Chen Chen* and Adrian Perrig

PHI: Path-Hidden Lightweight Anonymity Protocol at Network Layer

Abstract: We identify two vulnerabilities for existing high-speed network-layer anonymity protocols, such as LAP and Dovetail. First, the header formats of LAP and Dovetail leak path information, reducing the anonymity-set size when an adversary launches topological attacks. Second, ASes can launch session hijacking attacks to deanonymize destinations. HORNET addresses these problems but incurs additional bandwidth overhead and latency.
In this paper, we propose PHI, a Path-HIdden lightweight anonymity protocol that solves both challenges while maintaining the same level of efficiency as LAP and Dovetail. We present an efficient packet header format that hides path information and a new back-off setup method that is compatible with current and future network architectures. Our experiments demonstrate that PHI expands anonymity sets of LAP and Dovetail by over 30x and reaches 120 Gbps forwarding speed on a commodity software router.

Keywords: Anonymity, path-hidden protocols

DOI Editor to enter DOI
Received 2017-05-31; revised 2017-08-01; accepted 2017-08-01.

1 Introduction

Revelations about governments' mass-surveillance programs have demonstrated their capability of conducting pervasive surveillance on huge volumes of domestic and international traffic [4, 9]. Meanwhile, an increasing number of users have begun using anonymous communication software to protect their privacy. For instance, Tor [27] has on average 2 million active users per day [13]. However, most anonymity software today is built as an overlay network composed of end hosts' voluntarily-contributed nodes [6, 8, 14]. As a consequence, users experience poor performance due to long propagation delays and limited bandwidth, along with intrinsic queuing and retransmission delays of the protocols [28].

Recent research has proposed anonymity as a principal network function to benefit from short paths and high throughput of network devices [21, 34, 46]. These proposals demonstrate that it is viable to build lightweight cryptography into network routers to help anonymize the huge volumes of traffic accessible to mass surveillance programs today. We compare these protocols in Table 1.

Protocol	Throughput	Bandwidth Overhead	Path Info. Leakage	Session Hijacking	Trust in 1st hop	Hosts Control Paths
LAP	Low	Low	Yes	Yes	Yes	No
Dovetail	Low	Low	Yes	Yes	No	Yes
HORNET	Medium	Medium	No	No	No	Yes
PHI	Low	Low	No	No	No	No

Table 1. Comparison of high-speed network-layer anonymity protocols.

LAP [34], for example, adds an encrypted path into each packet, and LAP routers forward packets using only symmetric cryptography. However, a compromised first-hop Autonomous System (AS) can deanonymize both the source and the destination and thus immediately compromise anonymity because LAP's setup process leaks the destination address. Dovetail [46] overcomes the limitation by using an indirection node, called "match maker", to conceal the destination node from the first-hop AS. However, Dovetail requires that the source have full control over the traversed path, which harms its compatibility with current network architectures.

Furthermore, we focus on two vulnerabilities of LAP and Dovetail. First, their headers, even with the proposed defense to hide a path's length and routers' positions on a path [34], still leak such path information. Thus, an on-path router can reduce the size of the source's anonymity set based on publicly available network topology. Second, payload encryption is detached from the path in Dovetail, enabling a session hijacking attack to deanonymize destinations.

In contrast to LAP and Dovetail, HORNET [21] hides the path information by using an onion-encrypted data structure to embed path information and prevents the session hijacking attack. However, HORNET's solution incurs additional costs: first, HORNET's connection setup requires Elliptic-Curve Diffie-Hellman (ECDH) computation between a sender and each intermediate on-path nodes, adding computational la-

***Corresponding Author: Chen Chen:** Carnegie Mellon University / ETH Zurich, E-mail: chen.chen@inf.ethz.ch
Adrian Perrig: ETH Zurich, E-mail: adrian.perrig@inf.ethz.ch

tency; second, HORNET requires the sender to anonymously retrieve and verify the public keys of on-path nodes, introducing further latency and potential identity leakage vectors.

At a first glance, one is bound to an unfortunate choice between weaker anonymity and additional latency. In this paper, we demonstrate that it is possible to achieve the best of both worlds. We propose a Path-HIdden lightweight anonymity protocol, named PHI, that improves anonymity over LAP and Dovetail and is equally efficient.

PHI improves on LAP and Dovetail by introducing three new techniques. First, PHI places nodes' state in a pseudo-random order in a packet header to conceal information about node positions. Second, PHI leverages a back-off path construction method to eliminate the need for a source to fully control the path traversed. Third, PHI prevents session hijacking attacks by binding payload encryption to paths.

Our paper makes the following contributions:

1. We identify two attacks that reduce sizes of anonymity sets in LAP and Dovetail. In particular, we model and analyze the path information leakage when LAP and Dovetail intentionally obscure path information by using *variable-size segments* (see Section 2). In comparison, existing work, HORNET, only shows that LAP and Dovetail leak path information without such protection mechanism.
2. We propose a path-hidden header format that is more efficient than the ones used by onion routing protocols.
3. We present a new approach to establish an end-to-end path for a source node with no control over the path traversed.
4. We design the Path-HIdden lightweight anonymity protocol (PHI), an efficient network layer protocol that provides stronger anonymity properties than LAP and Dovetail with the same level of efficiency.
5. We evaluate PHI's security and performance. Evaluation results confirm that PHI's performance is comparable to, or more efficient than LAP and Dovetail, while expanding the sizes of anonymity sets.

2 Background

2.1 Network-layer Anonymity Protocols

Network-layer anonymity protocols assume that network infrastructure (e.g., switches and routers) perform anonymization operations when forwarding packets. They function at the network layer as a complementary or as an alternative option to the Internet Protocol (IP) to anonymize packets' sources and destinations. Compared to existing anonymity systems built on overlay networks such as Tor [27], network-layer anonymity protocols aim to offer low-latency and high-throughput packet forwarding and scale to handle high volumes of traffic seen in the Internet [21, 34, 46].

To achieve fast forwarding and high scalability on network devices whose per-packet computation and per-flow storage are usually extremely limited, network-layer anonymity protocols share two design choices: 1) that packet forwarding only needs symmetric crytography, and 2) that forwarding state should be carried by packets instead of stored on network devices. While using symmetric cryptography is also common in latest overlay-based anonymity systems, the second choice mainly distinguishes network-layer anonymity protocols. At the beginning of each connection, a source node and a destination node exchange setup packets that traverse each Autonomous System (AS) on the path. Within a setup packet, each on-path AS (or a *node*) creates a *path segment* containing its forwarding state. These path segments are carried by data packet headers so that a node can retrieve its state and know how to forward the packets.

Lightweight vs. onion-routing protocols. We divide existing network-layer anonymity protocols into onion-routing protocols and lightweight protocols based on their different requirements for computation and header overhead. An onion-routing protocol, like HORNET [21], requires an on-path network device to compute an expensive asymmetric crytographic operation for setting up a flow and applies per-hop authenticated encryption over every data packet. Each on-path node also needs to store necessary keys within its path segment, resulting in large packet headers. In comparison, lightweight anonymity protocols, such as LAP [34] and Dovetail [46], only require a network device to decrypt and verify a path segment that contains minimal information for forwarding packets, and use end-to-end packet encryption to offer confidentiality.

Topology-based attacks. Unlike a node in an overlay-based anonymity system that can forward packets to any other node, an AS can only forward packets according to its physical connections and business contracts with its neighbors. Therefore, network-layer anonymity protocols are inherently subject to so called "topology-based attacks". With publicly-available network topology information, a compromised node receiving a single packet from a victim can narrow down the victim's anonymity set. For example, in Figure 1, if we assume that AS2 is AS3's customer, by knowing the topology and receiving a packet from AS2, AS3 can conclude that the source node must be within {AS0, AS1, AS2, AS4, AS5}, which also forms the anonymity set. If the adversarial AS further discovers its AS-level distance from the source's AS, it can reduce the size of the victim's anonymity set. In Figure 1, if AS2 is AS3's customer and AS3 uncovers that the source of a packet from AS2 is 3 hops away, AS3 can infer that the source's

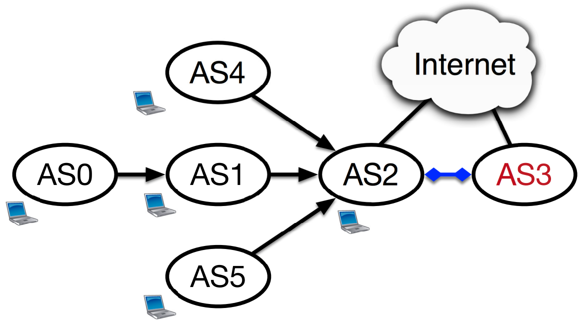


Fig. 1. Topology-based attacks. AS3 is a compromised AS and receives a packet form AS2 and conducts topology-based attacks to de-anonymize the packet's source node. The arrow from an AS points to its provider and the relationship between AS2 and AS3 can be peer-to-peer, customer-to-provider, and provider-to-customer.

anonymity set only contains AS0, which significantly harms the source's anonymity.

In both LAP and Dovetail, path segments are successively appended to a header by each on-path node, and each header tells the forwarding router which path segment to process. As a result, even a single compromised router can determine the number of hops between the end host and itself by counting path segments in the header. As demonstrated by Chen et al. [21], such an attack can successfully reduce the anonymity-set size of the source and destination. HORNET, on the other hand, avoids leaking such information by onion encrypting and shifting path segments in a header.

To defend against such attacks, LAP proposes using *Variable-Size Segments* (VSS) to hide path information. With VSS, each path segment is padded to a random size of multiple blocks chosen independently by on-path ASes. However, VSS doubles or triples the packet-header size, adding to bandwidth overhead. If the maximum number of segment blocks is A, the resulting VSS path in a packet header is A times larger than a path without VSS. In the following, we call A “path-size amplification factor”, or simply “amplification factor”.

