A Flexible In-Network IP Anonymization Service

Marc Mendonca
University of California
Santa Cruz, CA USA

Srini Seetharaman
Deutsche Telekom R&D Lab
Los Altos, CA USA

Katia Obraczka
University of California
Santa Cruz, CA USA

***Abstract*—User privacy on the Internet has been an increasing concern in recent years. With the proliferation and sophistication of information services, data mining, and search engines, a simple network address may be used to reveal a great deal of information about a user, including location, identity, and behavior. Existing approaches to privacy, however, make unacceptable tradeoffs between performance and anonymity. For example, Tor [5] attempts to provide strong anonymity by withholding trust from third-party relays. We believe an acceptable level of privacy can be provided to most users, with noticeably lower latency and throughput impact, by working with the network provider.**

In this paper, we introduce *AnonyFlow*, an in-network anonymization service designed to efficiently and seamlessly provide privacy to users as they communicate with other endpoints and services. We design, implement, and evaluate an OpenFlow-based prototype of *AnonyFlow* that achieves endpoint anonymity without compromising on throughput or latency.

I. Introduction

As data mining, geolocation services, targeted advertising, and data brokers become more pervasive, it is possible to learn a large amount from information flowing through the network, in particular network addresses stamped on each packet. This coupled with considerable increase in privacy and security breaches have sparked renewed interest in services and tools that provide user anonymity. However, as will become clear in Section III, current anonymization approaches typically incur prohibitive performance penalties.

We argue that the bigger threat to privacy for everyday Internet users is unscrupulous or overzealous endpoints and Web services, and not network infrastructure providers, who are typically restricted from disclosing data. In fact, our claim is that infrastructure providers would be quite willing to protect their users against any potential attacks and/or threats as they may be held accountable and liable for security breaches, such as identity theft and other intrusions of user privacy.

While there have been some previous attempts to enlist providers to offer privacy services [19], they required the deployment of specialized gateways and faced challenges arising from having multiple ingress/egress points in the network. Furthermore the previous approach forces temporary outages on long-lived flows because they are unable to keep track of per-flow state at the gateways beyond a fixed time period when keys rotate. Approaching this issue from a software-defined network viewpoint, the deployment is simplified and many of the problems facing earlier distributed solutions are solved by maintaining a centralized view.

In this paper, we address the issue of network privacy with this viewpoint in mind. We introduce *AnonyFlow*, an in-network anonymization service designed to efficiently and seamlessly provide privacy to users as they communicate with other endpoints and services. As an anonymization tool, *AnonyFlow* can also be used as a building block for a variety of services, such as PO Box [18], anonymization of network traces [17], as well as a way to provide separation between location and identification [13].

There exists several approaches to privacy as described in §III. Popular privacy-preserving tools, however, offer unacceptable delays. Figure 1 presents the results from our preliminary experiment involving the access of Wikipedia from a laptop in our Los Altos lab. In this figure, "proxy" represents the average of the page access delay seen with popular web proxy tools that offer anonymization[1]. This increase in latency motivated us to pursue other network-enabled approaches.

By enlisting cooperation from infrastructure providers and thus adopting an "in-network" approach to anonymization, we are able to provide endpoint privacy in a seamless, user-transparent way. Unlike approaches such as Onion Routing [8], *AnonyFlow* incurs negligible overhead; for example, as shown by our performance evaluation experiments in Section V, *AnonyFlow*'s impact on user-perceived

[1]Each delay measurement is in itself an average over 10 runs per method. The Tor experiment used a 3-relay path in each run. The proxies used were viz., anonymouse, zend2, proxify, hidemyass, kproxy, browser9. "BASE" refers to the non-anonymized web download.

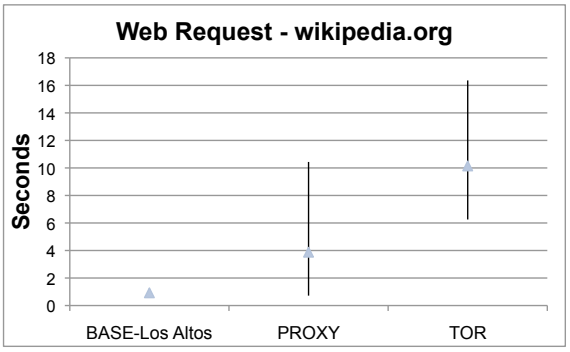


Fig. 1. Webpage load time using common privacy tools.

latency is close to zero. Additionally, *AnonyFlow* requires no changes to endpoints which facilitates its deployment considerably. Unlike Network Address Translation (NAT) based tools, *AnonyFlow* is able to provide intra-domain anonymity, as well as dynamic, on-demand addresses. This ability to provide disposable, flow-based identifiers prevents malicious endpoints from tracking behavior and launching attacks on users.

