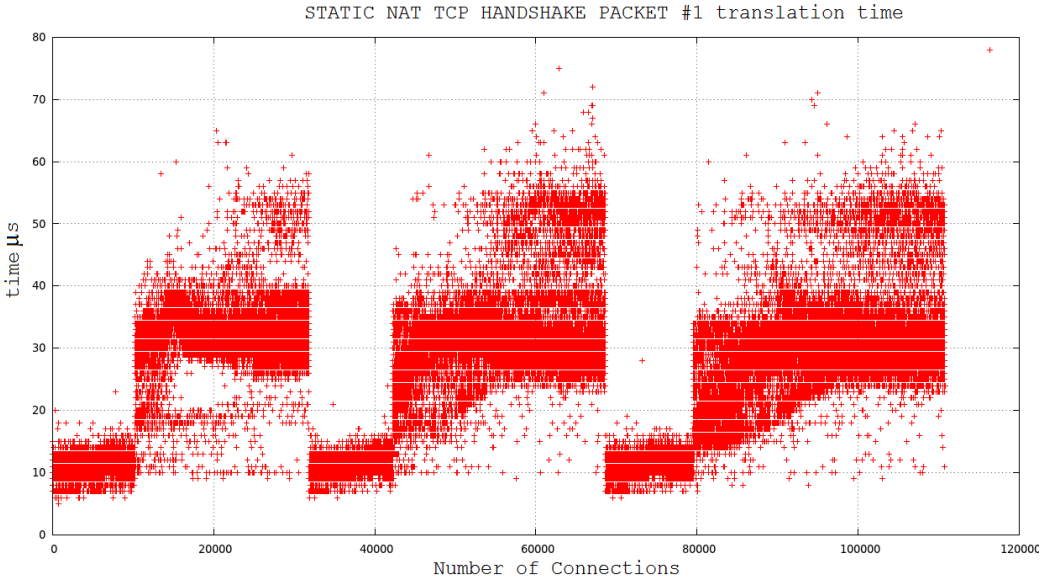


Figure B2b: Time taken by the first TCP handshake packet to traverse the static NAT gateway. Similar to Figure B2a but with faster times.