2.2 Challenges of Lightweight Protocols

We identify two attack vectors to existing lightweight protocols that can further reduce anonymity-set sizes of end hosts.

Probabilistic path information leakage. Although VSS helps obscure the number of path segments inserted in a packet header, it cannot fully hide the information. In fact, an on-path adversary can still determine the probability distribution of its position on the path and the path length. With the knowledge of the distribution, the adversary can increase the probability of identifying a victim and thus reduce *equivalent* anonymity-set size. For example, in Figure 1, if each AS can output a path segment of either 1 or 2 basic blocks, and AS3 observes that there are 6 basic blocks before its own path segment, AS3 can derive that the source node must reside in AS0. We will mathematically model the path information leakage of VSS in Section 6.

2.2.1 Session Hijacking Attacks

In both LAP and Dovetail, payload encryption is essential to prevent adversaries from deanonymizing either end by scrutinizing identity information in packet payloads. A common practice is to use DH key exchange to negotiate a shared symmetric key for encrypting subsequent payloads in a session. In order to thwart an adversary in hijacking the key exchange protocol by launching Man-in-the-Middle (MitM) attacks, both parties should verify the public key of the other end.

Nevertheless, when sender anonymity is at stake, end hosts can only conduct one-way authentication, i.e., a source verifies the public key of its destination but not vice versa. Consequently, an on-path adversarial node is able to hijack the key exchange session by replacing the source's public key with its own. Because the destination cannot verify that the public key is from an anonymous source or from an adversarial node, the adversary can communicate with the destination in place of the source, and compromise the destination's identity.

Note that in LAP an on-path adversarial node can directly obtain the destination from setup packets without launching session hijacking attacks. Dovetail, improving over LAP by using a helper node to conceal the destination's identity, is susceptible to the hijacking attack described above. A first-hop node, who knows the source by its advantageous position, can leverage such an attack to deanonymize the destination in the following way. The adversary waits until a source node establishes an end-to-end path using Dovetail's session setup, during which process the adversary can obtain the created path as well. The adversary then uses the intercepted path to communicate with the destination and sets up a secure channel by using the DH key exchange protocol. Because the destination cannot verify that the public key of the other end, it cannot tell whether it communicates with the actual source or an adversarial node. With this attack, the first-hop node can potentially compromise the destination's identity, defeating Dovetail's security goal.

3 Problem Definition

3.1 Network Model

We regard ASes as the network entity to route packets. Each AS maintains a set of interfaces through which all traversing traffic enters and exits the AS. Depending on the inter-domain routing protocol agreed upon by all ASes, an AS can route an incoming packet by either locally-stored forwarding information (e.g., BGP [45]) or routing information within the packet itself (e.g., NIRA [49], Pathlet [30], and SCION [50]). We also consider a model with loose AS boundaries. In particular, in Pathlet [30], an AS is further divided into virtual nodes (vnodes), each of which serves as an independent principal to forward packets. In this paper, we use the generic term *node* to denote an AS or a vnode in the case of Pathlet. We consider the fact that ASes in the inter-domain network follow various network policies, and our anonymity protocol should be versatile in supporting existing network policies.

For anonymity purposes, we assume that an AS upgrades its border routers to support necessary symmetric cryptography such as AES. Each AS also keeps a local secret key and updates it periodically. All the anonymity-supporting routers should share the secret key and synchronize the key updates.

3.2 Communication Model

A source can anonymously communicate with a destination through a session, which is composed of traffic that shares cryptographic state. Each session is divided into two phases: a setup phase and a forwarding phase.

The setup phase helps the source and destination to establish necessary state to communicate anonymously. First, the source creates a setup request based on the intended destination. Then the source sends the initial packet, called setup request, through the path. Routers of on-path ASes insert necessary forwarding state into the setup packet. When the setup request reaches the destination, the destination constructs a reply based on the request and sends it back to the source. As the reply reaches the source, the source can extract all cryptographic state required to communicate in the session.

The forwarding phase enables high-speed packet forwarding. The source and destination create packet headers for data packets belonging to that session. Processing the headers allows a router to determine the next router without learning information beyond the router's previous and next-hop nodes.

3.3 Threat Model

We target a curious but cautious adversary: the adversary aims to link an action (e.g., visiting a webpage) to an identity in a given anonymity set but wants to avoid detection. We allow the adversary to compromise any single AS, or any destination host in the network. By compromising an entity, the adversary obtains all the entity's cryptographic keys. The adversary can perform passive deep packet inspection or active traffic manipulation, such as dropping, injecting, and modifying packets that traverse a compromised entity. Furthermore, we assume that the adversary possesses full knowledge about the network topology. However, we do not consider an adversary that can perform passive or active traffic analysis. Our adversary model is in line with Dovetail and removes the assumption about trusting the first-hop node in LAP.

3.4 PHI's Security and Performance Goals

PHI achieves the following security goals:

- *Sender/sender-receiver anonymity*. PHI offers two types of anonymity properties: sender anonymity and sender-receiver anonymity, as defined by Pfitzmann and Köhntopp [44]. As a network-layer anonymity protocol, PHI considers an end host to be *anonymous* if its network location cannot be distinguished from that of a set of other hosts, called the anonymity set. However, because the first-hop AS is directly connected to the source node, PHI only offers *relationship anonymity* [44] against the first-hop adversary.
- *Session unlinkability*. Traffic from different sessions cannot be linked together by an adversary to de-anonymize end hosts.
- *Path-information confidentiality/authenticity*. An on-path compromised node cannot uncover the total number of nodes on the path or its distance from end hosts on either side. In addition, an adversary cannot modify an existing path or fake new paths so that resulting paths still traverse the network.

PHI also aims to offer a series of performance properties:

- *Low bandwidth overhead*. PHI adds a small header to each data packet.
- *Low latency*. Processing a PHI packet only incurs low additional latency.
- *Scalability*. PHI nodes maintain no per-flow state.

4 Design

PHI adopts two design features that are common among network-layer anonymity systems. First, for high-speed data-packet forwarding, it uses only symmetric cryptography for packet forwarding to achieve low end-to-end latency and high throughput. Second, to scale to large traffic volumes, PHI packet headers carry state required by routers to forward the packets, relieving routers from storing per-session state that can overflow their limited memory.

PHI offers stronger anonymity than Dovetail but only requires the same level of overhead, and is compatible with legacy network architectures. In particular, PHI defeats the identified attacks of existing lightweight protocols, i.e., that headers leak information about node positions and that they are vulnerable to session hijacking attacks.

We now describe the core ideas introduced in this paper. PHI's packet header design randomizes path-segment positions to prevent leaking topological information, and enlarges the expected anonymity set compared to VSS. (Section 4.2). Next, PHI uses a helper node to hide the destination from the nodes close to the source and allows the network itself to select the helper node using a back-off method (Section 4.3). Therefore, PHI is compatible with network architectures where dynamic global topology information is inaccessible, e.g., the Internet, not limiting itself to Pathlet. Lastly, during the path setup phase, PHI binds the source's public key to the established path, thwarting an on-path node from launching session hijacking attacks (Section 4.4).

4.1 PHI High-level Overview

Figure 2 shows an example of PHI path setup phase. Suppose a source node S wants to anonymously communicate with D using PHI. It starts by picking a third node M to establish a full path from S to D. In this paper, we call M the *helper node*. M can be either a service operated by an ISP or a server voluntarily run by another user. During the entire process, M knows D's network location but has no knowledge about S's. In addition, M only forwards on average fewer than 2 setup packets for each session.

First, S encrypts D's address using M's public key and creates a path to M. Based on M's address, each AS on the path between S and M independently decides how to forward the packet, encrypts the decision using a local secret, and places the resulting ciphertext in a pseudo-random position in the header determined by the session's identifier and the AS's local secret. In Figure 2, the full path between S and M is {AS0: $S \to a$, AS1: $b \to c$, AS2: $d \to e$, AS3: $f \to M$}. Each AS places its routing decision in a pseudo-random position in the header. For example, AS0 embeds $S \to a$ in the 3rd position in the header, AS1 inserts $b \to c$ into the 5th position, and etc. Because each AS independently selects a random position for its routing decision, it is possible that their selections are in conflict. We will address this issue in detail in Section 4.2.

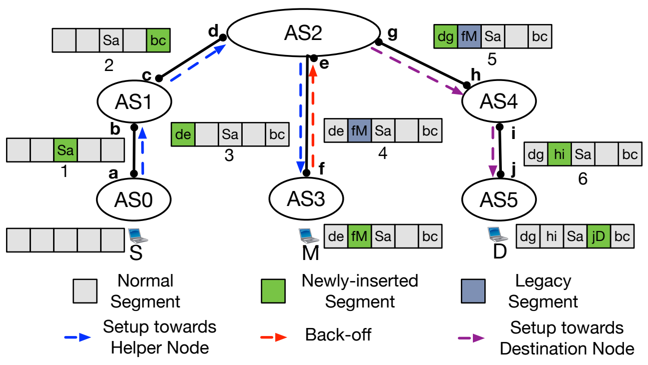


Fig. 2. An example of PHI session setup.

Inserted routing decision is indistinguishable from empty slots. Initially, the source generates random bits to fill the empty header. Each AS encrypts its routing decision with a local secret key before inserting routing decision into the header.

Once the setup packet reaches M, M decrypts D's address and initiates a back-off process that helps S find a midway node, e.g., AS2 in Figure 2. The midway node is responsible for forwarding all subsequent data packets in the forwarding phase between S and D. M uses the header received in the setup packets to route packets in the reverse direction. For instance, AS3 retrieves its routing decision $f \to M$ and forwards the back-off packet to AS2 through interface f.