To evaluate *AnonyFlow*'s functionality and performance in managed networks, we implemented a simple prototype using the OpenFlow platform [12]. OpenFlow enables execution of network-level services through a *controller*, which dictates the behavior and actions of switches under it's jurisdiction. This enables the implementation of in-network, on-the-fly, packet morphing actions. Our simple evaluation over the hardware-based testbed showed that *AnonyFlow* provides higher flexibility in IP anonymization with no impact on the end-to-end latency and a minor deterioration in TCP throughput in wide-area networks. By design, *AnonyFlow* places a degree of trust in the network infrastructure provider who operates the OpenFlow controller. As discussed above, network infrastructure providers have all the incentives to provide anonymity to its users in a transparent fashion, with minimal impact on performance.

II. Design Goals

Before describing our design goals, we briefly overview the threat model we employ.

Threat Model

Unlike systems that wish to offer full anonymity, we place a measure of trust in the managed network provider and the network privacy service. In our work, the main "adversary" is the other endpoint. To a lesser extent, we also hide information from third-party switches outside the managed network when traffic crosses the Internet.

In the network today, an intrusive endpoint may attempt to track user behavior based on the IP address of the connection. By correlating network logs with user actions, the "anonymity" that many users believe is implicit on the Internet is destroyed as usage patterns such as when the user connects, how often, to what services, etc. can be extracted. Furthermore, the proliferation of services such as WHOIS and IP geolocation allows third-party providers to learn a great deal about the location and possibly the real identity of the user. Besides passively monitoring user activity, active attacks may take place based on the information gleaned from IP address, where user experience may be altered or user access may even be blocked. While it is possible that user profiling may be used for a benign purpose, they can also lead to censorship or gross violations of user privacy. Our privacy service attempts to decouple network identifiers from location and identity in order to provide users with truly free and universal Internet experience.

Goals

AnonyFlow is designed to protect users primarily from endpoint logging, while minimizing its impact on performance.

Below, we list our main design goals:

- *Endpoint privacy* – the other endpoints should not be able to track source behavior or location.
- *Minimal performance impact* – no additional perceived latency on web traffic.
- *Network-based design* – should require no change to the endpoints.
- *Straightforward deployment* – the service should be easily deployed and managed, with a minimal amount of specialized network hardware.

We should also point out that *AnonyFlow* does not try to address the following issues:

- *Data security* – we allow applications to enforce their desired level of protection from eavesdropping and tampering by leaving the data encryption to the application layer. Likewise, we leave it to our users to select applications that are privacy-aware and will not leak identity information to the other endpoint.
- *Steganography* – we do not attempt to achieve unobservability nor hide the existence of messages within other data, such as digital media [4], [11].

- *Complete anonymity* – as previously highlighted, we assume that network providers are trusted entities and require their cooperation in concealing end-user identity.

III. Related Work

Simple anonymizing proxies [3] provide endpoint privacy but require trust in the proxy. Additionally, there is an overhead cost of working at a higher layer or through tunneling, as well as the delay of routing traffic through the proxy rather than following the most efficient route to the destination.

Traditional network address translation (NAT) also provides a certain degree of privacy, but the public IP address can still typically be traced back to a single household, organization, or ISP. Additionally, it provides no privacy benefit to intranetwork communication behind the NAT box.

Virtual Private Networks are another popular solution used to hide network identity from the opposite endpoint. While widely supported and deployed, it has similar drawbacks as an anonymizing proxy - the user must trust in the VPN service provider and traffic must flow through the provider network, which may not be the most efficient route to the destination.