TCP Handshake Packet #2 Forwarding Rate

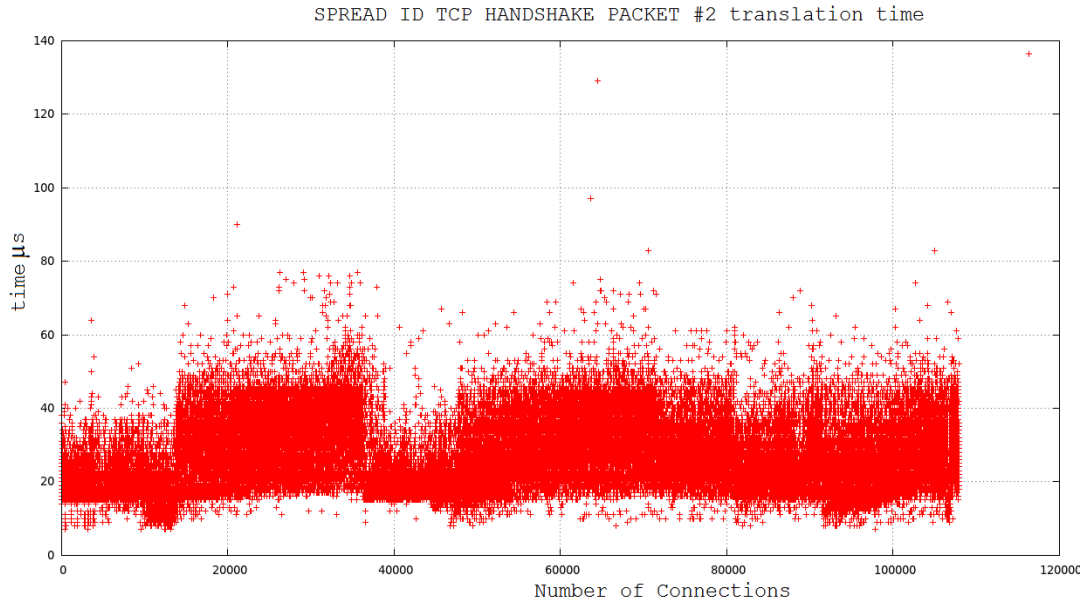
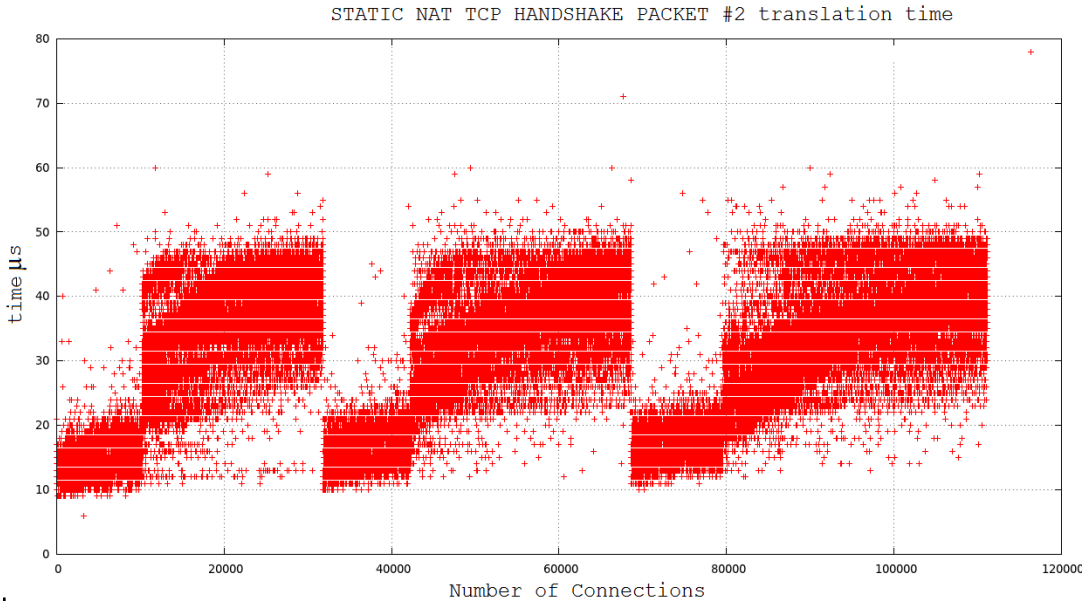


Figure B3a: Time taken by the second TCP handshake packet to traverse the SI Gateway, which time is representative of the remaining packets. As with Figures B1-B2, each band marks the beginning of a subnet, and times did not increase appreciably with more connections.



.

Figure B3b: Time taken by the second TCP handshake packet to traverse the static NAT gateway. Similar to Figure B3a.

Experimental Setup and Procedures

We summarize our experimental setup and procedures by explaining the SI gateway (Figure B4), giving pseudocode for Algorithm IPClient, and showing the connection topology (Figure B5).