The back-off process stops when an AS W on the path from S to M is willing to forward the packet to D through an interface that is different from the one where the first setup packet is received. We name this AS W the *midway node*. In Figure 2, AS2 can forward a packet to D through interface g, which is different from the interface d where the first setup packet is received.

W continues the setup process until the setup packet reaches D. D can retrieve the bi-directional end-to-end path between S and D from the setup packet's header. The rest of the setup phase and the data forwarding phase is essentially equivalent to LAP and Dovetail. D sends the created path back to S using the path in the reverse direction, and S and D can communicate anonymously using the newly established path.

4.2 Randomizing Path-segment Positions

The key idea behind hiding path information in PHI is to randomize the position of a node's segment in packet headers. As

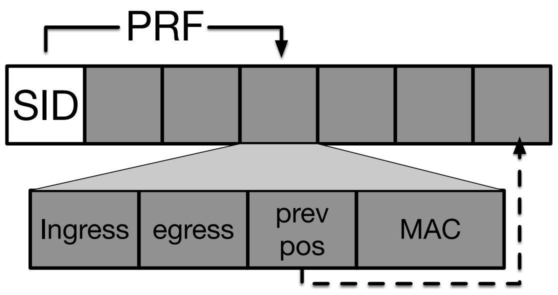


Fig. 3. Randomize a path segment's position in a packet header.

shown in Figure 3, a node n_i computes its segment's position in the header by using a Pseudo-Random Function (PRF) with its local secret k_i^{pos} as the key and the session's id *sid* as the input.

$$pos = \mathrm{PRF}(k_i^{pos}; sid) \quad (1)$$

In addition, the node can find its previous-hop node's segment using the "previous hop" field in the packet header, denoted as pos_{prev}.

If the packet is a setup request packet, the node n_i computes its path segment S_i using the routing information R_i, a local encryption key k_i^{enc}, a local MAC key k_i^{mac}, and a MAC M_{i-1} in the previous node's segment S_{i-1} as follows:

$$E_i = \mathrm{ENC}(k_i^{enc}; R_i \,\|\, pos_{prev} \,\|\, flags) \quad (2)$$
$$M_i = \mathrm{MAC}(k_i^{mac}; E_i \,\|\, M_{i-1}) \quad (3)$$
$$S_i = E_i \,\|\, M_i \quad (4)$$

Here, the actual form of R_i depends on the network architecture that PHI is based on. For instance, it can take the form of (*input link*, *output link*). n_i then replaces the current path segment in the packet request with the newly generated S_i.

If the packet is a data packet, n_i retrieves its path segment, decrypts the segment to find the routing information, and verifies the integrity of its path segment in the inverse process of Equations 2 to 4.

4.2.1 Resolving Collisions

For packet setup requests, because each node independently computes its path segment's position, it is possible that the position in the packet request is already occupied by segment of another node n_c. A collision in picking segment positions causes information loss for n_c's path segment. When the source uses the resulting packet header with collision to forward packets, the MAC verification will fail at n_c.

To avoid using a packet with collided path segments, the source sends more than one setup request packets to increase its success rate. Let the total number of on-path nodes be r, the maximum path segments in the header be m. To reach a success rate of c, the source can send N setup requests all together to satisfy the following condition:

$$1 - \left(1 - \frac{P_m^r}{m^r}\right)^N \geq c \quad (5)$$

where P_m^r computes the number of r-permutations of a set with n element.

Resolving local collisions. Because to each node, the position of its previous-hop node's segment is revealed by the field pos_{prev}, the node can further rule out one type of collision: its own segment's position collides with that of its previous node's segment. If such a collision happens, a node itself can resolve the collision without recourse to other nodes or end hosts. The node computes a new position as follows:

$$pos_{new} = PRF(s; sid \oplus ctr) \quad (6)$$

where ctr is the first value satisfying $pos_{new} \neq pos_{prev}$ starting from 1.

Resolving the local collision in the above manner poses a new challenge for retrieving the segment for a node. Since the node attempting to retrieve its segment from a packet header has no prior knowledge about where its previous-hop node's locates, it may end up with choosing the default position with $ctr = 0$. However, we remark that the node can detect this scenario by verifying the MAC contained in the segment. A failure indicates that there must be a local collision when setting up the packet header and the node should increment the ctr to locate a new position to retrieve its segment.

The resulting number of setup requests, N, satisfies:

$$1 - \left(1 - \frac{P_m^r}{m(m-1)^{r-1}}\right)^N \leq c \quad (7)$$

4.2.2 Probabilistic Header Sizes

The above method runs into a dilemma when the path length r is close to the maximal path segments that a header can contain m. For example, as Section 7.1 shows, when $m = 12$, the required number of setup packets to reach 90% success rate amounts to 4. We can increase m to reduce the required number of setup packets. When $m = 21$, the source only needs 3 setup requests to guarantee 90% success rate. However, with a larger m comes larger packet header for every data packet, which proportionally increases the bandwidth overhead.

However, we observe that the probability that a long path exists between a random pair of source and destination is low. According to our experiment in Section 7.1, the probability of a path that is longer than 7 AS hops is below 0.01%. As a result, we can leverage a probabilistic header size to mitigate the dilemma between the number of setup packets and the bandwidth overhead.

When the source finds the total path length r is above 7, the source always selects a large value for m, e.g., 48, to keep the number of sent packets small. When r is below 7, the source still chooses a large m with probability P. The source selects a small m for the rest of the paths. In the scenario where $P = 0$, the algorithm is equivalent to using separate header sizes for long and short paths, which leaks path length information by packet header sizes. The source can select larger values for P to add stronger protection to conceal the fact that a long path is used.

To estimate path lengths r, the source can download an AS-level topology, e.g., from CAIDA [38], and run a local simulator to calculate the path length. The resulting path length is in turn an input into the algorithm to determine the size of the path. Note that a wrong estimate of r due to inaccurate simulation can potentially increase the collision rate and increase setup latency when the estimated r is smaller than the real path length.

4.3 Back-off Path Setup

To hide destinations' addresses from ASes that are close to the source, PHI uses helper nodes as indirection for source nodes to reach their destinations. Helper nodes can either be end hosts or ISPs that provide PHI indirection service. The information about which nodes serve as helper nodes can be distributed to end hosts or retrieved by end hosts without compromising anonymity of communication that happen afterwards.

PHI's assumptions are weaker than Dovetail: source nodes in PHI do not have to possess control over the path that packets traverse. This relaxed assumption allows PHI to operate on architectures like the Internet where a source lacks control over packets' paths.

In this subsection, we discuss the back-off path setup employed by PHI to support the relaxed assumption. We then analyze the resulting anonymity-set size of the proposed technique.

4.3.1 Path Setup Using Helper Nodes

Figure 4 shows how to establish an end-to-end path using a helper node in PHI. To construct a PHI path to reach the destination D, a source node S from AS0 selects a helper node M in AS4 and establishes a first half path towards M. Like LAP, S only places M's address into the setup request, so that each

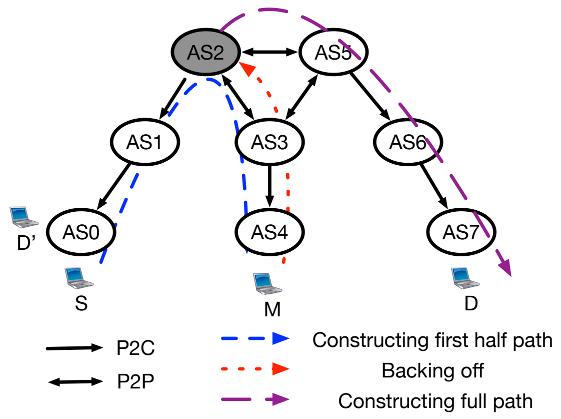


Fig. 4. Back-off path setup. The arrow standing for P2C relationship always points from the provider to its customer. The shaded node is the midway node. We call the path between S and M the first "half path" and the one between M and D the second "half path".

on-path AS can determine how to approach M. On-path nodes AS1 to AS4 follow the algorithm described in Section 4.2 to determine the positions of each individual path segment and insert them into the setup request (see Section 5).

In addition, S also encrypts D's address by using ECDH with M's public key. When M receives the setup request packet, it can obtain D's address and continue to establish the second half path to D.

Nevertheless, simply concatenating the first and second half paths to form the full path is undesirable. On one hand, M may not want to, or be able to, forward all traffic between S and D. On the other hand, the resulting long path incurs additional communication latency.

In PHI, M, on behalf of S, finds a node W on the first half path, called "midway node", that forwards data packets to D. In Figure 4, AS2 serves as the midway nodes that bridge the first half and second half paths together. Instead of traversing AS3 and AS4 twice, data packets can be directly forwarded towards D by AS4.

M uses a "back-off" technique to find the midway node with the network's help, while preserving each node's local policy. M sends a midway request packet (see Section 5) in the reverse direction of the first half path towards S. M also embeds D's address in the midway request packet. Each on-path node checks its previous hop and D's address, and decides whether it is the midway node based on its local policy. For example, AS4 cannot be the midway node because of the valley-freeness policy: it will not forward a packet from its provider AS4 back to AS4. In comparison, AS2 is the first node that can forward packets from AS1 to AS7 without breaking valley-freeness. Therefore, AS3 becomes the midway nodes. Once chosen, the midway node sends a path request to the D based on the received midway request.

PHI's header format inherently supports the above back-off technique because of its "automatic recycling" feature. When receiving a midway request packet containing the first half path, an on-path node can retrieve its previous path segment, verify its integrity, and decrypt information including the previous hop node. If the node decides not to become the midway node, its path segment in the header is automatically recycled by other nodes: another node can insert its path segment in the same position as an old segment to overwrite the legacy one.