Stronger overlay-based anonymity approaches include Onion routing [8], e.g., Tor [5], Web mixes, e.g., JAP [2], and Tarzan [7]. While they are still considered "low-latency" connection-oriented approaches compared to slower message-based anonymity systems [20], they still exhibit non-trivial overhead and noticeable delay (as observed in Figure 1. In particular, a recent usability study [6] of Tor found that DNS requests were 40 times slower than direct connections. The expected Web user cancellation rate was 6 times greater, indicating high degree of user dissatisfaction. Although these anonymity solutions still have their applications, we argue that lower-latency anonymity approaches, such as *AnonyFlow*, fill an important role both as stand-alone services as well as building blocks for applications that require endpoint anonymity.

The BLIND framework [22] provides location privacy in IP networks through the use of public key endpoint identifiers and NAT-based forwarding agents.

There are several approaches that use the idea of a location-independent identifier, including Mobile IP [18], the Host Identity Protocol [15], and the Locator Identifier Separation Protocol (LISP) [13]. Like LISP, we integrate a network-based location/identity split into our design.

[14] proposed an approach that assigned temporary, random, IP and MAC addresses to users requiring anonymity on the Internet. Similarly in wireless LANs, [9] presented a mechanism to enhance location privacy through the use of disposable interface identifiers. We expand on these ideas in our system to provide clients with temporary identifiers that are linked to specific flows.

Just as traditional telephone companies are able to provide "caller-ID blocking" as an additional service to their customers, ISP's should be able to offer a service that automatically hides the IPs of their users without modification at the client-side. The Address Hiding Protocol (AHP) [19] attempts to do this with a mechanism similar to CPP [21], a system that encrypts IPv6 address to provide location privacy. AHP requires specially designed gateways that use time-based keys to keep in sync, but can cause collisions in cases of long-lived flows, and requires a somewhat involved design for handling multiple ingress/egress points. We agree with the high-level goals of AHP, and hope to show how they can be more easily implemented and deployed in a software-defined networking environment.

IV. *AnonyFlow* Architecture and Operation

AnonyFlow is designed to provide endpoint privacy by concealing the source identifier from the other side of the connection. Additionally, the in-network approach allows a level of accountability that can be used to revoke access to malicious users.

Identifiers

Before presenting *AnonyFlow*'s architectural components, we describe the different identifiers *AnonyFlow* employs to represent users. Note that our work is equally applicable if we operate at Layer 2.

- *Machine IP address* – the address assigned to the machine, and what would normally be seen by the other endpoint if *AnonyFlow* is disabled. When using *AnonyFlow*, it is rewritten as soon as possible to another form, and only changed back when deliverying messages back to the machine.
- *AnonID* – the identifier that the other endpoint receives as it provides no information of the location or identity of the machine. They may be used on a temporary basis and discarded at the end of a flow; alternatively, it can be used as a more permanent identifier for a service that wishes to remain anonymous.

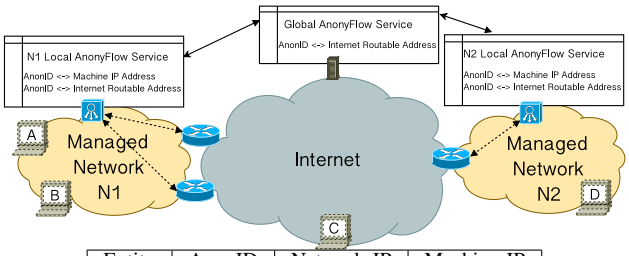
- *Network IP address* – this address is routable back to the managed network of the machine, and is necessary for traversing the Internet. It is associated with a given AnonID and can be changed on demand.

Components

Each managed network participating in the *AnonyFlow* service can be thought of as a *local domain*. The *AnonyFlow* service itself consists of a *local* and *global* component. At the border of each local domain, AnonIDs must be translated to an identifier (e.g., IP address) that would allow flows to traverse intermediate routers.

- *AnonyFlow conduit* – rewrites IP addresses to/from AnonIDs, and forwards resulting packets towards destination. Consults with the local *AnonyFlow* service.
- *Local AnonyFlow service* – handles user join/leave and local mappings, and communicates with global service.
- *Global AnonyFlow service* – handles network lookup for AnonIDs outside local managed network.

Example Operation



Entity	AnonID	Network IP	Machine IP
A	1	11.11.X.X	11.11.0.1
B	2	11.11.X.X	11.11.0.2
C	-	12.12.0.1	12.12.0.1
D	3	13.13.X.X	13.13.0.1

Fig. 2. Example of *AnonyFlow*'s Operation.