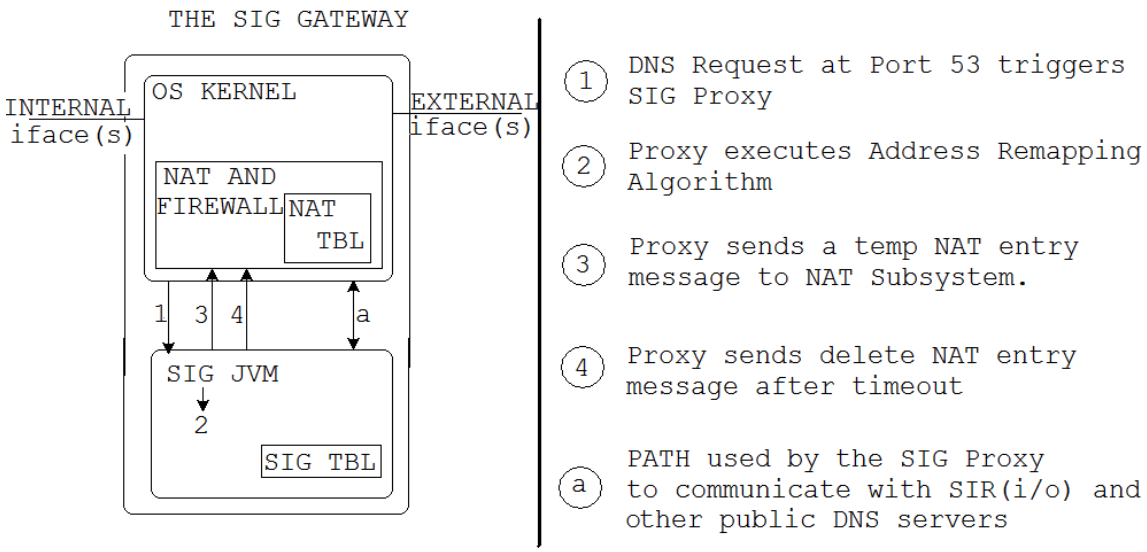


Figure B4: Summary of the structure and behavior of the spread identity gateway used in our DeterLab experiments. The left side of the figure summarizes the components of the gateway. The right side of the figure describes interactions of these components. Further optimizations (*e.g.,* parallel NATing) are possible and those can be expected to improve name resolution performance.

Algorithm IPClient (cIP, dev, URL, stdout)

This algorithm tries to connect to the given URL and measures both the time required to perform the DNS request, and the time taken to establish the connection, Output is stored into files, which are post-processed together with the internal- and outer-interface logs to produce graphs.

[sIP is server IP, cIP is ClientIP, dev is interface/link name, URL is URL, stdout is standard output stream]

[Dig(ip, dev, URL) : does a Namequery on URL using ip, dev as source address]

[Connect(cip, dev, sip) : Active connect from cIP to sIP]

[write(stream, record) : write record to stream]

[getIP(receiveData(h)) : Receive data from given handle, and then extract IP]

Begin

1. {sIP,queryTime} = Dig(cIP, dev, URL);
2. time = startTime();

   handle=Connect (cIP, dev, sIP);

   time = endTime() – time;
3. if (handle == INVALID)

   {

   write(stdout,{-1, time, queryTime, FAIL});

   return;

   }
4. IP=getIPFrom(receiveData(handle));

   write(stdout,{handle, time, queryTime, SUCCESS});

   return;

End

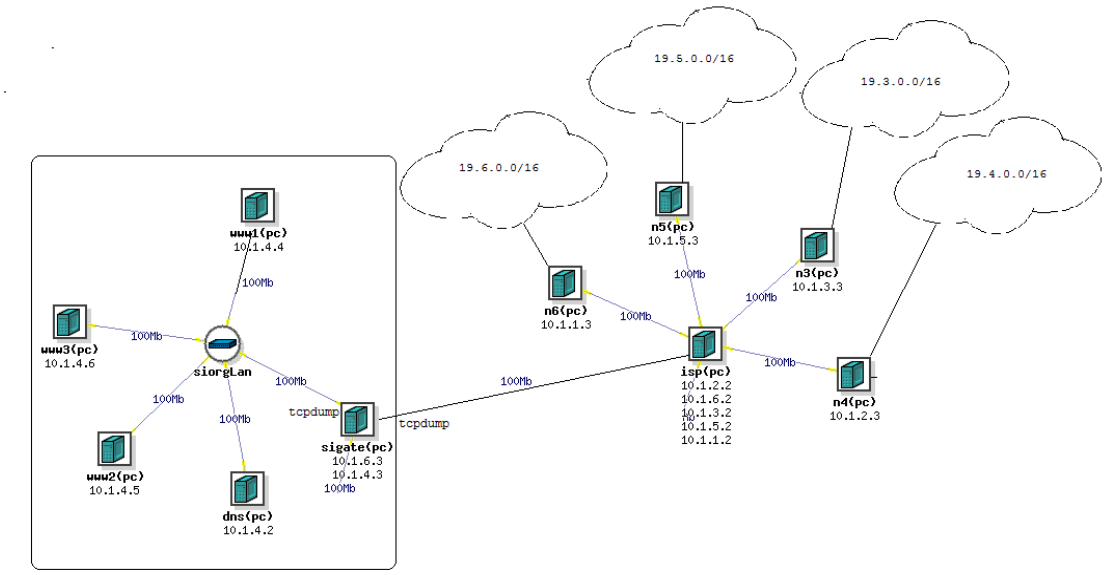


Figure B5: Network topology from our DeterLab experiments. The organization employing spread identity is bounded by a square on the left. On the right is a representative source organization that can generate more than 100,000 IP addresses. We generated these addresses using four machines (one per subnet) running multiple threads with IP aliasing.