Automatic recycling keeps header sizes small. As demonstrated in Section 4.2, to avoid setup failure caused by path-segment collisions, the header size increases exponentially with respect to the length of the path contained. With path-segment recycling, S only needs to consider the length of the longer path between the path from S to M and the one from M to D.

4.3.2 Maximizing Anonymity-set Size

To quantify each session's anonymity, we consider anonymity sets of source-destination pairs. Let $\mathscr{N}$ be the maximum anonymity set for source or destination that contains all possible end hosts, $\mathscr{A}(n)$, $n \in \mathscr{N}$ be the anonymity set of source-destination pairs that a compromised node n_i observes. If we further denote the anonymity set of the source node that an adversarial n observes $\mathscr{A}_S(n)$ and the anonymity set of the destination node that n observes $\mathscr{A}_D(n)$, we have the following relationship:

$$|\mathscr{A}(n)| = |\mathscr{A}_S(n)| \times |\mathscr{A}_D(n)| \tag{8}$$

where $|\cdot|$ is the size of a set.

We show that the midway node W, if compromised, can observe the smallest anonymity set. Consider $\mathscr{A}_S(n)$ for the first half path. It increases monotonically for the nodes in the order from S to M. For instance, in Figure 4, $\mathscr{A}_S(\text{AS3})$ is larger than $\mathscr{A}_S(\text{AS2})$, which in turn is larger than $\mathscr{A}_S(\text{AS1})$. For nodes in the second half path, $\mathscr{A}_S(n)$ is always $\mathscr{N}$. As for $\mathscr{A}_D(n)$, the resulting anonymity set is $\mathscr{N}$ for nodes before the midway point, but $\mathscr{A}_D(n)$ for nodes after M is $\{D\}$ because these nodes know D's address from the midway request and the second path request. Table 2 summarizes the anonymity-set size for each group of nodes.

Hence, we can use $\mathscr{A}(W)$ as a metric to measure the minimum level of anonymity that a session offers.

Choosing helper nodes. In fact, the source can determine which AS becomes the midway node by using AS-level network topology and AS-relationships available to the public [38], and obtain the anonymity set accordingly. Because the source node also possesses information about available middle nodes in the network, the source can improve the resulting anonymity-set size. The source first selects a subset of the helper nodes, computes the corresponding midway nodes and their respective anonymity-set sizes, and uses the helper node that yields the largest anonymity-set sizes.

However, it is dangerous for the source to use all or most of the available helper nodes to optimize the anonymity set, because an adversary located between the source and the midway node can reduce the anonymity set of the destination based on the selected helper nodes.

Node group	$\lvert\mathscr{A}_S\rvert$	$\lvert\mathscr{A}_D\rvert$
n **between** S **and** W	$1 \le \lvert\mathscr{A}_S(n)\rvert \le \lvert\mathscr{N}\rvert$	$\lvert\mathscr{N}\rvert$
W	$1 \le \lvert\mathscr{A}_S(W)\rvert \le \lvert\mathscr{N}\rvert$	**1**
n **between** W **and** M	$\lvert\mathscr{A}_S(W)\rvert \le \lvert\mathscr{A}_S(n)\rvert \le \lvert\mathscr{N}\rvert$	**1**
M	$\lvert\mathscr{N}\rvert$	**1**
n **between** W **and** D	$\lvert\mathscr{A}_S(W)\rvert \le \lvert\mathscr{A}_S(n)\rvert \le \lvert\mathscr{N}\rvert$	**1**

Table 2. Anonymity-set sizes observed by each group of nodes, if compromised.

4.4 Integrating Payload Encryption

PHI explicitly requires payload encryption. To defend against session hijacking attacks, PHI binds a source's DH public key to an established path. Upon setting up each session, the source node randomly generates a pair of DH keys for negotiating shared symmetric keys with the midway node and the destination. As opposed to Dovetail, PHI uses the hash of public key as the session identifier to bind the encryption key to the session itself. As a result, an adversary that hijacks the connection by simply replaying headers with the adversary's payload will be detected immediately by the destination.

Not only the destination, but also the midway node needs to check that the session identifier is associated with the source node's public key. If the midway node fails to verify that the session identifier matches the public key encrypting the destination's address, a node between the source node and the midway node can still hijack the session setup process: the midway node can replace the session identifier with the hash of its own public key and circumvent the destination's verification.

Note that the midway node and the on-path nodes after the midway node can still launch hijacking attacks. During setup sessions, they can replace the session identifiers with hashes of their own public keys. However, these nodes have no incentives to launch such attacks because they already obtain full knowledge about the destinations, shown in Table 2.

5 PHI Protocol Details

In this section, we describe PHI's packet format and its session setup phase in detail. We also briefly state how PHI achieves sender-receiver anonymity.

5.1 Packet Format

There are 4 types of packets in PHI: path requests, midway requests, path replies, and forward/backward data packets. As a network-layer protocol, all PHI headers are placed after the respective layer 2 headers. A path request allows a node to create or extend a path towards its intended entity while concealing its network location. A source node uses path request to create the first half path to the helper node that it chooses. A midway node uses the path request to extend the path to reach the destination to create a full path. A midway request, as its name indicates, is used by the helper node to perform the back-off method described in Section 4.3 to find the midway node. When a destination receives a full path established from the source, the destination uses a path reply to send the path back to the source so that they can start transmitting actual data using data packets. Appendix A.1 graphs a reference packet header format in detail.

5.2 Managing Keys

Each node in the network maintains a master secret key, from which it can derive three keys: a position key k^{pos}, an encryption key k^{enc}, and a MAC key k^{mac}. The position key is used by the node to determine the position of its path segment in a packet header. The encryption key encrypts the routing information in a path segment. The MAC key computes a MAC that helps a node to verify its path segment.

The destination node needs to generate a pair of DH keys. To communicate anonymously with the destination, a source node needs to obtain the destination's public key through an out-of-band channel. In order for a source to communicate with a destination in a new session, the source also needs to generate a new pair of DH keys for the session to negotiate with the destination a shared symmetric key to encrypt data payloads.

5.3 PHI Session Setup

There are four steps in PHI's session setup phase. First, the source node selects a helper node and establishes a half path to the helper node. Second, the helper node uses the back-off technique to search for a midway node. Third, the midway node extends the existing path to become a full end-to-end path between the source and its destination. Finally, the destination sends the created path back to the source. In this section, we walk through details of all four steps in an Internet setting. We remark the same process can be adapted to other network architectures.

Constructing a path to the helper node. To communicate with a destination node D, the source node S first selects a helper node M. If it is the first time for S to use M, S also needs to retrieve M's public key with its associate certificate. For example, S can obtain the information about the helper nodes' public keys from directory services like those employed in Tor [27]. S needs to verify M's public key before using it.

S then creates a path request. S first randomly generates a pair of DH keys for the session and uses the hash of the public key as the session identifier. Then, S generates random bits to initiate the path. S also encrypts the destination address using M's public key and puts the resulting ciphertext as the packet's payload. At last, S fills the address field in the path request header with M's address and sends it towards M.

Depending on the required success rate, S repeats the above process to create a number of setup packets in case that a single setup packet cannot achieve the desired success rate. S sends all setup packets to M together and then waits for responses. In addition, because the setup process can fail, S keeps a timer so that it can resend setup packets.

When receiving a path request, each on-path node follows the procedures described in Section 4.2. Based on the address specified in the packet header, an on-path node first creates its path segment following Equation 4. Then the node determines its path segment's position in the header field by Equation 1 and replaces the current path-segment with its own. Afterwards, the node can proceed to forward the packet towards M. The same process repeats itself until the packet reaches M.

Back-off to find the midway node. After M receives a path request from S, M first verifies that the session identifier is indeed the hash of the session's public key in the payload. If so, M decrypts D's address. M creates a midway request packet by placing D's address in the header's address field and using the path in the path request. M then sends the midway request back to the AS where the path request comes from.

An on-path node receiving the midway request first retrieves its segment inserted into the path. It decrypts the routing information and checks the corresponding MAC. If the MAC in the path segment successfully verifies, the node decides whether it is the midway node based on D's address, the previous-hop AS, and its local policy. If becoming a midway node violates its policy, it simply forwards the midway request toward S using the ingress port in the path segment.

Create a full path. If a node decides to be the midway node, the node creates a new path segment. It changes the egress port field to the one towards D, and sets the *midway* flag in the segment. Then, it replaces the current segment with its newly created one. Finally, it creates a new path request that addresses to D's and contains the path and the session's public key, and forwards it to D.

Nodes between the midway node and D process the path request in the same way as the nodes between S and M. They calculate the positions of their path segments, insert their new path segments accordingly, and forward the packet.

Path replies. When D receives the path request sent by the midway node, D first verifies that the public key matches the session identifier. Then, D conducts a DH key exchange operation to generate a symmetric key shared with S. The key is used as the master key to encrypt payloads of subsequent data packets. D uses the path composed in the forward path to send a path reply back to S. Every on-path node, including the midway node, retrieves its path segment, obtains its routing information, and forwards the packet towards S. Finally, when the path reply reaches S, S can retrieve the path and construct data packets with it.

5.4 Sender-Receiver Anonymity

PHI achieves sender-receiver anonymity defined by Pfitzmann and Köhntopp [44] using an indirection node, called rendezvous node. In order for a destination node to conceal its identity, it first chooses an available rendezvous node and establishes a session with it. After obtaining the path between the rendezvous node and itself, the destination node can publish the path together with the address of the rendezous node through an out-of-band channel, like a website, so that a source node intending to connect to the destination node can retrieve the path.

For a source to connect to the anonymous destination, after fetching the path and the rendezvous node's address, the source first establishes a path between the rendezvous node and itself. Then the source can place both two paths in its data packets sent to the destination node. When the rendezvous node receives a data packet from the source, it forwards the packet following the path established by the destination.