We use the scenario depicted in Figure 2 to illustrate *AnonyFlow*'s operation and examine the series of events that occur when host A opens a connection to the hidden service on host B. First, A sends a packet with source '11.11.0.1' and destination '2' that will pass through the *AnonyFlow* conduit on network $N1$. The conduit will consult with the *AnonyFlow*'s local service of $N1$ for the first packet of the flow. The local service determines that both the source address is associated with AnonID '1', and that the destination AnonID '2' is within the same network. The conduit will then rewrite the source address to '1' and the destination to '11.11.0.2' and set up rules to forward the flow to the destination.

Entity	AnonID	Network IP	Machine IP
B	X		
C		X	
D	X		
N1 Service	X	X	X
N2 Service	X	X	
Global Service	X	X	

TABLE I
IDENTIFIERS OF HOST A AS OBSERVED BY OTHER AGENTS.

If the destination AnonID is in another network, for instance, when host A communicates with host D, there are a few more steps. The local service must do a global lookup of the destination AnonID to determine an address routable to the destination network. It must also assign a routable address to the source, in a manner similar to NAT. Finally, when the flow arrives at a conduit in the destination network, it will rewrite the source address to AnonID '1' and the destination to the machine address of D.

In the final case of the destination being outside the *AnonyFlow* namespace, such as when host A communicates with host C, our system resembles a more flexible yet traditional NAT service. The conduit rewrites the source address with an address routable to the source network so that the destination is able to trace the message directly back to the source network.

In Table I, we summarize the identifiers of host A that each network entity is able to observe when host A communicates with host B, C, and D.

OpenFlow Platform

Although there are a number of ways to enable *AnonyFlow* in a managed network, we use the OpenFlow platform in our reference implementation. OpenFlow provides the means for rapid deployment and execution of network services by enabling a remote controller to modify the behavior of switches and routers. By providing direct access to the switch flow table, OpenFlow API allows services to achieve custom routing and packet processing.

In our OpenFlow implementation, each local *AnonyFlow* domain consists of a network managed by a single OpenFlow controller. *AnonyFlow* conduits correspond to OpenFlow-enabled switches, while the local *AnonyFlow* service is integrated with the OpenFlow controller. *AnonyFlow*'s global service exists outside of the OpenFlow infrastructure as a directory service.

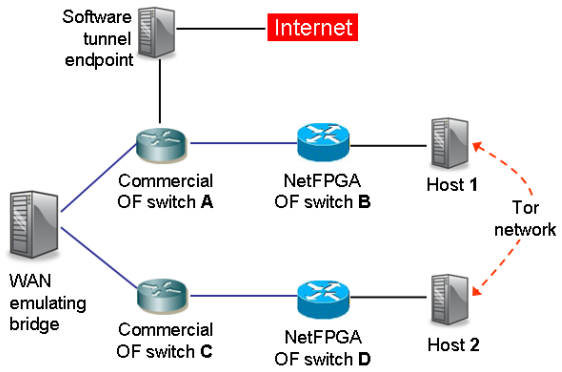


Fig. 3. Lab network used to emulate a wide-area OpenFlow-enabled network. The NetFPGA-based OpenFlow switches at the edge take care of the required header rewriting actions.

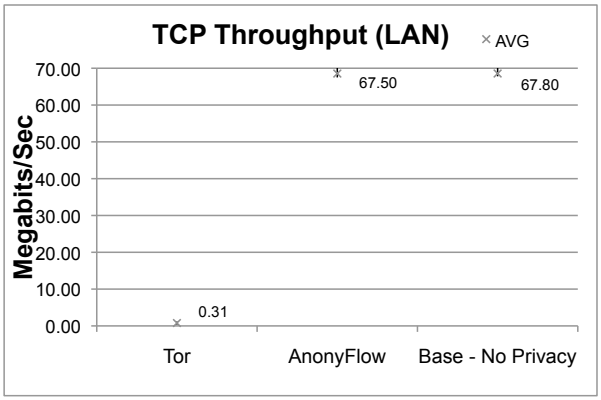


Fig. 4. TCP Throughput (`iperf`) between the 2 hosts in our testbed.