6 Security Analysis

In this section, we first quantitatively analyze in detail the security of PHI compared to VSS used in LAP and Dovetail under topology-based attacks. Then, we discuss PHI's defense mechanisms against various known active and passive attacks.

6.1 Defending Against Topology-based Attacks

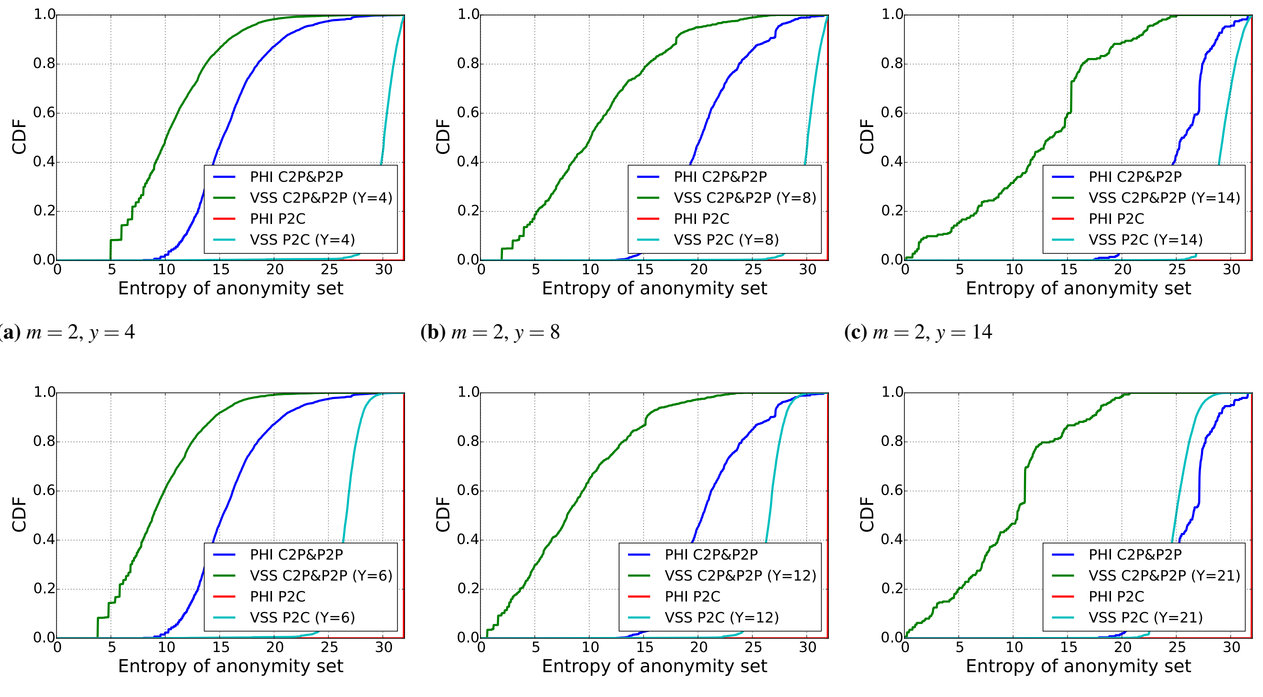
We compare the probability that an adversarial AS successfully discovers a packet's source node in PHI's segment-position randomization method and VSS. VSS differs from segment-position randomization in that it provides to the adversary additional information about the number blocks already inserted. To quantitatively compare between PHI and VSS's defenses against topology-based attacks, we conduct an experiment with the Internet topology considering the IPv4 address space.

Modeling anonymity-set size for VSS technique. We model the probability $P(x=S|Y=y,m=l)$ that an adversarial AS can locate a source node S given that the number of blocks in a header $Y=y$ and the maximal number of blocks that can be occupied by a path segment $m=l$. $P(D=d)$ is the distribution of distances between the source of adversarial ASes. $P(Y=y|D=d,m=l)$ is the conditional probability of d nodes output a sequence of path segments with total length y. $P(x=S|D=d)$ is the probability that the source can locate S if it is d hop away from S. d_{max} is the maximal diameter of the network. We use the following equation to compute $P(x=S|Y=y,m=l)$ from $P(D=d)$, $P(Y=y|D=d,m=l)$, and $P(x=S|D=d)$:

$$\begin{aligned} P(x=S|Y=y,m=l) &= \frac{P(x=S|m=l)}{P(Y=y|m=l)} \\ &= \frac{\sum_{d=1}^{d_{max}} P(x=S|D=d,m=l)P(D=d)}{\sum_{d=1}^{d_{max}} P(Y=y|D=d,m=l)P(D=d)} \\ &= \frac{\sum_{d=1}^{d_{max}} P(x=S|D=d)P(D=d)}{\sum_{d=1}^{d_{max}} P(Y=y|D=d,m=l)P(D=d)} \end{aligned} \quad (9)$$

We define the equivalent anonymity-set size observed by the adversarial AS as $\frac{1}{P(x=S|Y=y,m=l)}$.

Experiment setup. We use the CAIDA AS-relationship dataset [1] and the RouteView dataset [10] to construct an AS-level Internet topology annotated with both AS relationship and each AS's address space sizes. In our experiment, for each AS α, we compute the anonymity set observed by a neighboring AS β receiving packets from α in one of three cases: 1) β is α's provider (C2P), 2) β is α's peer (P2P), and 3) β is



(a) $m = 2, y = 4$ **(b)** $m = 2, y = 8$ **(c)** $m = 2, y = 14$

(d) $m = 3, y = 6$ **(e)** $m = 3, y = 12$ **(f)** $m = 3, y = 21$

Fig. 5. Cumulative Distribution Functions (CDF) of entropy in VSS and PHI. In VSS, we vary the maximal number of path-segment blocks per hop, denoted as m, and the observed total number of path-segment blocks Y. We also distinguish between the case where the adversary received a packet from its customer (C2P) or peer (P2P) and the case where the adversary received a packet from its provider (P2C). We consider IPv4's address space, so the maximal entropy is 32. Note 99.9% of paths are less than 8 hops long according to Section 7.

α's customer (P2C). In our experiment, we assume that every IPv4 address has equal probability to send the packet. As a result, the anonymity-set sizes are calculated as the sizes of IPv4 address spaces. We also assume that ASes obey the valley-free policy when forwarding packets. We compare PHI and VSS by the cumulative distribution of the anonymity-set sizes yielded by both schemes.

To compute anonymity-set sizes observed by AS β when using PHI, we add the anonymity-set sizes of all ASes that can send packets traversing the link from α to β without breaking the valley-free policy. For example, in Figure 1, if AS2 is α, AS3 is β, and α is β's customer, the anonymity-set size observed by β in this case is the sum of the IPv4-address-space sizes of all customers of AS2 including AS2 itself, i.e., {AS0, AS1, AS2, AS4, AS5}. For VSS, we compute the equivalent anonymity-set size observed by β when receiving a packet from α.

We can compute $P(x = S|D = d)$ directly from the topology by add up the anonymity-set sizes of the ASes that are d hops away from β and can send packets through link $\alpha \to \beta$ following the valley-free policy. For example, in Figure 1, if AS3 is AS2's provider and it is malicious, and $d = 3$, the anonymity-set size is simply the AS0's address-space size. $P(D = d)$ is inversely proportional to $P(x = S|D = d)$ because we assume that every IPv4 address has the same probability to originate the packet. Finally, if we additionally assume that each AS uniformly and independently picks the size of its path segment, we can derive $P(Y = y|D = d, m = l)$ by combinational algebra.

Results. Figure 5 graphs the resulting CDF of observed anonymity-set sizes in the form of entropy. Each sub-figure draws the CDF of PHI and VSS with different m and Y. We also vary inter-AS relationship between malicious ASes and their neighbors. The results from C2P and P2P are exactly the same, therefore we use the same line to represent both of them in each sub-figure. For fairness, when calculating PHI's CDF in each sub-figure, we only consider β who can actually observe that $Y = y$ when $M = m$.

In general, PHI provides 30–60000 times larger anonymity sets than VSS when the relationship is C2P or P2P, and 4–60 times larger anonymity sets than VSS when the relationship is P2C. The differences generally enlarge with larger values of m and y. When $m = 2$ and $y = 4$, PHI's anonymity-set size is around 30 times larger than VSS's for

C2P or P2P, and 4 times larger for P2C. When y increases, the gap grows wider. In fact, when $m = 2$ and $y = 14$, PHI offers a 250 times larger anonymity-set size than VSS for C2P or P2P, and 5-6 times larger for P2C. The gap further increases with a larger m. When $m = 3$ and $y = 21$, PHI provides an anonymity set that is almost 60000 times larger than those for C2P & P2P, and 70 times larger for P2C.

We also observe that PHI provides an anonymity-set size of almost 2^{32} when the AS relationship is P2C. This makes sense in a well connected network like the Internet. For example, in Figure 1, if AS2 is AS3's provider and AS3 is compromised, every packet that AS2 sends to AS3 can potentially come from the entire Internet with the valley-free policy preserved. The observation implies that choosing the destination node's providers or topological ancestors can be as secure as selecting an arbitrary third party end host without breaking the valley-freeness rule and introducing a longer path.

Impact of inaccurate simulation. As described in Section 4.3.2, the anonymity-set size depends on the anonymity-set size of the source node observed by the midway nodes. Thus, an inaccurate topology simulation might direct the source node to pick a wrong helper node, and thus a wrong midway node, which can potentially reduce the anonymity-set size.

6.2 Attacks Against Anonymity

With respect to the case where an adversary only controls a single AS, PHI can resist various passive or active attacks against anonymity besides the topology-based attacks discussed in Section 6.1. By passive attacks, we refer to the attacks that only require observation or logging of the traversing packets. In comparison, an adversary that conducts active actions can delay, drop, or modify traversing packets, to impact the anonymous traffic.

PHI defends against the following passive attacks in addition to topology-based attacks:

– **Observing packet payloads**. Packet payloads contain various identifiers, such as browser cookies, which can help correlate packet flows and deanonymize users. PHI defeats such attacks by requiring the source node to encrypt end-to-end all the traffic between itself and the destination node. In addition, all data packets are padded to a fixed size to prevent information leakage in PHI.
– **Session linkage**. If an adversary correlates sessions initiated by the same source node, the adversary can profile the user and subsequently deanonymize him or her. PHI prevents this attack by requiring the source to generate a random session identifier for each session. Hence, even if multiple sessions from the same source node follow the same path, the resulting path segments differ. Moreover, the source also generates a new pair of keys for end-to-end encryption to prevent session linkage from a set of colluding destinations.

PHI also defeats the following active attacks:

– **Session hijacking attacks**. A compromised on-path node can hijack a session by replacing the session's public key with its own. PHI derives the payload encryption key from the session's public key, which in turn is bound to the destination address. For nodes between a source node and a helper node, to whom the destination address is unknown, replacing the session's public key is detected by the helper node when decrypting the destination's address is unsuccessful.
– **Packet-header modification**. Adversarial nodes can modify packet headers to change the paths that the packets traverse. If the adversary directly modifies the path embedded in the packets, benign nodes can detect the changes because the verification of the MAC in its path segment fails. During the session setup, an adversary can also attempt to modify the address field in a path request or a midway request. Nevertheless, without colluding ASes, modifying the address field only disrupts the session setups, but does not reduce the observed anonymity set.
– **Replay packets**. An adversary node can replay packets. However, the replayed packets only traverse the same path that the previous packets traverse, yielding no new information about the either end of the session.
– **Pre-computation attacks**. An adversary can send many packets with different session identifiers to determine the positions of AS segments. PHI defeats the pre-computation attack by requiring that sources generate fresh session identifiers and ASes update their master keys periodically.

7 Evaluation

In this section, we first simulate and compare the overhead of PHI and VSS with Internet data traces from iPlane [7] and CAIDA [12]. Then we evaluate the performance of HORNET, LAP, Dovetail, and PHI on a high-speed software router.

7.1 Comparison of Overhead between PHI and LAP with VSS

We simulate the bandwidth overhead of both PHI and VSS-based LAP using iPlane traceroute data [7] and CAIDA anonymized Internet traces obtained by monitoring on high-speed backbone links [12]. Our CAIDA data traces include all traffic from a backbone link between 13:00 to 14:00 on Mar. 20th, 2014.

Path lengths. We first analyze the distribution of the number of setup packets required by PHI. We obtain the distribution of total number of hops between source and destination by using iPlane traceroute data. For each trace, we count the number of AS hops. Our results show that the mean AS-path length is 4.2, and 99.99% of paths are within 7 AS hops. Thus, in this paper, we choose 7 as the maximum length of a normal AS path, and paths with lengths larger than 7 are long paths. With regard to paths longer than 7, setting $m = 48$ can achieves 90% success rate when 5 setup packets are used.

Packet-header sizes. We use CAIDA anonymized Internet traces to compute the probability that a setup phase succeeds given different packet-header sizes. We vary the maximum number of path segments m and apply Equation 7 to each flow of AS path length r. Our result show that when 4 setup packets are used, a setup phase succeeds with probability 75% when $m = 8$. The probability of success increases to 90% when $m = 12$.

Bandwidth Overhead. We then evaluate the bandwidth overhead caused by larger headers in VSS compared to the overhead of additional setup requests in PHI. We conduct a trace-based simulation of both schemes using our CAIDA dataset. We assume all the flows in the dataset are converted to PHI and VSS sessions and we compute the extra bandwidth overhead introduced by both schemes with different configurations. Figure 6a demonstrates the resulting bandwidth cost. For PHI, we vary two parameters: the number of path segments that a normal header contains m and the probability that a larger header $m = 48$ is used for a short path P. We also test the bandwidth cost of VSS with different packet header sizes corresponding to an amplification factor of 2 to 4.

We observe that the bandwidth overhead increases as m and P get larger. With small headers where $m = 8$ and $P = 0.01$, PHI's bandwidth cost is 43% smaller than VSS with an amplification factor $A = 2$. When end hosts increase P to offer stronger security for sessions with longer paths, the bandwidth cost is still small. When $m = 8$ and $P = 0.2$, the bandwidth cost of PHI is only as large as that of VSS with $A = 2$ and at least 65% smaller than VSS with larger A. When latency of setup phases is concerned and $m = 12$ is used, the bandwidth cost of PHI is 11% smaller than VSS with $A = 2$ when $P = 0.01$. In the following evaluation, we choose $m = 12$ for a normal header and $m = 48$ for a large header.

7.2 Performance Evaluation

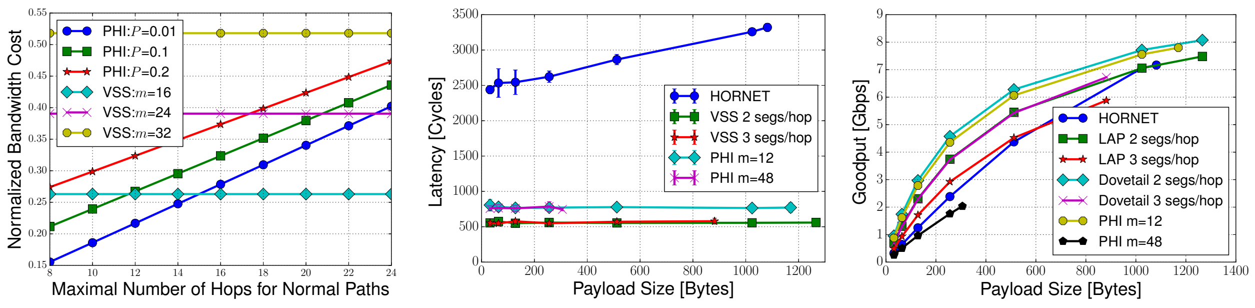
Implementation and testbed setup. We implement PHI on a high-speed software router using Data-Plane Development Kit (DPDK) (version 2.1.0) [3]. For comparison, we also implement HORNET, LAP, and Dovetail in the same setup.

To accelerate the underlying cryptography, we use Intel AESNI technology. In our implementation, we use 128-bit AES in CBC mode and 128-bit AES CBCMAC as our respective encryption and MAC functions. To implement the DH key exchange protocol and public key cryptography in HORNET, we use curve25519-donna library [2].

We test our implementation on a testbed composed of a high-speed software router and a Spirent TestCenter traffic generator/monitor [11], which are connected by twelve 10 Gbps links. Our software router has Intel Xeon E5-2680 CPU with 2 sockets and 8 cores per socket, 64 GB RAM, and 3 Intel 82599ES NICs supporting in total 120 Gbps maximal throughput. In all of our experiments, we configure our DPDK library to assign 1 CPU core to each port to process incoming packets.

Setup latency. We first measure the latency of processing setup requests in HORNET and in PHI. Table 3 shows the latency for processing setup requests for HORNET and PHI with different packet sizes. We observe that processing a HORNET session request requires almost 800 times more time than processing a PHI because the former needs a DH key operation but the latter only needs symmetric cryptography. Moreover, processing setup requests of different sizes incur similar latency, because the amount of computation needed is independent from header sizes. A single core can process PHI setup requests/replies at a speed of up to 3.76 Mpkt/s on a single core. According to our analysis of the CAIDA data trace used in Section 7.1, the average number of flow initiation on a 40 Gbps backbone link is 1721 per second. Hence, PHI can potentially satisfy the requirement of future traffic at today's budget.

Data-forwarding latency. We evaluate the latency for forwarding data packets with various payload sizes by a single core on a 10 Gbps link using PHI, VSS, and HORNET. We base our comparison on LAP using VSS to hide path information. The latency of Dovetail data forwarding should yield similar results because both LAP and Dovetail require decrypting a single block and check a MAC.



(a) Bandwidth cost **(b)** Latency **(c)** Goodput

Fig. 6. a) **Bandwidth cost of PHI under different configuration**. Lower is better. For PHI, we vary the number of path segments m that a header can contain for each line. Different lines have different probability P to select a large header for a short path. We fix the size of large headers so that they can contain 24 path segments. For comparison, we also draw VSS's bandwidth cost with different header sizes. All the bandwidth cost is normalized by dividing the bandwidth cost without anonymity. b) **Latency of processing data packets in HORNET, VSS, and PHI when different payload sizes vary**. Lower is better. c) **Goodput of HORNET, LAP, Dovetail, and PHI with different payload sizes on a 10 Gbps link**. Higher is better. For VSS, we test the configurations of 2 path segments per hop and 3 path segments per hop. For PHI, we evaluate the cases where m equals 12 and 48 respectively.

Scheme	Cycles	μs
HORNET	(6.60 $\pm$ 0.28) $\times 10^5$	(2.00 $\pm$ 0.09) $\times 10^2$
PHI ($m = 12$)	821 $\pm$ 83	0.249 $\pm$ 0.025
PHI ($m = 48$)	962 $\pm$ 132	0.291 $\pm$ 0.040

Table 3. Latency for processing setup packets in HORNET and PHI.

As shown in Figure 6b, in general, the latency needed by processing a HORNET data packet is 3 times higher than that of PHI. The latency increases linearly when the payload size grows, because HORNET data forwarding process encrypts or decrypts the entire header and payload. With respect to the comparison between PHI and VSS, PHI consumes on average 30% more time to forward a data packet than VSS as PHI requires additional PRF computation to locate the current path segment within the header. However, differing from HORNET, the computation required by PHI and VSS is constant and does not increase together with enlarging payloads.