V. Performance

To evaluate the performance of *AnonyFlow*'s OpenFlow-based implementation, we deploy it in a real network testbed with OpenFlow switches. We focus on IP identifier anonymization and compare the performance of *AnonyFlow* with other related approaches.

The testbed, as shown in Figure 3,has two logical sub-networks interconnected by a Linux bridge. Each subnetwork consists of two commercial OpenFlow-enabled switches and two NetFPGA [16]-based OpenFlow switches, all of which are controlled by a remote NOX [10]-based *AnonyFlow* controller. This testbed provides header rewriting capability at line-rate in the NetFPGA-based edge switches.

In Figure 1 we presented the latency characterization of common privacy-preserving tools. *AnonyFlow*'s performance is almost the same as the base case because the anonymization is done by an en-route switch. This happens at line speed and does not incur additional delays.

In Figure 4 we present the TCP throughput characterization performed over the above mentioned testbed. In this characerization, we ran `iperf` [1] between the two hosts (Host 1 and 2 in Figure 3) multiple times, with each run lasting 25 seconds. The mean, min and max values are shown in the two figures; the mean is denoted by a 'x' marker, and the vertical line around the mean value represents the min to max range. We observe that *AnonyFlow* does not experience a throughput deterioration, while Tor achieves throughput that is a few orders of magnitude lower than the direct route. This is because Tor selects additional Internet relay hops without regard to routing performance.

VI. Discussion

A. Attacks and Defenses

In this section, we examine possible attacks on *AnonyFlow* users and on *AnonyFlow* itself. As specified in our design goals, our system is designed to protect against a single adversary and not more exhaustive global attacks.

a) Passive attacks:

- *Network address tracking* – a static IP address allows users to be tracked, and may also leak information about identity and location. *AnonyFlow* uses disposable, flow-based identifiers when talking to endpoints within domains; otherwise, temporary IP addresses are assigned when communicating with traditional Web servers.
- *Timing correlation* – unlike high-latency anonymity systems that introduce delays to alter timing characteristics, *AnonyFlow* seeks to minimize performance impact; consequently, it does not hide information learned by the observation of packet inter-arrival times.
- *Content analysis* – packet payloads may contain identifying information. We leave it to the user to select applications that provide data privacy.

b) Active attacks:

- *Direct attacks on AnonIDs* – the use of temporary, flow-based identifiers offers a level of protection and indirection to endpoints as it is not possible to communicate with an identifier after it has been released or expired. This prevents attacks such as port scanning or denial of service.
- *Denial of service on service* – an adversary may attempt to overwhelm *AnonyFlow*'s mapping service with false requests. This may be dealt with, on-the-fly, by pushing rules that dump excessive flows from offenders at the conduits.

- *Router subversion* – if an intermediate router is compromised by the adversary, they may learn information about the hidden endpoint. In the worst case, the protection offered degenerates to a simple network address translation (NAT) service. To protect against subversion within domains, we require authentication and use encrypted communication between conduits and the privacy service.
- *Man-in-the-middle* – we leave it to the user to select applications that offer data privacy and integrity.
- *Governmental authority* – endpoint privacy is provided by the network infrastructure rather than the endpoints; therefore, it is subject to control by authoritative entities. This provides a level of accountability, and is well suited to protecting the identity of legitimate users, but means the service is ill-advised for abusive criminals or users living under oppressive regimes.

B. Future work

In summary, *AnonyFlow* offers a lightweight endpoint anonymity service with minimal performance impact. Directions for future work include:

- *Scalability* – currently, *AnonyFlow*'s global mapping service that maps AnonIDs to networks is implemented using a centralized architecture. As the number of users and domains grow, a distributed service would reduce possible bottlenecks.
- *Increased access to hidden services* – currently, only users from within *AnonyFlow* domains are able to access hidden services offered by other *AnonyFlow* users.
- *Placement* – it suffices to place OpenFlow switches with header rewriting capability in only a few strategic locations of the network. We plan to further investigate this option, which allows incremental deployment.

Features	NAT	Proxy	Tor	*AnonyFlow*
Hide source from destination	X	X	X	X
Hide source from relays			X	
Change identifier on-demand			X	X
L2 or Intra-domain privacy		X		X
Enable hidden services			X	X
Optimal routing	X			X

TABLE II
FEATURE COMPARISION OF ANONYMIZATION TECHNIQUES.