Goodput. We also measure the goodput of different schemes on a 10 Gbps link with respect to various payload sizes. Figure 6c graphs the results. There are two factors impacting the goodput of a protocol: forwarding speed and header sizes. Higher forwarding speed yields higher goodput while smaller header sizes help increase goodput. Yet, there is a third factor in our experiment that limits the goodput of LAP, Dovetail, and PHI: the maximal transmission rate on the link. In fact, LAP, Dovetail, and PHI can achieve 10 Gbps even with the payload size as small as 32. Hence, only the header size will influence their goodput.

We observe that PHI with $m = 12$ and Dovetail using 2 path segments per hop achieves the highest goodput among all the evaluated schemes because their efficient data forwarding and small headers. Compared to LAP with 2 path segments per hop, they achieve 18% and 12% more goodput. As larger headers are used, all three schemes' goodput decreases. On the other hand, because HORNET's throughput is lower than the maximal throughput, its goodput increases faster as the payload sizes increase. In general, HORNET's goodput is 60% lower than PHI with $m = 12$.

Maximum throughput. Lastly, we evaluate the maximum throughput on all 12 ports, which can possibly switch packets at the speed of 120 Gbps. We observe that LAP, Dovetail can fully saturate the 120 Gbps even with the smallest packets possible for individual protocols. In comparison, HORNET can only provide at maximum 90 Gbps with 1500-byte packets. But this number drops to around 50 Gbps when the packets contain payloads of 32 bytes. PHI's maximum throughput is very close to LAP and Dovetail. PHI can offer 106 Gbps with 32-byte payloads, and it can saturate 120 Gbps when using payloads larger than 128 bytes. In summary, PHI data forwarding can achieve similar maximum throughput as those of LAP and Dovetail, and can offer 35–200% speed-up against HORNET depending on packet sizes.

Limitations. The performance results from our single-node testbed indeed apply to a larger testbed, because the amount of computation required by forwarding is independent of traffic types or a node's position in the network. However, there are several limitations of our single-node evaluation compared to an end-to-end evaluation on a larger testbed. First, given the small computation latency of PHI, propagation latency will dominate end-to-end latency. An end-to-end evaluation will better demonstrate user-experienced latency. Second, the single-node evaluation cannot reflect real networks' heterogeneity such as connectivity and throughput.

8 Discussion

RTT-based timing attacks. The PHI protocol provides an efficient method to hide path information in packet headers. However, because PHI is a low-latency anonymity protocol, an on-path AS can potentially employ RTT-based timing attacks to measure its distance from the source or the destination [32]. Although PHI itself cannot defeat such a side-channel attack using RTT information, employing PHI poses a few challenges for such RTT-based timing attacks to function. First, RTT-based timing attacks require that the packets from vantage points to traverse the same path as the victim traffic. However, more and more ISPs begin to deploy load-balancing in the network [15] which undermines these attacks' assumption. Furthermore, network architectures, such as Pathlet [30] and SCION [50], provide high path diversity, which additionally reduces the possibility that victim traffic paths are traversed. Second, PHI enables sources and destinations to use other mechanisms to defeat RTT-based attacks. For example, a source and its destination can insert random delays into the timing information to obscure real RTTs. Finally, large-scale anonymity systems like PHI require the adversary to control vantage points located in a large number of ASes, which frustrates attackers with limited resource budgets.

Deployment of PHI. Prerequisites for deploying network-layer anonymity systems range from updating routing hardware to implementing Future Internet Architectures (FIA). On one end of the spectrum, as demonstrated in our evaluation, advances in incorporating cryptographic functions into hardware, such as Intel AESNI [5], have shown promising results for high-speed data forwarding, implying the possibility of deploying PHI on core routers. On the other end, unlike Dovetail and HORNET, PHI adapts well to both current BGP routing and FIAs by assuming no control of packet routes from end hosts, thus excluding the roadblock of FIA deployment.

Regarding deployment motivation, note that an ISP can advertise anonymity as a service to its customers as a competitive strategy. Both private and business customers may be willing to incur an additional cost for privacy, they might choose an ISP that offers privacy protection over one that does not. With respect to incremental deployment, a few ISPs that deploy PHI can establish tunnels between each other and begin to provide privacy services for their customers.

Integration with upper-layer anonymity tools. As a network-layer anonymity protocol, PHI concentrates on providing intermediate-level anonymity against topological attacks and achieving high throughput and low latency. Users who desire stronger anonymity can employ higher-layer anonymity tools, such as onion-routing networks [6, 8, 14], or Mix networks [19, 41]. In particular, when the source node is close to its intended destination in the network topology, e.g., they share the same AS, the source should at least use a VPN-like service to redirect its packets to avoid de-anonymization.

Attacks using multiple nodes. PHI is a network-layer anonymity system, which in general cannot defend against attacks using multiple colluding nodes. These attacks range from sophisticated traffic analysis [17, 33, 39, 40] (which are demonstrated effective against stronger anonymity protocol [27] by the research community), to identifier-based information aggregation (where identifiers about users, e.g., cookies, are used to filter and aggregate traffic belonging to the same user for more expensive analysis) [4].

We remark that PHI defends against the latter real-world attack by preventing identifier extraction. PHI employs end-to-end payload encryption to prevent application data from exposing application-level identifiers. In addition, PHI's session identifier can only be used to group traffic within the same flow, but cannot be linked to either a specific source node or its destination. By forcing adversaries to resort to more expensive techniques, PHI can help thwart mass surveillance.

9 Related Work

Anonymity systems using overlay networks. Chaum pioneered the first "Mix network" that anonymizes users' traffic in the Internet [20] using layered encryption, packet batching, and packet mixing. Several Mix networks have since been implemented and deployed based on Chaum's initial ideas [24, 25, 31, 41]. Typically, they offer strong anonymity properties but incur prohibitively high latency and computation for real-time applications.

Shifting the balance between security and usability, a number of low-latency onion-routing networks [16, 27] propose to create circuits that limit the usage of asymmetric cryptographic operations during circuit setup. But these systems are vulnerable to traffic analysis attacks [22, 40, 42, 43].

The research community also seeks to simultaneously achieve both strong anonymity and high usability. P^5 [47] and Tarzan [29] propose to organize onion routers as a P2P network, where neighboring routers obscure real traffic patterns by adding chaff traffic. Aqua [36] divides the network into an edge network and a core network. To thwart traffic analysis, it forms broadcast channels for users in each edge network and uses multi-path communication together with chaff traffic for the core network. Dissent [48] relies on DC-nets [18] and verifiable shuffles to defend against traffic analysis. Finally, Drac [23] and Herd [35] focus on offering traffic-analysis resistance for applications with low-volume communication.

Network-layer anonymity systems. Recently, the research community has advocated building the anonymous communication as a principal function of underlying network architectures, and a number of *network-layer anonymity systems* have been proposed since [21, 26, 34, 37, 46]. LAP [34] and Dovetail [46] provide a medium-level anonymity by requiring each router to encrypt its routing information in packet headers to conceal network locations of end hosts. The resulting anonymity guarantee is weaker than that offered by onion routing systems, due to lack of bit-content unlinkability. However, LAP and Dovetail enable high-speed data forwarding and can scale to support Internet-scale usage. Tor instead of IP [37] and HORNET [21] explore the possibility for routers to adopt onion routing to anonymize traversing traffic. HORNET demonstrated that it is viable to provide high-speed onion routing on network devices. Nevertheless, compared to LAP and Dovetail, HORNET introduces longer latency and requires more computation. In particular, HORNET's session setup requires public-key cryptographic operations that are expensive and can be used as a DoS vector.

10 Conclusion

We introduce PHI, a new lightweight network-layer anonymity protocol that enhances anonymity and compatibility over previous high-efficient protocols and maintains the same level of efficiency. PHI leverages a new cryptographic header format that efficiently hides nodes' positions, and a new back-off setup method that uses a helper node and requires no prior knowledge about network topology on end hosts. A prototype implementation of PHI demonstrates its capability to saturate 120 Gbps links on commodity hardware, evincing its potential to anonymize high volume of real-time traffic in the Internet.

11 Acknowledgments

We would like to thank our shepherd Boon Thau Loo and the anonymous PETS reviewers for their excellent suggestions. We thank Cristina Basescu for discussions about the initial ideas of packet formats and protocols. We are also grateful to Daniele E. Asoni, David Barrera, and the members of the ETH Zurich Network Security group for their inspiring feedback.

The research leading to these results has received funding from the European Research Council under the European Union's Seventh Framework Programme (FP7/2007-2013) / ERC grant agreement 617605. We also gratefully acknowledge support from ETH Zurich and from the Zurich Information Security and Privacy Center (ZISC).

A Appendix

A.1 Reference Packet Header Format

Figure 7 demonstrates a reference packet header formats for PHI headers, which we use in our prototype implementation.

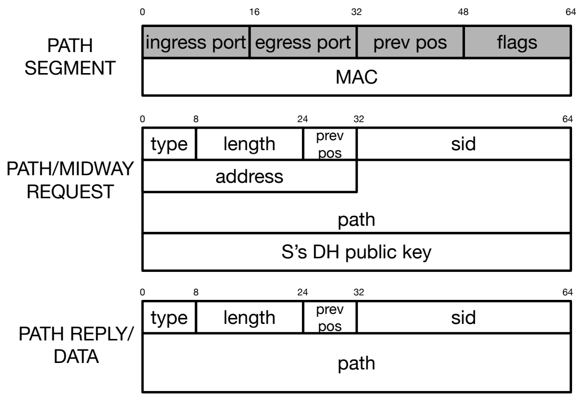


Fig. 7. Formats of a path segment and packet headers. The shaded fields are encrypted using a node's local secret key.

Common header fields. In all PHI packet headers, there are 4 common fields: type, length, previous hop's position, and a session identifier. The type field specifies which type the PHI packet header belongs and whether it uses a normal path or a long path. The length field describes how long the header is. The previous-hop-position field records the position of the last-hop node's path segment so that the current node can verify its path segment. Lastly, the session identifier helps identify which session the packet header belongs to and should be randomly generated by a source node.

Path. Another basic building block of PHI headers shared by all types of PHI packets is a PHI path. A PHI path contains a sequence of path segments. As shown in Figure 7, a path segment is a 16-byte block containing an 8-byte encrypted information block and an 8-byte MAC. An encrypted information block contains an ingress port field, an egress port field, a previous-hop position field, and flags. The first two fields help a node keep records of where to forward a packet, and the previous-hop field assists a node in finding the path segment of the previous-hop node so that the node can verify its MAC. Lastly, the flags help a node to keep record of its role. For example, the midway node sets the "midway flag" so that it remembers its role when processing a data packet.

Path/midway requests. In addition to common headers, both path and midway request headers also have a 4-byte address field, a path, and the source node's DH public key. For a path request from a source to a helper node, the source node fills the address field with the helper node's address so that on-path nodes can decide how to forward the path request. The actual destination address is encrypted using the shared key between the source node and the helper node, and appended to the path request. For a path request from a midway node to the destination, the address field contains the destination's address. The address field of a midway request also contains the destination address, which is used in the back-off technique for an on-path node to decide whether it is the midway node. With respect to the DH public key, both the helper node and the destination node use it to derive shared symmetric keys with the source node. We use ECDH key exchange protocol for small public key sizes.

Path replies/data packets. A path replies and a data packet share the same header format. The only field in addition to the common header is an end-to-end path. The selected destination and the midway node are already implicitly specified in a path. Therefore, there is no need to specify their addresses explicitly.

References

[1] CAIDA AS-relationship dataset. http://www.caida.org/data/as-relationships/.

[2] curve25519-donna. https://code.google.com/p/curve25519-donna/.

[3] DPDK: Data Plane Development Kit. http://dpdk.org/.

[4] Global surveillance disclosures (2013-present). https://en.wikipedia.org/wiki/Global_surveillance_disclosures_(2013%E2%80%93present).

[5] Intel AESNI Sample Library. https://software.intel.com/en-us/articles/download-the-intel-aesni-sample-library.

[6] The invisible internet project. https://geti2p.net/en/.

[7] iPlane dataset. http://iplane.cs.washington.edu/data/data.html.

[8] JonDonym anonymous proxy servers. https://anon.inf.tu-dresden.de/index_en.html.

[9] NSA collecting phone records of millions of verizon customers daily. http://www.theguardian.com/world/2013/jun/06/nsa-phone-records-verizon-court-order.

[10] RouteView project. http://www.routeviews.org/.

[11] Spirent TestCenter. http://www.spirent.com/~/media/Datasheets/Broadband/PAB/SpirentTestCenter/STC_Packet_Generator-Analyzer_BasePackage_datasheet.pdf.

[12] The CAIDA UCSD Anonymized Internet Traces 2015 - equinix-chicago 2015-01-20. http://www.caida.org/data/passive/passive_2015_dataset.xml.

[13] Tor metrics: Ddrect users by country. "https://metrics.torproject.org/userstats-relay-country.html. Retrieved on Nov.3, 2015.

[14] Tor project. https://www.torproject.org/.

[15] B. Augustin, X. Cuvellier, B. Orgogozo, F. Viger, T. Friedman, M. Latapy, C. Magnien, and R. Teixeira. Avoiding traceroute anomalies with Paris traceroute. In *ACM IMC*, 2006.

[16] O. Berthold, H. Federrath, and S. Köpsell. Web mixes: A system for anonymous and unobservable internet access. In *PETS*, 2001.

[17] S. Chakravarty, M. V. Barbera, G. Portokalidis, M. Polychronakis, and A. D. Keromytis. On the effectiveness of traffic analysis against anonymity networks using flow records. In *PAM*, 2014.

[18] D. Chaum. The dining cryptographers problem: Unconditional sender and recipient untraceability. *Journal of cryptology*, 1(1):65–75, 1988.

[19] D. Chaum, F. Javani, A. Kate, A. Krasnova, J. de Ruiter, and A. T. Sherman. cMix: Anonymization by high-performance scalable mixing. Technical report, 2016. https://eprint.iacr.org/2016/008.

[20] D. L. Chaum. Untraceable electronic mail, return addresses, and digital pseudonyms. *Commun. ACM*, 24(2):84–88, Feb. 1981.

[21] C. Chen, D. E. Asoni, D. Barrera, G. Danezis, and A. Perrig. HORNET: High-speed onion routing at the network layer. In *ACM CCS*, 2015.

[22] G. Danezis. The traffic analysis of continuous-time mixes. In *PETS*, 2004.

[23] G. Danezis, C. Diaz, C. Troncoso, and B. Laurie. Drac: An architecture for anonymous low-volume communications. In *PETS*, 2010.

[24] G. Danezis, R. Dingledine, and N. Mathewson. Mixminion: Design of a type III anonymous remailer protocol. In *IEEE S&P*, 2003.

[25] G. Danezis and I. Goldberg. Sphinx: A compact and provably secure mix format. In *IEEE S&P*, 2009.

[26] S. DiBenedetto, P. Gasti, G. Tsudik, and E. Uzun. ANDaNA: Anonymous named data networking application. *arXiv preprint arXiv:1112.2205*, 2011.

[27] R. Dingledine, N. Mathewson, and P. Syverson. Tor: The second-generation onion router. In *USENIX Security*, 2004.

[28] R. Dingledine and S. J. Murdoch. Performance improvements on Tor or, why Tor is slow and what we're going to do about it. *Online: http://www.torproject.org/press/presskit/2009-03-11-performance.pdf*, 2009.

[29] M. J. Freedman and R. Morris. Tarzan: A peer-to-peer anonymizing network layer. In *ACM CCS*, 2002.

[30] P. Godfrey, I. Ganichev, S. Shenker, and I. Stoica. Pathlet routing. *ACM SIGCOMM CCR*, 39(4):111–122, 2009.

[31] C. Gülcü and G. Tsudik. Mixing email with babel. In *NDSS*, 1996.

[32] N. Hopper, E. Y. Vasserman, and E. Chan-Tin. How much anonymity does network latency leak? *ACM Transactions on Information and System Security*, 13(2), February 2010.

[33] A. Houmansadr, N. Kiyavash, and N. Borisov. RAINBOW: A robust and invisible non-blind watermark for network flows. In *NDSS*, 2009.

[34] H. C. Hsiao, T. H. J. Kim, A. Perrig, A. Yamada, S. C. Nelson, M. Gruteser, and W. Meng. LAP: Lightweight anonymity and privacy. In *IEEE Security & Privacy*, 2012.

[35] S. Le Blond, D. Choffnes, W. Caldwell, P. Druschel, and N. Merritt. Herd: A scalable, traffic analysis resistant anonymity network for VoIP systems. In *ACM SIGCOMM*, 2015.

[36] S. Le Blond, D. Choffnes, W. Zhou, P. Druschel, H. Ballani, and P. Francis. Towards efficient traffic-analysis resistant anonymity networks. In *ACM SIGCOMM*, 2013.

[37] V. Liu, S. Han, A. Krishnamurthy, and T. Anderson. Tor instead of ip. In *ACM HotNets*, 2011.

[38] P. Mahadevan, D. Krioukov, M. Fomenkov, B. Huffaker, X. Dimitropoulos, k. claffy, and A. Vahdat. The Internet AS-level topology: Three data sources and one definitive metric. *ACM SIGCOMM CCR*, 36(1):17–26, Jan 2006.

[39] N. Mathewson and R. Dingledine. Practical traffic analysis: Extending and resisting statistical disclosure. In *PETS*, 2005.

[40] P. Mittal, A. Khurshid, J. Juen, M. Caesar, and N. Borisov. Stealthy traffic analysis of low-latency anonymous communication using throughput fingerprinting. In *ACM CCS*, 2011.

[41] U. Möller, L. Cottrell, P. Palfrader, and L. Sassaman. Mixmaster protocol (version 2). IETF Internet Draft, July 2003.

[42] S. J. Murdoch and G. Danezis. Low-cost traffic analysis of Tor. In *IEEE S&P*, 2005.

[43] L. Overlier and P. Syverson. Locating hidden servers. In *IEEE S&P*, 2006.

[44] A. Pfitzmann and M. Köhntopp. Anonymity, unobservability, and pseudonymity - a proposal for terminology. In *PETS*, 2001.

[45] Y. Rekhter and T. Li. A border gateway protocol 4 (BGP-4), 1995.

[46] J. Sankey and M. Wright. Dovetail: Stronger anonymity in next-generation internet routing. In *PETS*, 2014.

[47] R. Sherwood, B. Bhattacharjee, and A. Srinivasan. P^5: A protocol for scalable anonymous communication. In *IEEE S&P*, 2002.

[48] D. I. Wolinsky, H. Corrigan-Gibbs, B. Ford, and A. Johnson. Dissent in numbers: Making strong anonymity scale. In *Usenix OSDI*, 2012.

[49] X. Yang, D. Clark, and A. W. Berger. NIRA: a new inter-domain routing architecture. *IEEE/ACM Transactions on Networking*, 15(4):775–788, 2007.

[50] X. Zhang, H.-C. Hsiao, G. Hasker, H. Chan, A. Perrig, and D. G. Andersen. SCION: Scalability, control, and isolation on next-generation networks. In *IEEE S&P*, 2011.