VII. Concluding Remarks

In this work we presented *AnonyFlow*, an in-network endpoint anonymization service designed to provide privacy to users. Through experiments on a real network testbed, we show that our proof-of-concept OpenFlow-based prototype of *AnonyFlow* delivers similar performance when compared to non-anonymized network access, while providing more features than the other schemes (Table II).

References

[1] iperf tool. http://iperf.sourceforge.net.
[2] O. Berthold, H. Federrath, and S. Kopsell. Web mixes: A system for anonymous and unobservable internet access. In *Designing Privacy Enhancing Technologies*, pages 115–129. Springer, 2001.
[3] J. Boyan. The anonymizer. *CMC Magazine*, 1997.
[4] S. Burnett, N. Feamster, and S. Vempala. Chipping away at censorship firewalls with user-generated content. In *Proc. 19th USENIX Security Symposium, Washington, DC*, 2010.
[5] R. Dingledine, N. Mathewson, and P. Syverson. Tor: The second-generation onion router. In *Proceedings of the 13th conference on USENIX Security Symposium-Volume 13*, pages 21–21. USENIX Association, 2004.
[6] B. Fabian, F. Goertz, S. Kunz, S. Müller, and M. Nitzsche. Privately waiting–a usability analysis of the tor anonymity network. *Sustainable e-Business Management*, pages 63–75, 2010.
[7] M. Freedman and R. Morris. Tarzan: A peer-to-peer anonymizing network layer. In *In Proc. 9th ACM Conference on Computer and Communications Security*, 2002.
[8] D. Goldschlag, M. Reed, and P. Syverson. Onion routing. *Communications of the ACM*, 42(2):39–41, 1999.
[9] M. Gruteser and D. Grunwald. Enhancing location privacy in wireless lan through disposable interface identifiers: a quantitative analysis. *Mobile Networks and Applications*, 10(3):315–325, 2005.
[10] N. Gude, T. Koponen, J. Pettit, B. Pfaff, M. Casado, N. McKeown, and S. Shenker. Nox: towards an operating system for networks. *ACM SIGCOMM Computer Communication Review*, 38(3):105–110, 2008.
[11] T. Heydt-Benjamin, A. Serjantov, and B. Defend. Nonesuch: a mix network with sender unobservability. In *5th ACM workshop on Privacy in electronic society*, 2006.
[12] N. McKeown, T. Anderson, H. Balakrishnan, G. Parulkar, L. Peterson, J. Rexford, S. Shenker, and J. Turner. Openflow: enabling innovation in campus networks. *ACM SIGCOMM Computer Communication Review*, 38(2):69–74, 2008.
[13] D. Meyer. The locator identifier separation protocol (lisp). *The Internet Protocol Journal*, 11(1):23–36, March 2008.
[14] C. Molina-Jiménez and L. Marshall. True anonymity without mixes. In *wiapp*, page 32. Published by the IEEE Computer Society, 2001.
[15] R. Moskowitz and P. Nikander. Host Identity Protocol (HIP) Architecture. RFC 4423, May 2006.
[16] Netfpga platform. http://netfpga.org.
[17] R. Pang, M. Allman, V. Paxson, and J. Lee. The devil and packet trace anonymization. *SIGCOMM Comput. Commun. Rev.*, 36:29–38, January 2006.
[18] C. Perkins. IP Mobility Support for IPv4, Revised. RFC 5944, Nov. 2010.
[19] B. Raghavan, T. Kohno, A. Snoeren, and D. Wetherall. Enlisting isps to improve online privacy: Ip address mixing by default. In *Privacy Enhancing Technologies*, pages 143–163. Springer, 2009.
[20] A. Serjantov and P. Sewell. Passive attack analysis for connection-based anonymity systems. *Computer Security–ESORICS 2003*, pages 116–131, 2003.
[21] J. Trostle, H. Matsuoka, M. Tariq, J. Kempf, T. Kawahara, and R. Jain. Cryptographically protected prefixes for location privacy in ipv6. In *Privacy Enhancing Technologies*, 2005.
[22] J. Ylitalo and P. Nikander. Blind: A complete identity protection framework for end-points. In *Security Protocols*, pages 163–176. Springer, 2006.